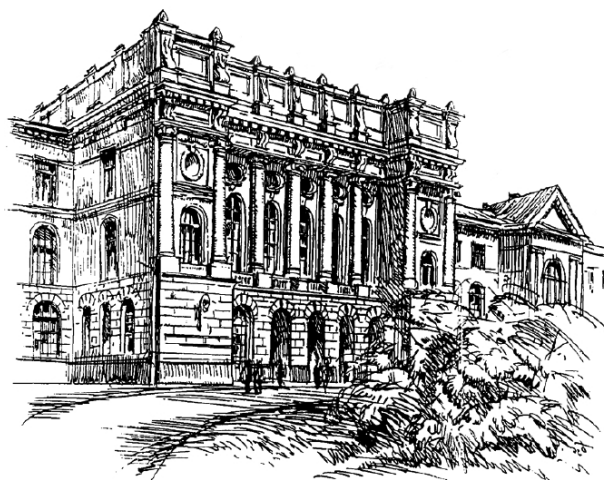


Министерство науки и высшего образования Российской Федерации
САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

Сборник материалов
научно-практической конференции студентов,
аспирантов и молодых ученых

22–23 апреля 2026 года



ПОЛИТЕХ-ПРЕСС

Санкт-Петербургский
политехнический университет
Петра Великого

Санкт-Петербург
2026

УДК 004
ББК 32.973
С56

Современные технологии в теории и практике программирования :
сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, 22–23 апреля 2026 г. – СПб. : ПОЛИТЕХ-ПРЕСС, 2026. – 411 с.

В сборнике публикуются материалы докладов студентов, аспирантов и молодых ученых, представленные на научно-практической конференции, проводимой Санкт-Петербургским политехническим университетом Петра Великого и организованной Институтом компьютерных наук и кибербезопасности при поддержке компании YADRO. Доклады отражают уровень подготовленности участников конференции в области применения современных средств и технологий разработки программного обеспечения.

Материалы сборника представляют интерес для специалистов в области компьютерных наук и информационных технологий, преподавателей, а также студентов и аспирантов технических вузов. В издании рассматриваются современные методы разработки программного обеспечения, включая подходы к созданию сложных систем и средств автоматизации инженерного проектирования.

Материалы сборника включают в себя актуальные вопросы реализации программных проектов различного назначения для теоретиков и практиков.

Редакционная коллегия:

Директор ВШПИ ИКНК *П. Д. Дробинцев*, профессор *И. Г. Черноруцкий*

ISBN 978-5-7422-9397-2

© Санкт-Петербургский политехнический
университет Петра Великого, 2026

Приветствие от компании YADRO

Дорогие участники!

Компания YADRO рада приветствовать вас на страницах сборника материалов научно-практической конференции студентов, аспирантов и молодых учёных «Современные технологии в теории и практике программирования».

В мире, где технологии меняют всё — от повседневной жизни до глобальных процессов, — именно молодые умы, полные идей и амбиций, становятся двигателем прогресса. Ваши исследования – важный шаг на пути к будущим решениям ключевых задач.

Поддержка молодых учёных важна для компании YADRO, поскольку научные исследования являются одним из ключевых элементов инженерной культуры. Мы убеждены в неисчерпаемости любознательности и исследовательского потенциала студентов и уверены, что данный сборник станет следующим значимым этапом в вашем профессиональном развитии.

Желаем всем участникам конференции дальнейших успехов в научной деятельности, полезных дискуссий и новых открытий!

Евгений Викторович Максимов,

Директор по развитию экосистемы и образовательных инициатив YADRO.

Тезисы докладов конкурса-конференции

Секция «Прикладная программная инженерия: продукты, сервисы и системы»

УДК 004.89

Н.А. Авдеева (4 курс бакалавриата),
А.В. Самочадин, к.т.н., доцент

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО АГЕНТА ДЛЯ ПОМОЩИ В ПЛАНИРОВАНИИ ПУТЕШЕСТВИЙ

Планирование путешествия почти всегда связано с рутинной работой: приходится подбирать даты и направление, проверять бюджет, искать отели и достопримечательности, сравнивать несколько сайтов и вручную собирать маршрут по дням. Имеющиеся сервисы решают отдельные части задачи – бронирование, навигацию или справочную информацию. Однако пользователю по-прежнему нужно самому формулировать запросы в виде форм, переключаться между разными системами и фактически формировать полную информацию о поездке из множества источников. Это плохо масштабируется и особенно неудобно, когда важны личные предпочтения и ограничения по времени [1], [5].

В работе рассматривается подход, при котором интеллектуальный агент оказывает помощь при планировании путешествия. Пользователь описывает свои планы обычным текстом на русском языке. Задача агента – понять такой запрос, уточнить недостающие параметры, подобрать подходящие точки интереса и собрать из них осмысленный маршрут с расписанием по дням. Ключевой идеей является не только автоматизация поиска, но и диалоговый формат взаимодействия, ближе к общению с живым помощником, чем к заполнению сложной формы [2].

Помимо этого, в работе формулируются и решаются несколько связанных задач. Во-первых, проводится анализ существующих туристических сервисов и на его основе формируется набор требований к агенту: поддержка естественного языка, персонализация, объяснимость рекомендаций и единое представление маршрута. Во-вторых, проектируется модуль обработки текста, который извлекает из пользовательских сообщений город, даты (включая общие формулировки), примерный бюджет, интересующие типы активности и количество людей. Модуль должен уметь работать с неполными запросами, хранить контекст диалога и внятно задавать уточняющие вопросы [6].

Третье направление – рекомендательная подсистема. Она использует открытые картографические и справочные источники для получения списка достопримечательностей в выбранном городе, после чего отбирает и ранжирует объекты с учётом предпочтений пользователя и бюджета. Для каждой выбранной точки агент формирует короткое текстовое

объяснение, почему именно это место попало в маршрут. Это может быть интересующая тематика, рейтинг, стоимость, а также расположение относительно других объектов [4].

Результат работы агента представляется пользователю в виде списка подобранных мест, расписания по дням и отображения на интерактивной карте. Расписание и карта позволяют наглядно увидеть предложенный план поездки без необходимости вручную сводить данные из разных источников [3].

Технически система проектируется как веб-приложение с разделением на клиентскую и серверную части и обменом данными по REST API в формате JSON. На сервере выделяются модули для обработки диалога, анализа текста, получения и фильтрации достопримечательностей, формирования расписания по дням и работы с хранилищем данных. Клиентская часть предоставляет чат-интерфейс для общения с агентом, карту для визуализации маршрута и представление расписания в табличной форме.

В качестве критериев качества разрабатываемого решения рассматриваются: насколько полно и корректно агент извлекает параметры из естественно-языковых запросов, насколько подобранные рекомендации соответствуют заявленным интересам и бюджету, насколько понятны пользователю объяснения рекомендаций, а также какова скорость ответа при обращении к внешним API.

ЛИТЕРАТУРА

1. Кумова Д. М. Использование платформ на базе искусственного интеллекта в сфере туризма // Сервис в России и за рубежом. — 2021. — Т. 15, № 3. — С. 18–26.
2. Жижилева К. А. Роль искусственного интеллекта в культуре туристской индустрии современного мира // Дизайн. Искусство. Индустрия. — 2024.
3. Богатырёв И. Н. Сервисные технологии и экотуризм: гармония природы и высоких технологий // Актуальные исследования. — 2025. — № 22 (257). — С. 15–18.
4. Волченко Т. А. Влияние цифровизации туризма на развитие индустрии питания, гостеприимства и рынков потребительских товаров // Trade, Service, Food Industry. — 2024. — Т. 4, № 2.
5. Видищева Е. В. Цифровая трансформация туристской отрасли: тенденции, вызовы и перспективы развития в России // Экономика и управление: проблемы, решения. — 2026.
6. Никулин Д. Ю., Краснов С. В. Интеллектуальные технологии в туризме // Журнал Волжского университета им. В. Н. Татищева. — 2017. — Т. 10, № 2.

УДК 004.42

П.Г. Авдышева (4 курс бакалавриата),
Л.П. Котлярова, ст. преподаватель

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА ПЛАТФОРМЕ ANDROID ДЛЯ КОРРЕКЦИИ ЦВЕТОВОСПРИЯТИЯ ВИЗУАЛЬНОГО КОНТЕНТА У ПОЛЬЗОВАТЕЛЕЙ С НАРУШЕНИЯМИ ЦВЕТОВОГО ЗРЕНИЯ

По статистике около 8% мужчин и 0,5% женщин обладают какой-либо формой цветовой слепоты. В современном мире визуальный контент является эффективным способом передачи информации. Однако люди, имеющие нарушения цветового зрения, могут сталкиваться с трудностями, так как они получают цветовую информацию в искаженном виде, что приводит к неверной интерпретации визуальных данных [1]. Проведенный анализ существующих решений, таких как DaltonicPointer, Color Blind Pal, Color Blind Test и ColorBlindClick Picker, выявил их функциональную ограниченность: они предлагают лишь узкий набор функций (например, только диагностику, только симуляцию или только определение цвета), которые могут помочь создать доступную среду для людей с цветовой слепотой. Ни одно из них не является всеобъемлющим инструментом, включающим одновременно коррекцию цветовосприятия, симуляцию зрения дальтоников, диагностику и идентификацию цветов в

едином приложении. Кроме того, при разработке интерфейсов также необходимо учитывать возможные особенности зрения людей. Это показывает важность создания комплексного решения, которое поможет скорректировать цветовосприятие визуального контента дальтониками.

Целью работы является разработка мобильного приложения под операционную систему Android, позволяющего корректировать восприятие цветов пользователями с различными формами дальтонизма, а также симулировать их видение цветов для людей без нарушений в целях повышения осведомленности и создания инклюзивного визуального контента. В работе решаются задачи анализа предметной области, выбора технологического стека, проектирования архитектуры и реализации ключевого функционала приложения.

Выбор технологического стека был произведен с упором на современные технологии. В качестве основного языка программирования выбран Kotlin [2], являющийся приоритетным для Android-разработки и обеспечивающий возможность мультиплатформенного расширения в будущем. Пользовательский интерфейс разрабатывается с применением современного декларативного фреймворка Jetpack Compose [3], который оптимизирует обновление экрана через механизм рекомпозиции. Для обработки статичных изображений используется CPU-подход с матричными преобразованиями цветов, тогда как для видеопотока в реальном времени задействуется графический процессор через OpenGL ES и GLSL-шейдеры [4], что обеспечивает более высокую производительность.

Проектирование приложения базируется на принципах чистой архитектуры (Clean Architecture). Это позволяет разделить проект на независимые слои: слой данных, отвечающий за источники данных и внешние API, доменный слой, содержащий бизнес-логику, модели данных и сценарии использования, а также слой представления, где реализован паттерн MVVM (Model-View-ViewModel) [5] с использованием ViewModel от Jetpack для обеспечения жизнеспособности данных при смене конфигурации устройства. Такая многослойность гарантирует слабую связанность компонентов, гибкость тестирования и возможность масштабирования проекта без рефакторинга существующей логики (рисунок 1).

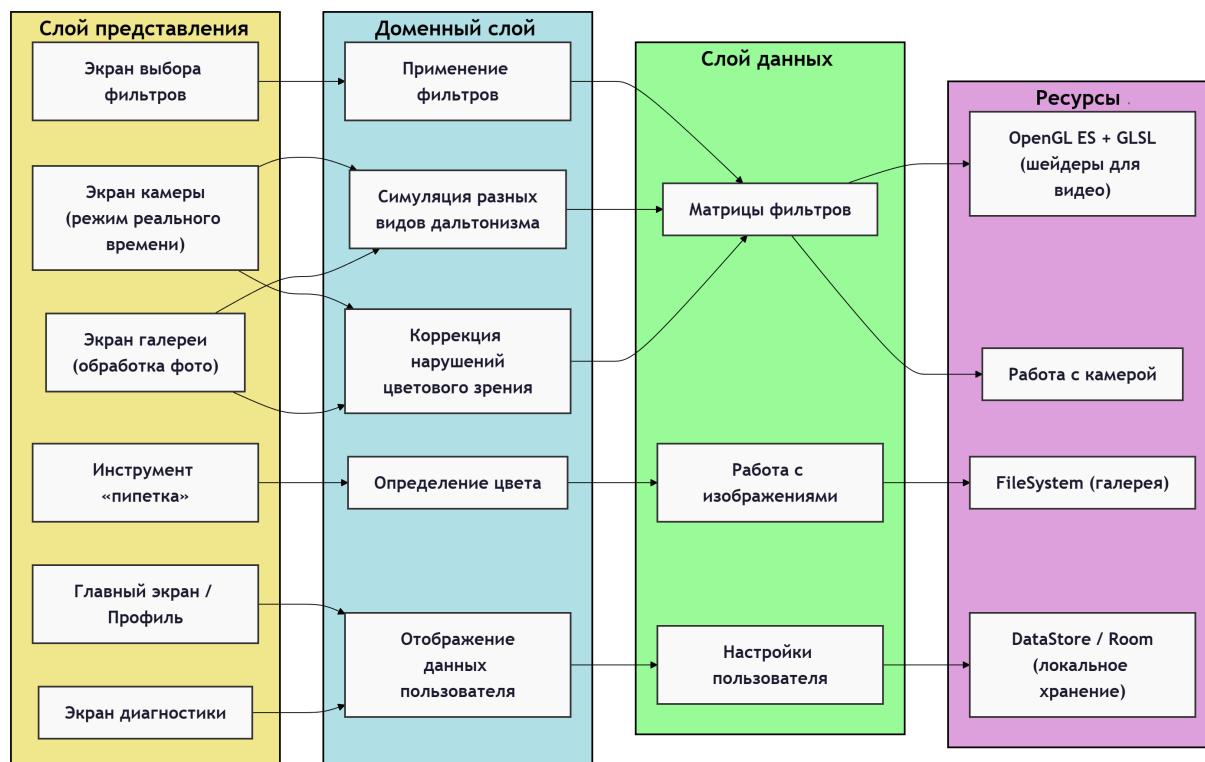


Рисунок 1 — Архитектура системы

Приложение поддерживает три основных типа дальтонизма: протанопию (нарушение восприятия красного цвета), дейтеранопию (нарушение восприятия зеленого цвета) и

тристанопию (нарушение восприятия синего цвета). При этом имеется возможность применить фильтры как к фотографиям из галереи, так и видеопотоку напрямую в приложении. Для определения цвета на фотографии реализован инструмент “пипетка”, который определяет код цвета в точке касания и отображает его название на естественном языке.

Результатом работы является прототип мобильного приложения на платформе Android. Уникальность заключается в разработке решения, объединяющего сразу несколько функций, необходимых для комфортной жизнедеятельности людей с нарушениями цветового зрения.

ЛИТЕРАТУРА

1. Almustanyir, A. A Global Perspective of Color Vision Deficiency: Awareness, Diagnosis, and Lived Experiences // Healthcare. – 2025. – URL: <https://www.mdpi.com/2227-9032/13/16/2031> (дата обращения: 10.03.2026).
2. Разрабатывайте приложения для Android с помощью Kotlin // Android Developers. – URL: <https://developer.android.com/kotlin?hl=ru> (дата обращения: 09.03.2026).
3. Введение в Jetpack Compose // METANIT. – URL: <https://metanit.com/kotlin/jetpack/1.1.php> (дата обращения: 09.03.2026).
4. GLSL Шейдеры // Mozilla MDN. – URL: https://developer.mozilla.org/ru/docs/Games/Techniques/3D_on_the_web/GLSL_Shaders (дата обращения: 09.03.2026).
5. Маринин, А. К. Выбор архитектуры для мобильных приложений // Известия Кабардино-Балкарского научного центра РАН. – 2024. – URL: <https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=izkab&paperid=902> (дата обращения: 10.03.2026).

УДК 004.415.2

Е.В. Алексеенко (4 курс бакалавриата),
А.В. Самочадин, к.т.н., доцент

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ СИСТЕМЫ ВОПРОСОВ И ОТВЕТОВ С ПОДДЕРЖКОЙ ПРОГРАММНОГО КОДА И МАТЕМАТИЧЕСКИХ ФОРМУЛ

В последние годы наблюдается активное развитие интернет-платформ коллективного обмена знаниями. Системы вопросов и ответов (Q&A-платформы) используются для обсуждения технических задач, обмена опытом и формирования баз знаний в различных областях науки и разработки программного обеспечения [2]. Подобные платформы формируют крупные сообщества пользователей, в которых коллективные знания используются для решения практических задач и поиска оптимальных ответов на поставленные вопросы [3].

Наиболее популярная платформа для разработчиков — Stack Overflow — предоставляет развитые инструменты публикации программного кода и подсветки синтаксиса для множества языков программирования [1]. Однако она не поддерживает нативный рендеринг математических формул в формате LaTeX. В то же время специализированные платформы, такие как Mathematics Stack Exchange, обеспечивают корректное отображение математической нотации, но ориентированы преимущественно на теоретические обсуждения и не предоставляют развитых инструментов работы с программным кодом. Таким образом, в современных веб-сервисах наблюдается функциональный разрыв между поддержкой программного и математического контента.

Целью данной работы является разработка веб-приложения системы вопросов и ответов, обеспечивающей интегрированную поддержку публикации программного кода и математических формул.

В рамках исследования был проведён анализ существующих Q&A-платформ, включая Stack Overflow, Mathematics Stack Exchange, Reddit, Quora и ResearchGate. На основе сравнительного анализа их функциональных возможностей были сформированы требования

к разрабатываемой системе, включающие поддержку Markdown-разметки, подсветку синтаксиса программного кода, рендеринг математических выражений в формате LaTeX, систему голосования и механизм тегирования контента.

Для реализации системы была выбрана многоуровневая клиент-серверная архитектура, представленная на рисунке 1. Клиентская часть реализована с использованием библиотеки React [5] и языка TypeScript, что обеспечивает компонентный подход к разработке пользовательского интерфейса и реактивное обновление данных. Для обработки технического контента используются библиотеки react-markdown для визуализации Markdown-разметки, Prism.js для подсветки синтаксиса программного кода и KaTeX для рендеринга математических формул [4].

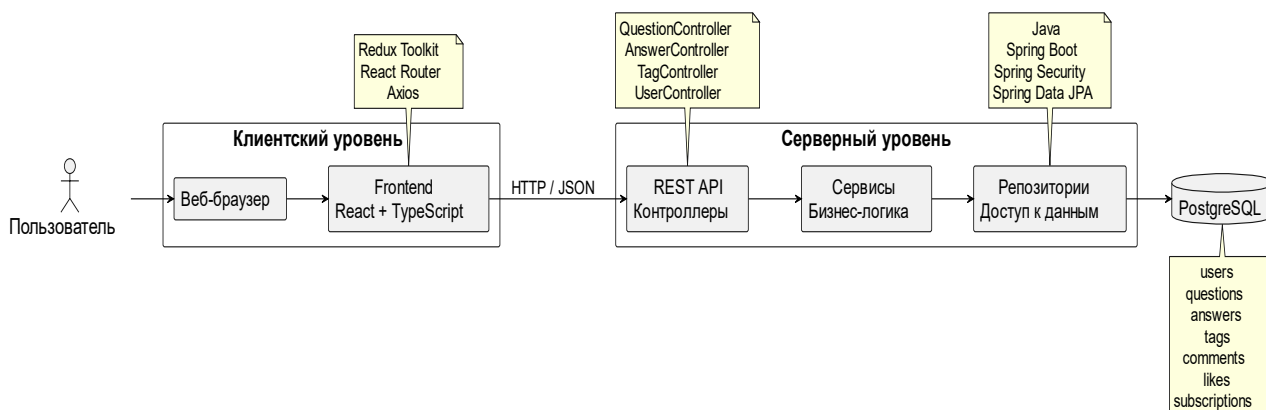


Рисунок 1 – Архитектура системы

Серверная часть реализована на языке Java с использованием фреймворка Spring Boot [6], обеспечивающего обработку HTTP-запросов, реализацию бизнес-логики и взаимодействие с базой данных через REST API. Для обеспечения безопасности применяется механизм аутентификации и авторизации на основе JWT-токенов. В качестве системы управления базами данных используется PostgreSQL [7].

Разработанная архитектура включает основные сущности системы: пользователей, вопросы, ответы, теги, комментарии и оценки. Контент публикаций хранится в формате Markdown с возможностью вставки программного кода и математических выражений в синтаксисе LaTeX. Визуализация контента выполняется на стороне клиента, что обеспечивает гибкость интерфейса и снижает нагрузку на сервер.

В рамках работы разработана демонстрационная версия системы, включающая 10 сущностей базы данных (пользователи, вопросы, ответы, теги, комментарии, оценки и подписки), REST API серверной части и клиентский интерфейс на React с поддержкой рендеринга LaTeX-формул через KaTeX, подсветки синтаксиса программного кода через Prism.js и управления состоянием через Redux Toolkit. Система поддерживает три роли пользователей: обычный пользователь, модератор и администратор.

В результате выполненной работы проведён сравнительный анализ Q&A-платформ и выявлен функциональный разрыв в области интеграции программного и математического контента; сформированы требования к системе и обоснован технологический стек; спроектирована многоуровневая клиент-серверная архитектура и реализовано веб-приложение, обеспечивающее поддержку подсветки программного кода и рендеринга LaTeX-формул в едином интерфейсе.

ЛИТЕРАТУРА

1. Calefato F., Lanubile F., Novielli N., Conte T. Community evolution on Stack Overflow // PLOS ONE. – 2021. – Т. 16, № 6. – e0253010. – DOI: 10.1371/journal.pone.0253010.
2. Anderson A., Huttenlocher D., Kleinberg J., Leskovec J. Discovering value from community activity on focused question answering sites: A case study of Stack Overflow // Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – 2012. – P. 850–858. – DOI: 10.1145/2339530.2339665.

3. Zoratto V., Godoy D., Aranda G.N. A study on influential features for predicting best answers in community question-answering forums // Information. – 2023. – Vol. 14, № 9. – P. 496. – DOI: 10.3390/info14090496.
4. KaTeX: the fastest math typesetting library for the web [Электронный ресурс]. – URL: <https://katex.org/>
5. React Documentation [Электронный ресурс]. – URL: <https://react.dev/>
6. Spring Boot Documentation [Электронный ресурс]. – URL: <https://spring.io/projects/spring-boot>
7. PostgreSQL: Documentation [Электронный ресурс]. – URL: <https://www.postgresql.org/docs/>

УДК 004.453

В.И. Андреев, Г. Мамедова (4 курс бакалавриата),
И.В. Зайцев, ст. преподаватель

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ФОРМИРОВАНИЯ РАСПИСАНИЙ ОБРАЗОВАТЕЛЬНЫХ УЧРЕЖДЕНИЙ

Составление учебного расписания в образовательных учреждениях представляет собой одну из центральных задач управления учебным процессом и требует при его формировании одновременного внимания к большому количеству параметров, среди которых учебные планы, нагрузка преподавателей, доступность аудиторий. При увеличении числа учебных групп и дисциплин задача составления расписания становится трудоёмкой [1].

На практике в образовательных организациях используются специализированные программные системы управления учебным процессом, однако их внедрение и полноценное использование требуют значительных финансовых затрат. В результате используются ограниченные версии продуктов. Поэтому расписание составляется вручную, что приводит к значительным временным издержкам: основное внимание уделяется обеспечению непротиворечивости, тогда как возможности оптимизации используются ограниченно [2].

С учетом этого актуальной является разработка программных средств, автоматизирующих процесс формирования расписания. Задача данной работы заключается в автоматизированном составлении учебного расписания для подразделения университета.

Целью работы является снижение трудоемкости подготовки расписания за счет разработки программной системы автоматизированного формирования учебных расписаний. Для достижения указанной цели поставлены следующие задачи:

1. Анализ существующих подходов к автоматизации формирования расписания;
2. Разработка архитектуры программной системы генерации расписания;
3. Реализация алгоритмов автоматизированного распределения занятий [3];
4. Экспериментальная оценка эффективности созданного решения.

Разрабатываемая система представляет собой программный комплекс с клиент-серверной архитектурой. При этом клиентская часть разработана с использованием фреймворка Angular, серверная часть построена на FastAPI. Хранение данных производится в реляционной СУБД PostgreSQL. Взаимодействие клиента и сервера обеспечивается при помощи REST API. Импорт исходных данных осуществляется через обработку Excel-файлов. В архитектуру включены следующие основные модули:

- модуль генерации расписания [4];
- модуль импорта и экспорта данных;
- система хранения данных;
- веб-интерфейс пользователя.

Наиболее трудоемкий из них – модуль генерации расписания, реализующий распределение учебных занятий по временным слотам и аудиториям, с учетом жёстких и мягких ограничений.

К жёстким ограничениям относятся:

- для одной учебной группы невозможно одновременно проводить разные занятия (если группа не разделена на подгруппы);
- преподаватель не может одновременно вести несколько занятий;
- аудитория не может использоваться более чем для одного занятия в один момент времени.

Нарушение жёстких ограничений делает расписание неприемлемым.

Мягкие ограничения направлены на повышение удобства расписания и включают минимизацию нежелательных ситуаций (например, «окон» между занятиями у студентов, сочетания очных и дистанционных занятий в один день), а также учёт предпочтений преподавателей [5]. Качество полученного расписания оценивается при помощи штрафной функции, измеряющей несоответствие мягким ограничениям. Оптимизация заключается в получении допустимого расписания, удовлетворяющего всем жестким ограничениям и минимизирующего общее значение штрафной функции.

В разрабатываемой системе генерация расписания осуществляется с использованием метаэвристических методов оптимизации. На первом этапе формируется начальное допустимое расписание, после чего его улучшение выполняется при помощи генетического алгоритма. В алгоритме применяются операции селекции, кроссовера и мутации, а также механизм элитизма для сохранения лучших решений. Для повышения устойчивости поиска используется элемент имитации отжига, позволяющий принимать некоторые ухудшающие решения на ранних этапах оптимизации. После завершения работы генетического алгоритма проводится дополнительная оптимизация расписания методом локального поиска [6].

Автоматизация процесса формирования учебных расписаний значительно снижает трудоемкость подготовки и упрощает корректировку расписания при изменении исходных данных, что способствует созданию более удобного расписания для всех участников образовательного процесса за счет уменьшения количества нежелательных ситуаций по сравнению с вариантами, созданными вручную. Система также выполняет анализ загрузки аудиторий, распределения нагрузки преподавателей и полноты учёта их предпочтений, что может быть полезным для выявления «узких мест» и принятия управленческих решений.

ЛИТЕРАТУРА

1. Mallari C. B., San Juan J. L., Li R. The university coursework timetabling problem: An optimization approach to synchronizing course calendars // *Computers & Industrial Engineering*. – 2023. – Vol. 184. – P. 109561. – DOI: 10.1016/j.cie.2023.109561. – Режим доступа: <https://www.sciencedirect.com/science/article/pii/S0360835223005855> (дата обращения: 15.03.2026).
2. Diallo F. P., Tudose C. Optimizing the Scheduling of Teaching Activities in a Faculty // *Applied Sciences*. – 2024. – Vol. 14, No. 20. – P. 9554. – DOI: 10.3390/app14209554. – Режим доступа: <https://doi.org/10.3390/app14209554> (дата обращения: 13.10.2025).
3. Goh S. L., Kendall G., Sabar N. R. Improved local search approaches to solve the post enrolment course timetabling problem // *European Journal of Operational Research*. – 2017. – Vol. 261, No. 1. – P. 17–29. – DOI: 10.1016/j.ejor.2017.01.040. – Режим доступа: <https://www.sciencedirect.com/science/article/pii/S0377221717300759> (дата обращения: 10.03.2026).
4. Abdipoor S., Yaakob R., Goh S. L., Abdullah S. Meta-heuristic approaches for the University Course Timetabling Problem // *Intelligent Systems with Applications*. – 2023. – Vol. 19. – P. 200253. – DOI: 10.1016/j.iswa.2023.2000789. – Режим доступа: <https://www.sciencedirect.com/science/article/pii/S2667305323000789> (дата обращения: 15.03.2026).
5. Nguyen V. D., Nguyen T. An SHO-based approach to timetable scheduling: a case study // *Journal of Information and Telecommunication*. – 2021. – Vol. 5, No. 4. – P. 421–439. – DOI: 10.1080/24751839.2021.1935644. – Режим доступа: <https://doi.org/10.1080/24751839.2021.1935644> (дата обращения: 15.03.2026).

6. Bashab A., Ibrahim A. O., Tarigo Hashem I. A., Aggarwal K., Mukhlif F., Ghaleb F. A., Abdelmaboud A. Optimization Techniques in University Timetabling Problem: Constraints, Methodologies, Benchmarks, and Open Issues // Computers, Materials & Continua. – 2023. – Vol. 74, No. 3. – P. 6462. – DOI: 10.32604/cmc.2023.034051. – Режим доступа: <https://doi.org/10.32604/cmc.2023.034051> (дата обращения: 11.02.2026).

УДК 004.415.2

А.А. Антонов (2 курс магистратуры),
А.Д. Ковалев., к.т.н., доцент

МЕТОД ВЫБОРА СТРАТЕГИИ РЕНДЕРИНГА WEB-ПРИЛОЖЕНИЙ

Современные Web-приложения предъявляют высокие требования к производительности, масштабируемости, поисковой оптимизации и уровню сопровождения. В условиях разнообразия стратегий рендеринга контента — Client-Side Rendering (CSR), Server-Side Rendering (SSR), Static Site Generation (SSG) и Incremental Static Regeneration (ISR) — отсутствует универсальный подход к их выбору. В данной работе предлагается метод выбора стратегии рендеринга Web-приложения на основе формализации нефункциональных требований, архитектурных ограничений и эксплуатационных характеристик.

При CSR HTML-разметка формируется в браузере пользователя после загрузки JavaScript-приложения и данных с сервера. Такой подход широко использовался в ранних SPA-архитектурах и обеспечивает высокий уровень интерактивности. Однако исследования показывают, что CSR негативно влияет на показатели первичной загрузки и воспринимаемую производительность, особенно на устройствах с ограниченными ресурсами [1]. Кроме того, CSR усложняет индексацию контента поисковыми системами, несмотря на развитие headless-рендеринга.

SSR предполагает генерацию HTML-страницы на сервере для каждого пользовательского запроса. Данный подход улучшает показатели First Contentful Paint и упрощает задачу поисковой оптимизации [1][2]. С точки зрения архитектуры распределённых систем SSR увеличивает нагрузку на сервер и усложняет горизонтальное масштабирование, что требует более строгого управления ресурсами и кэшированием [3].

SSG основывается на предварительной генерации HTML-страниц на этапе сборки. Такой подход минимизирует задержки при доставке контента и хорошо согласуется с CDN-ориентированными архитектурами. Согласно практическим исследованиям производительности, SSG демонстрирует лучшие показатели стабильности и отказоустойчивости, однако ограничен в сценариях с динамически изменяемыми данными [1].

ISR представляет собой развитие SSG и позволяет частично регенерировать страницы после развертывания. Этот подход снижает компромисс между актуальностью данных и производительностью доставки контента [4]. Тем не менее, вопросы согласованности данных и предсказуемости поведения системы при регенерации страниц требуют дополнительного архитектурного анализа [3].

Выбор стратегии рендеринга является многокритериальной задачей, затрагивающей нефункциональные требования, такие как производительность, масштабируемость, стоимость разработки и сложность сопровождения [3]. Отсутствие формального подхода приводит к архитектурным решениям, которые сложно оптимизировать и сопровождать в долгосрочной перспективе.

Для формализации задачи предлагается рассматривать стратегию рендеринга как функцию, оптимизируемую по вектору нефункциональных характеристик (1), включающих показатели производительности, актуальности данных, масштабируемости и стоимости эксплуатации [1], [5]. Предлагается использовать вектор критериев, где каждый параметр нормализуется в диапазоне [0,1]:

$$R = \langle SEO, TTFB, TTI, Freshness, Scalability, Cost, Complexity \rangle \quad (1)$$

Сравнительный анализ стратегий рендеринга может быть представлен в виде матрицы, отражающей степень соответствия каждой стратегии заданным критериям. Пример подобного анализа представлен в таблице 1. Подобные методы широко применяются при оценке архитектурных альтернатив в программной инженерии [3].

Таблица 1 – Сравнительный анализ стратегий рендеринга

Стратегия	SEO	TTI	Freshness	Scalability	Cost
CSR	0.2	0.6	0.9	0.9	0.8
SSR	0.9	0.7	0.9	0.6	0.5
SSG	1.0	0.9	0.2	1.0	0.9
ISR	0.9	0.85	0.7	0.85	0.8

Нормализованные значения критериев в матрице не являются результатом точных измерений, а представляют собой экспертную оценку, применяемую для сравнительного анализа архитектурных альтернатив. Используемые критерии носят качественный или смешанный характер и не обладают универсальными абсолютными метриками, что делает прямое количественное измерение невозможным.

Числовые значения приведены к нормализованной шкале [0,1] и отражают относительное положение стратегий рендеринга друг относительно друга, а не их абсолютные характеристики. Например, низкое значение SEO для CSR указывает на существенно меньшую эффективность поисковой индексации по сравнению с SSR и SSG из-за клиентской генерации контента, тогда как высокое значение для SSR обусловлено серверной генерацией HTML доступного поисковым системам при первом запросе.

Важно отметить, что конкретные числовые значения не являются критичными для корректности метода: при сохранении относительных соотношений между стратегиями итоговый результат выбора остаётся устойчивым. Матрица может быть откалибрована под конкретный домен, уточнена эмпирическими метриками или расширена с использованием формальных методов экспертной оценки. Основная ценность предложенного подхода заключается не в самих числах, а в формализации и воспроизводимости процесса выбора стратегии рендеринга в условиях промышленной разработки.

Использование взвешенной суммы критериев (2) позволяет формализовать процесс принятия архитектурного решения и сделать его воспроизводимым в различных проектных контекстах [3].

$$Score(S) = \sum_{i=1}^n w_i \cdot R_i(S) \quad (2)$$

где w_i — веса критериев, определяемые требованиями продукта.

Предложенный метод может быть интегрирован в процессы архитектурного анализа и технического ревью, что соответствует современным практикам управления жизненным циклом программного обеспечения [3]. Такой подход способствует снижению архитектурного долга и повышению предсказуемости эксплуатационных характеристик Web-приложений [1][5].

ЛИТЕРАТУРА

1. Iskandar T. F., Lubis M., Kusumasari T. F., Lubis A. R. Comparison between client-side and server-side rendering in the web development // IOP Conference Series: Materials Science and Engineering. — 2020. — Vol. 801. — P. 012136. DOI: 10.1088/1757-899X/801/1/012136.
2. Vallamsetla K. The Impact of Server-Side Rendering on UI Performance and SEO // International Journal of Scientific Research in Computer Science, Engineering and Information Technology. — 2024. — Vol. 10, No. 5. — P. 795–804.
3. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. — Sebastopol: O'Reilly Media, 2020. — 422 p. ISBN 978-1-492-04345-4.
4. Vercel Inc. Incremental Static Regeneration (ISR) in Next.js. — 2026. — Электронный ресурс. Режим доступа: <https://vercel.com/docs/incremental-static-regeneration> (дата обращения: 10.02.2026).
5. Google Developers. Web Vitals. — 2020. — Электронный ресурс. Режим доступа: <https://web.dev/vitals/> (дата обращения: 10.02.2026).

РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ ДЛЯ АНАЛИЗА И ПРОГНОЗИРОВАНИЯ
ИНЦИДЕНТОВ НА ОСНОВЕ ЗАЯВОК HELPDESK

Современные IT-отделы испытывают значительную нагрузку из-за роста объёма неструктурированных заявок, поступающих через системы поддержки пользователей HelpDesk. Это приводит к увеличению времени обработки обращений и простоем оборудования. Существующие системы управления (ITSM), как правило, ориентированы на реактивную модель, а внедрение интеллектуальных модулей затруднено высокой стоимостью лицензии (например, ServiceNow, Jira Service Management), либо необходимостью передачи данных во внешние облачные сервисы, что недопустимо в рамках политики информационной безопасности [1].

Эффективное управление IT-инфраструктурой требует перехода от реактивной модели устранения сбоев к проактивной и предиктивной аналитике, что позволит заранее прогнозировать и выявлять пики обращений, нагрузки, увеличить скорость обработки инцидентов и оптимизировать распределение ресурсов [2]. Задача классификации и прогнозирования инцидентов осложняется высокой вариативностью тематик обращений - от технических сбоев до запросов на доступ к ПО, - что требует применения гибких методов машинного обучения, адаптированных к специфическому контексту организации.

Таким образом, целью данной работы является разработка автономного программного комплекса для интеллектуального анализа и прогнозирования потока заявок HelpDesk в локальном контуре с использованием передовых технологий.

Для достижения поставленной цели реализованы следующие задачи:

- Анализ предметной области и обзор существующих подходов.
- Проектирование архитектуры, сбор и подготовка данных.
- Реализация NLP-модуля для обработки текстовых описаний заявок.
- Разработка, обучение и оценка качества прогнозных моделей.
- Реализация программного комплекса с модульной архитектурой.

В качестве инструмента классификации инцидентов и оценки их критичности выбран алгоритм градиентного бустинга CatBoost, демонстрирующий высокую точность на категориальных и текстовых признаках без сложной предобработки и превосходящий ряд традиционных методов, в частности SVM [3]. Прогнозирование временных рядов и выявление сезонных колебаний нагрузки реализовано аддитивной моделью Prophet [4]. NLP-конвейер реализован на базе связки TF-IDF и scikit-learn [5]. Такой выбор обусловлен вычислительной эффективностью при обработке текстов на русском языке на стандартных офисных серверах, в отличие от ресурсоёмких трансформерных моделей (например, BERT).

Практическая значимость работы заключается в интеграции механизмов активного обучения (Human-in-the-Loop) и мониторинга дрейфа данных (Data Drift) в единый программный контур. Предложенное решение позволяет техническому специалисту верифицировать рекомендации как «сложные примеры» с помощью отзывов и возможности инициирования переобучения моделей без перерыва в работе системы для эволюционирования моделей, при достижении заданного порогового сигнала. Это решает проблему деградации качества прогнозирования в условиях динамично изменяющейся IT-среды.

По результатам экспериментального применения разработанного комплекса достигается автоматизация до 80% типовых заявок при значении метрики F1-score > 0.8 для ключевых категорий инцидентов и сокращение среднего времени реакции на критические инциденты за счет раннего обнаружения аномалий в потоке обращений [6]. Дополнительным преимуществом является возможность эксплуатации системы в локальной инфраструктуре

без использования облачных сервисов, что обеспечивает соответствие требованиям информационной безопасности и минимизирует стоимость владения.

Таким образом, разработанное программное средство представляет собой эффективный инструмент интеллектуальной поддержки принятия решений в организациях с высокой изменчивостью ИТ-процессов.

ЛИТЕРАТУРА

1. Блинникова А. В., Нестерова Ю. О. Управление инцидентами в ITSM с использованием искусственного интеллекта // Информационные технологии и математическое моделирование в управлении сложными системами. - 2020. - №7. - С.1-9. [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/upravlenie-intsidentami-v-itsm-s-ispolzovaniem-iskusstvennogo-intellekta> (дата обращения: 20.01.2026).
2. Никулин В. В., Шибайкин С. Д., Соколова М. Применение методов машинного обучения для автоматизированной классификации и маршрутизации в библиотеке ITIL // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. - 2022. - №1.- С.42-52. - DOI: 10.24143/2073-5529-2022-1-42-52.
3. Prokhorenkova L., Gusev G., Vorobev A., Dorogush A. V., Gulin A. CatBoost: unbiased boosting with categorical features // Advances in Neural Information Processing Systems 31 (NeurIPS 2018). - 2018. [Электронный ресурс]. URL: <https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html> (дата обращения: 02.02.2026).
4. Груздев А.В. Прогнозирование временных рядов с помощью Facebook Prophet, ETNA, sktime и LinkedIn Greykite: практическое руководство. - М.: ДМК Пресс, 2023 - ISBN 978-5-93700-212-9.
5. Дементьев В. Е., Киреев С. Х. Выбор алгоритмов машинного обучения для классификации текстовых документов // Прикаспийский журнал: управление и высокие технологии. - 2022. - №2. - DOI: 10.24412/2782-2141-2022-2-22-52.
6. Трунов Е. Алгоритм обнаружения аномального поведения пользователей автоматизированных систем на основе машинного обучения // Вестник Астраханского технического университета. Серия: Управление, вычислительная техника и информатика. - 2025. - №4.- С.33-52. - DOI: 10.24143/2072-9502-2025-4-33-42.

УДК 004.415.25

А.В. Богданова (2 курс бакалавриата),
И.В. Зеленчук, ст. преподаватель

СОЗДАНИЕ МОБИЛЬНОГО ПОМОЩНИКА ДЛЯ НАПИСАНИЯ СТИХОВ

Идея проекта сформирована самостоятельно на основе личной потребности: отсутствия на рынке удобного инструмента для начинающих поэтов, которое подбирает рифмы и анализирует метрику стихов. *Целью* работы является создание мобильного приложения, упрощающего процесс стихосложения. Целевая аудитория работы — «ленивые поэты», которым важна простота, скорость и минимизация ручных действий при работе со стихотворным текстом.

В ходе работы были поставлены и решены следующие задачи:

1. Провести анализ предметной области и существующих решений. Изучить функциональные возможности существующих приложений и сервисов для поэтов, выявить их преимущества и недостатки, сформировать требования к разрабатываемому продукту с учётом потребностей целевой аудитории [1].
2. Обосновать выбор технологического стека и спроектировать архитектуру приложения. Определить подходящие инструменты разработки, выбрать архитектурный подход, разработать структуру модулей и схему взаимодействия компонентов [2][3].
3. Реализовать базовый функционал приложения [2][3].

4. Обеспечить надёжное хранение данных и стабильность работы [2][3].

5. Подготовить приложение к публикации и сопровождению [4][5].

Предлагаемое решение реализовано в виде мобильного приложения «РифмоПульт» для операционной системы Android. Разработка ведётся с использованием языка программирования Kotlin и среды Android Studio. Приложение распространяется через магазин приложений RuStore и активно поддерживается. Для подбора рифм приложение автоматически обращается к внешнему веб-сайту, специализирующемуся на этой задаче. При выделении слова в редакторе пользователь может вызвать команду «Рифма к слову», после чего приложение извлекает сто наиболее подходящих вариантов и отображает их в панели. Дополнительно реализована возможность перетаскивания рифм из панели в текст, что также ускоряет процесс. Ещё одна функциональность - подсчёт слогов, необходимый для соблюдения ритма стиха. Он реализован как подсчёт гласных. Подсчитанные слоги отображаются в конце строки и для ненавязчивости имеют серый цвет.

Функциональность проверялась как вручную, так и модульным тестированием. Среди исправленных проблем — накопление маркеров слогов при повторном открытии заметки, некорректное поведение кнопок отката и сохранения, а также отображение даты последнего изменения.

Демонстрационная версия приложения «РифмоПульт» (версия 1.1) имеет размер установочного пакета 4,89 МБ, что обеспечивает быструю загрузку и установку даже при ограниченных ресурсах телефона. Время отклика интерфейса не превышает 100 мс, что достигается за счёт использования корутин Kotlin для асинхронного выполнения операций и выноса сетевых запросов за пределы основного потока. Выбор архитектуры MVVM с паттерном Repository обусловлен необходимостью чёткого разделения ответственности: слой представления отвечает за отображение данных, доменный слой — за бизнес-логику, а репозиторий абстрагирует источники данных, что упрощает тестирование и дальнейшее расширение функционала. Для работы с сетью применена библиотека Ktor — легковесное и гибкое решение, поддерживающее асинхронные запросы и легко интегрируемое с корутинами. Парсинг HTML-ответов внешнего сервиса реализован через JSoup, что позволяет надёжно извлекать данные без избыточной сложности. Локальное хранение заметок организовано с помощью Room — официальной библиотеки для работы с SQLite, обеспечивающей типобезопасный доступ к данным и автоматическую миграцию схем.

Инновационная ценность решения заключается в интеграции функции подбора рифм непосредственно в контекст редактора: пользователь получает до 100 вариантов рифмы без перехода во внешние приложения, а механизм drag-and-drop позволяет мгновенно вставлять выбранное слово в текст.

Эффективность применённого стека технологий подтверждается стабильной работой приложения, отсутствием утечек памяти при длительном использовании и положительными отзывами первых пользователей в RuStore.

В ходе выполнения работы получены следующие результаты:

- Разработано и опубликовано мобильное приложение «РифмоПульт» для платформы Android;
- Реализованы базовые, но востребованные функции: подбор рифм через парсинг веб-ресурса, подсчёт слогов, редактор с историей изменений и экспорт чистого текста;
- Проект активно поддерживается: уже готовится к выпуску версия 1.2 с улучшениями интерфейса и дополнительными удобствами.

Материалы, иллюстрирующие выполнение работы:

- Скриншоты, видео-демонстрация, доступны на странице приложения в RuStore: Мобильное приложение «РифмоПульт» в каталоге RuStore. [Электронный ресурс] URL: <https://www.rustore.ru/catalog/app/com.example.rifmopult> (дата обращения: 11.03.2026).
- Исходный код проекта опубликован на GitHub: Исходный код проекта «РифмоПульт» [Электронный ресурс] URL: <https://github.com/AV-Bog/RifmoPult> (дата обращения: 11.03.2026).

– Техническая документация: Архитектура проекта «РифмоПульт» [Электронный ресурс] URL: <https://github.com/AV-Bog/RifmoPult/blob/main/ARCHITECTURE.md> (дата обращения: 11.03.2026).

Решение находится в активной разработке, планируется его дальнейшее развитие, включая улучшение алгоритмов анализа стиха и расширение возможностей редактора.

ЛИТЕРАТУРА

1. Фитцпатрик Р. Спроси маму: как общаться с клиентами и подтвердить правоту своей стартап-идеи / Р. Фитцпатрик ; пер. с англ. – Москва : Манн, Иванов и Фербер, 2020. – 240 с.
2. Язык программирования Kotlin. [Электронный ресурс] URL: <https://kotlinlang.org/docs/home.html> (дата обращения: 11.03.2026).
3. Руководство по языку Kotlin. [Электронный ресурс] URL: <https://metanit.com/kotlin/tutorial/> (дата обращения: 11.03.2026).
4. Политика в отношении контента для разработчиков Google Play. [Электронный ресурс] URL: <https://play.google/intl/ru/developer-content-policy/> (дата обращения: 11.03.2026).
5. Справка для разработчиков RuStore. [Электронный ресурс] URL: <https://www.rustore.ru/help/developers> (дата обращения: 11.03.2026).

УДК 004.45

Д.А. Божко (3 курс бакалавриата),
И.В. Зайцев, ст. преподаватель

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ КУРСА ПО ЯЗЫКУ SQL НА ОСНОВЕ СУБД POSTGRESQL

Интерес к языку SQL в наши дни значительно повысился благодаря расширяющемуся применению аналитики данных в различных областях, и поэтому включается в программы обучения все большего числа специалистов, перед которыми ставятся в первую очередь задачи извлечения и обработки данных (в отличие от разработчиков ПО). Особенно активно в нашей стране растет применение СУБД PostgreSQL [2]. Поэтому возникает необходимость в создании удобного онлайн-ресурса для тренировки и тестирования навыков работы с языком SQL, необходимых в сфере аналитики данных. Ресурс планируется использовать с создаваемыми преподавателями учебными базами данных (соответствующими профилю обучаемых специалистов, например, туризм, торговое дело и т.п.) совместно с существующими системами управления учебными курсами (такими, как Moodle), благодаря чему нет необходимости создавать аккаунты студентов и обрабатывать их персональные данные. При использовании расширенного модуля тестирования для Moodle, разрабатываемого в параллельном проекте, возможен учет не только правильности результата запроса, но и его эффективности [1].

Таким образом, цель работы – создание веб-ресурса SQL Trainer, позволяющей студентам выполнять SQL-запросы с анализом производительности (EXPLAIN ANALYZE) [2], а преподавателям – загружать соответствующие специальности учебные базы и использовать их для тренировочных и тестовых заданий в существующей системе управления курсами (Moodle).

Для достижения цели решаются следующие задачи:

1. Проектирование безопасной архитектуры с изоляцией сессий.
2. Реализация бэкенда на Java (Servlets, JSP) [3] с PostgreSQL через JDBC.
3. Разработка модуля анализа запросов с визуализацией планов выполнения [2].
4. Создание механизма загрузки SQL-скриптов преподавателем.
5. Реализация мониторинга сессий и экспорта CSV с электронной подписью.
6. Контейнеризация Docker для упрощения развертывания [4].

Архитектура платформы представлена на рисунке 1. Бэкенд на Java 17 [3] взаимодействует с PostgreSQL 15 [2] через JDBC. Для безопасности используются отдельные роли БД: students (выполнение только запросов разрешенного вида в рамках настраиваемых преподавателями ограничений, например, не более 3 сек на выполнение запроса и не более 20 первых строк в таблице результата) и teacher_role (загрузка и удаление своих баз данных и настройка их видимости и ограничений, например, разрешенного времени доступа).

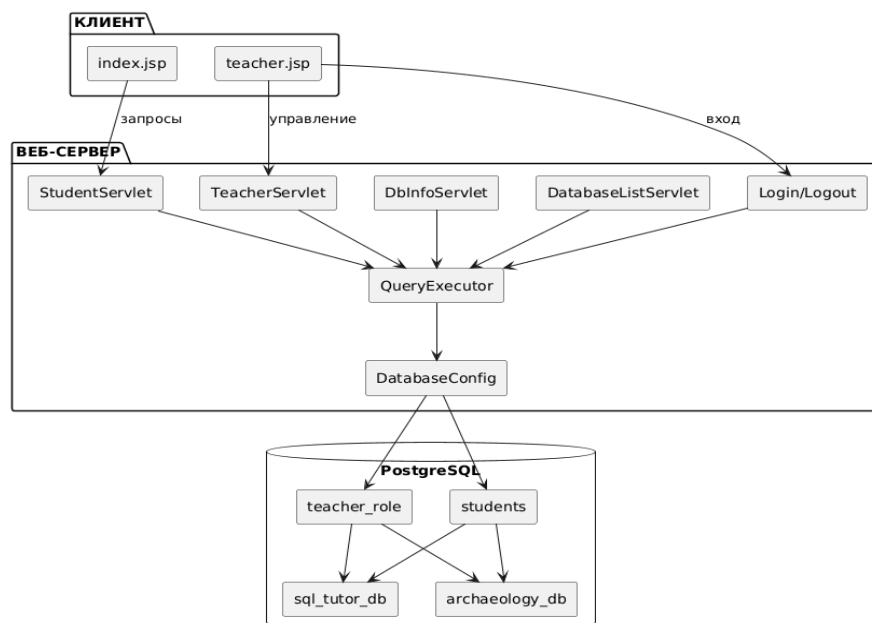


Рисунок 1 – Архитектура платформы SQL Trainer

Фронтенд разработан на чистом HTML, CSS и JavaScript с использованием Tailwind для стилизации, что обеспечивает минималистичный и отзывчивый интерфейс. Взаимодействие с сервером осуществляется через асинхронные fetch-запросы.

Ключевой особенностью платформы является возможность анализа эффективности запросов. Для каждого выполненного запроса автоматически формируется вывод EXPLAIN ANALYZE [2], позволяющий студентам видеть план выполнения и оценивать влияние индексов на производительность. Разработанное приложение контейнеризировано с помощью Docker [4], что позволяет развернуть его одной командой как на локальной машине, так и на сервере.

Для повышения корректности результатов тестирования знаний предусмотрен экспорт результатов запросов в CSV-файл, содержащий метаданные: время выполнения запроса, количество строк, дату генерации и уникальную электронную подпись [1], однако для большей надежности требуются дополнительные меры по исключению доступа тестируемых студентов к нейросетям и поисковым системам (например, работа ресурса в локальной сети без доступа в интернет и запрет мобильных устройств). Основы проектирования баз данных и принципы работы СУБД были учтены при разработке структуры учебных баз [5].

ЛИТЕРАТУРА

1. Швец В. Л., Вагин Д. В. Разработка системы мониторинга SQL запросов для учебных баз данных // Современные технологии в теории и практике программирования. – СПб.: Изд-во Политехн. ун-та, 2025. – С. 45-48.
2. PostgreSQL Documentation. Chapter 14. Performance Tips [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/15/performance-tips.html> (дата обращения: 01.03.2026).
3. Johnson R., Hoeller J. Professional Java Development with the Spring Framework. – Wiley Publishing, 2023. – 800 p.
4. Docker Documentation. Get Started with Docker [Электронный ресурс] Режим доступа: <https://docs.docker.com/get-started/> (дата обращения: 01.03.2026).
5. Кузнецов С. Д. Основы современных баз данных. – 3-е изд. – М.: Интернет-Университет Информационных Технологий, 2022. – 488 с.

РАЗРАБОТКА AI-АГЕНТА ДЛЯ ТРЕКИНГА ПИТАНИЯ С ИСПОЛЬЗОВАНИЕМ
ГИБРИДНОГО ВЕКТОРНОГО ПОИСКА

Современные системы мониторинга питания, например, FatSecret и MyFitnessPal, требуют от пользователя ручного ввода данных, что снижает долгосрочную вовлеченность [1]. Кроме того, они используют статические формулы расчета расхода энергии (TDEE), не учитывающие метаболическую адаптацию организма, и скрывают источники происхождения нутриентных баз данных.

Целью данной работы является разработка интеллектуального программного агента, обеспечивающего автоматизированный парсинг приемов пищи на естественном языке, прозрачную интеграцию международных и отечественных баз данных и динамический расчет метаболизма пользователя.

Разработанная система представляет собой микросервисное приложение, взаимодействующее с пользователем через интерфейс мессенджера Telegram. Основная архитектура представлена следующими компонентами:

1. Оркестрация LLM реализована на фреймворке LangGraph [2], он используется для построения графа состояний агента, который работает по пайплайну *extract, search, save, format*. Архитектура является модельнезависимой, что позволяет использовать как облачные решения, так и локальные развертывания для соблюдения требований ФЗ 152 о защите персональных данных.

2. Для сопоставления пользовательского ввода, например, «вареная куриная грудка 153 грамма», с нормализованной базой данных используется СУБД Qdrant. Применяется гибридный поиск (Hybrid Search), объединяющий плотные векторы (Dense Embeddings) для семантического сходства и разреженные векторы (Sparse Vectors) для точного совпадения ключевых слов [3]. В качестве мультиязычных dense-эмбеддингов могут использоваться модели семейства multilingual-e5 [4], а в качестве sparse-компоненты – BM25 [5]. Это позволило исключить ресурсоемкий этап LLM-перанкинга и сократить время поиска.

3. База данных PostgreSQL с использованием типа данных JSONB для обеспечения мультиязычности названий продуктов и нутриентов.

Для решения проблемы статических формул разработан алгоритм динамической оценки TDEE. Алгоритм использует метод экспоненциального сглаживания (EMA) для фильтрации ежедневных колебаний веса пользователя. На основе дельты сглаженного веса и усвоенных калорий за период (N дней) система вычисляет реальный расход энергии по формуле (1):

$$TDEE = Intake - (\Delta Weight \times 7700) \quad (1)$$

где 7700 – это энергетическая ценность 1 кг жировой ткани [6].

Архитектура построена с возможностью расширения и внедрения инновационного подхода к рекомендательным системам, которая подразумевает, что при запросе пользователя на подбор рациона под остаток калорий система использует метод Retrieval-Augmented Generation (RAG) [7], где в качестве контекста выступает не общая база рецептов, а векторная выборка из истории питания конкретного пользователя. Это обеспечивает высокую персонализацию рекомендаций.

Разработанный прототип демонстрирует высокую эффективность применения NLP и векторных баз данных в задачах HealthTech. Реализованный гибридный поиск и адаптивный метаболический движок решают ключевые проблемы существующих аналогов, обеспечивая низкий порог входа для пользователя и высокую точность нутрициологических расчетов. В дальнейшем планируется интеграция модуля компьютерного зрения для распознавания КБЖУ

с этикеток и автоматического обогащения профиля продукта микронутриентами из баз данных продуктов.

ЛИТЕРАТУРА

1. Kheirmandparizi, M., Gouin, J.-P., Bouchaud, C. C., Kebbe, M., Bergeron, C., Madani Civi, R., Rhodes, R. E., Farnesi, B.-C., Bouguila, N., Conklin, A. I., Lear, S. A., Cohen, T. R. Perceptions of self-monitoring dietary intake according to a plate-based approach: A qualitative study // PLOS ONE. – 2023. – Vol. 18, No. 11. – e0294652. – DOI: 10.1371/journal.pone.0294652.
2. LangChain. LangGraph overview [Электронный ресурс]. – URL: <https://docs.langchain.com/oss/python/langgraph/overview> (дата обращения: 10.03.2026).
3. Qdrant. Hybrid Queries [Электронный ресурс]. – URL: <https://qdrant.tech/documentation/concepts/hybrid-queries/> (дата обращения: 10.03.2026).
4. Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., Wei, F. Multilingual E5 Text Embeddings: A Technical Report // CoRR. – 2024. – abs/2402.05672.
5. Robertson, S. E., Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond // Foundations and Trends in Information Retrieval. – 2009. – Vol. 3, No. 4. – P. 333–389. – DOI: 10.1561/15000000019.
6. Hall, K. D. What is the required energy deficit per unit weight loss? // International Journal of Obesity. – 2008. – Vol. 32, No. 3. – P. 573–576. – DOI: 10.1038/sj.ijo.0803720.
7. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. – 2020. – Vol. 33

УДК 004.42

Ж.А. Бонилья Тельес (2 курс магистратуры),
А.Д. Ковалев, к.т.н., доцент

РАЗРАБОТКА ВЕБ-РЕДАКТОРА ПОЧТОВЫХ СООБЩЕНИЙ С ПОДДЕРЖКОЙ DRAG-AND-DROP И API ДЛЯ ПОЛЬЗОВАТЕЛЬСКИХ ВИДЖЕТОВ

В современных информационных системах электронная почта остаётся одним из ключевых каналов коммуникации с пользователями. Почтовые рассылки активно применяются в маркетинге, сервисных уведомлениях, образовательных платформах и корпоративных информационных системах. Повышение требований к визуальному оформлению писем, адаптивности и персонализации контента приводит к необходимости использования специализированных инструментов для их создания и редактирования [1].

Существующие веб-редакторы почтовых сообщений, как правило, ориентированы на пользователей без технической подготовки и предоставляют ограниченные возможности расширения функциональности. В то же время разработчики и компании-интеграторы заинтересованы в возможности добавления собственных компонентов, бизнес-логики и динамических элементов в структуру письма [2]. Именно поэтому актуальной задачей является достижение баланса между простотой использования редактора, его гибкостью и расширяемостью.

Одним из способов решения данной проблемы является разработка веб-редактора почтовых сообщений с поддержкой механизма drag-and-drop, позволяющего визуально формировать структуру письма, а также предоставление программного интерфейса (API) для подключения пользовательских виджетов [3]. Такой подход позволяет абстрагироваться от ручного редактирования HTML-кода, упростить процесс создания писем и одновременно обеспечить возможность расширения функциональности редактора под конкретные задачи.

Таким образом, целью данной работы является снижение трудоёмкости разработки и редактирования почтовых сообщений за счёт создания веб-редактора с поддержкой drag-and-drop и API для пользовательских виджетов, обеспечивающего гибкость, масштабируемость и удобство интеграции в существующие веб-системы.

Для достижения поставленной цели в работе необходимо решение следующих задач:

1. Проведение обзора существующих веб-редакторов почтовых сообщений и подходов к их реализации.
2. Анализ архитектурных решений и механизмов расширения функциональности подобных редакторов.
3. Проектирование архитектуры веб-редактора с поддержкой drag-and-drop и модульной структурой компонентов.
4. Разработка API для подключения и управления пользовательскими виджетами.
5. Реализация прототипа веб-редактора и демонстрация его функциональных возможностей.

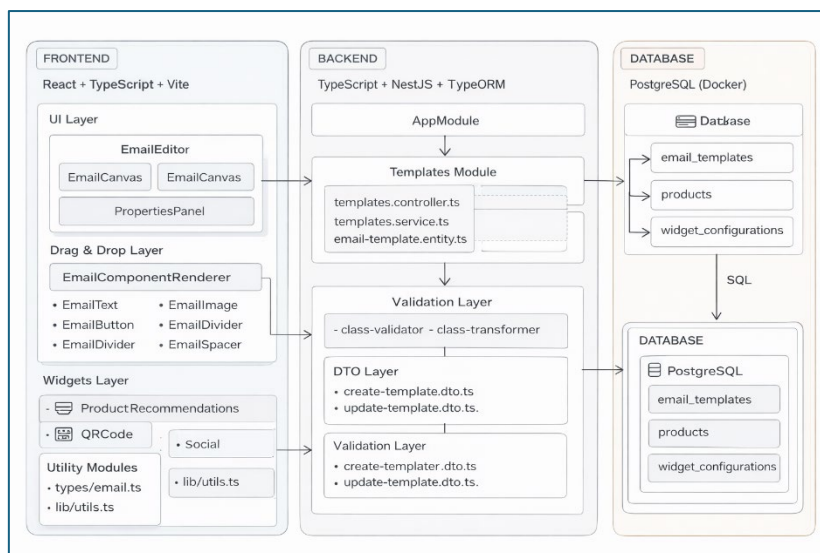


Рисунок 1 – Архитектура системы

Ключевым компонентом разработанного решения является модульная клиент-серверная архитектура. Frontend реализован на React [4] с использованием библиотеки @dnd-kit для высокопроизводительного drag-and-drop. Backend построен на NestJS [5] с применением TypeORM для работы с PostgreSQL [6]. Типобезопасность обеспечена повсеместным использованием TypeScript.

Архитектура системы представлена на рисунке 1. Пользователь взаимодействует с визуальным редактором (канвас, палитра компонентов, панель свойств). Состояние шаблона сохраняется в JSON-формате и передаётся на сервер для персистентного хранения. Динамические виджеты (рекомендации товаров, QR-коды, социальные сети) получают данные через REST API в реальном времени.

Разработанное решение демонстрирует снижение трудоёмкости процесса создания профессиональных электронных рассылок, предоставляя маркетологам мощный инструмент визуального конструирования с открытой архитектурой расширения, что открывает перспективы его применения в промышленных системах email-маркетинга.

ЛИТЕРАТУРА

1. Nielsen J. Usability Engineering. – San Francisco: Morgan Kaufmann, 1994. – 362 p.
2. Chaffey D., Ellis-Chadwick F. Digital Marketing. – Pearson Education, 2019. – 672 p.
3. Melnikov A. HTML Email Design Guide. – Smashing Magazine, 2020.
4. React. [Электронный ресурс]. Режим доступа: <https://react.dev/> (дата обращения: 10.02.2026).
5. NestJS. [Электронный ресурс]. Режим доступа: <https://nestjs.com/> (дата обращения: 10.02.2026).
6. PostgreSQL. [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/> (дата обращения: 10.02.2026).

ПРОЕКТИРОВАНИЕ МАСШТАБИРУЕМОЙ СЕРВЕРНОЙ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ ДЛЯ ИНДУСТРИИ КРАСОТЫ

Одной из наиболее динамично развивающихся сфер в стране является индустрия красоты. Существующие цифровые решения охватывают лишь часть потребностей мастера этой сферы. CRM-системы (Customer Relationship Management), как например, DIKIDI, YCLIENTS, автоматизируют запись, но лишены визуальной составляющей, с другой стороны, социальные сети обеспечивают охват аудитории, но не предоставляют механизмов управления расписанием и клиентской базой [4, 5]. Дополнительной проблемой является отсутствие способов оценки надёжности как клиентов, так и мастеров, что не позволяет заранее определить степень их добросовестности [6]. Данная работа посвящена проектированию и разработке серверной части мобильного приложения «make», устраняющего указанные ограничения за счёт интеграции функциональности социальной сети и CRM-системы в едином сервисе.

Из-за необходимости одновременной поддержки различных типов нагрузки: частых операций чтения ленты, взаимодействия в чате в режиме реального времени, а также операций бронирования – была выбрана микросервисная архитектура [2]. Система включает семь специализированных сервисов: «User Service» управляет профилями, подписками и ролевой моделью; «Post» отвечает за публикацию работ мастеров, лайки, хештеги и полнотекстовый поиск; «Timeline» формирует персонализированную ленту; «Booking» реализует управление расписанием и онлайн-записью с листом ожидания; «Stories» обслуживает временный медиаконтент; «Chat» обеспечивает обмен сообщениями в реальном времени; «Notification» доставляет push-уведомления и рассылки. Единой точкой входа служит API-Gateway NAPIroxy, осуществляющий маршрутизацию HTTP-соединений. Архитектура системы представлена на рисунке 1.

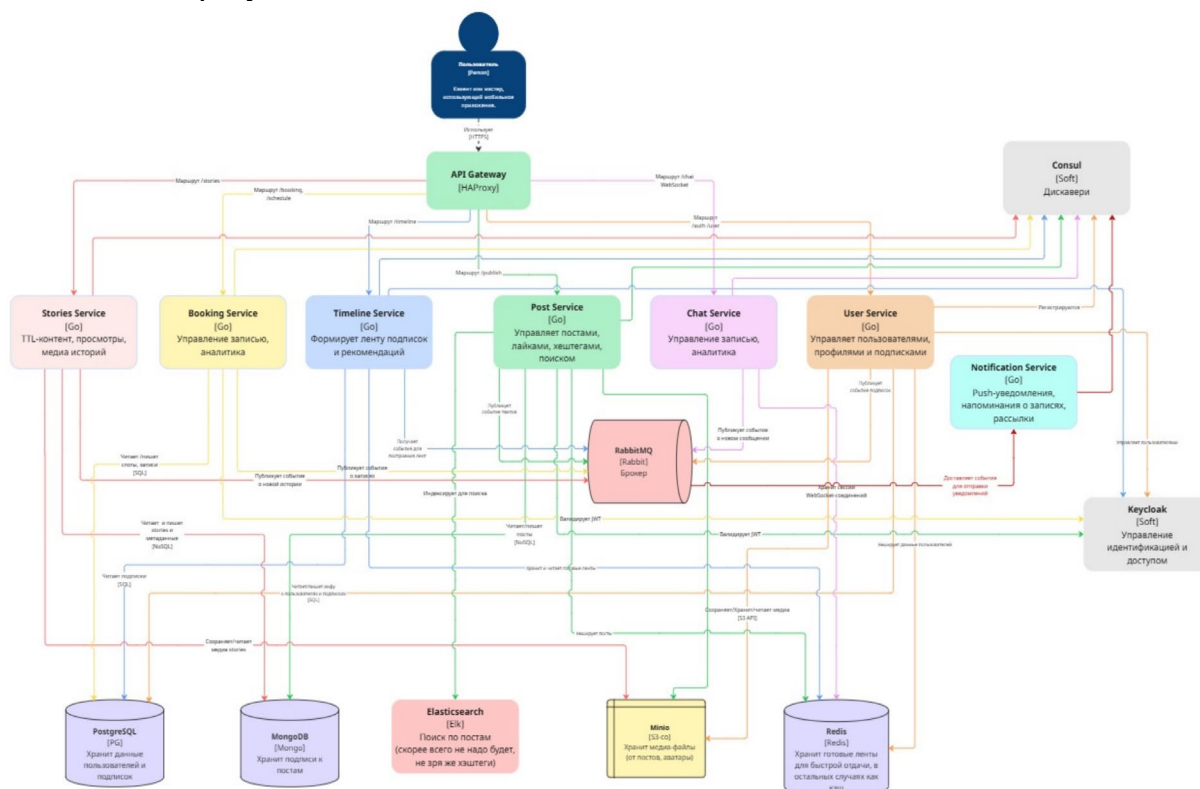


Рисунок 1 – Архитектура системы

Все сервисы реализованы на Golang [1]. Для хранения данных используется несколько баз данных: PostgreSQL гарантирует ACID-транзакции для критически важных данных, таких как учётные записи пользователей, расписания и записи на процедуры, исключая коллизии двойного бронирования слотов; MongoDB хранит посты, теги и метаданные контента благодаря гибкой документоориентированной модели; Redis используется как кеш лент, снижая нагрузку на основные базы данных при большом количестве запросов; Minio предоставляет S3-совместимое объектное хранилище медиафайлов. Полнотекстовый и нечёткий поиск по постам и хештегам реализован на базе Elasticsearch. Управление идентификацией и выдача JWT-токенов централизованы в Keycloak. Асинхронное межсервисное взаимодействие организовано через RabbitMQ, что позволяет сделать сервисы независимыми друг от друга и обрабатывать фоновые задачи – push-уведомления, рассылки по листу ожидания без блокировки основных потоков.

В ходе реализации были решены две сложные технические задачи. Первая касается формирования персонализированной ленты: Timeline Service строит её на основе графа подписок пользователя и его поведенческой активности (лайки, просмотры), после чего результат кешируется в Redis для обеспечения быстрого времени ответа при повторных запросах. При публикации нового поста Post Service отправляет событие в RabbitMQ, Timeline Service, получив его, удаляет устаревшие ленты подписчиков данного мастера из кеша, гарантируя, что при следующем обращении лента будет пересчитана с учётом актуальных данных. Вторая задача связана с проектированием Chat Service: постоянное соединение реализовано через WebSocket, история сообщений сохраняется в MongoDB. Нетривиальной задачей является обработка офлайн-состояния получателя: при разрыве соединения сервис не теряет сообщение, а инкрементирует в Redis счётчик непрочитанных сообщений для данного пользователя. При следующем подключении клиент получает актуальное значение счётчика непрочитанных сообщений для отображения в интерфейсе, не выполняя дополнительных запросов к базе данных.

Конкурентным преимуществом разрабатываемого сервиса является уникальное сочетание возможностей, плохо реализуемое в рамках существующих аналогов. Рейтинговая система оценки клиентов, в отличие от стандартной оценки только мастеров, позволяет минимизировать риски недобросовестного поведения. Автоматизированный лист ожидания устраняет простои в расписании: при отмене записи система автоматически предлагает освободившийся слот следующему клиенту в очереди через Notification Service. Персонализированная лента рекомендаций обеспечивает привлечение клиентов для мастера без необходимости платного продвижения. С инженерной точки зрения применение событийно-ориентированной (event-driven) архитектуры с разделением зон ответственности создаёт горизонтально масштабируемый и отказоустойчивый фундамент, пригодный для промышленной эксплуатации при росте пользовательской базы.

ЛИТЕРАТУРА

1. Донован А. А. Язык программирования Go / А. А. Донован, Б. У. Керниган. – М.: Вильямс, 2016. – 432 с.
2. Ньюмен С. Создание микросервисов / С. Ньюмен. – СПб.: Питер, 2016. – 304 с.
3. Кудинов А. CRM: Российская практика эффективного бизнеса / А. Кудинов. – М.: 1С-Паблишинг, 2019. – 463 с.
4. Афанасиади Е. А. Автоматизация бизнес-процессов деятельности предприятия малого и среднего предпринимательства beauty-сферы / Е. А. Афанасиади, С. Э. Малахова // Студенческий проект года 2025 : сборник статей Международного научно-исследовательского конкурса, Петрозаводск, 21 мая 2025 года. – Петрозаводск: МЦНП «Новая Наука», 2025. – С. 69-76.
5. Горнякова Д. А. Формирование бизнес-процессов с помощью ИТ-технологий в бьюти-сервисе / Д. А. Горнякова // Туризм, сервис и гостеприимство: перспективы развития : сборник статей Всероссийской научно-практической конференции с международным участием, Ялта, 23–29 сентября 2024 года. – Херсон: ХГПУ, 2024. – С. 140-143.

6. Силкина Н. С. Разработка Android-приложения для организации работы самозанятых специалистов / Н. С. Силкина, А. А. Валиулин, М. Д. Зусман // Наука ЮУрГУ : Секции технических наук : материалы 73-й научной конференции, Челябинск, 20–22 апреля 2021 года. – Челябинск: Издательский центр ЮУрГУ, 2021. – С. 458-464.

УДК 004.738.5

К.А. Быкова (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА ОБУЧАЮЩЕГО САЙТА-СИМУЛЯТОРА ИНВЕСТИЦИЙ ДЛЯ НАЧИНАЮЩИХ ПОЛЬЗОВАТЕЛЕЙ

В современном быстро меняющемся мире все более актуальны вопросы о том, как рационально распоряжаться личными финансами, как приумножить и не потерять деньги. Банковские и финансовые организации активно продвигают образовательные программы, направленные на обучение финансовой грамотности [1, 2].

Одной из форм такой работы являются выездные занятия в школах и вузах. Если для школьников достаточно базовых объяснений, связанных с накоплением и управлением карманными средствами, то старшеклассники и студенты чаще проявляют интерес к инвестициям и задают вопросы о вложении средств и работе фондового рынка. Преподаватели также отмечают возрастающую значимость данной темы.

При этом инвестиционная деятельность является сложной областью: для принятия решений необходимо понимать особенности финансовых инструментов, уметь анализировать изменения и учитывать неопределенность рынка. Попытки обучения исключительно на теоретических материалах или презентациях часто оказываются недостаточными. В то же время реальная торговля на бирже для новичков опасна финансовыми потерями, а существующие платформы с демо-счетами часто перегружены функциональностью, сложны в освоении и не содержат понятных обучающих подсказок [3].

Таким образом, возникает потребность создать безопасную, доступную и наглядную среду, которая позволит начинающим пользователям освоить основы инвестирования через практику без риска потери реальных средств.

Целью данной работы является разработка веб-приложения – обучающего симулятора инвестиций для начинающих инвесторов.

Для достижения поставленной цели необходимо решение следующих *задач*:

1. Проанализировать существующие программные решения и онлайн-платформы, предназначенные для обучения инвестициям и работы на финансовых рынках, выявить их достоинства и недостатки с точки зрения начинающего пользователя.

2. Исследовать особенности обучения финансовой грамотности и инвестициям, определить требования к содержанию и форме подачи материала.

3. Сформулировать функциональные и нефункциональные требования к разрабатываемому веб-приложению.

4. Спроектировать архитектуру веб-приложения, структуру базы данных и пользовательский интерфейс обучающего симулятора инвестиций.

5. Реализовать веб-приложение с использованием современных веб-технологий, обеспечивающее моделирование основных инвестиционных операций и отображение результатов.

6. Разработать механизм обучающих подсказок и обработки типичных ошибок пользователя при совершении виртуальных сделок.

7. Провести тестирование разработанного симулятора и оценить его пригодность для использования в образовательных целях.

Разработанное приложение реализует клиент–серверную архитектуру. Пользовательский интерфейс ориентирован на минимализм и поэтапное обучение. На первом

этапе пользователю доступны только базовые операции — выбор актива, покупка и продажа виртуальных лотов. После освоения начальных навыков интерфейс расширяется дополнительными инструментами управления рисками, что позволяет постепенно усложнять процесс обучения.

Ключевой особенностью симулятора является модуль анализа выбранного временного периода. Пользователь выбирает на графике изменения цены выбранной акции две точки: начало и конец периода и получает количественную и качественную оценку поведения цены на данном отрезке. По результатам анализа система вычисляет основные характеристики периода, включая изменение цены, направление тренда, уровень волатильности и показатель риска, а также формирует пояснения и подсказки, которые направлены на объяснение причин полученного результата и логики инвестиционного решения, что позволяет пользователю интерпретировать динамику рынка и использовать полученные знания при реальных покупках [4, 5].

Информация о периоде формируется двумя способами. Часть пояснений и подсказок составляются с использованием искусственного интеллекта, что позволяет каждый раз получать уникальные результаты по особенностям выбранного периода [6]. Для второго способа в программе имеется файл со списком заранее заготовленных пояснений, которые выводятся в зависимости от параметров анализируемого периода. Это сделано для того, чтобы у пользователя всегда имелась корректная информация по его периоду вне зависимости от работы интеллектуального анализа.

Дополнительно в приложении реализован встроенный словарь инвестиционных терминов, обеспечивающий оперативное получение разъяснений используемых понятий непосредственно в процессе обучения.

После того, как пользователь проанализирует различные периоды и поймет, какой характер динамики цены связан с минимальным уровнем риска, он может попробовать купить выбранную акцию по текущей цене и через несколько дней оценить прибыль и сделать выводы. Для этого реализована простая регистрация и виртуальный портфель.

Практическая значимость разработанного веб-приложения заключается в возможности его использования в образовательных учреждениях в рамках курсов по финансовой грамотности и основам инвестирования [3, 4]. Симулятор позволяет безопасно отрабатывать инвестиционные решения, анализировать ошибки и формировать осознанное отношение к риску без необходимости использования реальных денежных средств.

В результате работы был создан функциональный прототип обучающего симулятора инвестиций, соответствующий требованиям прикладной программной инженерии и ориентированный на использование в учебном процессе. Разработанный прототип демонстрирует перспективность применения симуляционных методов для формирования базовых инвестиционных навыков у начинающих пользователей [5].

ЛИТЕРАТУРА

1. OECD. PISA 2022 Results (Volume IV): How Financially Smart Are Students? — Paris : OECD Publishing, 2024. — 247 p. — DOI: 10.1787/5a849c2a-en.
2. OECD. Shaping Students' Financial Literacy: The Role of Parents and Socio-Economic Backgrounds // PISA in Focus. — 2024. — No. 126. — 9 p. — Paris : OECD Publishing. — DOI: 10.1787/c3f3dc74-en.
3. Cannistrà M., De Beckker K., Agasisti T., Amagir A., Pöder K., Vartiak L., De Witte K. The impact of an online game-based financial education course: Multi-country experimental evidence // Journal of Comparative Economics. — 2024. — Vol. 52, iss. 4. — P. 825–847. — DOI: 10.1016/j.jce.2024.08.001.
4. Pichop G. N. Integrating the Stock Market Simulation Into the Core Curriculum of a Business Program: Evidence of the Impact on Learning From a Longitudinal Study // Journal of Applied Business and Economics. — 2025. — Vol. 27, no. 1. — DOI: 10.33423/jabe.v27i1.7503.
5. Chang S. E., Luo H., Tseng L. Improving the Effectiveness of a Stock Simulation Trading Course via Blockchain and Social Networking: A Taiwanese Study // Electronics. — 2024. — Vol. 13, no. 22. — Art. 4338. — DOI: 10.3390/electronics13224338.

6. Merino-Campos C. The Impact of Artificial Intelligence on Personalized Learning in Higher Education: A Systematic Review // Trends in Higher Education. — 2025. — Vol. 4, no. 2. — Art. 17. — DOI: 10.3390/higheredu4020017.

УДК 004.42

Ю. Ван (2 курс магистратуры),
В.В. Амосов, к.т.н., доцент

ОНЛАЙН-ЧАТ НА ОСНОВЕ SPRING BOOT И VUE С ВОЗМОЖНОСТЬЮ ИЗМЕНЕНИЙ ДЛЯ НУЖД ПОЛЬЗОВАТЕЛЯ

Системы мгновенного обмена сообщениями являются важным компонентом современных информационных систем. В корпоративной среде, онлайн-обучении и цифровых сообществах востребованы решения, обеспечивающие обмен сообщениями в реальном времени и допускающие дальнейшее функциональное развитие. В связи с этим разработка модульной и расширяемой онлайн-системы чата представляет практический интерес для программной инженерии [1].

Целью работы является проектирование прототипа онлайн-системы чата на основе микросервисной архитектуры, пригодной для последующего расширения и адаптации под различные прикладные сценарии.

Система реализуется по модели разделения фронтенда и бэкенда. Клиентская часть разрабатывается на основе Vue3, серверная — на основе Spring Boot и Spring Cloud. Доступ и маршрутизация запросов осуществляются через API-шлюз, а взаимодействие между сервисами поддерживается механизмом регистрации и обнаружения сервисов [1]. Архитектура включает пользовательский сервис, сервис контактов, сервис групп, сервис чата и файловый сервис.

Для передачи сообщений в реальном времени используется WebSocket. Базовая функциональность системы включает регистрацию и вход пользователей, управление контактами и группами, а также обмен сообщениями. Хранение данных организовано с использованием MySQL. Аутентификация реализуется на основе JWT, а Redis применяется для хранения и проверки токенов, что обеспечивает управление сессиями и контроль доступа. Структура микросервисной архитектуры онлайн-системы чата представлена на рисунке 1.

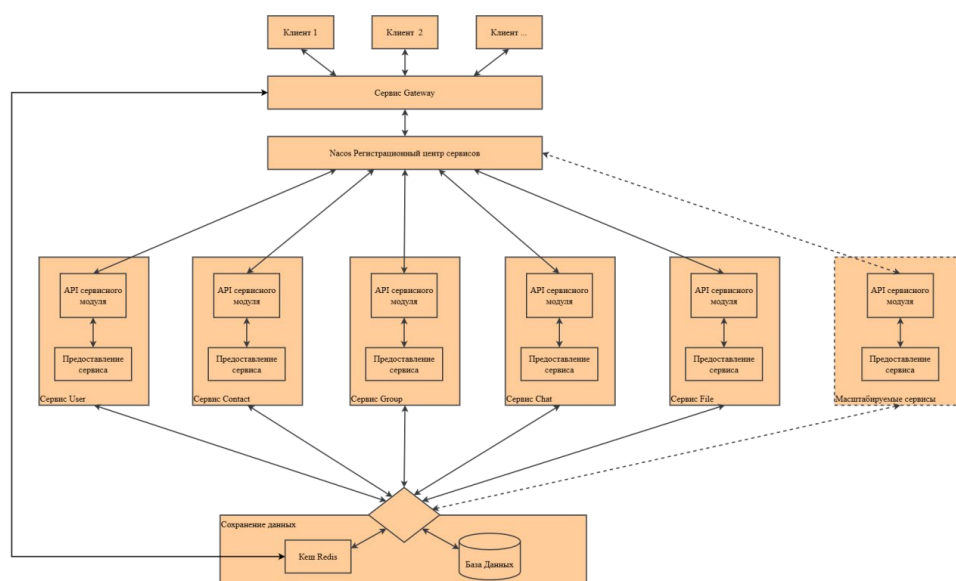


Рисунок 1 – Схема микросервисной архитектуры онлайн-системы чата

В качестве направления дальнейшего развития рассматривается интеграция технологий искусственного интеллекта [2][3]. Перспективными являются реализация чат-бота на основе

моделей обработки естественного языка [4][5], а также перевод сообщений в реальном времени с применением моделей машинного перевода [6][7].

Таким образом, предлагаемый прототип онлайн-системы чата сочетает базовые средства мгновенной коммуникации и расширяемую микросервисную архитектуру, что делает его пригодной основой для дальнейшей разработки прикладной коммуникационной платформы.

ЛИТЕРАТУРА

1. Schwarz G.-D., Bauer A., Riehle D., Harutyunyan N. A taxonomy of microservice integration techniques // Information and Software Technology. 2025. Vol. 183. Art. 107723. DOI: 10.1016/j.infsof.2025.107723.
2. Moreschini S., Pour S., Lanese I. et al. AI Techniques in the Microservices Life-Cycle: a Systematic Mapping Study // Computing. 2025. Vol. 107. Art. 100. DOI: 10.1007/s00607-025-01432-z.
3. John M. M., Holmström Olsson H., Bosch J. An empirical guide to MLOps adoption: Framework, maturity model and taxonomy // Information and Software Technology. 2025. Vol. 183. Art. 107725. DOI: 10.1016/j.infsof.2025.107725.
4. Singh S. U., Namin A. S. A survey on chatbots and large language models: Testing and evaluation techniques // Natural Language Processing Journal. 2025. Vol. 10. Art. 100128. DOI: 10.1016/j.nlp.2025.100128.
5. de Oliveira H. T. A., Sobrinho Á. A. D. C. C., Dias A. R. C., Araújo A. M. C., Araújo R. D., Matos D. D. M. C., Galvez S. M.-N. Large Language Models for chatbot applications handling sensitive information // Expert Systems with Applications. 2026. Vol. 299, Part C. Art. 130145. DOI: 10.1016/j.eswa.2025.130145.
6. Ataman D., Birch A., Habash N., Federico M., Koehn P., Cho K. Machine Translation in the Era of Large Language Models: A Survey of Historical and Emerging Problems // Information. 2025. Vol. 16, No. 9. Art. 723. DOI: 10.3390/info16090723.
7. Choudhary R., Hida R., Hamada M., Futami H., Sekiya T. Exploring Context Strategies in LLMs for Discourse-Aware Machine Translation // Findings of the Association for Computational Linguistics: EMNLP 2025. 2025. P. 24382–24391. DOI: 10.18653/v1/2025.findings-emnlp.1324.

УДК 004.65

Н.А. Васин (4 курс бакалавриата),
В.Э. Шмаков, к.т.н., доцент

СИСТЕМА АВТОМАТИЗИРОВАННОГО ВЗАИМОДЕЙСТВИЯ С ЗАКАЗЧИКОМ В УПРАВЛЕНИИ ПРОЕКТОМ

В современных условиях разработки программного обеспечения значительное влияние на успешность реализации проектов оказывает эффективность взаимодействия между заказчиком и исполнителем на ранних этапах жизненного цикла программного продукта. Для небольших команд разработчиков и самозанятых специалистов данная стадия реализации проекта особенно критична, поскольку в условиях ограниченных ресурсов значительная часть рабочего времени затрачивается на коммуникацию, уточнение требований и сбор исходных материалов. Это приводит к снижению производительности разработки и повышению вероятности возникновения ошибок в проектной документации [1].

Анализ практики взаимодействия с заказчиками показывает наличие ряда системных проблем:

- Семантический разрыв между представлениями заказчика и исполнителя о содержании будущего программного продукта.
- Неструктурированность исходных проектных артефактов.
- Изменение и расширение требований на ранних этапах разработки.
- Субъективность первичной оценки сроков и стоимости разработки.

Несмотря на широкое распространение систем управления проектами, таких как Jira, monday.com и аналогичных решений, их функциональные возможности в основном ориентированы на управление задачами внутри команды разработки. Подобные системы позволяют структурировать требования и прикреплять связанные файлы, однако они не предназначены для активного взаимодействия с заказчиком на этапе формирования требований. Кроме того, в большинстве случаев отсутствуют механизмы автоматической проверки корректности требований и их соответствия установленным методологическим критериям.

В связи с этим, целью данной работы является создание системы автоматизированного взаимодействия с заказчиком и управления проектом, ориентированной на поддержку ранних стадий проектирования программных решений.

Для достижения поставленной цели были сформулированы следующие задачи:

- анализ существующих инструментов автоматизации взаимодействия с заказчиком;
- формализация функциональных и нефункциональных требований к разрабатываемой системе;
- проектирование архитектуры программного решения и структуры базы данных;
- разработка прототипа программного приложения;
- проведение тестирования системы и анализ полученных результатов.

Предлагаемая система реализует диалоговый механизм взаимодействия с заказчиком, обеспечивающий структурированный сбор требований в процессе последовательного уточнения параметров проекта. В ходе диалога система анализирует формулировки требований и проверяет их соответствие методологическим критериям SMART. При выявлении недостаточно определённых или противоречивых формулировок система автоматически инициирует уточняющие вопросы, направленные на повышение качества требований [2]

Дополнительной функцией системы является сбор и классификация проектных артефактов, включая текстовые документы, графические материалы и другие файлы, необходимые для разработки. На основе собранной информации система формирует структурированный проектный бриф, содержащий сведения о проекте, перечень согласованных требований, список связанных артефактов и предварительную оценку трудозатрат. Для реализации функций интеллектуального анализа требований используется интеграция с современными языковыми моделями, позволяющими выявлять семантические несоответствия и пробелы в формулировках требований.

Архитектура разрабатываемой системы включает пользовательский веб-интерфейс, серверный API-слой, модуль обработки искусственного интеллекта и систему хранения данных. Серверная часть реализована на языке программирования Python с использованием фреймворка FastAPI. Для хранения данных применяется система управления базами данных PostgreSQL, дополненная расширением pgvector для работы с векторными представлениями данных и реализации семантического поиска. Обработка асинхронных задач выполняется с использованием системы очередей Redis и Celery. Данный технологический стек обеспечивает масштабируемость, расширяемость и возможность интеграции с внешними интеллектуальными сервисами [3,4].

Ожидается, что применение разработанной системы позволит повысить структурированность взаимодействия с заказчиком, снизить объем рутинной коммуникации и улучшить качество формируемых требований. Это, в свою очередь, должно привести к уменьшению количества изменений требований на поздних этапах разработки и повышению предсказуемости сроков и стоимости реализации программных проектов.

ЛИТЕРАТУРА

1. Wang Y., Luo L., Liu H. Bridging the semantic gap between customer needs and design specifications using user-generated content // IEEE Transactions on Engineering Management. – 2022. – Vol. 69, No. 4. – P. 1622–1634. – URL: <https://doi.org/10.1109/TEM.2020.3021698>

2. User Intention Recognition and Requirement Elicitation Method for Conversational AI Services / Y. Zhang, X. Li, J. Wang. – arXiv:2009.01509. – 2020. – URL: <https://arxiv.org/abs/2009.01509>
3. How to Build an AI-Powered Semantic Search in PostgreSQL with pgvector. – Redgate, 2025. – URL: <https://www.red-gate.com/simple-talk/databases/postgresql/how-to-build-an-ai-powered-semantic-search-in-postgresql-with-pgvector/>
4. Huang J. A soft prompts-based approach with pretrained language models for customer needs analysis. – 2024. – URL: <http://qlab.ieda.ust.hk/upload/paper170.pdf>

УДК 004.42

Д.А. Вещикова, И.А. Касьянова (4 курс бакалавриата),
Е. В. Комарова, ст. преподаватель

РАЗРАБОТКА АЛГОРИТМА СЕМАНТИЧЕСКОЙ СЕГМЕНТАЦИИ ТОПОГРАФИЧЕСКИХ КАРТ НА ОСНОВЕ МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ

Развитие геоинформационных технологий и оцифровка архивных картографических материалов привели к значительному увеличению объема доступных геопространственных данных [1], а в частности, топографических карт, которые являются ценным ресурсом для геопространственного анализа. Ручная интерпретация топографических материалов сопровождается существенными временными и трудовыми затратами, следовательно, существует необходимость в разработке автоматизированных методов извлечения семантической информации. Безусловно, методы глубокого обучения и компьютерного зрения уже применяются в этой области [2], но, как показывают исследования, данные методы весьма ограничены из-за специфики топографических карт [3].

Целью данной работы является разработка алгоритма семантической сегментации топографических карт на основе методов глубокого обучения для автоматического выделения типов местности.

Топографическая карта представляет собой растровое или векторное изображение с многоуровневой семантикой: цветовые коды, текстурные элементы, штриховки, условные обозначения. Всё это в совокупности обеспечивает отображение различных типов местности и объектов инфраструктуры. Современные исследования показывают, что методы глубокого обучения в задачах семантической сегментации, демонстрируют высокую эффективность [4][5], но работа с геопространственными данными все еще остается сложной задачей из-за высокой вариативности внутри классов и сложности разделения объектов с похожими текстурными характеристиками [3]. Следовательно, автоматическая обработка топографических карт требует дополнительных этапов подготовки данных и постобработки.

Предлагаемый алгоритм предназначен для автоматического выделения типов местности на основе анализа растровых топографических карт. Алгоритм реализует последовательный пайплайн, включающий этапы подготовки данных, семантической сегментации и постобработки результатов (см. рисунок 1).

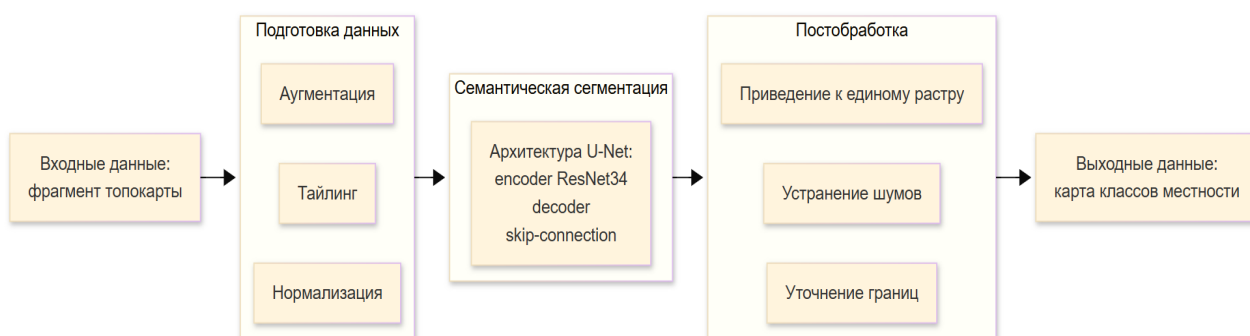


Рисунок 1 – Пайплайн семантической сегментации топографических карт

На вход алгоритма подается растровое изображение топографической карты со встроенными метаданными геопривязки. На первом этапе выполняется предварительная обработка исходного изображения: выполняется извлечение фрагментов фиксированного размера с возможностью перекрытия тайлов для устранения артефактов на границах. Производится нормализация цветового пространства к диапазону $[0, 1]$ и аугментация данных, включающая геометрические преобразования (повороты, отражения) и цветовые искажения с учётом специфики картографических изображений.

Основной этап семантической сегментации реализуется на основе архитектуры свёрточной нейронной сети. В ходе работы были изучены современные архитектуры: U-Net [6], DeepLabV3+ [7] и FCN [8], в итоге в качестве базовой была выбрана U-Net благодаря наличию skip-connections, позволяющих сохранять пространственную информацию о границах объектов на разных уровнях абстракции признаков. Кодировщик (на базе ResNet34) постепенно снижает пространственное разрешение, увеличивая число каналов для извлечения иерархических признаков. Трансферное обучение компенсирует ограниченность выборки топографических карт. Декодировщик с транспонированными свёртками и skip-connections восстанавливает разрешение до исходного размера. Выходной слой сети применяет функцию активации softmax для многоклассовой классификации, формируя тензор вероятностей принадлежности каждого пикселя к одному из N классов местности.

Этап постобработки заключается в выполнении морфологических операций для устранения шумов классификации и артефактов на границах сегментов. Формируется финальная растровая карта классов местности в формате GeoTIFF с сохранением системы координат и геопривязки исходной карты, что обеспечивает совместимость с форматами геопространственных данных и возможность дальнейшей интеграции в ГИС.

Алгоритм реализован на языке программирования Python с использованием фреймворка глубокого обучения PyTorch. Для работы с геопространственными данными применяются библиотеки GDAL и rasterio, которые позволяют сохранять геопривязку на всех этапах обработки и обеспечивать совместимость с распространенными форматами геоданных. Интеграция с библиотекой segmentation-models-pytorch предоставляет доступ к готовым реализациям архитектур-бэкбонов (ResNet) и механизмов декодирования.

Оценка эффективности разработанного алгоритма будет проводиться на специально сформированном датасете фрагментов топографических карт масштаба 1:50 000 с ручными масками сегментации. В качестве основных метрик качества приняты: коэффициент пересечения над объединением (Intersection over Union, IoU) для каждого класса, средний IoU по всем классам (mIoU) и F1-мера для комплексной оценки точности сегментации [8].

Таким образом, разработанный алгоритм семантической сегментации создаёт методическую основу для автоматизированного анализа топографических карт с использованием методов глубокого обучения. Дальнейшие исследования будут направлены на формирование размеченного датасета и экспериментальную валидацию предложенного подхода на реальных геопространственных данных.

ЛИТЕРАТУРА

1. De Paor D. G., Whitmeyer S. J. The digital revolution in geologic mapping // GSA Today. – 2010. – Vol. 20, No. 4. – P. 4–10. – DOI: 10.1130/GSATG70A.1.
2. Maxwell A., Warner T., Fang F. Implementation of machine-learning classification in remote sensing: An applied review // International Journal Of Remote Sensing - 2018. - Vol. 39, NO. 9, 2784–2817. – DOI: <https://doi.org/10.1080/01431161.2018.1433343>
3. L. Huang, B. Jiang, S. Lv, Y. Liu and Y. Fu, Deep-Learning-Based Semantic Segmentation of Remote Sensing Images: A Survey // *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, - 2024, - Vol. 17, - P. 8370-8396, - DOI: 10.1109/JSTARS.2023.3335891
4. Lv J, Shen Q, Lv M, Li Y, Shi L and Zhang P. Deep learning-based semantic segmentation of remote sensing images: a review // *Sec. Environmental Informatics and Remote Sensing* – 2023, - Vol. 11, - DOI: 10.3389/fevo.2023.1201125

5. Saeedimoghaddam M., Stepinski T. F. Automatic extraction of road intersection points from USGS historical map series using deep convolutional neural networks // International Journal of Geographical Information Science. – 2020. – Vol. 34, No. 5. – P. 947–968. – DOI: 10.1080/13658816.2019.1696968.
6. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation, - 2015, DOI: 10.48550/arXiv.1505.04597
7. Chen, LC., Zhu, Y., Papandreou, G., Schroff, F., Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation // Computer Vision – ECCV 2018, - 2018, - Vol. 11211, - DOI: https://doi.org/10.1007/978-3-030-01234-2_49
8. Long J., Shelhamer E., Darrell T. Fully Convolutional Networks for Semantic Segmentation // CVPR, - 2015, - DOI: <https://doi.org/10.1109/CVPR.2015.7298965>

УДК 004.42

А.А. Вяземский (2 курс магистратуры),
Т.В. Леонтьева, к.т.н., доцент

ПОИСК ОПТИМАЛЬНЫХ РАСПОЛОЖЕНИЙ МОДЕЛИ ДЛЯ 3D-ПЕЧАТИ

В настоящее время технологии 3D-печати набирают все большую популярность. Детали, напечатанные на 3D-принтере, уже внедряются в массовое производство.

Существует множество разных технологий 3D-печати, у каждой есть свои особенности. Один из распространенных методов 3D печати – моделирование методом послойного наплавления (FFF – Fused Filament Fabrication). Эта технология подразумевает создание изделий с помощью последовательного нанесения слоев расплавленного материала (пластиковой нити). Все рассматриваемые далее алгоритмы предполагают использование именно этой технологии.

Модель подготавливается к печати в специальных программах – слайсерах. Программа по 3D-модели вычисляет траектории движения принтера. В слайсерах можно задать множество параметров, от которых зависит результат печати [6]. Одним из самых важных параметров является расположение детали на столе принтера. Оно влияет на весь процесс печати: качество и прочность итогового изделия, скорость печати, количество израсходованного материала и т.д. Таким образом, поиск оптимального расположения является очень важной задачей.

Некоторые модели заранее проектируются с расчетом на определенное расположение плоскости печати. Но, на практике, сделать так получается далеко не всегда. Геометрия моделей, в основном, довольно сложная, поэтому заранее предсказать положение плоскости печати будет трудно. Также, для большинства моделей, идеального расположения просто не существует, а значит необходимо выбирать между несколькими вариантами. У каждого есть свои преимущества и недостатки, поэтому финальное решение должно зависеть от конкретных целей пользователя, сценариев использования изделия, особенностей 3D-принтера и т.п.

Существуют программы, которые позволяют автоматически находить расположение плоскости печати [1]. Их основным недостатком является отсутствие гибкой настройки параметров и визуализации критериев, по которым оценивается найденная плоскость печати. Некоторые решения вообще не предполагают задание каких-либо параметров: алгоритм принимает на вход 3D-модель и выдает единственный вариант расположения детали.

Разрабатываемое приложение-плагин для САПР КОМПАС-3D [5] позволяет оценивать различные варианты расположения модели по нескольким критериям и визуализировать полученные значения.

Первым и основным критерием является площадь и объем нависающих элементов. 3D-печать по технологии FFF ведется по слоям и каждый последующий слой должен хотя бы частично опираться на предыдущий (или на стол принтера). Вследствие этого существует

максимальный угол, под которым нависающий элемент напечатается качественно. Если этот угол превышен, то слайсер сгенерирует поддерживающие структуры, количество которых стремятся минимизировать. При их печати расходуется время, материал, ухудшается качество поверхности изделия.

Вторым важным критерием является площадь нижней грани. От этой площади сильно зависит стабильность печати. Чем больше площадь соприкосновения детали со столом принтера, тем лучше она будет держаться на столе. Детали с небольшой площадью нижней грани часто отделяются от стола во время печати. А это вынуждает начинать весь процесс заново.

Алгоритм оценивает значения всех критериев для нескольких вариантов расположения. Для их выбора вокруг модели описывается сфера. Пусть каждая точка сферы, а точнее нормаль в этой точке будет соответствовать одному варианту расположения детали. Его также можно однозначно задать одним вектором.

Среди рассчитанных значений можно выбрать наименьшее по одному определенному критерию, или учитывая сразу несколько критериев в сумме. Этот вектор и будет оптимальным расположением плоскости печати.

Автоматически найденный вектор расположения далеко не всегда будет самым удачным. Поэтому пользователю предлагается визуализация всех критериев оценки плоскости печати. По рассчитанным на сфере значениям составляется тепловая карта (рисунок 1), по которой хорошо видно, какие из вариантов расположения являются наилучшими. Благодаря такому визуальному представлению пользователь сможет наглядно увидеть слабые места в модели, и, возможно, перепроектировать их. Также по карте видна динамика изменения значений критерия.

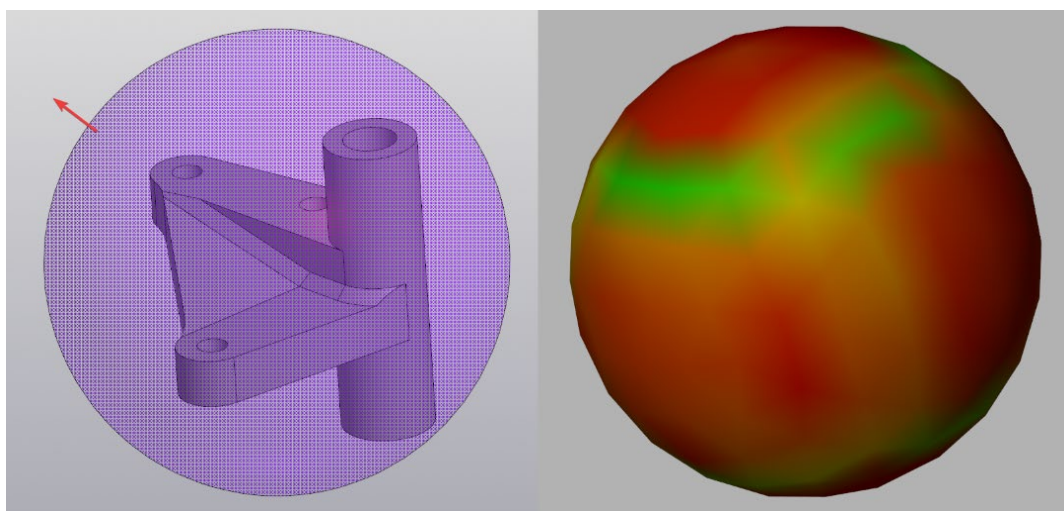


Рисунок 1 – Визуализация критериев оценки расположения плоскости печати

Разрабатываемое приложение позволяет находить несколько оптимальных вариантов расположения плоскости печати по различным критериям. Также помимо автоматического поиска предлагается набор настраиваемых визуализаций, благодаря которым пользователь сможет выбрать расположение детали, которое будет наилучшим и оптимальным для решения его конкретной задачи.

ЛИТЕРАТУРА

1. Schranz, Christoph. (2016). Tweaker - Auto Rotation Module for FDM 3D Printing. 10.13140/RG.2.2.27593.36966.
2. Jayakody, Don Pubudu Vishwana Joseph & Deng, Bailin & Goonetilleke, Ravindra & Thomas-Seale, Lauren & Kim, Hyunyoung. (2025). Part orientation fused shape optimization for minimisation of print time and material waste in extrusion-based 3D printing. Computer-Aided Design. 192. 104029. 10.1016/j.cad.2025.104029.

3. Голованов, Н. Н. Геометрическое моделирование : методическое пособие / Н. Н. Голованов. - Москва : ДМК Пресс, 2020. - 406 с. - ISBN 978-5-97060-806-7.
4. Ezair, Ben & Massarwi, Fady & Elber, Gershon. (2015). Orientation Analysis of 3D Objects Toward Minimal Support Volume in 3D–printing. Computers & Graphics. 51. 10.1016/j.cag.2015.05.009.
5. Система трехмерного моделирования КОМПАС-3D: обзор программы – URL: <https://ascon.ru/products/kompas-3d/> (дата обращения: 10.03.2024)
6. Вяземский А.А., Леонтьева Т.В. Разработка приложения, выполняющего оптимизацию модели для 3D печати // Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, 24–25 апреля 2024 г. – СПб. : ПОЛИТЕХ-ПРЕСС, 2024. – С. 29-31

УДК 004.42

А.А. Вяземский, Н.С. Онуфриенко (2 курс магистратуры)
Т.В. Леонтьева, к.т.н., доцент

ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ РАССТАНОВКИ МЕБЕЛИ ПО НАБОРУ ПРАВИЛ ДЛЯ BIM СРЕДЫ ПРОЕКТИРОВАНИЯ

В современном мире для проектирования зданий используются инструменты цифрового трехмерного моделирования [1]. Они значительно упрощают процесс проектирования, уменьшают вероятность ошибок и обеспечивают соблюдение заданных норм и ограничений.

Однако у дизайнеров и архитекторов остается ряд задач, требующих выполнения рутинных действий, например расстановка мебели по заданным ограничениям в сотнях похожих помещений. К примеру, в здании школы необходимо расставить мебель (парты, стулья, доски, рабочее место учителя и т.д.) в множество почти одинаковых кабинетов. При расстановке мебели необходимо учитывать ряд правил и норм, например, в случае школы, это расстояние между партами, расстояние от первой пары до доски, расположение мебели относительно окон (направление лучей естественного освещения).

Такая задача потребует довольно много времени работы проектировщиков. Также существует вероятность появления ошибок. Предлагается решение, которое позволит автоматически расставлять мебель в 3D-модели здания. Разработанный алгоритм сэкономит время инженеров, исключит все ошибки и обеспечит соблюдение всех заданных норм.

Программа состоит из трех независимых блоков. Такое архитектурное решение позволяет абстрагироваться от среды проектирования – основной алгоритм расстановки распространяется в виде python-скрипта с минимальными зависимостями. Остальные два блока небольшие, и в них происходит интеграция с конкретной средой проектирования. В нашем случае был выбран Autodesk Revit [2].

Первый модуль с помощью API [3] среды проектирования собирает информацию о всех помещениях в модели. В результате работы модуля формируется json-файл, в котором для каждого помещения записано его название, координаты стен, расположение и размеры окон и дверей.

Второй модуль, реализованный на языке Python, содержит основной алгоритм и работает независимо от среды проектирования. В нем производится анализ каждой комнаты: при помощи библиотеки alphashape [4] определяется контур комнаты и по положению окон задается ее ориентация, что позволяет установить параметры для перевода из локальной системы координат в глобальную для каждой из комнат. Затем для них вызываются функции расстановки мебели. В функциях задается паттерн расстановки в рамках локальной системы координат комнаты. Паттерн расстановки и список мебели, которую необходимо расставить в каждой из комнат, являются входными данными алгоритма. Результатом работы модуля является json-файл с информацией для расстановки мебели (название, координаты и ориентация). Также выдается визуализация расстановки, приведенная на рисунке 1.

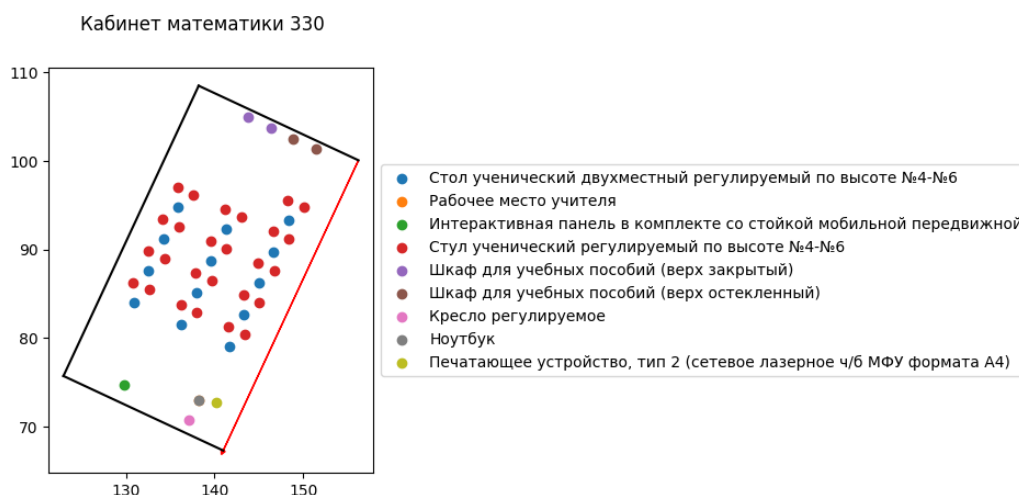


Рисунок 1 – Пример расстановки мебели в помещении

Третий модуль по подготовленному на предыдущем шаге json-файлу создает все объекты требуемой мебели в модели помещения. Результат работы модуля приведен на рисунке 2.

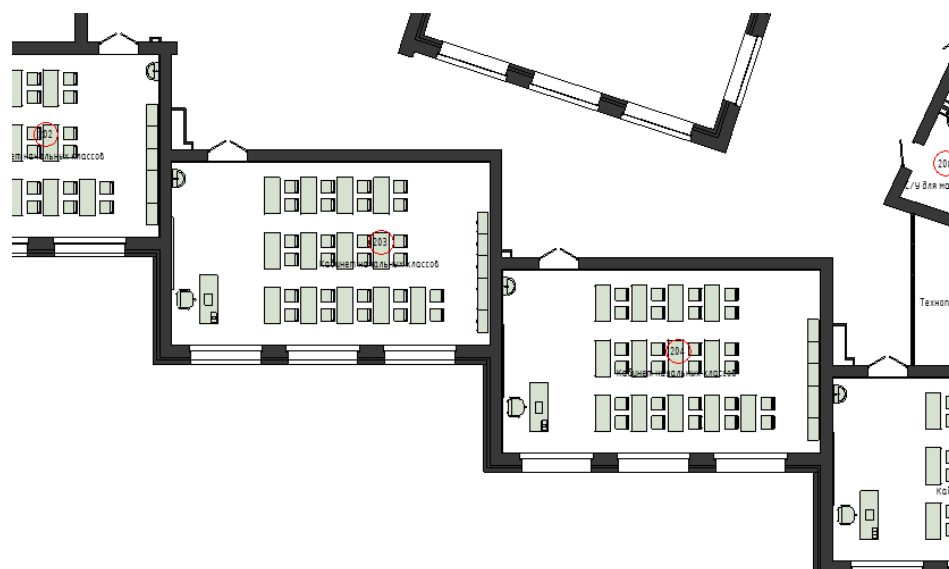


Рисунок 2 – Полученная 3D-модель

Разработанная программа может быть легко портирована для другой среды информационного моделирования, например Renga [5]. Для этого необходимо реализовать только первый и третий модули, которые являются самыми простыми в реализации. Паттерны расстановки мебели также можно легко адаптировать для решения нужной задачи. Так как они являются простыми функциями, описывающими шаблоны расстановки в рамках локальной системы координат комнаты.

ЛИТЕРАТУРА

1. Просто о BIM: что такое, как работает, где применяется [Электронный ресурс]. Режим доступа: <https://bim-info.ru/articles/prosto-o-bim-cto-takoe-kak-rabotaet-gde-primenyaetsya>
2. Autodesk Revit [Электронный ресурс]. Режим доступа: <https://www.autodesk.com/products/revit/overview>
3. Документация Revit API [Электронный ресурс]. Режим доступа: <https://www.revitapidocs.com/>
4. Библиотека Alphashape [Электронный ресурс]. Режим доступа: <https://github.com/bellockk/alphashape>
5. Renga [Электронный ресурс]. Режим доступа: <https://rengabim.com>

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ ПОДСЧЁТА ТРУБ

В период активного внедрения цифровых технологий в промышленность компании сталкиваются с рядом трудностей. Нехватка специалистов – одна из главных причин, замедляющих рост производительности. В таких условиях неэффективное распределение времени сотрудников, например, ручной подсчет продукции, становится роскошью. Для решения этой проблемы было создано клиент-серверное приложение для подсчёта и учёта труб, главной особенностью которого является внедрение сверточной нейронной сети для автоматического извлечения полезных данных из изображений.

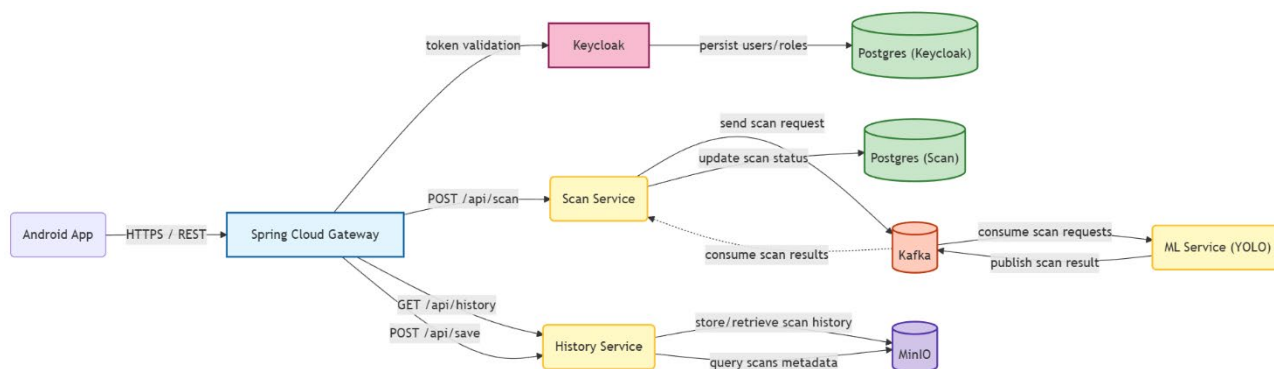


Рисунок 1 – Архитектура системы

На рисунке 1 показано, что приложение построено на основе микросервисной архитектуры [1]. Микросервисный подход при проектировании программных продуктов дает несколько важных преимуществ: отказоустойчивость, масштабируемость, гибкость и изоляция. В качестве основного фреймворка для построения микросервисной архитектуры был выбран Spring [2] на базе языка программирования Java. Данный фреймворк был выбран исходя из его богатой экосистемы и широкого инструментария, что позволяет легко обеспечить связь между различными технологиями в проекте.

Важнейшей частью работы является обеспечения всей системы достаточным уровнем отказоустойчивости. Для этого в проект был интегрирован распределенный брокер сообщений с открытым исходным кодом Apache Kafka [3]. Так как в данном проекте существуют тяжелые процессы сканирования изображений с помощью нейронной сети, критически важно, чтобы результат этого сканирования не был утерян. Использование брокера сообщений позволяет решить эту проблему, так как Kafka дает гарантии доставки сообщений между микросервисами.

Другой важной примененной технологией для обеспечения масштабируемости и отказоустойчивости является контейнеризация при помощи Docker [4]. Этот инструмент позволяет проводить изолированный запуск микросервисов отдельно от всей системы и настроить политики повторного запуска программы при возникновении ошибок. Таким образом, компоненты системы становятся изолированными друг от друга, их можно масштабировать в ширину, запуская новые копии такого же компонента.

Ключевым компонентом проекта, содержащим основной функционал, является ML-модуль, который содержит сверточную нейронную сеть для детекции объектов на изображении на базе алгоритма YOLO [5]. Сам алгоритм YOLO имеет ряд преимуществ в скорости обработки, точности распознавания объектов и простоте использования. Модель одновременно находит координаты ограничивающих рамок и классы объектов, что позволяет обрабатывать изображения в один проход. Так же, при всей своей точности, алгоритм

относительно не требователен к вычислительным ресурсам и может использоваться для детекции объектов в реальном времени.

Кроме того, были задействованы другие инструменты, например масштабируемое хранилище файлов с открытым исходным кодом MinIO [6], которое используется для учета сканирований. Spring Cloud Gateway используется для маршрутизации запросов от клиента к конкретному микросервису. Spring Cloud Eureka внедрен для создания кластеров микросервисов конкретного типа и распределения нагрузки между ними. Keycloak – сервис для аутентификации с открытым исходным кодом, необходим для учета пользователей в системе и управления токенами доступа.

Подводя итоги, удалось разработать программный продукт, обладающий практически всеми характеристиками качественного ПО: отказоустойчивостью, масштабируемостью, производительностью, простотой использования. По результатам измерений, у заказчика данного решения улучшилась статистика некоторых показателей: время отгрузки продукции сократилось на 37%, а скорость инвентаризации увеличилась на 45%. К тому же, данное решения имеет перспективы развития в других сферах, так как его легко внедрить в другие отрасли, где требуется вести учет однотипных объектов, обучив модель на других датасетах.

ЛИТЕРАТУРА

1. Ньюмен С. Создание микросервисов. 2-е изд. – СПб.: Питер, 2023. – 624 с.
2. Xiong Q., Li W. Design and Implementation of Microservices Gateway Based on Spring Cloud Zuul // CIBDA 2022: 3rd International Conference on Computer Information and Big Data Applications. – Wuhan, 2022. – P. 1-5.
3. Баженов А.Э., Райков А.В., Нехаев М.В., и др. Оптимизация взаимодействия микросервисов с использованием асинхронных вызовов и брокеров сообщений (Kafka, RabbitMQ) // Программные системы и вычислительные методы. – 2025. – № 4. – С. 77-93.
4. Adaptive load balancing and fault-tolerant microservices architecture for high-availability web systems using docker and spring cloud / [D. Han et al.] // Discover Applied Sciences. – 2025. – Vol. 7, No 7. – P. 705.
5. Рогозин А. А., Шатских В. В. ВЛИЯНИЕ ГИПЕРПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ НА КАЧЕСТВО РАСПОЗНАВАНИЯ ОБЪЕКТОВ В РЕАЛЬНОМ ВРЕМЕНИ // Материалы Всероссийской научно-практической конференции «ПРОБЛЕМНЫЕ ВОПРОСЫ МОДЕЛИРОВАНИЯ СИСТЕМ И ПРОЦЕССОВ». Воронежский государственный лесотехнический университет имени Г.Ф. Морозова. 2025. С. 190-194.
6. Han D., Lee S., Kim K., Park S. Improving Access to Object Storage on Disaggregated Storage // 2024 13th Non-Volatile Memory Systems and Applications Symposium (NVMSA). – IEEE, 2024. – P. 1-2.

УДК 004.42

В.И. Домбровская (4 курс бакалавриата),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЯ ДЛЯ ВЕДЕНИЯ ПАРОДОНТОЛОГИЧЕСКОГО ЛЕЧЕНИЯ ПАЦИЕНТА

В настоящее время в стоматологической практике все еще преобладает ведение бумажной пародонтологической документации. Несмотря на развитие цифровых технологий, многие клиники продолжают использовать бумажные амбулаторные карты и журналы учета работы врача [1]. Такой подход повышает вероятности ошибок при заполнении документации, усложняет обработку данных, что ведет к невозможности оперативного анализа динамики состояния полости рта пациента [2].

Пародонтологическая карта представляет собой сложную структуру обширного ряда клинических показателей таких как: глубина зондирования, уровень десневого края, подвижность, кровоточивость, наличие зубных отложений и так далее [3]. Для комплексной диагностики врачу необходимо рассчитывать ряд индексов, формулы которых включают

десятки параметров каждого зуба. На практике подобные расчеты часто выполняются вручную, а порой и вовсе не производятся из-за их трудоемкости.

Целью работы является разработка мобильного Android-приложения, которое бы позволило автоматизировать процесс ведения пародонтологической карты пациента, выполнять автоматический расчет диагностических индексов, хранить историю осмотров и визуализировать динамику состояния полости рта.

Для достижения поставленной цели были выделены следующие задачи:

- реализация удобного интерфейса для заполнения пародонтологической карты;
- разработка механизма автоматизации расчета диагностических индексов;
- разработка структуры базы данных для хранения информации об осмотрах пациентов;
- формирование рекомендаций по лечению на основе рассчитанных индексов;
- обеспечение безопасного хранения и передачи медицинских данных.

Разрабатываемая система реализована в виде клиент-серверного программного комплекса, включающего мобильное приложение и серверную часть.

Мобильное приложение разработано для операционной системы Android, что обусловлено распространенностью устройств на данной платформе и возможностью использования мобильных устройств непосредственно во время осмотра пациента.

Клиентская часть реализована на языке Kotlin, который является официальным языком разработки Android-приложений и обеспечивает высокую надежность и читаемость кода [4]. Для построения пользовательского интерфейса использовался декларативный фреймворк Jetpack Compose [5], позволяющий эффективно реализовывать сложные интерактивные экраны, такие как визуализация пародонтологической карты.

Для организации архитектуры приложения был выбран паттерн MVVM (Model-View-ViewModel), обеспечивающий разделение бизнес-логики, представления и обработки данных. Такой подход упрощает тестирование и сопровождение проекта [6].

Взаимодействие мобильного клиента с серверной частью осуществляется посредством REST API. Для сетевого взаимодействия используются библиотеки Retrofit [7] и OkHttp [8], обеспечивающие надежную обработку HTTP-запросов и интеграцию с API.

Для хранения пользовательских настроек используется DataStore, а защита конфиденциальной информации обеспечивается библиотекой AndroidX Security Crypto, позволяющей шифровать данные на устройстве.

Серверная часть реализована на основе фреймворка Spring Boot также на языке Kotlin для повышения согласованности с клиентской частью. Хранение данных осуществляется в реляционной базе данных PostgreSQL, управление схемой базы данных выполняется с помощью Liquibase. Для обеспечения безопасности и аутентификации пользователей применяется механизм JWT-токенов и средства Spring Security.

Архитектура системы, представленная на рисунке 1, предполагает взаимодействие мобильного клиента с серверным приложением через REST-интерфейс. Пользователь авторизуется, заполняет информацию о состоянии пациента, полученную в ходе приема. На стороне клиента выполняется анализ введенных данных и подсчет диагностических индексов, которые передаются на сервер для хранения и последующего обращения к ним.

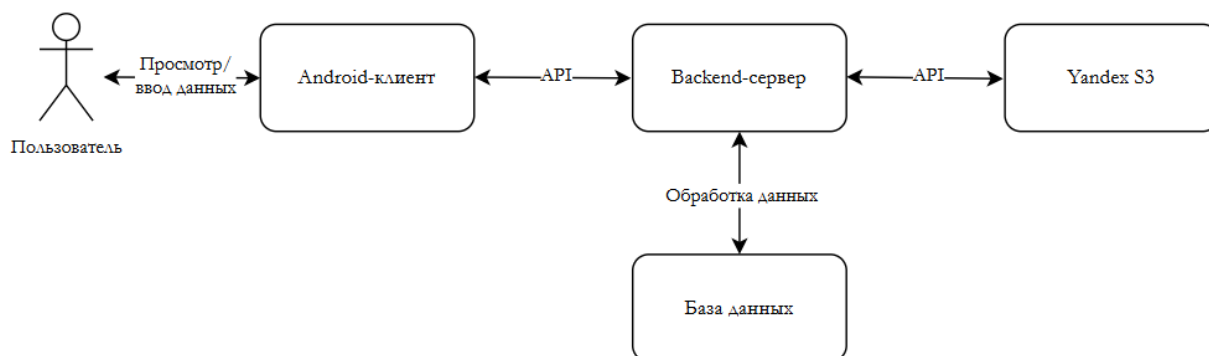


Рисунок 1 – Диаграмма архитектуры контейнерного уровня

На текущем уровне разработки реализован функциональный прототип мобильного приложения, включающий авторизацию врача-стоматолога, заполнение интерактивной пародонтологической карты, автоматический расчет диагностических индексов и хранение результатов осмотров. Реализованный прототип позволяет сократить время заполнения документации во время приема, уменьшить вероятность ошибок при расчетах и повысить объективность оценки состояния пациента.

ЛИТЕРАТУРА

1. Бутова В. Г., Янушевич О. О., Кузнецов А. В. Организация ведения медицинской документации в стоматологической практике // Российская стоматология. – 2015. – Т. 8, № 3. – С. 32–34.
2. Материалы 22-го Всероссийского стоматологического форума, 4–6 марта 2025 г., Москва // Российская стоматология. – 2025. – Т. 18, № 1. – С. 46–108. – DOI: 10.17116/rosstomat20251801146.
3. Параметры интактного пародонта, регистрируемые с помощью клинических, функциональных и лучевых методов исследования (обзор литературы) // Российская стоматология. – 2024. – Т. 20, № 4. – С. 46–54. – DOI: 10.18481/2077-7566-2024-20-4-46-54.
4. Kotlin for Android [Электронный ресурс] // Android Developers. – Режим доступа: <https://developer.android.com/kotlin> (дата обращения: 15.03.2026).
5. Jetpack Compose [Электронный ресурс] // Android Developers. – Режим доступа: <https://developer.android.com/jetpack/compose> (дата обращения: 15.03.2026).
6. Ramachandrappa N. C. MVVM Design Pattern in Software Development // International Journal of Computer Trends and Technology. – 2024. – Vol. 72, No. 9. – P. 114–119. – DOI: 10.14445/22312803/IJCTT-V72I9P117.
7. Retrofit: Type-safe HTTP client for Android and Java [Электронный ресурс] // Square. – Режим доступа: <https://square.github.io/retrofit/> (дата обращения: 15.03.2026).
8. OkHttp: HTTP client for Android and Java applications [Электронный ресурс] // Square. – Режим доступа: <https://square.github.io/okhttp/> (дата обращения: 15.03.2026).

УДК 004.418

Р.Ю. Домингес Молина (2 курс магистратуры),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА ПРОТОТИПА СИСТЕМЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ НА БАЗЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Системы электронной коммерции, ориентированные на массовую аудиторию, требуют высокой масштабируемости программного обеспечения, гибкости и отказоустойчивости. С увеличением числа пользователей и объема обрабатываемых данных возрастает необходимость в архитектурных решениях, способных эффективно распределять нагрузку между компонентами системы.

Традиционно многие приложения разрабатывались с использованием монолитной архитектуры, в которой все компоненты входят в одно приложение и развёртываются как единое целое. Такой подход может поддерживать большое количество пользователей за счёт вертикального масштабирования, то есть увеличения ресурсов сервера. Однако у него есть ряд ограничений. В монолитной архитектуре изменения в одном модуле обычно требуют пересборки и повторного развёртывания всей системы, что усложняет поддержку и замедляет процесс разработки. Кроме того, при росте нагрузки необходимо масштабировать всё приложение целиком, даже если увеличение трафика связано только с одной функциональной частью системы [1].

В качестве альтернативы используется архитектура микросервисов, которая предполагает разделение системы на несколько независимых сервисов, каждый из которых отвечает за определённую бизнес-функцию. В системе электронной коммерции это могут быть

отдельные сервисы для управления пользователями, каталогом товаров, корзиной покупок, заказами, платежами и складскими запасами. Каждый сервис реализует собственную бизнес-логику и может использовать собственную базу данных. Такой подход уменьшает связанность компонентов и позволяет разрабатывать, развёртывать и масштабировать сервисы независимо друг от друга [2].

Взаимодействие между микросервисами осуществляется через сетевые интерфейсы. Для операций, требующих немедленного ответа, используются синхронные протоколы, такие как HTTP или gRPC. Однако во многих случаях применяется асинхронное взаимодействие на основе событий. В этом случае сервисы публикуют сообщения, которые затем обрабатываются другими компонентами системы через брокеров сообщений, например RabbitMQ или Apache Kafka [6]. Такой подход повышает гибкость архитектуры и снижает прямую зависимость между сервисами.

Важным компонентом микросервисной архитектуры является API Gateway, который служит единой точкой входа для клиентских приложений. Пользователи взаимодействуют только с этим компонентом, тогда как микросервисы остаются внутри внутренней инфраструктуры системы. API Gateway выполняет маршрутизацию запросов, балансировку нагрузки, а также обеспечивает аутентификацию и авторизацию пользователей, упрощая взаимодействие между клиентами и распределёнными сервисами [3].

Масштабируемость и отказоустойчивость являются ключевыми требованиями для систем с высокой нагрузкой. Использование технологий контейнеризации и оркестрации, таких как Kubernetes, позволяет автоматизировать развёртывание, обновление и горизонтальное масштабирование сервисов. В такой архитектуре каждый микросервис может динамически увеличивать количество своих экземпляров в зависимости от нагрузки. Это позволяет обслуживать большое количество пользователей и сохранять работоспособность системы даже при сбое отдельных узлов [4].

Цель данной работы — разработка прототипа системы электронной коммерции на основе микросервисной архитектуры и демонстрация принципов её функционирования.

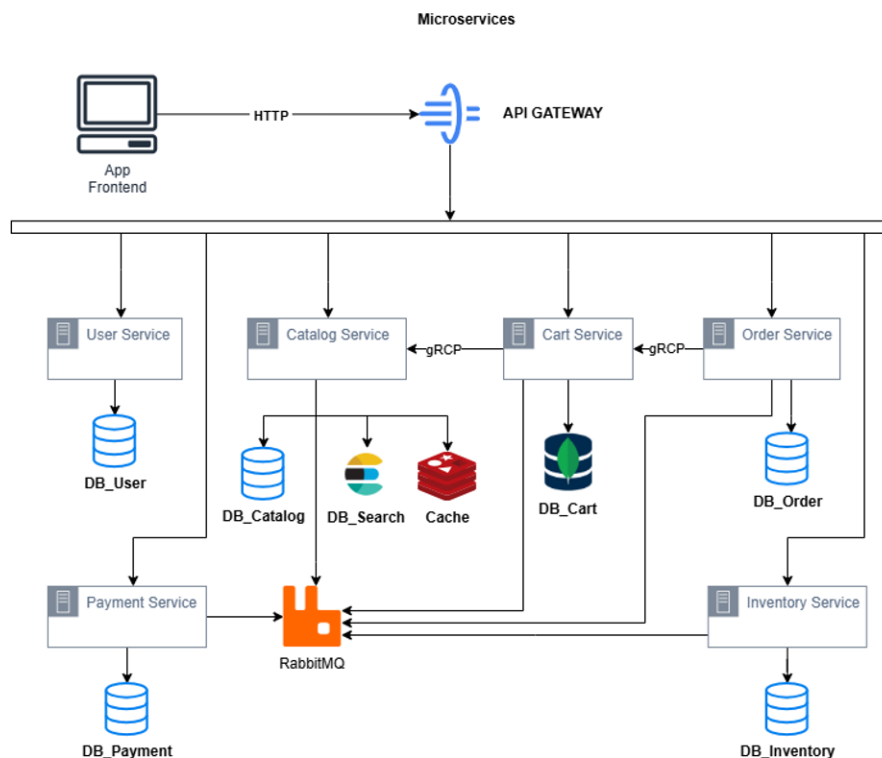


Рисунок 1 – Архитектура системы

В рамках исследования был разработан прототип системы электронной коммерции на основе микросервисной архитектуры. Архитектура системы (рисунок 1) включает несколько независимых сервисов, отвечающих за различные бизнес-функции: управление

пользователями, каталог товаров, корзину покупок, обработку заказов, платежи и управление складскими запасами. Взаимодействие между сервисами осуществляется через API Gateway и брокер сообщений RabbitMQ.

Для повышения производительности системы используются дополнительные компоненты, такие как кеширование данных и специализированный сервис поиска товаров. Каждый сервис может развёртываться и масштабироваться независимо, что позволяет гибко управлять нагрузкой и повышать отказоустойчивость системы.

Разработанный прототип реализован на языке программирования Go и демонстрирует основные принципы построения микросервисной архитектуры, включая разделение функциональности на независимые сервисы, взаимодействие через сетевые интерфейсы и использование контейнеризации для развёртывания системы. Полученные результаты подтверждают возможность применения микросервисного подхода при разработке масштабируемых систем электронной коммерции.

ЛИТЕРАТУРА

1. Al-Debagy O., Martinek P. A Comparative Review of Microservices and Monolithic Architectures // 18th IEEE International Symposium on Computational Intelligence and Informatics (CINTI). – Budapest, 2018. – P. 149–154.
2. Newman S. Building Microservices: Designing Fine-Grained Systems. – Sebastopol: O'Reilly Media, 2015.
3. Fowler M. Microservices Patterns: With Examples in Java. – Shelter Island: Manning Publications, 2019.
4. Shkuro Y. Mastering Distributed Tracing: Analyzing Performance in Microservices and Complex Systems. – Birmingham: Packt Publishing, 2019.
5. Rajesh R. Microservices with Go. – Birmingham – Mumbai: Packt Publishing, 2022.
6. Тришин Е. А. Исследование брокеров сообщений в приложениях с микросервисной архитектурой // Международный научный журнал «Вестник науки». – 2024. – № 6 (75). – Том 3. – С. 214–219.

УДК 004.91

Б.С. Дубинин (1 курс магистратуры),
О.Н. Петров, к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ АНАЛИЗА УЧЕБНЫХ ПЛАНОВ НОВОГО ПОКОЛЕНИЯ

Целью работы являлась разработка приложения в рамках системы автоматизированного проектирования образовательного процесса для работы с документами учебных планов в формате rlx и в новом формате визуального представления в xlsx [4], а также предоставление статистической информации по дисциплинам, семестрам и учебному плану в целом. [5]

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к ПО (исходные условия задачи). Эти требования подвергаются анализу.

В составе приложения с точки зрения решаемой задачи необходимо выделить основные объекты, такие как:

- приложение, которое принимает и генерирует запросы для модуля преобразования учебных планов, [1]
- модуль преобразования учебных планов, осуществляющий всю работу с Xml-данными, их обработкой и сбором статистики для формирования новых учебных планов. [3]

Приложение содержит несколько связанных между собой окон Windows Forms, каждое из которых выполняет определенную функцию.

Окно «Система формирования новых учебных планов» является основой приложения, состоит из текстовой части и кнопки, при нажатии на которую открывается диалоговое окно

открытия файлов. При выборе файла `plx` данный файл будет передан в модуль преобразования учебных планов.

Окно «Полученная из учебного плана статистика» представляет собой часть приложения, которая сохраняет информацию и выводит ее на верхнюю половину окна. Ниже выводится вторая таблица с данными, которая высчитывается по общей информации модулем обработки собранной статистики и выводит различную статистическую информацию по плану.

Под двумя таблицами находится дерево, содержащее иерархическую структуру всех модулей (блоков) учебного плана, с помощью которого можно указывать желаемые для сбора статистической информации модули. Пример дерева модулей приведен на рисунке 1.

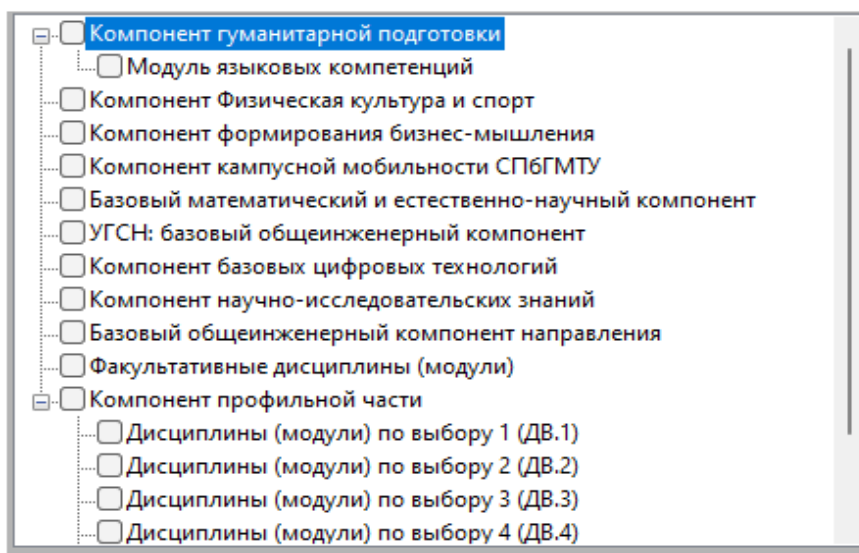


Рисунок 1 – Пример дерева модулей в приложении

Справа от данного дерева находится кнопка с названием «Вывести статистику по модулям». Данная кнопка заполняет таблицу искомой информацией по выбранным элементам.

Модуль преобразования учебных планов состоит из нескольких частей, каждая из которых выполняет различные функции в обработке полученного учебного плана.

Первой частью является функция прочтения XML файла и его десериализация, в которой происходит разбор полученного файла-плана на детали и их заполнение в существующий класс `Document`. Класс `Document` представляет собой разобранный из XML файла учебного плана (`plx`) класс для хранения данных.[2]

Второй частью является объявление всех данных для их использования в последующих частях модуля. В этой части происходит заполнение размеров необходимых для разбора данных. Размеры были получены из открытого учебного плана.

Третьей частью является циклический перебор элементов модуля (блока) для нахождения соответствия между кодом в наименовании и кодом в часах с учетом типа данных учебного плана.

Четвертой частью модуля является циклический перебор элементов модуля (блока) для выявления соответствия по коду наименования и коду вида работ.

Модуль анализа статистической информации также содержит несколько подклассов, связанных только с обработкой или выдачей статистики.

Метод «`AddNode`» добавляет в элемент `TreeView` (дерево) значение, определяемое уровнем вложения. К примеру: «Модуль языковых компетенций» имеет родителя «Компонент гуманитарной подготовки», у которого нет других родителей; «Русский язык как иностранный» имеет родителя «Модуль языковых компетенций».

Метод «`allChosenData`» позволяет собрать всю информацию по дереву с выбранными в нем элементами, игнорируя не выбранные.

Метод «chosenStats_Click». Он вызывает метод «allChosenData» и собирает всю информацию по выделенным ветвям в дереве. Затем он создает и заполняет все компоненты пользовательского интерфейса.

ЛИТЕРАТУРА

1. Обзор для разработчиков на C# - Visual Studio (Windows) - Режим доступа: <https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/visual-studio-ide?view=vs-2022>.
2. Справочник по API службы DocumentFormat.OpenXml – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/documentformat.openxml?view=openxml-3.0.1>.
3. Одиночкина, С. В. Основы технологий XML : учебное пособие / С. В. Одиночкина. — Санкт-Петербург : НИУ ИТМО, 2013. — 56 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/43573>. — Режим доступа: для авториз. пользователей.
4. Виноградов, Г. П. Визуальное программирование в MS Excel : учебное пособие / Г. П. Виноградов, Н. В. Кирсанова. — Вологда : Инфра-Инженерия, 2024. — 188 с. — ISBN 978-5-9729-1886-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/427817>. — Режим доступа: для авториз. пользователей.
5. Учебные планы – Возможности – Режим доступа: <https://docs.mmis.ru/plany/possibilities>.

УДК 004.91

Б.С. Дубинин (1 курс магистратуры),
О.Н. Петров, к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ВИЗУАЛИЗАЦИИ УЧЕБНЫХ ПЛАНОВ НОВОГО ПОКОЛЕНИЯ

Целью работы являлась разработка приложения в рамках системы автоматизированного проектирования образовательного процесса для работы с документами учебных планов в формате rlx и в новом формате визуального представления в xlslx.

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к ПО (исходные условия задачи). Эти требования подвергаются анализу.

В составе приложения с точки зрения решаемой задачи необходимо выделить основные объекты, такие как:

- приложение, которое принимает и генерирует запросы для модуля преобразования учебных планов, [1]
- модуль, обеспечивающий взаимодействие модуля-генератора с таблицами Excel и их генерацией.

Приложение содержит несколько связанных между собой окон Windows Forms, каждое из которых выполняет определенную функцию.

Окно «Система формирования новых учебных планов» является основой приложения, состоит из текстовой части (описания для объяснения пользователю о необходимости ввода в программу учебного плана для преобразования и сбора статистической информации), а также кнопку, при нажатии на которую открывается диалоговое окно открытия файлов. [5]

Модуль преобразования представляет собой несколько функций, которые преобразуют изначальные данные из формата rlx в формат, который позволяет работать с приложением и использоваться в Excel. [2]

Первой частью является функция прочтения XML файла и его десериализация, в которой происходит разбор полученного файла-плана на детали и их заполнение в существующий класс Document. [3]

Второй частью является объявление всех данных для их использования в последующих частях модуля.

Третьей частью модуля является циклический перебор элементов модуля (блока) для нахождения соответствия между кодом в наименовании и кодом в часах с учетом типа данных учебного плана.

Четвертой частью модуля является циклический перебор элементов модуля (блока) для выявления соответствия по коду наименования и коду вида работ.

Модуль взаимодействия с табличными документами содержит несколько подклассов, каждый из которых предназначен для манипулирования определенной частью документа Excel. [4]

Первая часть модуля включает проверку на наличие файла, в который должна производиться загрузка нового учебного плана. Если такого файла нет, то он будет создан. Если такой файл есть, то он будет использован. Также тут присутствует проверка на использование файла, т.к. при работе данного модуля нельзя держать открытым тот табличный документ, в который будет происходить запись. Другие документы держать открытыми можно.

Во второй части модуля метод принимает исходный обработанный документ и добавляет ему стиль документа при его отсутствии или использует уже присутствующий.

В третьей части модуля происходит настройка ширины колонок в табличном документе, она необходима отдельно от всех методов, т.к. ее необходимо объявлять независимо от рядов и ячеек документа.

В четвертой части модуля происходит циклический перебор данных всех семестров. Если массив данных, полученный модулем преобразования учебных планов, не имеет ни одной записи в данном семестре, то этот семестр пропускается. Идет выбор цвета ячейки на основе родительского кода элемента учебного плана.

Финальным действием всего модуля является сохранение документа Excel. Пример учебного плана в формате xlsx приведен на рисунок 1.

	Семестр 1	Семестр 2	Семестр 3	Семестр 4	Семестр 5	Семестр 6	Семестр 7	Семестр 8
1	Введение в специальность	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
2	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
3	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
4	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
5	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
6	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
7	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
8	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
9	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
10	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
11	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
12	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
13	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
14	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
15	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
16	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета
17	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета	Основы профессионального этикета

Рисунок 1 – Пример учебного плана

ЛИТЕРАТУРА

1. Обзор для разработчиков на C# - Visual Studio (Windows) - Режим доступа: <https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/visual-studio-ide?view=vs-2022>.

2. Справочник по API службы DocumentFormat.OpenXml – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/documentformat.openxml?view=openxml-3.0.1>.
3. Одиночкина, С. В. Основы технологий XML : учебное пособие / С. В. Одиночкина. — Санкт-Петербург : НИУ ИТМО, 2013. — 56 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/43573>. — Режим доступа: для авториз. пользователей.
4. Виноградов, Г. П. Визуальное программирование в MS Excel : учебное пособие / Г. П. Виноградов, Н. В. Кирсанова. — Вологда : Инфра-Инженерия, 2024. — 188 с. — ISBN 978-5-9729-1886-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/427817>. — Режим доступа: для авториз. пользователей.
5. Учебные планы – Возможности – Режим доступа: <https://docs.mmis.ru/plany/possibilities>.

УДК 004.932

Р.И. Думаев (3 курс аспирантуры),
С.А. Молодяков, д.т.н., профессор

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ОБЪЯСНИМОЙ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ДЛЯ АНАЛИЗА КТ-ИЗОБРАЖЕНИЙ

Развитие методов анализа медицинских изображений в последние годы привело к появлению большого количества систем компьютерной диагностики и инструментов визуализации, интегрированных с алгоритмами машинного обучения [1, 2]. Однако большинство существующих решений сосредоточено либо на разработке моделей анализа изображений, либо на создании инфраструктуры хранения и просмотра медицинских данных, что приводит к недостаточной интеграции аналитических результатов с пользовательским интерфейсом.

Одним из наиболее масштабных проектов внедрения алгоритмов машинного обучения является московский эксперимент по компьютерному зрению в радиологии, реализованный в рамках платформы mosmed.ai [3]. Несмотря на успешную интеграцию алгоритмов в клиническую инфраструктуру и масштаб применения, интерфейсная составляющая таких решений остается ограниченной. Основное взаимодействие врача с результатами анализа сводится к просмотру цветowych меток и автоматически сформированных текстовых заключений. Недостатком подобного подхода является слабая интеграция визуализации результатов алгоритма с аналитическими объяснениями и количественными характеристиками обнаруженных объектов, что ограничивает интерпретируемость результатов. Дополнительные аспекты использования систем рассматриваются в работе [4], где проводится независимая оценка нескольких ИИ-сервисов. В исследовании показано, что алгоритмы предоставляют вероятностные оценки наличия патологий и сегментации очагов, однако интерфейс взаимодействия врача с результатами остается ограниченным. В частности, отсутствуют механизмы объяснимости, позволяющие понять вклад различных признаков в итоговое решение модели.

Отдельное направление исследований связано с разработкой веб-платформ визуализации. Одним из наиболее известных решений является ONIF Viewer, базовая архитектура которого не обеспечивает встроенной связи между аналитическими показателями модели и визуальным представлением данных [5].

Таким образом, целью данной работы является повышение интерпретируемости результатов автоматического анализа за счет объединения 3D-визуализации медицинских данных, структурированного отчета и концептуальных объяснений моделей машинного обучения в едином интерактивном интерфейсе. Для достижения указанной цели поставлены и решены следующие задачи:

1. Разработка модуля визуализации КТ-данных,
2. Реализация двусторонней синхронизации между отчетом и изображением,
3. Интеграция объяснимой системы поддержки принятия решений (СППР).

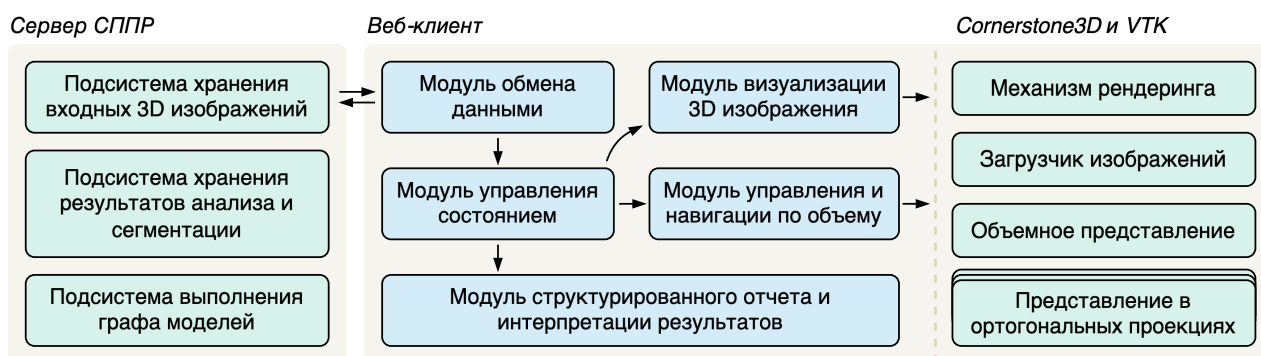


Рисунок 1 – Архитектура системы

Архитектура системы представлена рисунке 1. Разработанная система включает модуль визуализации, обеспечивающий инициализацию и настройку механизма рендеринга, регистрацию загрузчиков медицинских изображений и применение предустановленных параметров визуализации за счет использования библиотек обработки и рендеринга Cornerstone3D [5] и VTK [6]. Интерфейс реализует синхронизированный просмотр в объемном и ортогональных представлениях. Все представления работают с общим объектом состояния, благодаря чему поддерживается согласованная навигация по объему и синхронное изменение параметров отображения.

Дополнительные модули обеспечивают отображение структурированного отчета с результатами анализа СППР, где интерпретируемые признаки и их вклад в оценку целевого признака визуально связаны с соответствующими объектами на изображении. Интерфейс системы реализован как веб-приложение с компонентной архитектурой, где логика взаимодействия с данными и управление состоянием разделены на независимые модули. На рисунке 2 приведен разработанный интерфейс предлагаемой системы.

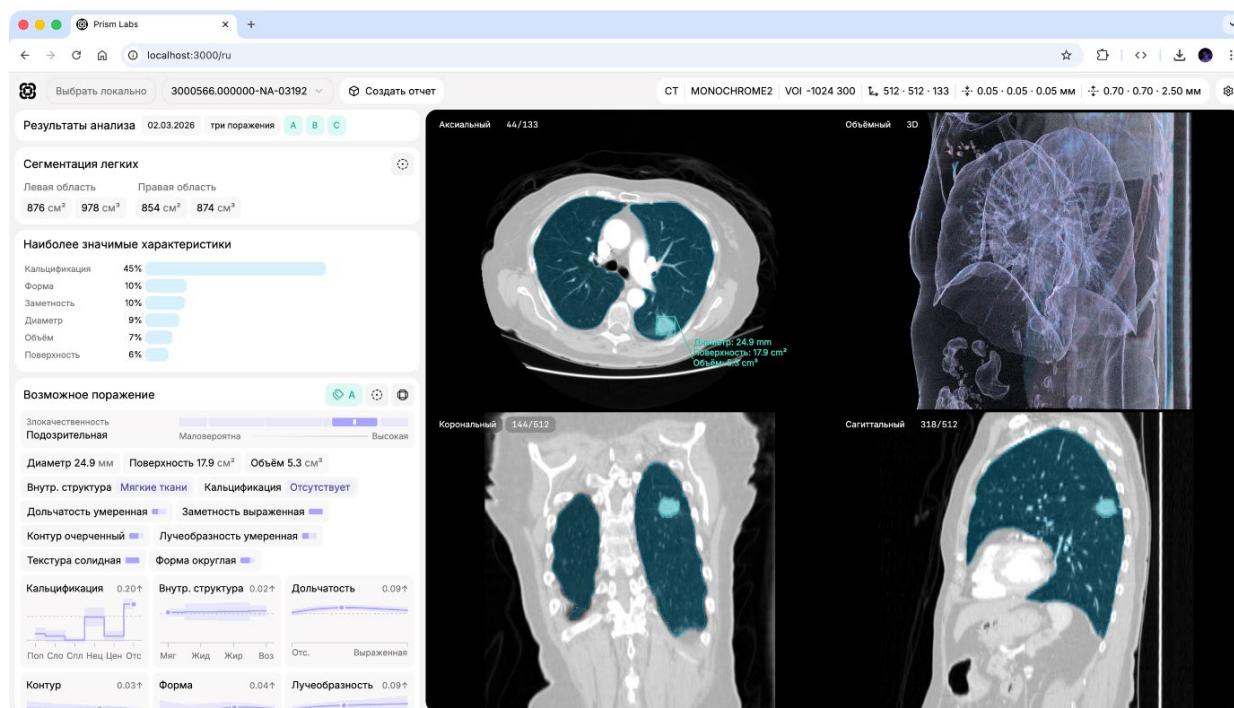


Рисунок 2 – Разработанный пользовательский интерфейс СППР

Предлагаемая в работе система и пользовательский интерфейс направлены на устранение указанных ограничений за счет интеграции объемной визуализации, структурированного отчета и концептуальных объяснений моделей машинного обучения в

едином интерактивном интерфейсе, что обеспечивает повышенную интерпретируемость результатов.

ЛИТЕРАТУРА

1. Dumaev R. I., Molodyakov S. A., Utkin L. V. A Model for the Explainable Malignancy Assessment of Pulmonary Nodules on CT Images //Scientific and Technical Information Processing. – 2025. – Т. 52. – №. 6. – С. 760-768.
2. Abbas Q., Jeong W., Lee S. W. Explainable AI in clinical decision support systems: a meta-analysis of methods, applications, and usability challenges // Healthcare. – MDPI, 2025. – Т. 13. – №. 17. – С. 2154.
3. Vasilev Y., Vladzmyrskyy A., Voloshin R., Arzamasov K., Omelyanskaya O., Reshetnikov R. Building a National AI Radiology Ecosystem: The Moscow Model // Imaging Intelligence in Practice. – 2026. – Т. 26, № 2.
4. Arzamasov K., Vasilev Y., Zelenova M., Pestrenin L., Busygina Y., Bobrovskaya T., Chetverikov S., Shikhmuradov D., Pankratov A., Kirpichev Y., Sinitsyn V., Son I., Omelyanskaya O. Independent evaluation of the accuracy of 5 artificial intelligence software for detecting lung nodules on chest X-rays // Quantitative Imaging in Medicine and Surgery. – 2024. – Т. 14. – № 8. – С. 5288–5303.
5. Ziegler E., Urban T., Brown D., Petts J., Pieper S. D., Lewis R., Hafey C., Harris G. J. Open Health Imaging Foundation Viewer: An Extensible Open-Source Framework for Building Web-Based Imaging Applications to Support Cancer Research // JCO Clinical Cancer Informatics. – 2020. – Т. 4. – С. 336–345.
6. Jourdain S., Li F., Martin K., Finet J., Lam L., Girault A., Wittenburg S., McCormick M., Belmadi A., Macron L., Jhaveri S., Boucaud A., Avery P., Ziegler E., Thompson D., Chassagnard M., Dickson M. Kitware/vtk-js: v21.1.2 // Zenodo (CERN European Organization for Nuclear Research). 2021.

УДК 004.93`12

О.А. Дьяконов, М.В. Пушкин (2 курс магистратуры),
М.В. Болсуновская, к.т.н., доцент

ГИБРИДНАЯ ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА РЕКОМЕНДАЦИИ СТИЛЕЙ ОДЕЖДЫ STUDQR: АРХИТЕКТУРА И ПЕРВЫЕ РЕЗУЛЬТАТЫ

Настоящая работа посвящена разработке интеллектуальной гибридной системы рекомендаций для офлайн-ритейла одежды, призванной решить проблему «разрыва коммуникации» между товаром и покупателем в условиях рационализации потребительского поведения [1, 2]. Предлагаемое решение STUDQR основано на интеграции QR-кодификации товаров с веб-платформой, однако ключевым научно-техническим компонентом является модуль автоматической классификации стилей одежды, построенный на основе трансферного обучения с использованием предобученной мультимодальной модели CLIP [3].

Архитектура интеллектуальной системы. Система реализована в виде микросервиса с REST API, интегрируемого с основным веб-сайтом STUDQR. Архитектура включает три взаимосвязанных модуля: 1) модуль извлечения признаков на основе предобученной модели CLIP ViT-B/32, отвечающий за преобразование изображения товара в векторное представление размерности 512; 2) модуль классификации на основе логистической регрессии, обучаемый на авторском датасете; 3) модуль ранжирования, который на основе предсказанного стиля и правил сочетаемости формирует финальные рекомендации.

Для решения задачи классификации был собран авторский датасет «Fashion Style Dataset», содержащий 850 изображений, распределённых по 11 стилистическим классам: Casual, Formal, Ethnic, Party, Smart Casual, Sports, Home, Travel, Eclectic, Military, Grunge. Изображения отбирались вручную из открытых источников с лицензией CC0 (Unsplash, Pexels, Pixabay) с последующей экспертной верификацией соответствия классу. Все изображения приведены к разрешению не менее 224×224 пикселей.

Использован подход трансферного обучения: предобученная модель CLIP ViT-B/32 [3] с замороженными весами применялась как экстрактор признаков (512-dim, L2-нормализация), на которых обучался классификатор логистической регрессии (scikit-learn).

При разделении данных 80/20 (stratified) и оценке на тестовой выборке (170 изображений) достигнуты следующие результаты: Accuracy — 92,35%, F1 (macro) — 92,56%, кросс-валидация (5-fold) — $92,12\% \pm 5,19\%$. Классы Ethnic, Formal, Home, Military, Party, Travel показали $F1 \geq 0,96$. Наиболее низкие результаты у классов Casual ($F1=0,71$) и Smart Casual ($F1=0,83$), что объясняется субъективностью границы между этими стилями.

На рисунке 1 изображена последовательность работы гибридной системы STUDQR.

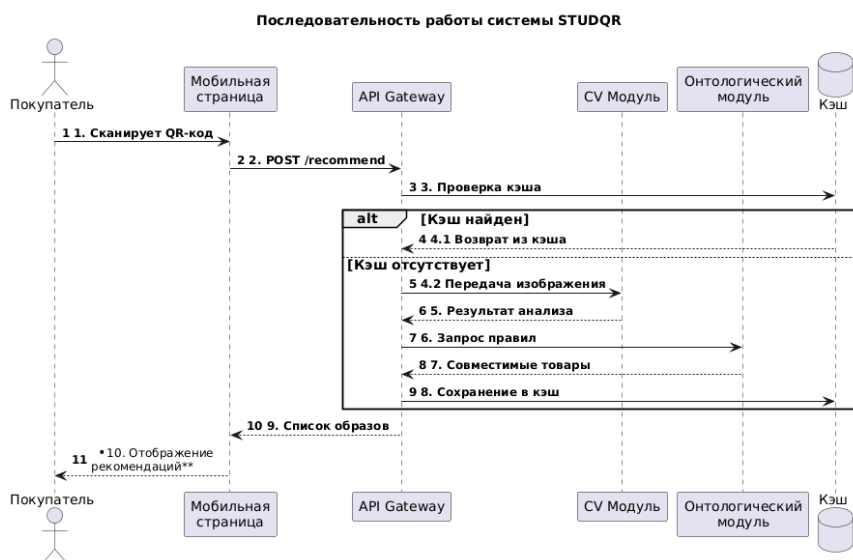


Рисунок 1 – Диаграмма последовательности

Ключевыми направлениями дальнейших исследований являются: 1) интеграция онтологической модели стилей для обеспечения семантической корректности рекомендаций, отсутствующей у чистых CV-подходов [4]; 2) проведение тестирования полноценной гибридной системы.

Гибридный подход, сочетающий мощь предобученных мультимодальных моделей с интерпретируемыми классификаторами и предметными знаниями, обладает значительным потенциалом для создания масштабируемых и осмысленных рекомендательных систем в ритейле.

ЛИТЕРАТУРА

1. Дьячкова А. В., Андриянова Е. В., Дмитриева Т. Ю. Влияние макроэкономических факторов на потребительское поведение в розничной торговле // Вестник университета. – 2022. – № 5. – С. 758–765.
2. Депутатова М. В., Перельман Е. Л. Стратегии повышения конкурентоспособности торгового предприятия в условиях неопределенности // Российское предпринимательство. – 2021. – Т. 22. – № 1. – С. 93–110.
3. Radford A., Kim J. W., Hallacy C. et al. Learning Transferable Visual Models From Natural Language Supervision // Proceedings of the 38th International Conference on Machine Learning (ICML). – 2021. – P. 8748–8763.
4. Sabiri B., Khtira A., El Asri B., Rhanoui M. Hybrid Quality-Based Recommender Systems: A Systematic Literature Review // Journal of Imaging. – 2025. – Vol. 11. – P. 12–20.
5. Liu Z., Luo P., Qiu S. et al. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2016. – P. 1096–1104.

РАЗРАБОТКА МОДУЛЯ УПРАВЛЕНИЯ ДОКУМЕНТАМИ ДЛЯ AI-АССИСТЕНТА НА ОСНОВЕ LLM И RAG

Искусственный интеллект проник во все сферы нашей жизни, начиная от помощников для технической поддержки клиентов и заканчивая помощью врачам в проведении клинических исследований [1]. Современные информационные системы используют AI-ассистентов на базе LLM и RAG для автоматизации рутинных задач, например, для качественного сопровождения продукта на стороне заказчика, уменьшения времени на обучение новых сотрудников и поддержки базы знаний документов в актуальном состоянии и доступной для всех сотрудников [2-5].

Однако на данный момент существует вопрос о корректности полученного ответа, поэтому для многих ассистентов были добавлены функции, демонстрирующие пользователю процесс размышлений и ссылки на источники информации, что позволяет самостоятельно контролировать качество. Также большинство предприятий и компаний имеют внутренние документы, которые невозможно найти в открытом доступе, но они играют важную роль при формировании ответа. Без информации из этих источников корректность ответа снижается.

Таким образом, можно отметить, что технологии хороши, но обладают существенным недостатком, который заключается в отсутствии возможности управления документами, информация из которых используется для ответа, самостоятельно на стороне пользователя. Это влечет недостоверную информацию в ответе и противоречивые ответы на один и тот же вопрос. Поэтому актуальным является решение задачи по управлению документами и созданию персональной базы знаний для каждого заказчика, использующего AI-ассистента. В связи с этим в рамках проекта будет разработан инструмент по управлению документами, позволяющий заказчикам поддерживать базу знаний в актуальном виде.

Целью работы является разработка и внедрение инструмента для управления документами в базе знаний, с которой связан AI-ассистент, для уменьшения времени взаимодействия технического писателя с продуктом путем автоматизации процессов управления документами.

Успешное достижение поставленной цели предполагает решение следующих задач:

1. Изучить существующие методы управления документами.
2. Провести сравнительный анализ существующих инструментов, с помощью которых реализованы методики управления документами в информационных системах.
3. Предложить подход, предоставляющий хранение информации в векторном представлении, поиск и удаление повторяющейся информации, поиск противоречащей друг другу информации, возможность выбрать приоритетную информацию.
4. Реализовать предложенный подход в виде отдельного модуля в рамках AI-ассистента.
5. Провести экспериментальные исследования с целью демонстрации уменьшения времени взаимодействия технического писателя с продуктом.

Рассматриваемый в работе AI-ассистент основан на технологиях LLM и RAG и использует в ответах документы из внутренней базы знаний. Эта база знаний будет формироваться исходя из документации конкретной компании, что повысит качество ответов системы.

Общая UML-схема всей системы представлена на рисунке 1.

Модуль анализа качества состоит из модуля контроля качества, который оценивает качество входных данных, релевантных документов и ответа LLM, и модуля визуализации статистики, который будет демонстрировать наиболее актуальные темы вопросов пользователей, темы, на которые система не может дать ответ, и аналогичную статистику по

продукту. Модуль управления документами отвечает за автоматизацию процессов, связанных с загрузкой, обновлением и просмотром документов в базе знаний.

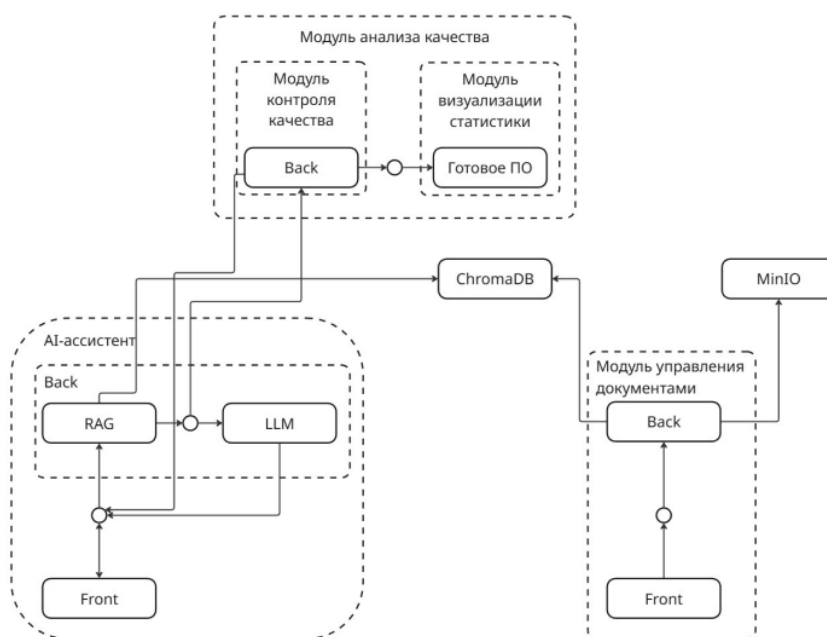


Рисунок 1 – UML-схема системы

Внедрение AI-ассистентов на базе LLM и RAG в современные информационные системы открывает значительные возможности. Однако текущий уровень развития таких систем связан с проблемой недостаточной прозрачности и контролируемости источников информации, что может приводить к снижению корректности ответов и появлению противоречивых результатов. Ключевая проблема, выявленная в работе, заключается в отсутствии у заказчика полноценных средств управления собственными документами, используемыми в базе знаний AI-ассистента. Это обуславливает необходимость разработки специализированного инструмента, позволяющего заказчикам самостоятельно формировать и поддерживать персональную базу знаний, устранять дублирующую и противоречивую информацию и устанавливать приоритеты для источников. В перспективе дальнейшее развитие подобных решений может привести к созданию более ответственных и адаптивных AI-ассистентов, тесно интегрированных в корпоративные процессы и поддерживающих доверие пользователей к автоматически генерируемым ответам.

ЛИТЕРАТУРА

1. Thomas J., Yaqoob F. Data Governance in the Era of Big Data: Challenges and Solutions Farzana Yaqoob. (2022). Data Governance in the Era of Big Data: Challenges and Solutions. Data Governance in the Era of Big Data: Challenges and Solutions. 2023. № October.
2. Опыт создания гибридных программ по решениям «1С» / А. Н. Амбражей, А. В. Валюхова, Н. М. Головин [и др.] // Новые информационные технологии в образовании : Сборник научных трудов XXIV Международной научно-практической конференции, Москва, 30–31 января 2024 года. – Москва: ООО "1С-Публишинг", 2024. – С. 255-258. – EDN MJCAAC.
3. Penelova M. Access Control Models // Cybern. Inf. Technol. 2021. Vol. 21, № 4. P. 77–104.
4. Mehra T. The Critical Role of Role-Based Access Control (RBAC) in securing backup, recovery, and storage systems. 2024. № December.
5. Свидетельство о государственной регистрации программы для ЭВМ № 2025613621 Российская Федерация. Программа для онлайн-тренажера для освоения сквозных бизнес-процессов предприятия : заявл. 28.01.2025 : опубл. 13.02.2025 / А. Н. Амбражей, В. А. Касилов, Е. Жарко [и др.] ; заявитель федеральное государственное автономное образовательное учреждение высшего образования “Санкт-Петербургский политехнический университет Петра Великого”. – EDN NATSGG.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ БЛОК-СХЕМ

Целью работы является решение задачи определения состава и последовательности элементов алгоритма по программному коду, а также отображение их в графическом виде, формируя блок-схему. Разработанное приложение предназначено для эффективного оформления программной документации по разрабатываемым продуктам на различных языках программирования.

Актуальность темы обусловлена растущей потребностью в автоматизации инструментов проектирования программного обеспечения. Несмотря на существование аналогичных решений, многие из них требуют значительных ручных доработок или ограничены в применении.

Исследовательская работа предназначена для решения следующих задач:

- Выбор необходимых средств программирования (среды программирования, языка и библиотек и т.д.).
- Разработка приложения.
- Сохранение полученной информации в виде текстовых файлов формата XML или другого формата.
- Формирование наглядной блок-схемы алгоритма.
- Выполнение печати полученной блок-схемы.

Для разработки приложения были рассмотрены современные технологии, которые могут быть использованы в процессе разработки. Использование связки Python + NiceGUI [1] для настольного оффлайн-приложения по генерации блок-схем алгоритма становится всё более оправданным, особенно когда важен современный и понятный интерфейс. Такой подход даёт кроссплатформенность и простоту развертывания: не нужно устанавливать дополнительные библиотеки или системные зависимости, кроме самого Python. Это позволяет запустить приложение на любом ПК с браузером, включая компьютеры в университетских лабораториях без доступа к Интернету.

Graphviz [2] – лучший вариант для вывода блок-схем, потому что он предоставляет мощные средства для визуализации графов с минимальными усилиями. Благодаря Python-оберткам, таким как rpygraphviz или rpydot, можно легко создавать, редактировать и отображать блок-схемы. Эта технология идеально подходит для работы с диаграммами, обеспечивая гибкость, поддержку различных форматов и простоту интеграции в проект.

Алгоритм работы приложения:

1. Загрузка файла, содержащего код: пользователь выбирает из файлов компьютера файл с кодом. Поддерживаются только файлы поддерживаемых приложением языков программирования.
2. Формирование блок-схемы: программа формирует блок-схему по представленному коду.
3. Сохранение блок-схемы: после нажатия кнопки, пользователю будет предложено выбрать формат файла.
4. Выбор формата файла: пользователь может выбрать формат изображения или формат векторной графики.
5. Сохранение файла в выбранном формате в каталог, выбранный пользователем.

Работа приложения выполняется следующим образом.

После загрузки файла надо выбрать язык программирования. Это делается через выпадающий список «Выберите язык».

После выбора языка необходимо нажать на кнопку «Построить блок-схему». После нажатия кнопки уменьшенная копия блок-схемы отобразится в правой части приложения (рисунок 1).

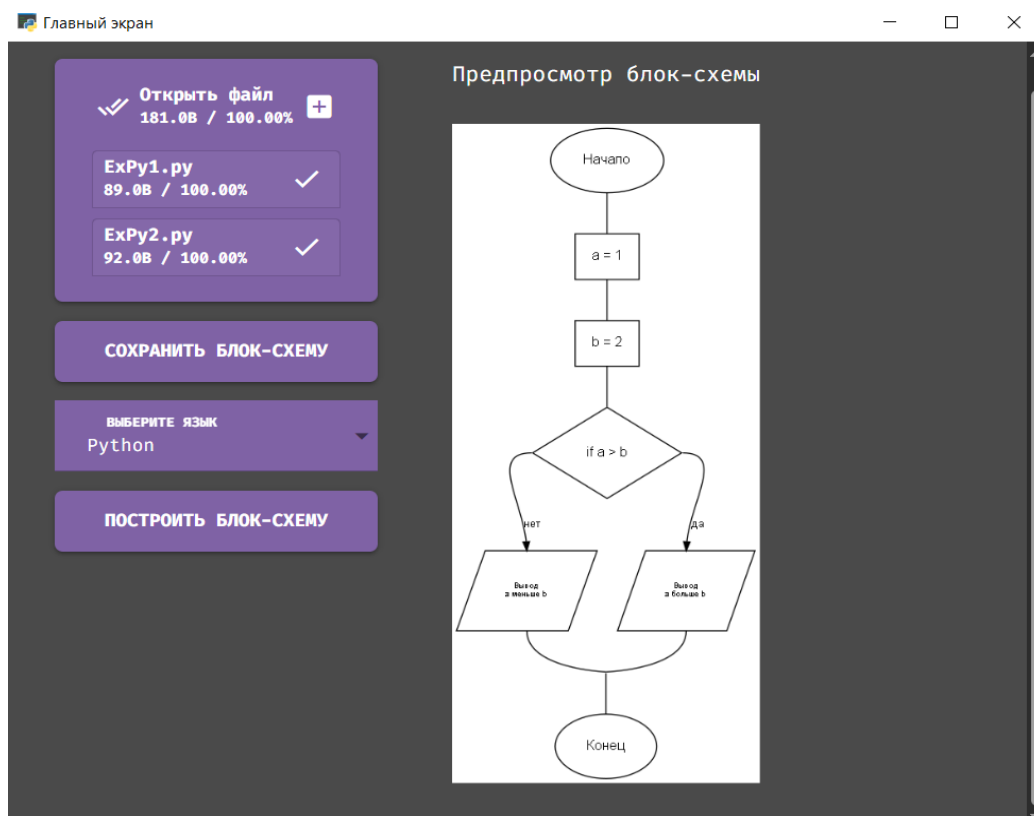


Рисунок 1 – Экран работы приложения

Результатом проделанной работы стало полностью функционирующее приложение, работающее в оффлайн-режиме без необходимости подключения к сети Интернет. Проведено тестирование основных функций приложения на примерах кода различной сложности. Таким образом, разработанное программное средство соответствует поставленным требованиям, упрощает процесс оформления алгоритмической части программной документации и может быть эффективно использовано в учебной и проектной деятельности.

ЛИТЕРАТУРА

1. Как купить домен и использовать доменное имя для размещения сайта или своих сервисов через NGINX/ [URL]: <https://habr.com/ru/articles/716002/> (дата обращения: 22.05.2025).
2. Graphviz. [URL]: <https://graphviz.org/> (дата обращения: 10.04.2025).

УДК 004.4

А.Д. Ефремов (4 курс бакалавриата),
И.В. Шошмина, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА «МОБИЛЬНЫЙ КОНТРОЛЕР» ДЛЯ ГАЗОВОЙ КОМПАНИИ

В газовой компании для сбора данных абонентов контролеры осуществляют обход на местах, традиционно контролерам выдаются бумажные акты с заданием, контролеры вручную заполняют акты на объектах, впоследствии заполненные акты передаются в офис, где их вносят в корпоративную систему учёта (АИС РНГ или АИС «Расчёт с населением за газ»). Такой процесс обслуживания приводит к большим затратам на ввод данных, увеличивает риск потери документов, замедляет общий процесс обработки актов. В результате в газовой

компании было принято решение автоматизировать этот процесс, снизив влияние человеческого фактора, увеличив оперативное взаимодействие офисных и полевых сотрудников.

Целью моей работы является разработка программного комплекса, поддерживающего работу контролеров газовой компании от создания заявок до генерации актов их загрузки в АИС учета. Задачи включали анализ бизнес-процесса [3], формирование технического задания, выбор архитектуры, реализацию ключевых модулей и их дальнейшее внедрение.

Для реализации системы используется платформа 1С [1]. На рисунке 1 приведено положение комплекса «Мобильный контролер» в общей архитектуре автоматической корпоративной системы. Комплекс связан с внутренней автоматизированной информационной системой «Расчёт с населением за газ» (АИС РНГ) и внешним сервисом «МТС Мобильные сотрудники» [4]..



Рисунок 1 – Положение сервиса «Мобильный контролер» в архитектуре корпоративной системы.

Ключевые модули комплекса мобильный контролер приведены на рисунке 2: модуль формирования загрузочных файлов (экспорт в формат, требуемый сервису МТС) [5], модуль приёма и обработки JSON-результатов с мобильных устройств, включающий подмодуль генерации актов (PDF и Word) и подмодуль хранения данных [2]. Модуль формирования загрузочных файлов автоматизирует перевод реестра планируемых визитов из АИС РНГ в таблицу Excel. Модуль обработки получает JSON-файлы с данными (идентификатор задания, сведения абонента, фотографии) и формирует итоговые акты на основе корпоративных шаблонов, сохраняет документы в базе и отправляет письма абонентам. Обеспечены требования безопасности хранения персональных данных и аутентификации API.



Рисунок 2 – Модульная архитектура разрабатываемого комплекса.

На данное время комплекс «Мобильный контролер» введен в промышленную эксплуатацию в абонентских пунктах Санкт-Петербурга, Ленинградской области, Калининградской области и республики Карелии. Автоматизация позволила сократить время работы по наряд-заданиям, и снизить количество ошибок ручного ввода.

ЛИТЕРАТУРА

1. Документация по платформе 1С:Предприятие 8.3. – М.: Фирма «1С», 2024. – URL: <https://its.1c.ru/db/v8315doc>
2. PostgreSQL 16 Documentation. – The PostgreSQL Global Development Group, 2025. – URL: <https://www.postgresql.org/docs/16/index.html>
3. Особенности управления предприятием в эпоху цифровой трансформации. Сергиенко Д.Г. 2024. <https://cyberleninka.ru/article/n/osobennosti-upravleniya-predpriyatiem-v-epohu-tsifrovoy-transformatsii/>
4. Документация по работе с сервисом “МТС Мобильный сотрудники”. 2025. – URL: <https://developers.mts.ru/traveling-staff/documentation/21-autentifikatsiya-dostup-i-szhatie-dannih>
5. 1С:Предприятие 8.3. Практическое пособие разработчика. Используем 1С:EDT. М.Г. Радченко, Е.Ю. Хрусталева. 2023. – URL: <https://its.1c.ru/db/pubdevguideedt>

УДК 004.8

К.А. Инчина (2 курс магистратуры),
С.Н. Верзунов, д.т.н., доцент

ПРИМЕНЕНИЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ ДЛЯ КРАТКОСРОЧНОГО ПРОГНОЗИРОВАНИЯ СКОРОСТИ ВЕТРА

Экстремальные ветровые явления в горных районах представляют собой одну из наиболее сложных проблем прикладной метеорологии. Их возникновение обусловлено взаимодействием крупномасштабной атмосферной циркуляции и локальных факторов. Для территорий с выраженным горным рельефом такие процессы носят явный локальный характер и отличаются высокой пространственной и временной изменчивостью. К числу подобных регионов относится территория Кыргызской Республики, где влияние горных систем Тянь-Шаня и Памиро-Алая, а также внешней северной стороны Кыргызского хребта существенно трансформирует поле ветра [1, 2]. Несмотря на то, что экстремальные ветровые явления на территории Кыргызстана не являются преобладающими, их своевременное и точное прогнозирование имеет важное практическое значение, поскольку сильные порывы ветра способны вызывать значительный ущерб.

Таким образом, целью данной работы стало повышение эффективности прогнозирования силы ветровых явлений путем построения моделей нейронных сетей для поддержки принятия решений и снижения возможных последствий чрезвычайных ситуаций, связанных с сильными ветрами.

Предсказание класса силы ветра производилось с горизонтом прогноза 3 часа вперед. Распределение данных по классам было проведено двумя способами. При первом способе интенсивность была разделена на две категории: умеренный и сильный ветер. Во втором случае на три категории: слабый, умеренный и сильный ветер. При распределении на три класса значения слабого ветра были взяты в диапазоне [0, 1.5), умеренного – [1.5, 8), сильного – [8, ∞) м/с. При распределении на два класса значения слабого ветра были включены в класс умеренных ветровых условий. Существенной характеристикой используемого набора данных являлся выраженный дисбаланс классов, где из 61346 экземпляров сильных было лишь 134.

Для устранения классового дисбаланса использовалась функция потерь Focal Loss [3] (1). Параметр α – весовой коэффициент, балансирующий вклад классов в ошибку, вычислялся путем нахождения обратных частот классов с последующей их нормализацией (2). Значения были фиксированы на протяжении всего процесса обучения.

$$FL(p_t) = \alpha(1 - p_t)^\gamma \times CCE(y, p) \quad (1)$$

$$\alpha_c \propto \frac{1}{freq(c)}, \sum_{c=1}^c \alpha_c = 1 \quad (2)$$

Также, был имплементирован вариант функции потерь, использующей адаптивный параметр a (3), вычисляемый на каждой эпохе обучения на основе частоты класса и точности определения каждого класса.

$$\alpha_c \propto \frac{1}{freq(c)^\beta \times acc(c)^\delta}, \sum_{c=1}^c \alpha_c = 1 \quad (3)$$

Во избежание резких скачков и обеспечения более плавного изменения a , к параметру было применено дополнительное сглаживание (4).

$$\tilde{\alpha}_c^t \propto \mu \tilde{\alpha}_c^{t-1} + (1 - \mu) \alpha_c^t, \sum_{c=1}^c \tilde{\alpha}_c^t = 1 \quad (4)$$

В качестве моделей решения использовались два подхода:

- гибридная CNN-LSTM модель;
- комбинированная архитектура CNN-Mamba [4, 5].

Сравнительный анализ подходов показал, что модель CNN-Mamba, использующая в функции потерь адаптивный a , позволила предоставлять более точные прогнозы. Лучшие показатели наблюдались при разделении данных на два класса. В то же время, результаты подчеркнули сложность обучения в условиях сильного дисбаланса классов (таблица 1, рисунок 1).

Таблица 1 – Значения метрик классификации по результатам работы CNN-Mamba модели с динамическим параметром a в функции потерь

Количество классов	Категория ветра	Precision	Recall	F1-score	Accuracy
2	Умеренный	1.00	0.87	0.93	0.87
	Сильный	0.01	0.59	0.02	
3	Слабый	0.60	0.72	0.65	0.56
	Умеренный	0.51	0.36	0.42	
	Сильный	0.05	0.28	0.08	

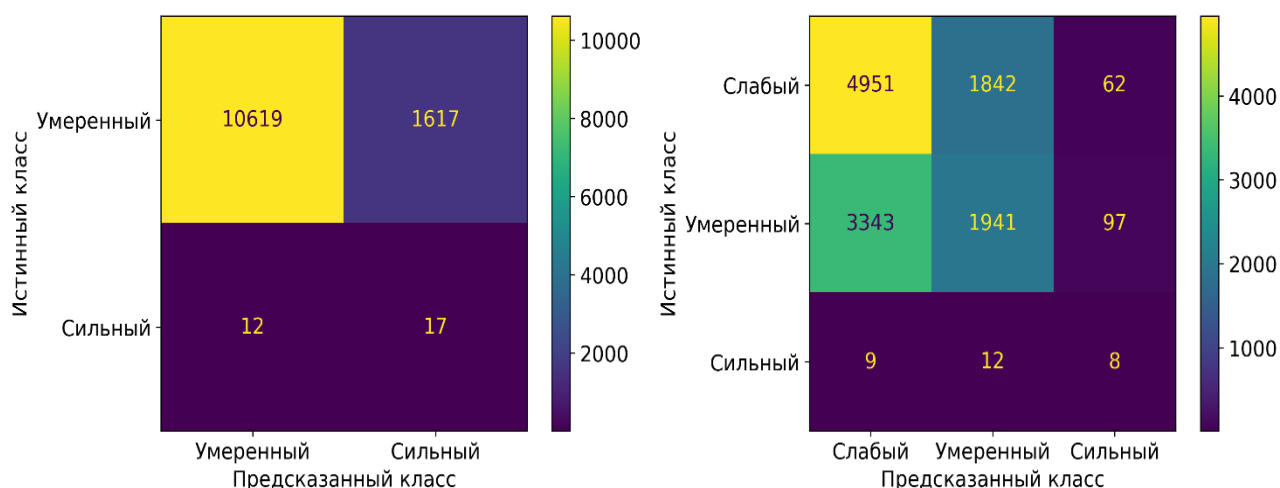


Рисунок 1 – Матрицы ошибок по результатам работы CNN-Mamba модели с динамическим параметром a в функции потерь

На основании полученных результатов можно заключить, что разработанная модель и использование динамического параметра a в функции потерь Focal Loss демонстрируют потенциал для повышения эффективности прогнозирования и перспективность своего применения в системах прогнозирования ветровых явлений.

ЛИТЕРАТУРА

1. Брусенская И. С., Подрезов О. А. Режим ветра на территории Северного, Северо-Западного Кыргызстана. – Бишкек: КРСУ, 2011. – 143 с.
2. Подрезов О. А. Современный климат Бишкека, Чуйской долины и северного склона Киргизского хребта. – Бишкек: КРСУ, 2013. – 202 с.
3. Lin T.-Y., Goyal P., Girshick R., He K., Dollár P. Focal Loss for Dense Object Detection. – 2018. – DOI: arXiv/1708.02002v2.
4. Gu A., Dao T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. – 2024. – DOI: 10.48550/arXiv.2312.00752.
5. Qin H., Chen Y., Jiang Q., Sun P., Ye X., Kin C. MetMamba: Regional Weather Forecasting with Spatial-Temporal Mamba Model. – 2024. – DOI: 10.48550/arXiv.2408.06400.

УДК 004.42

Р.Р. Загидулин (4 курс бакалавриата),
И.В. Зайцев, ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ АНАЛИЗА ДОСТИЖЕНИЙ СТУДЕНТОВ

Несмотря на неоднозначные оценки цифровизация процесса обучения является одной из главных тенденций эволюции в образовательной сфере. Как и в других областях, цифровизация создает новые возможности как для достижения полезных целей (например, большой объем и частота обратной связи для обучающегося), так и для обмана (специфика сферы образования состоит в том, что в большинстве случаев целью обмана является экономия собственного времени и интеллектуальных усилий, в результате студент может не получить нужных знаний и навыков и само осознание этого является мотивацией честно использовать новые средства обучения). Поэтому преподаватели все более широко используют системы управления учебными курсами, применяя их не только для организации дистанционного обучения, но и для очных курсов, расширяя возможности получения знаний и навыков для нацеленных на учебу студентов. С течением времени растёт количество заданий с автоматической проверкой, размещаемых преподавателями в системах управления учебными курсами. При этом растёт объём времени и усилий, затрачиваемых на регулярную обработку данных преподавателями, что, наиболее вероятно, имеет отрицательное влияние на эффективность обучения. При таких обстоятельствах возникает идея разработать программное обеспечение, позволяющее организовать регулярную статистическую обработку текущих результатов с предоставлением студентам отчетов (видимых в учебных аккаунтах студентов в системе Moodle) об их персональном абсолютном и относительном прогрессе в выполнении заданий изучаемого курса. При этом использование алгоритма, учитывающего сроки выполнения работ, позволит замотивировать студентов выполнять задания своевременно, а не дожидаться решений со стороны одногруппников, что, в том числе, поспособствует развитию дисциплины и тайм-менеджмента у обучающихся.

Среди представленных в сети Интернет литературы на тему образовательной аналитики стоит обратить внимание на публикацию “Translating learning into numbers: A generic framework for learning analytics” (Greller, W., Drachsler, H., 2012) [1]. Она была опубликована в журнале “Journal of Educational Technology & Society” (т. 15, № 3) и появилась в критический момент становления образовательной аналитики как дисциплины, когда сообществу требовалась объединяющая концептуальная основа. Авторы предложили первую исчерпывающую онтологическую структуру для образовательной аналитики, которая систематизировала разрозненные подходы и практики. Их фреймворк стал концептуальным мостом между, теоретическими исследованиями обучения, практическими ИТ-решениями, педагогическими потребностями, институциональными требованиями. Авторы также придерживаются определённой иерархии классификации целей аналитики и выделяют следующие этапы: мониторинг (описательная аналитика), анализ (диагностическая аналитика), прогноз (предиктивная аналитика), вмешательство (прескриптивная аналитика).

Разрабатываемая программа собирает данные из курсов системы Moodle [2] и на их основе проводит анализ результатов работ студентов. Результат анализа представляет собой список некоторых метрик по оценке. К примеру, следующая информация может представлять интерес для студента в ходе изучения курса: место студента в рейтинге, число студентов с аналогичным результатом, среднее значение оценки у всех студентов, число студентов с более успешным результатом, число студентов с менее успешным результатом.

По умолчанию, оценка студента рассматривается в разрезе всех результатов, полученных изучающими данный курс в текущем семестре. Если в системе Moodle у студента фиксируется принадлежность к какой-либо группе и/или какому-либо потоку учащихся в рамках текущего курса, то анализ проводится также в рамках группы и/или потока. Необходимо выводить время обновления, то есть время последней обработки результатов, чтобы пользователь мог судить об актуальности анализа и делать выводы об изменении своей успеваемости в течение обучения. Также анализ должен включать в себя как рассмотрение каждого конкретного задания, так и всего курса в целом, то есть сумму оценок курса, которая хранится на Moodle сервере.

В настоящее время общий результат студента в ходе прохождения курса обычно рассчитывается максимально прямолинейно – как сумма баллов, полученных за все задания и тесты. С точки зрения преподавателя может представлять интерес рейтинг общих результатов с учетом фактора времени по настраиваемому алгоритму.

В настоящий момент программа представляет собой консольное приложение, требующее токен для доступа к Moodle API [3] и адрес локального сервера Moodle. Выбор в пользу консольного приложения вместо API-сервера был сделан в целях упрощения взаимодействия программы с пользователем, упрощения автоматизации и уменьшения ресурсов вычислительной машины, требуемых для его работы. Все необходимые расчёты программа делает самостоятельно на основе разработанных функций, тем самым избавляя пользователя от необходимости заниматься их настройкой.

Практическая ценность описанного приложения состоит в решении ключевой проблемы, выявленной в ходе анализа существующих решений - отсутствия в системе Moodle стандартных средств регулярного сравнительного анализа в разрезе учебных групп и потоков (учитывающего сроки выполнения заданий) с предоставлением агрегированной информации самим студентам.

Работа непосредственно соответствует тренду на автоматизацию рутинных операций в педагогической деятельности. Система освобождает преподавателей от трудоемких ручных расчетов статистики, позволяя перераспределить временные ресурсы на индивидуальную работу со студентами и совершенствование учебного процесса.

Разработанное решение восполняет пробел в области сравнительного анализа успеваемости, обладает потенциалом для расширения и настройки алгоритмов обработки данных, может служить основой для дальнейших исследований в области применения систем управления учебными курсами. Внедрение такого инструмента в учебный процесс позволит мотивировать студентов не затягивать выполнение заданий курса, своевременно информируя их об абсолютном и относительном текущем прогрессе без увеличения затрат времени преподавателей на необходимую для такого информирования регулярную обработку данных системы Moodle.

ЛИТЕРАТУРА

1. Greller, W. Translating learning into numbers: A generic framework for learning analytics [Текст] / W. Greller, H. Drachsler // Journal of Educational Technology & Society. – 2012. – Vol. 15, № 3. – P. 42–57.
2. Moodle. Moodle 5.0.4 Documentation: [Электронный ресурс]. – URL: https://docs.moodle.org/500/en/Main_page
3. Moodle. Moodle Learning Analytics API [Электронный ресурс]. – URL: <https://moodle.com/us/functionality-with-moodle/learning-analytics>

РАЗРАБОТКА CRM-СИСТЕМЫ ДЛЯ ПРОИЗВОДСТВА

В современных условиях развития корпоративных информационных систем возрастает роль платформ, обеспечивающих централизованное управление клиентскими данными. При отсутствии единой платформы сведения о клиентах, сделках и коммуникациях распределяются между таблицами, почтой и мессенджерами, что приводит к потере истории взаимодействия и снижению прозрачности продаж. Существующие CRM-решения либо обладают избыточным функционалом, либо ограничивают возможности адаптации серверной логики и не соответствуют требованиям B2B-компаний к безопасности и контролю данных [1].

Целью данной работы является разработка операционной CRM-системы для внутреннего использования в компании, занимающейся поставкой оборудования и реализацией инженерных проектов. Для достижения цели решены следующие задачи: изучены существующие CRM-решения (Bitrix24, amoCRM, ELMA365 CRM, RetailCRM, SberCRM); проведён их сравнительный анализ; обоснован выбор технологического стека разработки.

В качестве архитектурного подхода выбрана микросервисная архитектура [2, 3], предполагающая декомпозицию системы на слабо связанные независимые сервисы, каждый из которых реализует отдельную бизнес-функцию. Серверная часть реализуется на языке Python с использованием фреймворка FastAPI. Для хранения структурированных данных применяется PostgreSQL, слабоструктурированных – MongoDB; такой подход соответствует принципу polyglot persistence [4] и является типичным для микросервисных систем [5]. Дополнительно используются Redis для кэширования и MinIO как S3-совместимое объектное хранилище. Взаимодействие сервисов осуществляется через REST API [6], развёртывание – посредством Docker.

В результате работы сформирована архитектурная модель серверной части CRM-системы и обоснован выбор технологического стека. Проведённый сравнительный анализ показал, что ни одно из рассмотренных решений не удовлетворяет требованиям в полной мере: универсальные платформы содержат избыточный функционал, облачные решения ограничивают кастомизацию, а специализированные системы не адаптируются под B2B-процессы с длинными циклами сделок.

Разработанная система обеспечивает централизованное управление клиентской базой, сделками и коммуникациями и позволяет адаптировать бизнес-процессы под специфику B2B-поставок оборудования и реализации инженерных проектов.

ЛИТЕРАТУРА

1. Buttle F., Maklan S. Customer Relationship Management: Concepts and Technologies. – 4th ed. – Routledge, 2019.
2. Ньюмен С. Создание микросервисов. – 2-е изд.. – СПб.: Питер, 2022.
3. Ричардс М., Форд Н. Фундаментальный подход к программной архитектуре. – СПб.: БХВ-Петербург (Springer), 2024.
4. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. – СПб.: БХВ-Петербург (Springer), 2024
5. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга. – СПб.: Питер, 2021.
6. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures. – UC Irvine, 2000.

РАЗРАБОТКА МОДЕЛИ ДЛЯ РАСПОЗНАВАНИЕ ЗУБОЧЕЛЮСТНЫХ ДЕФЕКТОВ НА
ИЗОБРАЖЕНИЯХ В ФОРМАТЕ DICOM

Компьютерная томография (далее – КТ) является незаменимым помощником в современной стоматологии при лечении/удалении зубов, протезировании и имплантации. Формат DICOM играет ключевую роль как стандарт хранения, передачи и обработки медицинских изображений и сопутствующей метаинформации [1]. Однако его обработка – сложная задача, требующая значительных временных затрат и специализированных программ, которые есть далеко не у всех медицинских учреждений [2]. Актуальность работы обусловлена растущей нагрузкой на врачей и увеличением объемов проводимых исследований, повышающими риск пропуска аномалий при ручном просмотре специалистом [3]. Внедрение методов автоматического обнаружения и выделения патологий позволят избежать данной проблемы.

Цель работы разработать модель для обнаружения и визуализации дефектов на изображениях в формате DICOM. Для достижения этой цели необходимо решение следующих задач:

1. Обзор существующих подходов для обработки DICOM формата
2. Проведение сравнительного анализа имеющихся решений
3. Предложение автоматического подхода обнаружения патологий на срезах
4. Реализация данного подход
5. Проведение экспериментов

Развитие методов машинного обучения и глубоких нейронных сетей позволяет автоматизировать задачи сегментации и выявления аномальных зон на медицинских изображениях, ранее выполняемых вручную стоматологами [4].

На данный момент существуют аналоги разрабатываемой системы: 3D Slicer, Second Openion, DentIX AI, Overjet. Автоматическая сегментация патологий выполняется с использованием искусственного интеллекта. Такой подход позволяет избежать фактора человеческой ошибки [5].

Архитектура онлайн-системы для обнаружения зубочелюстных дефектов на изображениях в формате DICOM представлена на рисунке 1. Приложение работает по следующему алгоритму: пользователь загружает исследование КТ челюсти (в виде папки с DICOM-файлами) через веб-интерфейс. Метаданные пациента не играют решающей роли в процессе сегментации. Система автоматически анонимизирует все файлы — удаляет персональные данные, очищает private tags и генерирует новые UID. Затем каждый срез проходит предобработку: извлекается Pixel Data, масштабируется (rescale), адаптируется оконным преобразованием (windowing), нормализуется, пересобирается в виде тензора, после чего поступает на вход обученной модели U-Net [6].

Модель строит карту признаков, размечая границы зубов, ткани и воздух. Используется несколько слоев свертки с функцией активации ReLU и пулингом, которые позволяют перейти от мелких деталей к общему контексту изображения. Затем процесс идет в обратном порядке, и карта признаков восстанавливается до исходного размера, где каждый пиксель - вероятность наличия аномалии, например кариеса, от 0 до 1. На основе вероятности система автоматически отбирает 10 наиболее “подозрительных” срезов и отображает их пользователю. При необходимости врач может скачать отображенные срезы.

Подход был реализован с использованием языка программирования Python [7], который имеет подходящие библиотеки для машинного обучения (далее – ML) и работы с медицинскими данными. Для процесса автоматической сегментации задействованы ML библиотеки PyTorch + MONAI, позволяющие работать с медицинскими данными с помощью

архитектурного подхода U-Net – наиболее используемого для задач биомедицинской сегментации, когда набор данных для обучения ограничен и несбалансирован. Работа с DICOM форматом осуществляется с использованием библиотеки pydicom [8].

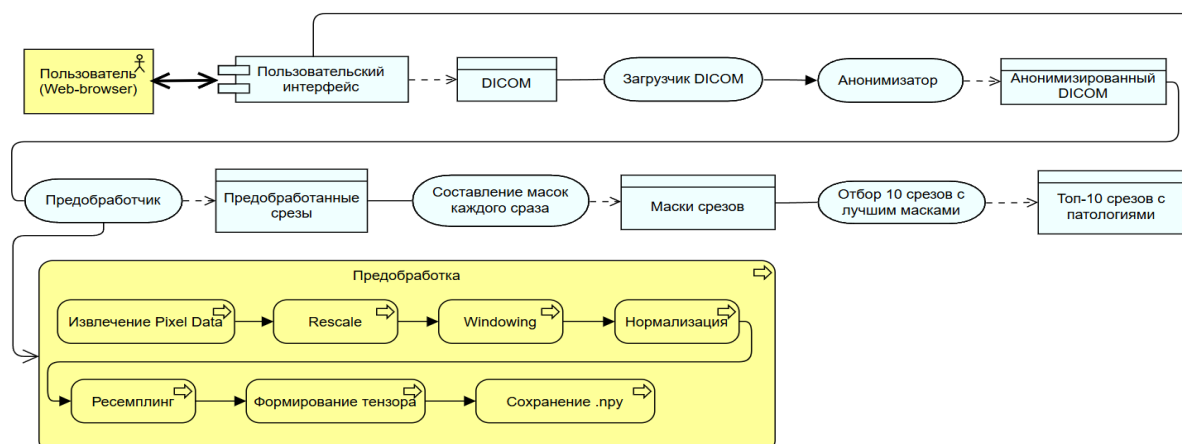


Рисунок 1 – Архитектура онлайн-системы для обнаружения зубочелюстных дефектов на изображениях в формате DICOM

ЛИТЕРАТУРА

1. Учеваткин А.А., Юдин А.Л., Афанасьева Н.И., Юматова Е.А. Оттенки серого: как и почему мы ошибаемся // Медицинская визуализация. – 2020. - №24. – Т. 3. – С. 123–145. - DOI 10.24835/1607-0763- 2020-3-123-145.
2. Ahammed Muneer K., V., Rajendran, V., & K., P. Glioma Tumor Grade Identification Using Artificial Intelligent Techniques // Journal of Medical Systems. -2019. – No 43. – P. 1-12. - DOI 10.1007/s10916-019-1228-2.
3. Larobina, M. Thirty Years of the DICOM Standard // Tomography. – 2023. No 9. – P. 1829-1838. - DOI 10.3390/tomography9050145.
4. Najjar R. Redefining Radiology. A Review of Artificial Intelligence Integration in Medical Imaging // Diagnostics (Basel). – 2023. – No 13. – P. 2760. - DOI 10.3390/diagnostics13172760.
5. Najjar R. Redefining Radiology. A Review of Artificial Intelligence Integration in Medical Imaging // Diagnostics (Basel). – 2023. – No 13. – P. 2760. - DOI /10.3390/diagnostics13172760.
6. Python 3.14.3 documentation. [Электронный ресурс] Режим доступа: <https://docs.python.org/3/>.
7. Shobayo O, Saatchi R. Developments in Deep Learning Artificial Neural Network Techniques for Medical Image Analysis and Interpretation. Diagnostics. - 2025. - Т. 15. -DOI 10.3390/diagnostics15091072.
8. pydicom documentation [Электронный ресурс] Режим доступа: <https://pydicom.github.io/pydicom/stable/>.

УДК 004.453

Д.Д. Кириллова (1 курс магистратуры),
И.В. Никифоров., к.т.н., доцент

АВТОМАТИЗАЦИЯ ПРЕОБРАЗОВАНИЯ НЕСТРУКТУРИРОВАННОГО ТЕКСТА ПРЕЗЕНТАЦИЙ В ФОРМАТ MARKDOWN ДЛЯ ИНТЕГРАЦИИ В КОРПОРАТИВНУЮ БАЗУ ЗНАНИЙ

Современные компании сталкиваются с проблемой эффективного использования накопленного неструктурированного контента, в частности презентаций в форматах PDF и PPTX. Эти файлы содержат ценную информацию (инструкции, аналитику), однако их сложно индексировать, искать и интегрировать в корпоративные базы знаний. Ручная обработка и перевод каждого документа, особенно технически сложного, крайне трудоемка и медленна,

что ограничивает масштабируемость процесса [1]. Как отмечается в современных исследованиях по управлению знаниями, эффективность корпоративных систем напрямую зависит от качества структурирования и доступности контента для сотрудников [2, 3]. В работах, посвященных построению корпоративных инженерных порталов, подчеркивается, что именно базы знаний выступают ключевым элементом интеграции разнородной информации, что подтверждает актуальность выбранного подхода [4].

Для решения задач структурирования и унификации корпоративного контента широко применяется формат Markdown [5]. Его ключевые преимущества заключаются в простоте синтаксиса, удобстве версионирования посредством Git, поддержке полнотекстового поиска и возможности интеграции с платформами совместной работы, такими как GitHub. Однако процесс преобразования исходных презентаций в качественный Markdown-контент требует решения ряда сложных задач: корректное извлечение текста и графики, сохранение иерархической структуры документа, адаптация форматирования и локализация содержимого.

Большинство существующих инструментов конвертации (например, Pandoc) обеспечивают базовое преобразование форматов, но не учитывают специфику корпоративных требований: необходимость двуязычного представления, сохранение семантических связей между текстом и изображениями, поддержку сложных технических терминов [6]. Кроме того, стандартные конвертеры не предоставляют механизмов для автоматизированного перевода контента и его последующей публикации в структурированном виде.

Для решения описанной проблемы требуется разработка комплексного подхода к преобразованию презентационных материалов, учитывающего как технические аспекты конвертации, так и требования корпоративных баз знаний.

Таким образом, опираясь на современные подходы к организации корпоративных знаний [2, 3] и выявленные ограничения существующих инструментов, целью данной работы является снижение трудоемкости процесса наполнения корпоративных баз знаний за счет разработки методики автоматизированного преобразования неструктурированных презентационных материалов в структурированный двуязычный Markdown-контент с последующей интеграцией в репозиторий GitHub.

На начальном этапе работы был проведен анализ существующих решений и выполнено ручное преобразование пилотной выборки презентаций. В ходе данного этапа обрабатывались материалы общим объемом более 500 слайдов, пример результата преобразования одного из слайдов в структурированный Markdown с переводом представлен на рисунке 1.

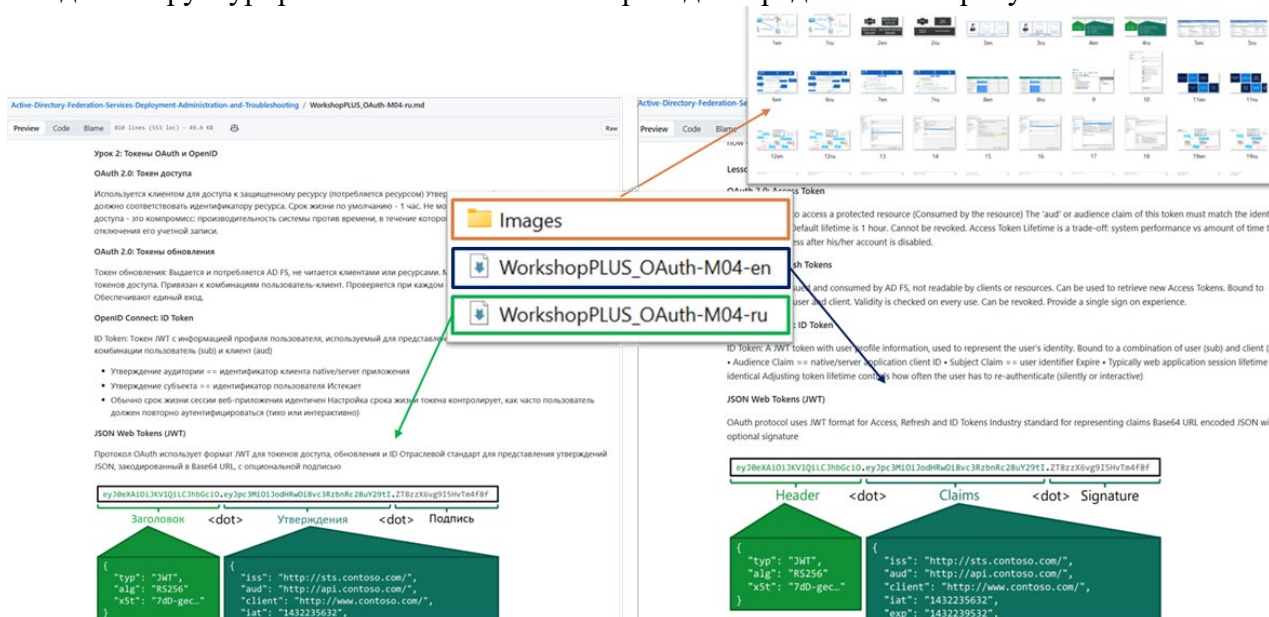


Рисунок 1 – Пример ручного преобразования слайда презентации в двуязычный формат Markdown

Анализ полученных результатов позволил выявить основные проблемы и узкие места процесса: высокую временную затратность ручного перевода, обработка одного слайда занимала в среднем 5–7 минут, сложность унификации терминологии и необходимость повторного выполнения однотипных операций при обработке каждого документа. Как видно из рисунка 1, значительная часть слайдов содержала повторяющуюся структуру, что делает их подходящими для автоматизации. Полученный опыт лег в основу формулирования требований к автоматизированному решению, что согласуется с существующими методиками разработки инструментов автоматизации обработки документов [7].

В качестве дальнейшего развития работы планируется разработка архитектуры специализированного программного средства, автоматизирующего выделенные этапы. Планируемое решение будет включать функциональные компоненты, перечисленные ниже.

1. Модуль извлечения контента (Parser Module) — для корректного парсинга PDF и PPTX с выделением текстовых блоков (заголовки, списки, абзацы) и графических элементов с сохранением их семантической взаимосвязи.

2. Модуль структурирования (Markdown Generator) — для автоматического преобразования извлеченного текста в валидный синтаксис Markdown: определение уровней заголовков, форматирование списков, оформление таблиц и блоков кода, генерация ссылок на изображения.

3. Модуль локализации (Translation Manager) — интеграция с API сервисов машинного перевода (Yandex.Translate) и большими языковыми моделями для автоматического перевода контента, а также построение глоссария для унификации технических терминов.

4. Модуль публикации (Git Publisher) — формирование целевой структуры репозитория согласно корпоративным стандартам, добавление YAML-метаданных и автоматическая загрузка файлов в GitHub.

Предполагается, что предложенная архитектура, разработанная с учетом теоретических основ управления знаниями [2, 3], обеспечит сквозную автоматизацию процесса преобразования презентационных материалов, минимизируя необходимость ручного вмешательства и позволяя масштабировать наполнение корпоративной базы знаний.

ЛИТЕРАТУРА

1. Rothman S. L. P. Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining / S. L. P. Rothman. – Hoboken, NJ : Wiley, 2014. – 288 p. – ISBN 978-1-118-83473-2.
2. Днепровская Н. В. Менеджмент знаний в организации : учебник. – Москва : КноРус, 2026. – 207 с. – ISBN 978-5-406-15347-5.
3. Уринцов А. И. Управление знаниями в организации : учебник и практикум для вузов / А. И. Уринцов, С. О. Акимов, В. В. Дик [и др.] ; под ред. А. И. Уринцова. – Москва : Юрайт, 2023. – 254 с. – (Высшее образование). – ISBN 978-5-9916-9039-3.
4. Ключкова, Л. И. Интеграция инструмента Grafana в инженерный портал компании / Л. И. Ключкова, В. А. Ромашов, И. В. Никифоров // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 24–25 апреля 2024 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. – С. 278-280. – EDN BNXLFT.
5. Markdown Guide. Электронный ресурс [Режим доступа]: <https://www.markdownguide.org/> (Дата обращения: 16.03.2026).
6. Pandoc Documentation. Электронный ресурс [Режим доступа]: <https://pandoc.org/> (Дата обращения: 16.03.2026).
7. Пономарев А. А. Методика разработки инструментов автоматизации обработки документов / А. А. Пономарев, И. В. Никифоров // Современные технологии в теории и практике программирования : сборник материалов конференции. – Санкт-Петербург, 2024. – С. 290-292.

РЕАЛИЗАЦИЯ СЕРВЕРНОЙ ЧАСТИ ВЫСОКОНАГРУЖЕННОГО ВЕБ-ПРИЛОЖЕНИЯ

В современном мире веб-разработки монолитные приложения теряют актуальность, уступая место микросервисной архитектуре, которая обеспечивает гибкость, масштабируемость и независимость компонентов [1]. Традиционный монолитный подход, при котором все компоненты приложения тесно связаны и проектируются как единое целое, перестает отвечать современным потребностям проектов, особенно работающих с высокими нагрузками [2]. Целью данной работы является анализ микросервисного подхода при проектировании серверной части социальной сети, а также реализация каждого сервиса.

Основная проблема монолитной архитектуры заключается в тесной связанности ее компонентов [1]. Это замедляет ввод новых функций, сильно усложняет масштабирование и тестирование, а также снижает отказоустойчивость всего приложения [7]. В контексте социальной сети это означает, что изменения в модуле, отвечающем за поиск, могут повлечь сбой в модулях постов и профилей.

Микросервисная архитектура предлагает разбиение приложения на множество независимых сервисов, каждый из которых реализует определенную логику [2]. Эта архитектура позволяет устранить указанные недостатки монолита, благодаря чему и стала стандартом современной разработки [8]. Рассмотрим разбиение серверной части социальной сети на пять логических сервисов:

1. Gateway: Выступает единой точкой входа для всех запросов, поступающих от любых клиентских приложений. Задачей сервиса является маршрутизация запросов к соответствующим сервисам и проверка подлинности пользователя. Клиентскому приложению не нужно знать адрес каждого отдельного сервиса, вместо этого он отправляет запросы к Gateway. Это облегчает будущее масштабирование, а также повышает простоту интеграции [3].

2. Auth Service: Отвечает за регистрацию, вход в систему и управление сессиями пользователей. Для безопасности приложения критически важно было выделить этот сервис в отдельный компонент. Например, утечка данных или их сбой не скомпрометирует учетные данные всех пользователей [2].

3. Post Service: Управляет созданием, хранением и выдачей постов пользователей. Он изолирует всю логику работы с постами, которые создают наибольшую нагрузку на веб-приложение [3].

4. Profile Service: Управляет созданием, хранением публичной информации о пользователях: имя, аватар, количество подписок, подписчиков, а также посты, созданные конкретно данным пользователем.

5. Search Service: Обеспечивает полнотекстовый поиск по всем постам. Так как поиск является ресурсоемкой операцией, этот сервис вынесен в отдельный, для избежания дополнительной нагрузки на сервис постов. Для этих целей используется специализированный инструмент Elasticsearch [4].

Одним из самых важных аспектов при проектировании такой системы является организация взаимодействия между этими сервисами [3]. В архитектуре создаваемого веб-приложения используется два типа взаимодействия:

1. Синхронное взаимодействие (REST): Применяется для запросов от клиента, проходящих через Gateway. Например, запрос на получение ленты постов или профиля [5].

2. Асинхронное взаимодействие (RabbitMQ): Применяется для взаимодействия сервисов между собой. В данном случае не используется синхронное взаимодействие для повышения отказоустойчивости. Например, при сбое одного из сервисов, данные, посылаемые в него, не

пропадут, а сохранятся в брокере сообщений RabbitMQ, который доставит их при восстановлении ключевого сервиса. [6]

Анализ показал, что применение микросервисной архитектуры при создании серверной части социальной сети позволяет решать ключевые проблемы масштабирования и надежности [7]. Каждый сервис может быть масштабирован горизонтально независимо, распределяя таким образом пиковую нагрузку, а отказ одного из сервисов не приводит к полной недоступности веб-приложения [2].

В ходе работы была спроектирована архитектура серверной части социальной сети, обоснован отказ от монолитного подхода в пользу микросервисов. Определены функциональные границы пяти ключевых сервисов и описан механизм их взаимодействия. Предложенный подход позволяет создать отказоустойчивую, легко масштабируемую и технологически гибкую систему.

ЛИТЕРАТУРА

1. Fowler M., Lewis J. Microservices [Электронный ресурс] // martinowler.com. – Режим доступа: <https://martinowler.com/articles/microservices.html> (дата обращения: 14.03.2026).
2. Ньюмен С. Создание микросервисов. — СПб.: Питер, 2016. — 304 с.
3. Richardson C. Microservices Patterns. — Manning Publications, 2018. — 520 p.
4. Gormley C., Tong Z. Elasticsearch: The Definitive Guide — O'Reilly Media, 2015. — 724 p.
5. Fielding R. Architectural Styles and the Design of Network-based Software Architectures (Dissertation) [Электронный ресурс] // University of California, Irvine. – Режим доступа: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата обращения: 14.03.2026).
6. RabbitMQ Documentation [Электронный ресурс] // VMware. – Режим доступа: <https://www.rabbitmq.com/documentation.html> (дата обращения: 14.03.2026).
7. Fowler M. Microservice Trade-Offs [Электронный ресурс] // martinowler.com. – Режим доступа: <https://martinowler.com/articles/microservice-trade-offs.html> (дата обращения: 14.03.2026).
8. Newman S. Building Microservices: Designing Fine-Grained Systems. — 2nd ed. — O'Reilly Media, 2021. — 490 p.

УДК 004.4'234

А.Ю. Коба (4 курс бакалавриата),
Н.В. Воинов, к.т.н., доцент

УНИВЕРСАЛЬНЫЙ МЕХАНИЗМ ПРОСМОТРА ФАЙЛОВ В ИНТЕРФЕЙСЕ 1С:ПРЕДПРИЯТИЕ

Платформа «1С:Предприятие» занимает доминирующее положение на рынке корпоративных информационных систем России и стран СНГ: свыше 2 миллионов предприятий используют конфигурации на её основе для автоматизации документооборота и управления бизнес-процессами [2]. Стандартная подсистема «Работа с файлами» Библиотеки стандартных подсистем (БСП) не предоставляет встроенного механизма просмотра документов в интерфейсе приложения: пользователи вынуждены скачивать файлы на локальные компьютеры и открывать их во внешнем зарубежном программном обеспечении [6]. Это снижает эффективность работы, повышает нагрузку на сетевую инфраструктуру при частых загрузках, создаёт зависимость от иностранных продуктов на рабочей станции и риск утечки конфиденциальных данных за периметр корпоративной системы в нарушение требований Федерального закона № 152-ФЗ «О персональных данных». В условиях государственного курса на импортозамещение устранение данной зависимости приобретает стратегическое значение.

Таким образом, целью работы является разработка расширения конфигурации 1С:Предприятие, реализующего универсальный механизм просмотра файлов форматов PDF, DOCX, ODF, текстовых и табличных данных непосредственно в интерфейсе приложения без

загрузки на клиентский компьютер. Для достижения поставленной цели решаются следующие задачи: анализ и сравнение подходов к просмотру документов в корпоративных системах; исследование открытых JavaScript-библиотек рендеринга; проектирование модульной архитектуры системы с разделением клиентской и серверной логики; разработка механизма безопасной передачи файлов через временное хранилище и реализация Docker-контейнера для хостинга статических библиотек.

В ходе анализа подходов выделено четыре подхода (таблица 1). Установлено, что клиентский рендеринг на основе JavaScript-библиотек учитывает все требования безопасности, автономности и кроссплатформенности.

Таблица 1 – Сравнительный анализ подходов к просмотру документов

Подход	Требования к клиенту	Безопасность	Совместимость с 1С
Внешние приложения	Установка и лицензии	Данные копируются на устройство	Зависит от ОС и ПО
Облачные сервисы	Не требуется	Передача данных третьим лицам	Зависит от доступа в сеть
Серверный рендеринг	Не требуется	Данные на сервере; сложная конвертация	Средняя (требуются сервисы конвертации)
Клиентский рендеринг	Не требуется	Нет локальных файлов; контроль доступа в 1С	Высокая (HTML-поле)

Технической основой решения служит элемент формы «Поле HTML-документа», использующий движок WebKit [7]. Применение JavaScript-технологий для создания корпоративных веб-интерфейсов на платформе 1С:Предприятие рассмотрено в работе [1]. Разработанное решение реализовано в виде расширения конфигурации и включает:

- серверный слой (встроенный язык 1С): получение файла из информационной базы и подготовка данных для клиентской части;
- клиентский слой HTML/JavaScript: рендеринг без создания локальных копий файлов;
- диспетчер форматов: автоматический выбор обработчика;
- контейнеризованный хостинг статических библиотек: nginx-сервер в Docker-контейнере для кроссплатформенного автономного развертывания [3] [4].

Архитектура решения представлена на рисунке 1.

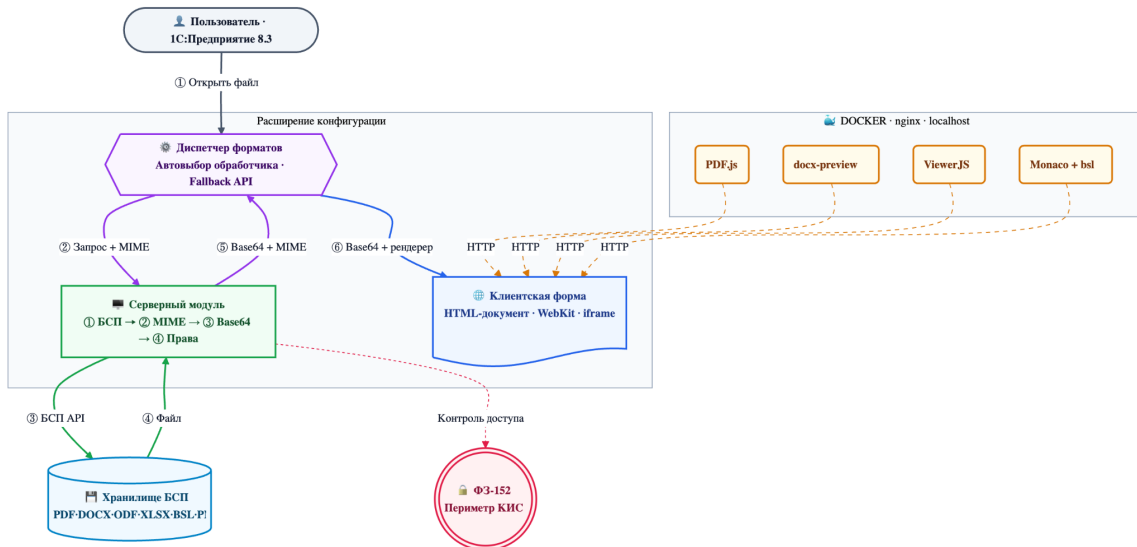


Рисунок 1 – Архитектура решения

Рендеринг поддерживаемых форматов реализован на открытых библиотеках без зарубежного проприетарного ПО: PDF.js – для PDF, docx-preview – для DOCX, ViewerJS – для

ODF, Monaco Editor – для текстовых и программных файлов. Интеграция с ролевой моделью доступа БСП исключает выход документов за периметр корпоративной системы, что соответствует требованиям Федерального закона № 152-ФЗ и стандартам защиты корпоративных данных [5].

По результатам функционального тестирования подтверждена корректность отображения документов всех поддерживаемых форматов, включая материалы с кириллицей, таблицами и изображениями. Реализация в виде расширения конфигурации обеспечивает подключение к типовым конфигурациям без изменения их исходного кода, а модульная архитектура обеспечивает расширение перечня форматов без доработки базовой функциональности. Разработанное решение устраняет зависимость рабочего места от зарубежного лицензионного ПО, тем самым обеспечивая *практический вклад* в задачу импортозамещения.

ЛИТЕРАТУРА

1. Кедрин В. С., Родюков А. В. Адаптивные технологии быстрого проектирования веб-интерфейсов корпоративных образовательных систем в рамках платформы «1С:Предприятие 8.3» // Информатика и образование. – 2022. – № 5(37). – С. 5–14. – DOI 10.32517/0234-0453-2022-37-5-5-14.
2. Акиндинов К. В., Картечина Н. В., Акиндинов В. В. Сущность программирования в экосистеме «1С:Предприятие» и его роль в цифровой трансформации бизнеса // Наука и образование. – 2025. – Т. 8, № 3. – С. 115.
3. Шейнман В., Стариков Д. Д., Тюменцев Д. В., Вавилов Г. Д. Повышение эффективности процессов разработки программного обеспечения: контейнерные технологии // Программные системы и вычислительные методы. – 2024. – № 4. – С. 151–161. – DOI 10.7256/2454-0714.2024.4.72015. – EDN JXRTJC.
4. Bondarenko A. S., Zaytsev K. S. Using container management systems to build distributed cloud information systems with microservice architecture // International Journal of Open Information Technologies. – 2023. – Vol. 11. – No. 8. – P. 17–23. – ISSN 2307-8162.
5. Ozman F. M. Systematic literature review on 'secure document management systems (DMS)' // World Journal of Advanced Engineering Technology and Sciences. – 2025. – Vol. 5, No. 1. – P. 1460–1469. – DOI 10.30574/wjaets.2025.15.1.0146.
6. Библиотека стандартных подсистем, редакция 3.1 // ИТС 1С [Электронный ресурс]. Режим доступа: <https://its.1c.ru/db/metod8dev#content:5856:hdoc> (дата обращения: 20.02.2026).
7. Ганиев С. Поле HTML-документа: для чего можно использовать // Хабр [Электронный ресурс] Режим доступа: https://habr.com/ru/companies/lad_/articles/813177/ (дата обращения: 07.03.2026).

УДК 004.932.2

О.А. Косминская (4 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

НЕЙРОСЕТЕВАЯ ОБРАБОТКА ПОКАЗАНИЙ ПРИБОРОВ В РЕАЛЬНОМ ВРЕМЕНИ

На современных предприятиях при работе сложных систем требуется контроль большого количества процессов с целью сбора статистики и быстрого реагирования на критические отклонения, что вручную часто невозможно из-за их обилия и труднодоступности. Хотя существует множество решений данной проблемы (от OpenCV до нейросетей), они узкоспециализированы: их эффективность зависит от моделей приборов, условий среды и выбранного метода распознавания, каждый из которых имеет свои ограничения по скорости и внедрению.

После анализа различных готовых решений для распознавания показаний приборов на основе нейросетей и компьютерного зрения (Viso.ai, Noema и пр.) были сделаны следующие выводы: большая часть открытых систем распознаёт конкретные приборы, а именно,

стрелочные манометры с круговой шкалой. Также большая часть из них недоступна для покупки в РФ.

Исходя из вышеизложенного, была сформулирована следующая цель работы: создание модуля системы для считывания показаний с манометров (полукруглая шкала с радиально расположенной стрелкой) и ленточных регистраторов (прямоугольная шкала с линейно перемещающимся указателем (самописец)) с помощью нейронных сетей в реальном времени. Типы приборов были выбраны в соответствии с требованиями заказчика, им же был предоставлен видеоматериал для считывания показаний. Система должна упростить механизм параллельного отслеживания показаний приборов и ускорить процесс реагирования на их нестандартные изменения (отклонение от нормы).

Архитектура спроектированной системы включает три модели нейронных сетей типа YOLO[1], дообученных на предоставленных заказчиком, размеченных средствами Roboflow [2] и CVAT [3], датасетах. Первая модель, типа YOLO Detection, отвечает за определение класса прибора на изображении — класс манометра или ленточного регистратора. Вторая и третья, типа YOLO Pose, отвечают за нахождение ключевых точек шкал и стрелок приборов на манометрах и самописцах соответственно.

Для обучения сетей и последующей обработки полученных данных был выбран язык программирования Python [4]. В частности, фреймворк для машинного обучения PyTorch, библиотеки OpenCV, NumPy и Matplotlib. Средой для обучения сетей был выбран Google Colab (в связи с возможностью подключения к удалённым средам выполнения, таким как «Графический процессор T4» (GPU Tesla T4) и TPU v5e-1). Для дальнейшей обработки данных, с использованием уже готовых обученных сетей была выбрана среда выполнения PyCharm (Community Edition).

Логика обработки полученных данных основана на различных подходах из нескольких статей, адаптированных под решаемую задачу. Во многих статьях перед распознаванием шкалы и стрелки проводится выравнивание изображения по некоторым ключевым точкам. Например, в [5] находятся наибольшие деления шкалы, затем через них проводится эллипс, и трансформация состоит в том, чтобы преобразовать его в окружность, а в работе [6] для подобных целей находятся и применяются углы ограничивающего прямоугольника прибора. Так как известно, что форма корпуса прибора представляет собой окружность, которая, соответственно, вписывается в правильный многоугольник (квадрат), и приведение полученного прямоугольника к квадрату, соответственно, выравнивает и окружность циферблата. Первые два метода были признаны неподходящими по следующим причинам: ключевые точки в обозначенных местах на шкале есть только у рассмотренных в статье моделей приборов и подобных им, а выравнивание по делениям шкалы перестаёт правильно работать, как только хотя бы одно из них становится видно менее чётко, чем другие. Соответственно, сеть его не распознаёт, и выстроенный по полученным точкам эллипс уже не является окружностью. Таким образом, для класса манометра был выбран способ с выравниванием по углам найденной рамки (bounding box), и приведением полученной части изображения к квадрату. В случае же самописца опытным путём было выяснено, что данный способ не подходит, поэтому выравнивание производилось на следующем этапе. Следующий этап – распознавание делений шкалы и вычисление показаний. Для класса манометров они вычисляются соответственно алгоритму, подобному описанному в [7]. Для регистраторов находилась более ровная из шкал путём вычисления дисперсии координат найденных точек на её делениях, производилось выравнивание этой шкалы по оси ОХ с помощью линейной интерполяции и расчёт значений в зависимости от близости концов стрелок, найденных на изображении как отдельные классы, к делениям шкалы.

Полученная система была протестирована на синтетических видео с несколькими приборами разных классов в кадре. Для тестовой версии была реализована визуализация найденных точек и значений в кадре самого видео в процессе запуска (рисунок 1), а для итоговой версии прототипа, с параллельной обработкой нескольких видеопотоков, был создан дашборд с помощью подключенной платформы мониторинга и анализа данных Grafana.

В результате для рассмотренных приборов была достигнута точность в 1-2 деления шкалы, т. е., погрешность в полградуса-градус для самописца, а также в 0,05-0,1 МПа для манометров, т. о., погрешность около 0,5-1,0 деления

Описанная методика может быть применена и к другим типам приборов, с уточнением для конкретного вида шкалы.

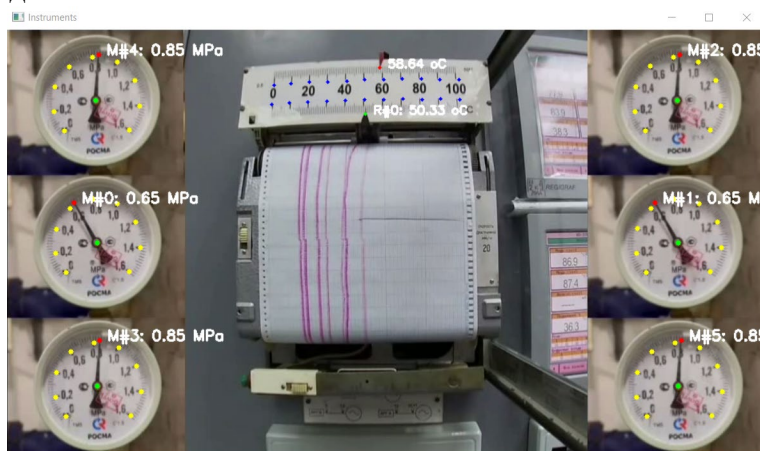


Рисунок 1 – демонстрация работы модуля

ЛИТЕРАТУРА

1. Документация Ultralytics YOLO. [Электронный ресурс] – Режим доступа: <https://docs.ultralytics.com/ru/> (дата обращения: 05.02.2025)
2. Roboflow Docs [Электронный ресурс] – Режим доступа: <https://docs.roboflow.com/> (дата обращения: 05.02.2023)
3. CVAT Overview. [Электронный ресурс] – Режим доступа: https://docs.cvat.ai/docs/getting_started/overview/ (дата обращения: 05.02.2025)
4. Python Documentation. [Электронный ресурс] – Режим доступа: <https://docs.python.org/3/> (дата обращения: 05.02.2023)
5. Reitsma M., Keller J., Blomqvist K., Siegwart R. Under pressure: learning-based analog gauge reading in the wild [Электронный ресурс] – Режим доступа: <https://arxiv.org/abs/2404.08785> (дата обращения: 04.02.2023)
6. Howells B., Charles J., Cipolla R. Real-time analogue gauge transcription on mobile phone [Электронный ресурс] – Режим доступа: <http://www.jjcvision.com/publications/howellsCVPR2021.pdf> (дата обращения: 04.02.2023)
7. Gallagher J. Turn Analog Dials into Digital Data with Computer Vision [Электронный ресурс] – Режим доступа: <https://blog.roboflow.com/read-analog-dials-computer-vision/> (дата обращения: 04.02.2023)

УДК 004.5:004.89

Б.Н. Костюков (4 курс бакалавриата),
О.А. Юсупова, ст. преподаватель.

РАЗРАБОТКА МОДУЛЯ ОТОБРАЖЕНИЯ ТЕГОВ НА ОСНОВЕ AI-АНАЛИЗА ОТЗЫВОВ

Розничная торговля активно перемещается в онлайн-среду, однако стандартные списки пользовательских отзывов зачастую создают информационную перегрузку, что негативно сказывается на конверсии из посетителей в покупателей [1]. Применение методов искусственного интеллекта для автоматического анализа массива отзывов и предоставления персонализированной выжимки в виде тегов становится ключевым инструментом повышения эффективности сервиса [2]. Реализация такого подхода требует разработки новых высокопроизводительных интерфейсных модулей, интегрируемых в существующую инфраструктуру мобильного приложения.

Целью настоящей работы являлась разработка клиентского модуля отображения тегов, генерируемых на основе AI-анализа текстов отзывов в мобильном приложении. В ходе исследования были решены следующие задачи: проведен сравнительный анализ существующих решений и паттернов проектирования пользовательских интерфейсов, изучена архитектура целевого Android-приложения, разработан план интеграции модуля, включающий проектирование слоев данных, бизнес-логики и пользовательского интерфейса, осуществлена программная реализация, проведено тестирование стабильности работы и анализ полученных результатов.

Технологическая реализация модуля базируется на современном стеке инструментов, обеспечивающих производительность и отказоустойчивость. В качестве основного языка программирования использован Kotlin, для построения пользовательского интерфейса применен декларативный фреймворк Jetpack Compose [3]. Асинхронная обработка данных и управление потоками выполняются с помощью Coroutines [4] и StateFlow [5]. Сетевое взаимодействие с внешним API Diginetica AI для получения результатов анализа отзывов реализовано посредством Ktor Client. Архитектурной основой модуля послужили принципы Clean Architecture с разделением на уровни данных (Data), бизнес-логики (Domain) и представления (Feature), а также паттерн MVVM в UI-слое [6].

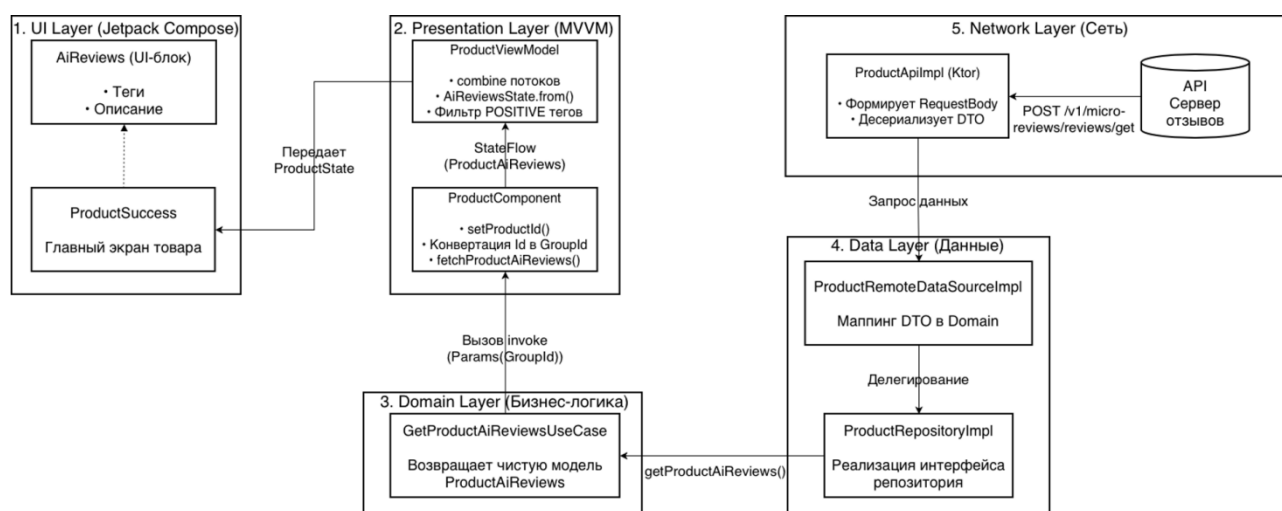


Рисунок 1 – Архитектура проекта и точка интеграции модуля

В результате проведенной работы был успешно интегрирован клиентский модуль, архитектура и место которого в общей структуре приложения представлены на рисунке 1. Реализован сетевой уровень для получения данных от внешнего сервиса AI-анализа, их последующей фильтрации и трансформации в отдельном компоненте бизнес-логики (UseCase) перед передачей в пользовательский интерфейс. Обеспечена асинхронность выполнения операций, исключая блокировку UI-потока. Комплексное тестирование подтвердило стабильность работы модуля: данные сохраняются при изменении конфигурации устройства (повороте экрана) благодаря использованию ViewModel, а анализ профилировщиком не выявил утечек памяти.

Разработанный модуль успешно решает задачу снижения информационной перегрузки потребителя, предоставляя оперативный доступ к обобщенной сути отзывов. В настоящее время функционал запущен в режиме пилотного проекта и проходит апробацию на ограниченной товарной номенклатуре. Полученные в ходе тестирования результаты демонстрируют готовность решения к полноценному внедрению на весь каталог приложения, что позволит повысить его конверсионные характеристики и улучшить пользовательский опыт.

ЛИТЕРАТУРА

1. Яковлева Е. А. Роль технологий искусственного интеллекта в цифровой трансформации экономики / Е. А. Яковлева, А. Н. Виноградов, Л. В. Александрова, А. П. Филимонов // Вопросы

- инновационной экономики. – 2023. – Т. 13, № 2. – С. 707–726. – URL: <https://www.elibrary.ru/item.asp?id=54060710> (дата обращения: 19.03.2026).
2. Демяниенко А. В. Искусственный интеллект и большие данные в e-commerce / А. В. Демяниенко // Информатика и системы управления. – 2024. – № 1. – С. 34–41. – URL: <https://www.elibrary.ru/item.asp?id=82310708> (дата обращения: 19.03.2026).
 3. Усов М. А. Разработка Android-приложений на основе декларативного подхода с применением Jetpack Compose / М. А. Усов, Н. В. Воинов, И. В. Зайцев // Современные технологии в теории и практике программирования : сборник материалов конференции, Санкт-Петербург, 22 апреля 2021 года. – Санкт-Петербург : ПОЛИТЕХ-ПРЕСС, 2021. – С. 126–127. – URL: <https://www.elibrary.ru/item.asp?id=45781370> (дата обращения: 19.03.2026).
 4. Агафонова А. С. Параллельное программирование в языке Kotlin / А. С. Агафонова, А. В. Горбунов // Международный журнал гуманитарных и естественных наук. – 2020. – № 6–2 (45). – С. 9–12. – URL: <https://www.elibrary.ru/item.asp?id=42924711> (дата обращения: 19.03.2026).
 5. Блинов И. Н. Асинхронность и масштабируемость: обработка данных в Android-приложениях / И. Н. Блинов, Т. А. Лядинский // Сборник научных работ. – Минск : БГУ, 2025. – С. 409–415. – URL: <https://www.elibrary.ru/item.asp?id=88885896> (дата обращения: 19.03.2026).
 6. Раджабов И. Н. Чистая архитектура и паттерн MVVM в практике разработки Android-приложения / И. Н. Раджабов, М. Л. Рысин // Студенческий вестник. – 2022. – № 21 (213). – С. 68–71. – URL: <https://www.elibrary.ru/item.asp?id=54271020> (дата обращения: 19.03.2026).

УДК 004.93

Е.А. Ключко (4 курс бакалавриата),
А.М. Хахина, д.т.н., профессор

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОПРЕДЕЛЕНИЯ ПОЗИЦИИ И ТРАЕКТОРИИ МЯЧА: НА ПРИМЕРЕ ВОЛЕЙБОЛА

Современное развитие спортивной индустрии характеризуется активной цифровизацией. В рамках международной конференции AI Journey 2025 президент России Владимир Путин отметил, что использование искусственного интеллекта позволит сделать спорт прозрачным, более интересным и честным [1]. В ходе опроса, проведенного компаниями «iSprotConnect» и «Choicely», изучены мнения 289 руководителей из разных сфер спортивного бизнеса [2]. Исследование показало, что 71% специалистов считает вовлечение зрителей важным для достижения целей спортивных организаций, в которых они работают. Технологии на основе искусственного интеллекта рассматриваются как ключевой фактор трансформации в данной области в ближайшие годы. Около 29% респондентов считают ИИ революционным в сфере взаимодействия с болельщиками.

Существующие системы Hawk-Eye [3] и BOLT6 [4] имеют высокую стоимость эксплуатации, что ограничивает распространение в любительских и полупрофессиональных лигах, и недоступны на российском рынке.

Целью разрабатываемого приложения является создание доступной системы детекции и трекинга спортивных объектов на основе модели компьютерного зрения. Предлагаемое решение обеспечивает прозрачность судейства и автоматический сбор игровых данных.

Для достижения поставленной цели необходимо решить следующие задачи:

- Провести сравнительный анализ существующих методов и моделей детекции мелких высокоскоростных объектов.
- Спроектировать архитектуру программного обеспечения.
- Реализовать клиент-серверное приложение с применением современного технологического стека.
- Провести экспериментальную оценку работы прототипа системы на реальных видеозаписях волейбольных матчей.

Реализация единой точки входа серверной части и сервисов бизнес-логики использовался фреймворк FastAPI на языке программирования Python. Его преимуществом

является высокая производительность благодаря асинхронной архитектура, а также автоматическая генерация документации на основе OpenAPI.

Сервис загрузки видеопотока реализован на языке программирования Go, обеспечивающий высокую пропускную способность и эффективную работу с конкурентными потоками. Сервис инференса свёрточной нейронной сети реализован на Python с использованием библиотеки PyTorch. В качестве модели выбрана TrackNetv5 [5], преимуществом которой является анализ нескольких последовательных кадров для обнаружения объекта в моменты коллизий и перекрытий.

Для долговременного хранения пользовательских данных, параметров камер и позиций мяча используется реляционная база данных PostgreSQL с расширением TimescaleDB. Для кэширования и хранения JWT токенов применяется Redis, обеспечивающий минимальные задержки при проверке аутентификации и доступе к часто запрашиваемым данным.

Использование Apache Kafka обеспечивает слабую связность между сервисами детекции и обработки видеопотока.

Мониторинг состояния системы возможен благодаря использованию Prometheus для сбора и хранения метрик и Grafana для их визуализации. Это позволяет администраторам наблюдать за производительностью каждого сервиса в реальном времени, а также определять узкие места системы.

Для обеспечения идентичности сред разработки, тестирования и эксплуатации используется Docker. Эта платформа позволяет создавать легковесные и переносимые контейнеры, которые эффективнее используют ресурсы по сравнению с виртуальными машинами [6].

Архитектура клиент-серверного приложения представлена на рисунке 1.

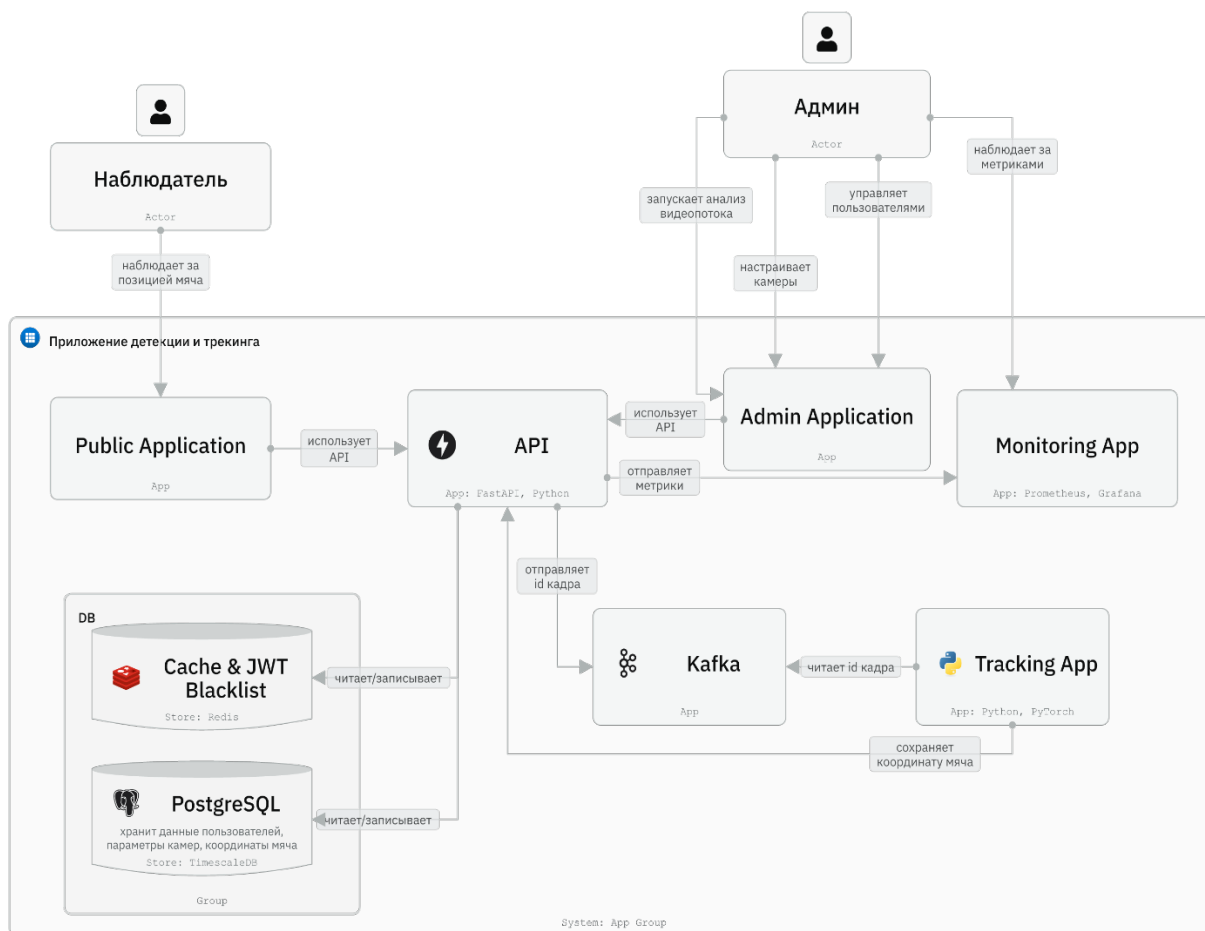


Рисунок 1 – Архитектура клиент-серверного приложения

Таким образом, предложенная микросервисная архитектура с асинхронным взаимодействием на базе Kafka и специализированными инструментами хранения (TimescaleDB, Redis) обеспечивает высокую масштабируемость и отказоустойчивость системы.

ЛИТЕРАТУРА

1. Использование ИИ в судействе делает спорт более честным, заявил Путин – РИА Новости, 19.11.2025. [Электронный ресурс]. Режим доступа: <https://ria.ru/20251119/putin-2056103157.html?in=t> (дата обращения: 15.12.2025)
2. The State of Sports Fan Engagement 2023, by iSportConnect and Chocicely. [Электронный ресурс]. Режим доступа: <https://www.chocicely.com/state-of-sports-fan-engagement-2023> (дата обращения: 23.10.2025)
3. Sony Group Portal - Making Sports More Exciting Through Visualization Technology. [Электронный ресурс]. Режим доступа: <https://www.sony.com/en/SonyInfo/technology/stories/entries/Hawk-Eye/> (дата обращения: 20.10.2025)
4. Home | BOLT6. [Электронный ресурс]. Режим доступа: <https://www.bolt6.ai/> (дата обращения: 13.10.2025)
5. Tang H. et al. TrackNetV5: Residual-Driven Spatio-Temporal Refinement and Motion Direction Decoupling for Fast Object Tracking //arXiv preprint arXiv:2512.02789. – 2025.
6. Gudelli V. R. Containerization technologies: ECR and Docker for microservices architecture //International Journal of Innovative Research in Management, Pharmacy and Sciences (IJIRMPs). – 2023. – Т. 11. – №. 3.

УДК 004.89

В.В. Краснов (3 курс бакалавриата),
О.В. Александрова, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ КРАТКОСРОЧНОГО ПРОГНОЗИРОВАНИЯ ГИПОГЛИКЕМИИ НА ОСНОВЕ ВРЕМЕННЫХ РЯДОВ НЕПРЕРЫВНОГО МОНИТОРИНГА ГЛЮКОЗЫ (CGM)

Гипогликемия — это опасное падение уровня глюкозы (сахара) в крови, вызывающее слабость, голод, потливость, дрожь и спутанность сознания, может наступить кома. Неонатальная гипогликемия (НГ) - снижение концентрации глюкозы в плазме венозной крови $< 2,6$ ммоль/л в периоде новорожденности независимо от срока гестации [5]. Для контроля уровня глюкозы в крови и предупреждения развития гипогликемии используются специальные датчики, производится непрерывный мониторинг глюкозы (НМГ) – Continuous Glucose Monitoring (CGM) [4]. Система НМГ позволяет контролировать один из самых важных параметров гликемии – время нахождения в целевом диапазоне, или time in range (TIR) [5]. Врожденный гиперинсулинизм (высокое содержание инсулина) приводит к падению содержания глюкозы, и если не предпринимать никаких мер, то высока вероятность развития гипогликемии [6].

В работе используются предоставленные Государственным Педиатрическим Медицинским Университетом г. Санкт-Петербурга данные НМГ пациентов с врожденным органическим гиперинсулинизмом, они предварительно обезличены, для НМГ используется система FreeStyle Libre. Данные предоставляются в виде csv файлов со множеством столбцов, каждый файл – это выгрузка по одному пациенту. Одновременно в каждой строке заполнены столбцы времени события, тип события (если это измерение, то тип измерения) и, в зависимости от типа измерения заполнен один из столбцов с показаниями уровня глюкозы. Так же есть событие типа «прием пищи» и «прием лекарств», но эти типы событий почти не предоставлены в записях, возможно потому, что должны заноситься в программу вручную без использования какой-либо автоматики.

Видно, что используемых характеристик не так много, фактически, есть только время измерения и само значение измерения, а также возможные производные от этих двух, такие как скорость изменения показания, время суток или другие временные интервалы в течении суток и, возможно, ускорение изменения самого содержания глюкозы [5]. По характеристике ускорения есть сомнение, что его можно получить с необходимой точностью, т.к. измерения от прибора идут с интервалами от 5 до 15 минут и в течении таких интервалов времени (между точек данных в записях файла) измерения могут, например, прекратить расти и начать падать, а за тем снова начать расти. Таким образом, уже скорость является достаточно усредненной характеристикой, а ускорение теряет в точности еще больше.

Различные типы ML-моделей для предсказаний предполагают использование данных для обучения в различных представлениях, например, для использования методов деревьев решений и случайного леса необходимо, что бы каждая строка содержала все характеристики, по которым будет приниматься решение [1]. В нашем случае, при использовании временных рядов, будет необходимо предоставить в одной строке несколько идущих подряд через равные промежутки времени измерений, временную характеристику какого-то из этих измерений, например, начального, и само предсказываемое значение через определенный интервал времени. Таким образом, перед осуществлением ML-обучения для конкретного метода необходимо из существующих данных создать подходящий для обучения набор данных. Так же необходимо решить проблему привязки предсказания к конкретному времени, чтобы модель не считала, что временная метка строго определяет предсказание, а что бы передаваемая очередная строка с измерениями была применима к некоторому интервалу времени вокруг реально произведенного измерения.

После подготовки данных проводится обучение ML-модели на некоторой большей части данных, оставшаяся часть данных будет использоваться для валидации модели [1]. При этом необходимо сделать так, чтобы данные конкретного пациента целиком находились либо только в наборе тренировочных данных, либо только в данных валидации. Иначе есть большая вероятность того, что ML-модель запомнит при обучении паттерны изменения одного пациента и при валидации найдет эти же паттерны. Это приведет к тому, что метрики, получаемые при валидации обученной модели, могут ошибочно указывать на хорошее обучение.

Для предсказаний числа можно использовать линейную/полиномиальную регрессию, экспоненциальное сглаживание, ARIMA/SARIMA, Random Forest, рекуррентные нейросети (LSTM / GRU).

В задачах предсказания некоторого числа используются следующие метрики.

MAE (Mean Absolute Error) - средняя абсолютная ошибка, считает среднюю разницу между предсказанием и фактом по модулю.

MSE (Mean Squared Error) – средняя квадратичная ошибка, считает средний квадрат разницы.

RMSE (Root Mean Squared Error) – корень из среднеквадратичной ошибки.

Коэффициент детерминации – показывает, какая доля дисперсии целевой переменной объясняется моделью.

ЛИТЕРАТУРА

1. Уткин Л.В. Машинное обучение. Лекции по курсу, СПбПУ, 2015 [Электронный ресурс] Режим доступа: <https://levutkin.github.io/teaching/machine-learning> (дата обращения: 27.02.2026)
2. Франсуа Шолле. Глубокое обучение на Python. – 2025. – С. 203-225.
3. Kaggle: Your Machine Learning and Data Science Community [Электронный ресурс] Режим доступа: <https://www.kaggle.com/> (дата обращения: 27.02.2026).
4. Development and Validation of a Machine Learning Model to Predict Near-Term Risk of Iatrogenic Hypoglycemia in Hospitalized Patients. JAMA Network Open | Diabetes and Endocrinology. – 2021.
5. Data-based modeling for hypoglycemia prediction: Importance, trends, and implications for clinical practice. [Электронный ресурс] Режим доступа: <https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2023.1044059/full> (дата обращения: 27.02.2026).

6. Hypoglycemia event prediction from CGM using ensemble learning. [Электронный ресурс] Режим доступа: <https://www.frontiersin.org/journals/clinical-diabetes-and-healthcare/articles/10.3389/fcdhc.2022.1066744/full> (дата обращения: 27.02.2026).

УДК 004.41

С.А. Крючков (4 курс бакалавриата),
О.А. Юсупова, ст. преподаватель

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ И ВИЗУАЛИЗАЦИИ ЛОГИСТИЧЕСКИХ ДАННЫХ

В современной логистике наблюдается острый дефицит доступных и эффективных инструментов для оперативного управления и визуализации данных о грузоперевозках [1, 5]. Существующие на рынке корпоративные системы (TMS) отличаются высокой стоимостью внедрения и избыточной сложностью для малого и среднего бизнеса [3, 4, 7], в то время как использование табличных процессоров не обеспечивает необходимой безопасности и целостности данных. Целью данной работы является разработка полнофункционального веб-приложения, обеспечивающего эффективное управление информацией о перевозках, анализ статистики и формирование отчетов [1, 2].

Архитектура разработанного приложения построена по принципу разделения клиентской и серверной частей (архитектура клиент-сервер). Клиентская часть разработана с использованием библиотеки React 18, что позволило создать адаптивный интерфейс с использованием компонентного подхода. Серверная часть построена на платформе Node.js с использованием фреймворка Express. Для обеспечения безопасности данных внедрена система аутентификации на основе JSON Web Tokens (JWT); пароли пользователей защищены с помощью алгоритма хеширования bcrypt.

Ключевой особенностью приложения является функциональный дашборд, реализованный с помощью библиотеки Recharts [1, 5]. Система визуализирует такие метрики, как общее количество перевозок, распределение по статусам и популярные маршруты [2, 5]. Кроме того, в приложении реализованы визуализация маршрутов на карте и многоуровневая система фильтрации данных [5, 6]. Для интеграции с бизнес-процессами предусмотрен функционал экспорта отчетов в формат Excel (.xlsx) с помощью библиотеки ExcelJS.

В текущей версии в качестве хранилища используется JSON-файл, что упрощает развертывание на этапе разработки. Однако архитектура приложения предусматривает слой абстракции (dataStore), что позволяет в будущем выполнить переход на реляционную СУБД (например, PostgreSQL) без изменения основной бизнес-логики [1, 6].

Практическая ценность работы заключается в создании масштабируемого решения, которое позволяет компаниям снизить операционные издержки за счет цифровизации логистических операций и повышения прозрачности процессов [1, 2, 5, 6, 7].

ЛИТЕРАТУРА

1. Web-Based Supply Chain Management System: Empowering Enterprises with Real-Time Data and Performance Dashboards [Электронный ресурс] // Journal of Engineering and Science. – 2025.
2. The Role of Data Analytics and Reporting in Transportation Management Systems [Электронный ресурс] // Manhattan Associates. – 2025.
3. Transportation Logistics Automation (TMS) [Электронный ресурс] // CDC Software Solutions.
4. 1С:Предприятие 8. TMS Логистика. Управление перевозками [Электронный ресурс]. – 1С, 2025.
5. TMS Analytics and Data Visualization Ease Logistics Decision-Making [Электронный ресурс] // CTSI Global. – 2021.
6. Unified Logistics Analytics: TMS + FMS + WMS Insights in One View [Электронный ресурс] // TachyonHub.
7. Logistics TMS Software: Streamline Your Tools [Электронный ресурс] // DAT Solutions. – 2024.

РАЗРАБОТКА ВЕБ-ПЛАТФОРМЫ ДЛЯ ВИЗУАЛИЗАЦИИ И АНАЛИЗА РЕЗУЛЬТАТОВ РАСПОЗНАВАНИЯ ДЕЙСТВИЙ ЧЕЛОВЕКА В ОФИСНОЙ СРЕДЕ

Современные системы видеонаблюдения и анализа поведения человека находят широкое применение в офисных пространствах для повышения безопасности, оптимизации рабочих процессов и оценки эффективности использования помещений. Системы автоматического распознавания действий сотрудников (работа за компьютером, совещания, перемещения) на основе методов глубокого обучения генерируют размеченные видео и статистические данные, однако для эффективного использования этих результатов необходимо удобное и наглядное представление. Существующие решения часто либо узкоспециализированы, либо сложны в настройке, что обуславливает актуальность создания универсальной веб-платформы визуализации. Целью работы является разработка архитектуры и реализация такой платформы, обеспечивающей просмотр видео с наложенными аннотациями, построение графиков активности, фильтрацию данных по различным параметрам и экспорт отчётов.

Разработанная платформа построена по микросервисной архитектуре. API Gateway, реализованный на Gin [1], принимает все запросы от клиентского приложения и маршрутизирует их к соответствующим сервисам. Как видно из рисунка 1, сервис статистики и сервис экспорта обращаются к базе данных PostgreSQL [2], а приложение на React [3] загружает видео непосредственно из MinIO [4] по подписанному URL.

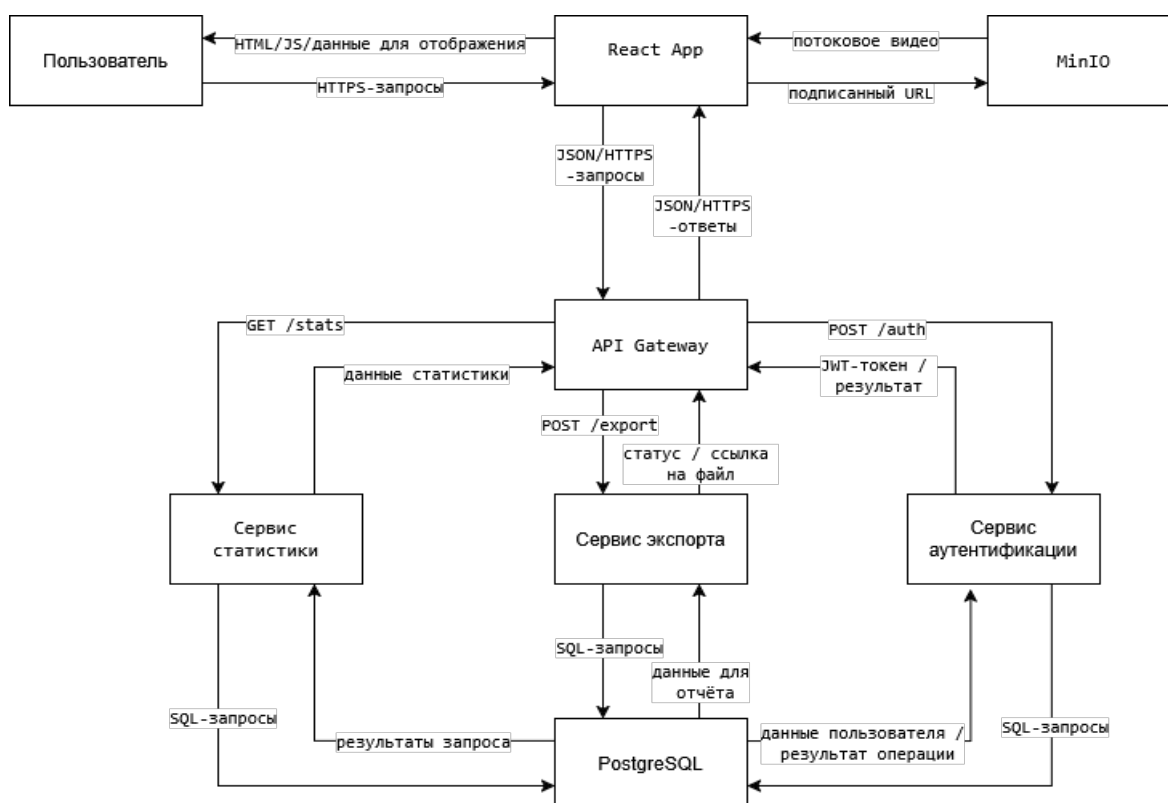


Рисунок 1 – Архитектура веб-платформы

Выбор технологий обусловлен требованиями к производительности и масштабируемости: Go обеспечивает высокую скорость обработки запросов и эффективную конкурентность, Gin – удобную маршрутизацию и middleware. PostgreSQL надёжно хранит данные с поддержкой JSON-полей. MinIO позволяет организовать отказоустойчивое хранение видео и потоковую передачу через HTTP Range. React с библиотеками Chart.js [5] даёт возможность строить интерактивные графики.

Реляционная схема базы данных включает таблицы: employees (сотрудники), sessions (сессии видеонаблюдения), action_types (справочник действий), actions (интервалы действий с временными метками), users (пользователи платформы). Такая структура позволяет эффективно выполнять агрегирующие запросы и фильтрацию.

REST API спроектировано с учётом потребностей клиента. Основные эндпоинты: /api/v1/auth/login – аутентификация, возвращает JWT; /api/v1/employees – получение списка сотрудников; /api/v1/sessions – список сессий с фильтрацией по дате, сотруднику, зоне; /api/v1/sessions/{id}/video – потоковая передача видео (перенаправление на подписанный URL MinIO [4]); /api/v1/stats/daily – ежедневная статистика действий; /api/v1/export/pdf – асинхронный запуск генерации PDF-отчёта. Асинхронный экспорт реализован через фоновые горютины с сохранением статуса задачи в таблице export_jobs. Клиент опрашивает статус и получает ссылку на готовый файл. Такой подход исключает блокировку пользователя при длительных операциях. Аутентификация основана на JWT, middleware проверяет токен и извлекает роль пользователя (admin, analyst, viewer) для разграничения доступа.

Клиентское приложение разработано на React и включает модуль аутентификации, дашборд с общей статистикой, страницу сессий с фильтрацией и списком записей, видеоплеер, страницу детальной аналитики с временными линиями действий и графиками распределения, интерфейс для запроса отчета и скачивания его. Видеоплеер отображает видео с наложенными bounding boxes и скелетными моделями, которые формируются на основе алгоритма детектирования объектов с использованием бинарных нейронных сетей [6].

Разработанная веб-платформа обеспечивает удобную визуализацию и анализ результатов распознавания действий человека в офисе. Применение микросервисной архитектуры на Go и современных фронтенд-технологий позволило создать масштабируемое и производительное решение. В дальнейшем планируется расширить возможности настройки отчётов.

ЛИТЕРАТУРА

1. Gin Web Framework [Электронный ресурс]. URL: <https://gin-gonic.com/> (дата обращения: 01.03.2026).
2. PostgreSQL Documentation [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения: 01.03.2026).
3. React – библиотека для пользовательских интерфейсов [Электронный ресурс]. URL: <https://reactjs.org/> (дата обращения: 01.03.2026).
4. MinIO Object Storage [Электронный ресурс]. URL: <https://min.io/> (дата обращения: 01.03.2026).
5. Chart.js [Электронный ресурс]. URL: <https://www.chartjs.org/> (дата обращения: 01.03.2026).
6. Королев Д. О., Малеев О. Г. Исследование эффективности применения бинарных нейронных сетей при детектировании объекта на изображении // Информатика и ее применения. – 2023. – Т. 17, № 3. – С. 88-92. – DOI 10.14357/19922264230312.

УДК 004.925

Я.С. Котова (4 курс бакалавриата),
А.В. Петров, ст. преподаватель

СОЗДАНИЕ ПЛАТФОРМЫ ОТРАСЛЕВОГО ПРОФИЛИРОВАНИЯ ДЛЯ ПОДГОТОВКИ ИНЖЕНЕРОВ

В современной системе высшего образования наблюдается значительный разрыв между содержанием учебных программ и требованиями промышленности к компетенциям инженеров [3]. Выпускники вузов часто обладают устаревшими или строго теоретическими знаниями, что приводит к дополнительным затратам работодателей на их дообучение. Существующие цифровые решения, такие как ERP-системы вузов (1С:Университет, Ядро ВАК) и платформы карьерного ориентирования (hh.ru, LinkedIn), не обеспечивают интеграции между образовательными программами, профессиональными стандартами и запросами рынка труда [2].

Предметная область исследования лежит на стыке цифровых технологий в образовании, управления персоналом и анализа больших данных для прогнозирования потребностей отрасли. Цель работы – разработать модель платформы отраслевого профилирования, позволяющую сократить время актуализации образовательных программ за счет семантического сопоставления компетенций вузов и работодателей в рамках ФГОС, утвержденных Минобрнауки РФ [4].

Для достижения цели решаются следующие задачи:

- анализ разрыва между программами по ФГОС, профессиональными стандартами (Минтруд РФ) и требованиями работодателей (например, Т-банк) [1];
- сравнение подходов к интеграции образовательных и профессиональных стандартов;
- разработка модели данных и алгоритма семантического сопоставления компетенций;
- создание прототипа модуля подачи заявок от работодателей.

Профилирование начинается с момента поступления, с активным взаимодействием работодателей с 3-го курса; 4-й курс посвящен финальной специализации и трудоустройству. Базовая программа первых трех лет соответствует ФГОС, обеспечивая фундаментальные компетенции; адаптивный трек 4-го года дополняется модулями, предлагаемыми платформой на основе запросов отрасли, без нарушения государственных стандартов [4].

Научная новизна заключается в методологии семантического сопоставления систем компетенций (ФГОС и профессиональных стандартов) и модели интеграции ERP-систем вузов с запросами работодателей, обеспечивающей адаптацию треков в рамках государственных требований. Практическая значимость: вузы повышают конкурентоспособность, работодатели снижают затраты на адаптацию кадров, государство оптимизирует кадровую политику; ключевые показатели – рост трудоустройства выпускников и сокращение времени поиска работы.

ЛИТЕРАТУРА

1. Профессиональные стандарты в инженерии (Минтруд РФ, 2026). - Режим доступа: profstandart.rosmintrud.ru.
2. 1С:Университет: интеграция с профстандартами // Вестник вузов. – 2025. – № 2. – С. 45–52.
3. Анализ разрыва компетенций в инженерии / Иванов А.А. // Инновации в образовании. – 2024. – Т. 15. – № 3. – С. 112–120.
4. Фальков В. Валерий Фальков провел установочное совещание по реализации пилотного проекта, направленного на изменение уровней профессионального образования // Официальный сайт Минобрнауки РФ. – 2026. – Режим доступа: <https://minobrnauki.gov.ru/press-center/news/novosti-ministerstva/99710/>.

УДК 004.4

А.А. Ларионенко (4 курс бакалавриата),
Е.С. Орлов, ст. преподаватель

МИКРОСЕРВИСНАЯ ПЛАТФОРМА ЦИФРОВЫХ УНИВЕРСИТЕТСКИХ СЕРВИСОВ С ЦЕНТРАЛИЗОВАННОЙ АУТЕНТИФИКАЦИЕЙ И МОДУЛЕМ КОНТРОЛЯ ПОСЕЩАЕМОСТИ

В современных университетских информационных системах одной из типичных проблем является разрозненность цифровых сервисов, сопровождаемая дублированием логики аутентификации, несогласованностью ролевых моделей и высокой стоимостью сопровождения интеграций. На практике это приводит к усложнению администрирования, росту числа точек отказа и снижению темпа развития пользовательских сервисов для студентов, преподавателей и сотрудников.

Цель работы заключается в разработке и внедрении прикладной платформы цифровых университетских сервисов, объединяющей студенческий портал, модуль заявок и модуль

контроля посещаемости в рамках согласованной микросервисной архитектуры [1, 2] с единым контуром идентификации, централизованным управлением учетными записями и формализованными межсервисными контрактами.

В рамках проекта реализована платформа, включающая пять backend-сервисов: main (студенческий портал), services (заявки), traffic (посещаемость), sso (централизованная аутентификация и provisioning) и schedule (шлюз доступа к данным расписания). Сервисы реализованы с использованием FastAPI [3] и SQLAlchemy [4]. Система поддерживает вход студентов через CAS, SSO-вход для сотрудников, преподавателей и администраторов, запуск мини-приложений по короткоживущему launch token, подачу и обработку заявок, а также проведение и сопровождение сессий посещаемости.

Центральным архитектурным элементом платформы выступает сервис sso, обеспечивающий единую модель идентичности для различных доменных ролей [2]. В нем реализованы механизмы централизованной аутентификации, выдачи токенов и provisioning API для создания и синхронизации учетных записей доменных сущностей с контролем связки app/role/entity_id. Такой подход позволяет согласованно управлять доступом в распределенной системе и исключает дублирование логики авторизации в прикладных сервисах. Для студентского контура дополнительно используется отдельный серверно-валидационный student-session token, что позволяет разделить пользовательские сценарии и усилить контроль доступа [7].

Наиболее прикладным компонентом платформы является модуль «Посещаемость», реализующий киоск-ориентированный сценарий проведения занятий. Регистрация устройства выполняется по одноразовому PIN-коду, после чего преподаватель запускает сессию занятия, связанную с аудиторией и расписанием. Отметка присутствия студента осуществляется через мобильный интерфейс посредством сканирования QR-кода. Для защиты от повторного использования QR-данных применяется механизм ротации короткоживущих токенов на основе HMAC-SHA256 с проверкой текущего и предыдущего временного окна [7]. Это повышает надежность фиксации присутствия в реальных условиях эксплуатации и делает сценарий пригодным для практического применения.

В модуле «Услуги» реализованы динамические формы заявок на основе JSON-описания полей, ролевой workflow обработки (pending, in_progress, completed, rejected) [2], поддержка вложений и история ответов по заявке. Такая организация позволяет расширять состав предоставляемых сервисов без переработки клиентской и серверной логики под каждый новый тип обращения, а также обеспечивает единообразный сценарий взаимодействия студентов и исполнителей.

Для изоляции интеграции с внешним источником данных расписания реализован отдельный сервис schedule, выступающий единым шлюзом доступа к соответствующим данным [1]. Это решение не является самостоятельной целью проекта, однако позволяет сократить связанность доменных компонентов, централизовать логику взаимодействия с внешним API и упростить сопровождение системы при изменении форматов или правил доступа к данным расписания.

С точки зрения эксплуатации в stateful-сервисах используется миграционная дисциплина на базе Alembic [5], а развертывание платформы осуществляется в контейнерной среде Docker Compose [6]. Это обеспечивает воспроизводимость запуска, предсказуемость изменений схем данных и упрощает развитие платформы в процессе расширения функциональности.

ЛИТЕРАТУРА

1. Newman S. Building Microservices. 2nd ed. Sebastopol: O'Reilly Media, 2021.
2. Richardson C. Microservices Patterns. Shelter Island: Manning Publications, 2018.
3. FastAPI Documentation [Электронный ресурс]. URL: <https://fastapi.tiangolo.com/> (дата обращения: 05.03.2026).
4. SQLAlchemy 2.0 Documentation [Электронный ресурс]. URL: <https://docs.sqlalchemy.org/> (дата обращения: 05.03.2026).

5. Alembic Documentation [Электронный ресурс]. URL: <https://alembic.sqlalchemy.org/> (дата обращения: 05.03.2026).
6. Docker Documentation [Электронный ресурс]. URL: <https://docs.docker.com/> (дата обращения: 05.03.2026).
7. OWASP Cheat Sheet Series: JSON Web Token [Электронный ресурс]. URL: <https://cheatsheetseries.owasp.org/> (дата обращения: 05.03.2026).

УДК 004.41:550.8

И.А. Кулагин (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА КОРПОРАТИВНОЙ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ ДЛЯ ОЦЕНКИ ВЕРОЯТНОСТИ ГЕОЛОГИЧЕСКОГО УСПЕХА (GPOS)

В настоящее время в нефтегазовой отрасли геологоразведочные работы носят очень рискованный характер с финансовой точки зрения, так как в мире крупные месторождения истощаются, а поиски новых сопровождаются значительной степенью неопределенности. Одним из ключевых этапов перед принятием решения о бурении скважин является оценка геологических рисков – расчет коэффициента геологического успеха (КГУ, или GPOS — Geological Probability of Success). Оценка основывается на вероятностном анализе пяти независимых факторов углеводородной системы: наличия и качества коллектора, существования ловушки, генерации углеводородов, их миграции и сохранности залежи [1].

Анализ существующей практики показал, что в большинстве случаев оперативная оценка КГУ осуществляется специалистами с использованием разрозненных электронных таблиц (например, Excel или P7). Популярность данного инструмента обусловлена его доступностью и высокой гибкостью: универсальность таблиц позволяет легко адаптировать формулы для проведения расчетов в строгом соответствии с локальной корпоративной методикой. С другой стороны, на рынке представлены мощные специализированные программные комплексы зарубежного производства (такие как GeoX от Schlumberger, решения Rose & Associates, модули оценки рисков в Petrel). Однако в текущих геополитических условиях их использование сопряжено с неприемлемыми санкционными рисками (невозможностью продления лицензий и получения технической поддержки). Это диктует острую необходимость импортозамещения и создания собственного ПО [2].

В текущих условиях усложнения добычи углеводородов возникает необходимость в стандартизации и автоматизации проведения и учета расчетов в рамках информационной системы компании с учетом особенностей корпоративной методики [1].

Целью данной работы является разработка интерактивного веб-приложения (ВІ-платформы) для стандартизации оценки вероятности геологического успеха и централизованного управления данными геологоразведочных проектов.

Для достижения поставленной цели необходимо решение следующих *задач*:

1. Провести обзор существующих программных решений расчета, хранения и ранжирования КГУ.
2. Изучить формализованную корпоративную методологию расчета КГУ и учесть её особенности для дальнейшей разработки интерфейсов и логики ПП.
3. Сформулировать функциональные и нефункциональные требования к разрабатываемому веб-приложению.
4. Спроектировать архитектуру приложения и структуру базы данных.
5. Спроектировать и реализовать пользовательский веб-интерфейс, включающий интерактивный аналитический инструмент и картографический модуль, представленный на рисунке 1.
6. Провести тестирование разработанного ПП и оценить его пригодность для использования в корпоративной среде.

Приложение реализует клиент-серверную архитектуру с использованием современного технологического стека. Серверная логика и интерфейсная часть реализованы на языке Python с использованием библиотеки Dash, которая позволяет создавать сложные аналитические приложения с развитой логикой и визуализацией без написания объемного кода на JavaScript [3]. Роль СУБД выполняет база данных PostgreSQL с применением механизма пула соединений, реализованного с помощью библиотеки psycopg2.

Ключевой функциональной особенностью платформы является интерактивная матрица оценки. Интерфейс динамически адаптируется под действия эксперта: при выборе уровня достоверности исходных данных система адаптирует дальнейший выбор параметров и автоматически блокирует нерелевантные ячейки.

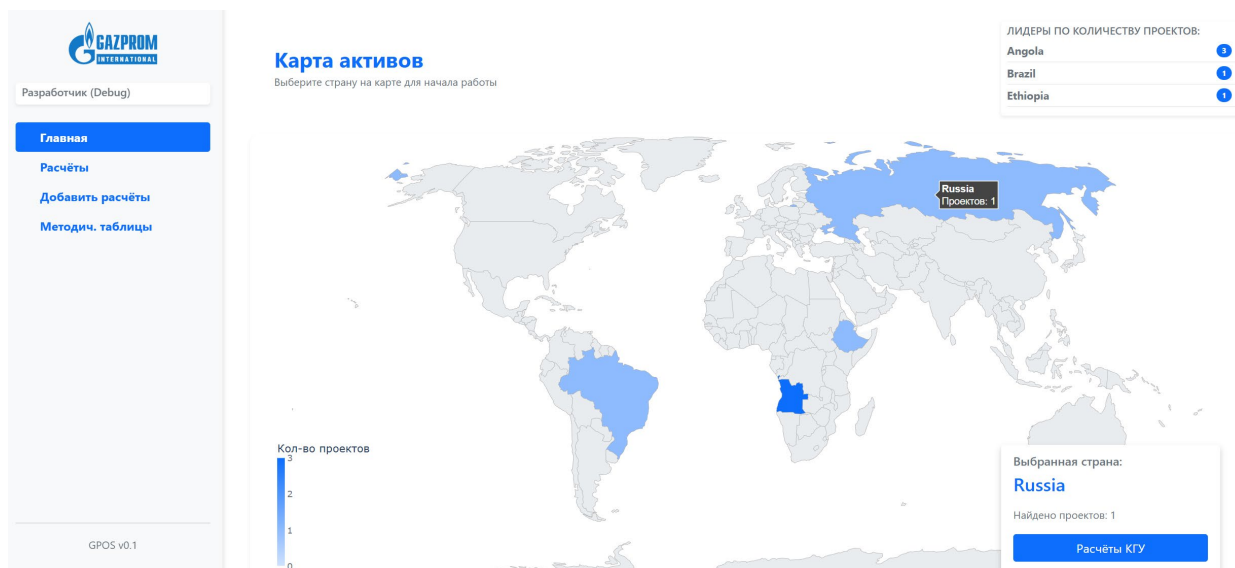


Рисунок 1 – Интерактивная карта проекта GPOS

Платформа позволяет сопоставлять расчеты КГУ в динамике, автоматически подсвечивая улучшение или ухудшение прогноза по каждому фактору после получения новой геолого-геофизической информации. Результаты расчетов могут быть экспортированы в формате отчетов Excel для дальнейшего использования в экономическом моделировании.

Практическая значимость работы заключается в создании функционального ИТ-продукта, который снижает влияние “человеческого фактора”, ускоряет процесс подготовки проектной документации и формирует единую, защищенную корпоративную базу знаний по геологическим рискам. Инструмент успешно решает задачу цифровизации и стандартизации расчетов КГУ, обеспечивая прозрачность принятия дорогостоящих управленческих решений в геологоразведке [5].

В результате работы внедрен полнофункциональный прототип ВІ-приложения, соответствующий требованиям промышленной программной инженерии. Инструмент успешно решает задачу стандартизации расчетов КГУ, обеспечивая прозрачность и обоснованность принятия дорогостоящих управленческих решений в геологоразведке.

ЛИТЕРАТУРА

1. Методика расчета коэффициента геологического успеха и вероятности открытия залежей УВ. — 2023. — 37 с. (Утвержденная корпоративная методика).
2. Поляков А. А., Мурзин Ш. М. Международный опыт анализа геологических рисков // Нефтегазовая геология. Теория и практика. — 2012. — Т. 7, № 4. — С. 1–22.
3. Dash Documentation. Plotly: Build Data Apps in Python. [Электронный ресурс] Режим доступа: <https://dash.plotly.com/introduction> (дата обращения: 10.03.2026)
4. Роуз П. Р. Анализ рисков и управление нефтепоисковыми проектами / пер. с англ. — М.-Ижевск: НИЦ «Регулярная и хаотическая динамика», Институт компьютерных исследований, 2011. — 304 с.

5. Карнаухов А. М. Направления развития «цифрового рывка» в геологоразведке // Нефтегазовая геология. Теория и практика. — 2019. — Т. 14, № 4. — С. 1–12.

УДК 004.75

А.А. Кумачев, Е.А. Башлачева (4 курс бакалавриата),
А.В. Гончаров, ст. преподаватель

ПРИМЕНЕНИЕ ISTIO ДЛЯ СБОРА ИНФОРМАЦИИ ОБ АКТИВНОСТИ ПОЛЬЗОВАТЕЛЕЙ В МИКРОСЕРВИСНОМ ПРИЛОЖЕНИИ И ОРГАНИЗАЦИИ SERVICE MESH

Современные микросервисные системы требуют эффективных средств наблюдаемости и анализа пользовательской активности. В традиционных архитектурах сбор информации о действиях пользователей реализуется непосредственно в коде сервисов, что усложняет поддержку и развитие системы. В условиях распределённой среды возникает задача централизованного получения данных об активности пользователей без внесения изменений в бизнес-логику приложений.

Одним из способов решения данной проблемы является использование Istio для сбора шаблонов активности пользователей. Применение Istio также позволит организовать Service Mesh, что обеспечивает прозрачное управление сетевыми взаимодействиями между сервисами на уровне инфраструктуры: это позволяет решить множество инфраструктурных задач без изменения бизнес-логики сервисов: балансировку нагрузки, канареечные релизы, автоматические проверки здоровья, а также реализовать паттерны отказоустойчивости, такие как retry, timeout, и circuit breaker [1].

Istio использует sidecar прокси Envoy, которые автоматически внедряются в каждый pod и перехватывают входящий и исходящий сетевой трафик. Это позволяет централизованно собирать информацию о пользовательской активности, и сохранять ее для дальнейшего анализа, а также реализовать функции маршрутизации, балансировки нагрузки, управления версиями сервисов, а также обеспечивать защищённое взаимодействие между сервисами с использованием механизмов mTLS [2][3].

Таким образом, целью данной работы является продемонстрировать реализацию service mesh и функционала сбора информации об активности пользователей, реализованного с использованием платформы Istio в среде Kubernetes.

Для достижения цели работы было разработано демонстрационное микросервисное приложение, моделирующее систему обработки платежей. Архитектура приложения включает три микросервиса: сервис управления учетными записями пользователей (Account Service), сервис обработки платежей (Payment Service) и сервис сбора и обработки информации о запросах пользователей – User Telemetry Service.

Account Service обеспечивает создание и хранение учетных записей пользователей. После регистрации пользователя сервис инициирует создание баланса посредством удалённого вызова процедур в Payment Service. Пользователь может взаимодействовать с сервисами посредством HTTP запросов. Взаимодействие между сервисами реализовано с использованием протокола gRPC.

Для управления сетевым взаимодействием между микросервисами в систему был интегрирован service mesh на базе Istio. В процессе настройки платформы была выполнена установка компонентов Istio в кластер Kubernetes и включено автоматическое внедрение sidecar прокси Envoy в pods микросервисов.

Для идентификации пользователей в системе в каждом запросе используется уникальный идентификатор, передаваемый в HTTP-заголовках. Архитектура данного решения представлена на рисунке 1 – перехват входящего трафика осуществляется на уровне ingress-шлюза Istio, который фиксирует параметры запроса, включая время его поступления и идентификатор пользователя. Полученные данные передаются в систему мониторинга. Определение временных интервалов активности пользователей осуществляется с

использованием подхода session timeout: последовательные запросы пользователя считаются принадлежащими одной сессии, если интервал между ними не превышает заданного порогового значения.

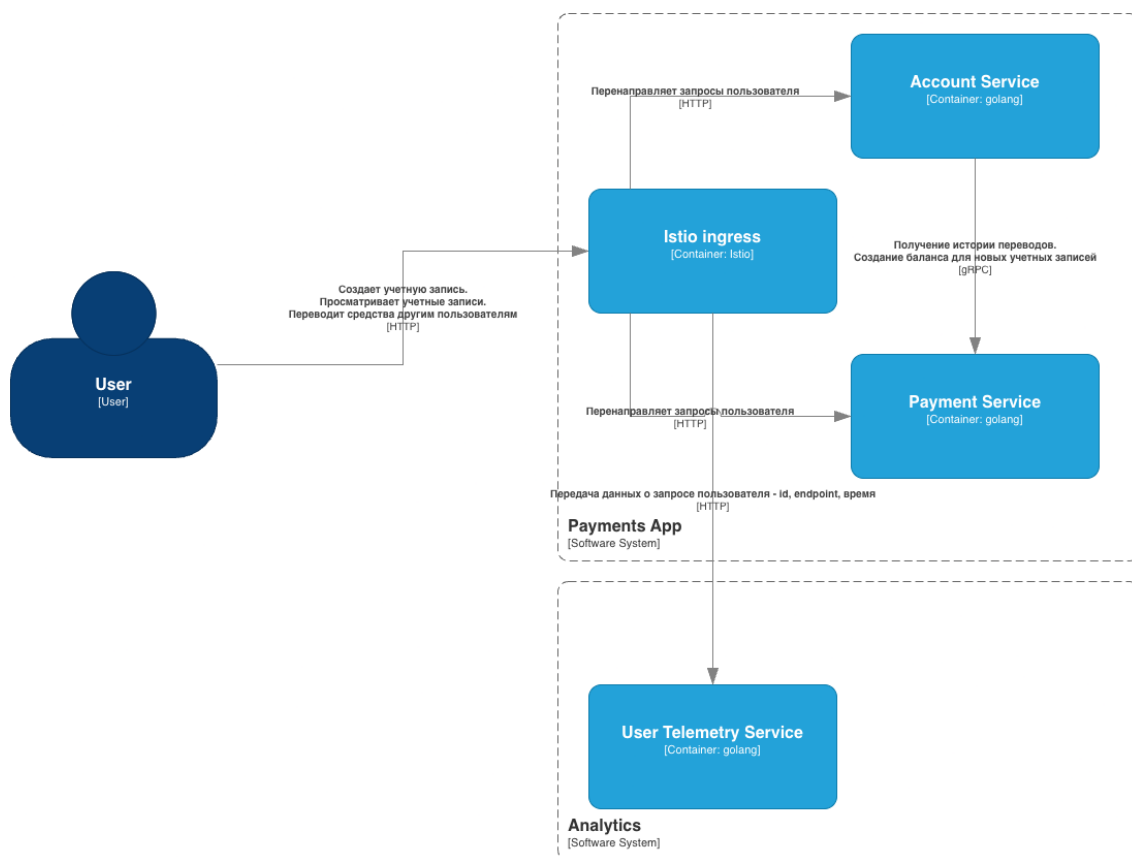


Рисунок 1 – Архитектура демонстрационного приложения

Сервисы были написаны на языке go [4][5], поскольку он имеет лаконичный синтаксис, хорошо подходящий для написания сетевых многопользовательских приложений, а также по причине того, что в результате сборки образуется один исполняемый файл, что упрощает контейнеризацию.

В приведенном приложении сетевые запросы к сервисам обрабатываются Istio, что позволяет собирать информацию об активности пользователей на уровне инфраструктуры без изменения логики микросервисов, а также обеспечивает сбор метрик и трассировку запросов, управление доступом к сервисам и дополнительное шифрование трафика внутри kubernetes при помощи mTLS, организацию канареечных релизов, а также повышенную отказоустойчивость.

ЛИТЕРАТУРА

1. Khatri A., Khatri V. Mastering Service Mesh: Enhance, Secure, and Observe Cloud-Native Applications with Istio, Linkerd, and Consul. – Packt Publishing Limited, 2020. – 626 с.
2. Sharma R., Singh A. Getting Started with Istio Service Mesh: Manage Microservices in Kubernetes. – New York: Apress, 2019. – 321 с.
3. Istio Service Mesh [Электронный ресурс]. – Режим доступа: <https://istio.io> (Дата обращения: 12.03.2026).
4. Никифоров И. В., Андрианов А. А., Александрова О. В. Введение в язык программирования Go = Introduction to the Go programming language: учебник. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – 363 с.
5. Go Programming Language [Электронный ресурс]. – Режим доступа: <https://go.dev> (Дата обращения: 12.03.2026).

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ROI-ОГРАНИЧЕНИЯ НА КАЧЕСТВО ИНТЕРПРЕТАЦИИ
МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ ДЛЯ МРТ ГОЛОВНОГО МОЗГА

В последние годы методы машинного обучения активно применяются для анализа медицинских изображений, включая МРТ головного мозга, однако их внедрение в клиническую практику по-прежнему ограничено недостаточной интерпретируемостью принимаемых решений [1]. Современные обзоры по объяснимому искусственному интеллекту (ХАИ) в медицинской визуализации отмечают, что визуальные объяснения важны не только для повышения доверия специалистов, но и для проверки того, действительно ли модель опирается на клинически значимые области изображения, а не на случайные артефакты или нерелевантные структуры [2].

Среди пост-хок методов визуальной интерпретации для изображений одними из наиболее распространенных являются Grad-CAM и LIME. Grad-CAM строит тепловую карту на основе градиентной информации последних сверточных слоев и позволяет выявлять области изображения, наиболее влияющие на предсказание модели [3]. LIME формирует локальную интерпретируемую аппроксимацию поведения модели и оценивает вклад отдельных суперпиксельных регионов изображения в итоговое решение [4]. Такое различие делает эти методы удобными для совместного рассмотрения: Grad-CAM лучше отражает глобальную пространственную локализацию внимания сети, тогда как LIME позволяет анализировать вклад локальных сегментов изображения [5].

Современные исследования указывают на ограниченную надежность широко используемых saliency-методов в радиологии [6],[7]. Для МРТ-изображений эта проблема усугубляется наличием обширного фона, артефактов и анатомически нерелевантных зон - алгоритм может обращать внимание именно на эти области, и тогда визуальное объяснение будет указывать на клинически незначимые участки, снижая доверие к интерпретации.

С практической точки зрения представляет интерес сравнение стандартных интерпретаций LIME и Grad-CAM с их вариантами, построенными при использовании предварительного выделения области интереса (ROI). В базовом случае интерпретация формируется по всему МРТ-изображению, включая фоновые области, тогда как при ROI-ограничении анализ выполняется только в пределах анатомически значимой области мозга. Для LIME это приводит к изменению пространства суперпикселей, по которым оценивается локальный вклад, а для Grad-CAM – к анализу карты значимости внутри клинически релевантной зоны.

Цель эксперимента – оценить влияние ограничения области анализа на пространственную структуру интерпретаций, получаемых методами ХАИ. Для этого были рассмотрены четыре варианта: стандартный LIME, LIME+ROI, стандартный Grad-CAM и Grad-CAM+ROI.

В работе использовался открытый датасет Brain MRI Segmentation [8], содержащий МРТ-срезы головного мозга с ручными масками. На данных Brain MRI Segmentation был обучен бинарный классификатор на базе ResNet18. Для интерпретации использовались предсказания этой модели, а методы LIME и Grad-CAM применялись к положительным срезам с последующим сравнением полученных карт значимости с ручными масками патологии.

Область интереса формировалась автоматически для каждого изображения на основе самого МРТ-среза. Сначала изображение переводилось в градации серого и нормировалось, затем вычислялся робастный порог по интенсивности, после чего формировалась бинарная маска предполагаемой области мозга.

Полученная маска дополнительно обрабатывалась с помощью морфологических операций закрытия и открытия, удаления малых объектов, заполнения отверстий и выделения

крупнейшей связной компоненты. Если после всех этапов ROI оказывалась пустой, использовалась полная область изображения, что предотвращало потерю среза на этапе последующей интерпретации.

В вариантах без ROI интерпретация строилась по исходному изображению целиком. В вариантах с ROI изображение предварительно маскировалось вне области мозга, и уже по этому ROI-ограниченному входу выполнялось построение LIME и Grad-CAM интерпретаций.

Для количественной оценки результатов использовался коэффициент Dice, измеряющий степень перекрытия двух бинарных масок и показывающий, насколько выделенная моделью область совпадает с истинной областью патологии.

По результатам эксперимента наилучшие средние значения пространственного перекрытия были получены для метода LIME+ROI: средний коэффициент Dice составил 0.1736. Для стандартного LIME этот показатель оказался ниже и составил 0.1248, что указывает на положительное влияние предварительного ограничения области анализа для данного метода.

Для Grad-CAM ситуация оказалась иной: стандартная версия показала более высокое значение Dice, чем Grad-CAM+ROI. Средний коэффициент Dice для Grad-CAM составил 0.1532, тогда как для Grad-CAM+ROI - 0.1227.

Полученные результаты показывают, что влияние ROI зависит от используемого метода интерпретации. Для LIME ограничение области анализа областью мозга способствует улучшению локализации значимых областей, тогда как для Grad-CAM аналогичный подход, напротив, сопровождался снижением значения коэффициента Dice.

ЛИТЕРАТУРА

1. Ahmed F., Naz N. S., Khan S., Rehman A. U., Ismael W. M., Khan M. A. Explainable artificial intelligence (XAI) in medical imaging: a systematic review of techniques, applications, and challenges // BMC Medical Imaging. – 2026. – Vol. 26, No. 1. – P. 37. – DOI: 10.1186/s12880-025-02118-w.
2. Borys K., Schmitt Y. A., Nauta M., Seifert C., Krämer N., Friedrich C. M., Nensa F. Explainable AI in medical imaging: An overview for clinical practitioners – Beyond saliency-based XAI approaches // European Journal of Radiology. – 2023. – Vol. 162. – P. 110786. – URL: <https://doi.org/10.1016/j.ejrad.2023.110786>
3. Selvaraju R. R., Cogswell M., Das A., Vedantam R., Parikh D., Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization // International Journal of Computer Vision. – 2020. – Vol. 128, No. 2. – P. 336–359. – DOI: 10.1007/s11263-019-01228-7.
4. Ribeiro M., Singh S., Guestrin C. Why Should I Trust You? : Explaining the Predictions of Any Classifier // Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. – San Diego, California: Association for Computational Linguistics, 2016. – P. 97–101. – DOI: 10.18653/v1/N16-3020.
5. Chaddad A., Hu Y., Wu Y., Wen B., Kateb R. Generalizable and Explainable Deep Learning for Medical Image Computing: An Overview // Current Opinion in Biomedical Engineering. – 2025. – DOI: 10.1016/j.cobme.2024.100567.
6. Singh Y., Hathaway Q. A., Keishing V., Salehi S., Wei Y., Horvat N., Vera-Garcia D. V., Choudhary A., Mula Kh A., Quaia E., Andersen J. B. Beyond Post hoc Explanations: A Comprehensive Framework for Accountable AI in Medical Imaging Through Transparency, Interpretability, and Explainability // Bioengineering. – 2025. – Vol. 12, No. 8. – P. 879. – URL: <https://doi.org/10.3390/bioengineering12080879>
7. Zhang J., Chao H., Dasegowda G., Wang G., Kalra M. K., Yan P. Revisiting the Trustworthiness of Saliency Methods in Radiology AI // Radiology: Artificial Intelligence. – 2024. – Vol. 6, No. 1. – P. e220221. – URL: <https://doi.org/10.1148/ryai.220221>
8. Brain MRI segmentation // Kaggle. – URL: <https://www.kaggle.com/datasets/mateuszbudalgg-mri-segmentation>

РАЗРАБОТКА СИСТЕМЫ ИНТЕРАКТИВНОЙ 3D-ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ СЕЙСМИЧЕСКОГО МОДЕЛИРОВАНИЯ НА СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМАХ

Современная сейсморазведка, используемая для поиска месторождений углеводородов, оперирует терабайтами данных, получаемых в ходе 3D- и 4D-съемок. Суперкомпьютерные технологии позволяют решать вычислительно сложные обратные задачи, такие как полноволновая инверсия (Full Waveform Inversion — FWI) и обращающая миграция во времени (Reverse Time Migration — RTM), сложность которых достигает 10^{17} – 10^{18} операций с плавающей точкой. Однако анализ получаемых многомерных результатов (волновые поля, кубы атрибутов) остается на уровне статичных 2D-срезов. Существует острый технологический разрыв между мощностью вычислений и инструментами для интерпретации, что не позволяет геофизикам в полной мере оценить пространственную структуру изучаемых объектов [1].

Целью данной работы является разработка программной системы на языке Python для интерактивного анализа и 3D-визуализации результатов сейсмического моделирования, выполняемого на высокопроизводительных системах. Для достижения цели необходимо решить следующие задачи: проанализировать геологическое строение целевого региона (на примере Восточно-Сибирского моря) для формирования требований к модели; спроектировать модульную архитектуру системы; реализовать модуль загрузки и обработки геофизических данных в форматах HDF5 и NumPy; разработать движок 3D-визуализации; адаптировать систему для работы в среде суперкомпьютера.

Основой для геологической модели послужили данные региональных исследований Восточно-Сибирского моря. В разрезе осадочного чехла выделяется 7 сейсмостратиграфических комплексов (ССК) — от мелового до четвертичного, разделенных региональными отражающими горизонтами (ESS1, ESS2, ESS3 и др.) [2]. Ключевым объектом для визуализации являются клиноформные тела в комплексах ССК4 и ССК5, перспективные для поиска литолого-стратиграфических ловушек углеводородов. В качестве вычислительного ядра для тестирования системы используются результаты решения прямой задачи — 2D акустического волнового уравнения методом конечных разностей.

Результаты работы были получены с использованием вычислительных ресурсов суперкомпьютерного центра Санкт-Петербургского политехнического университета Петра Великого (www.spbstu.ru) на узлах «Политехник-РСК Торнадо».

Система построена по модульному принципу. Модуль загрузки данных обеспечивает чтение данных из HDF5-файлов и их преобразование в массивы NumPy. Для работы с объемами, превышающими доступную оперативную память, используется библиотека Dask, реализующая ленивые вычисления. Движок 3D-визуализации реализован на базе библиотеки PyVista [3], предоставляющей высокоуровневый Python-интерфейс к инструментарию VTK. Движок позволяет отображать объемные модели скорости, накладывать волновые поля на геологическую основу и создавать анимацию их распространения. Модуль анализа включает инструменты для построения произвольных 2D-сечений, извлечения одномерных трасс и выделения изоповерхностей. Для интеграции с HPC-инфраструктурой разработаны сценарии пакетной обработки для запуска под управлением SLURM, что позволяет генерировать изображения непосредственно на вычислительных узлах.

Исследования производились на синтетических моделях геологии Восточно-Сибирского моря. Результаты демонстрируют высокую производительность при использовании GPU для объемного рендеринга, что позволяет достичь интерактивной частоты кадров (более 30 FPS) для моделей размером до 500×250 ячеек. Интеграция с Dask позволяет загружать и

обрабатывать наборы данных, в 3–4 раза превышающие объем оперативной памяти тестового узла.

Разработанная система предлагает эффективный подход к преодолению разрыва между суперкомпьютерным моделированием и геологической интерпретацией. Применение открытого стека технологий обеспечивает гибкость и масштабируемость решения. В дальнейшем планируется расширить систему поддержкой промышленного формата SEG-Y и добавить веб-интерфейс для удаленного доступа геологов.

ЛИТЕРАТУРА

1. Virieux J., Operto S. An overview of full-waveform inversion in exploration geophysics // *Geophysics*. – 2009. – Vol. 74. – No. 6. – P. WCC1-WCC26.
2. Кириллова Т.А. Изучение геологического строения и оценка перспектив нефтегазоносности осадочных бассейнов континентальной окраины Восточно-Сибирского моря // *Отчет ОАО «МАГЭ»*. – Мурманск, 2013.
3. Sullivan C., Kaszynski A. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit // *Journal of Open Source Software*. – 2019. – Vol. 4. – No. 37. – P. 1450.

УДК 004.4

С.Р. Литвин (4 курс бакалавриата),
Б.М. Медведев, к.т.н., доцент

МИКРОСЕРВИС ДЛЯ АВТОМАТИЧЕСКОЙ МОДЕРАЦИИ ПОЛЬЗОВАТЕЛЬСКОГО КОНТЕНТА НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

В современных условиях экспоненциального роста объёмов пользовательского контента ключевым вызовом для платформ, основанных на взаимодействии между людьми, становится обеспечение безопасности информационной среды. Актуальность работы обусловлена необходимостью комплексной защиты пользователей от нежелательного и опасного контента. Это подразумевает ограждение несовершеннолетних от информации категории 18+ в соответствии с Федеральным законом № 436-ФЗ и фильтрацию особо опасных материалов (например, террористической направленности) для предотвращения их демонстрации любым пользователям вне зависимости от возраста.

Современные исследования подтверждают эффективность использования методов глубокого обучения для автоматической модерации. В исследовании [1] предложен гибридный подход с использованием рекуррентных нейронных сетей для классификации текста по категориям оскорбительного и нежелательного контента. Для анализа визуального контента в исследовании [2] продемонстрирована эффективность комбинации лёгких свёрточных нейронных сетей (EfficientNet-B0, MobileNetV2) с LSTM для обнаружения сцен насилия. Работа [3] показывает, что fine-tuned ResNet50_V2 в паре с SVM достигает точности 92,80% для задач цензурирования контента. Важность контекстно-зависимых решений и мультимодального подхода подчёркивается в исследовании [4], где авторы предлагают комплексную систему оценки пользовательского контента.

Целью данной работы является разработка системы автоматической модерации контента, обеспечивающей выявление материалов категории 18+ в загружаемых пользователями изображениях и текстах с последующим применением цензуры в зависимости от возраста просматривающего пользователя и его настроек.

В рамках работы решаются следующие задачи: анализ архитектуры существующего приложения и формулирование требований к интеграции, выбор моделей машинного обучения для классификации текстового и визуального контента, проектирование архитектуры сервиса модерации с учётом требований асинхронности и масштабируемости, разработку программных средств модерации, включающих модуль классификации изображений на основе свёрточной нейронной сети, модуль анализа текста с использованием трансформерной

модели, модуль асинхронной обработки задач на базе брокера сообщений и модуль для работы с базой данных.

К разрабатываемым программным средствам предъявляются следующие основные требования: органичная интеграция в существующую микросервисную архитектуру на Java/Spring Boot, автоматическое выявление контента категорий 18+ в изображениях и тексте с использованием моделей машинного обучения, асинхронная обработка пользовательского контента без замедления работы клиентских приложений, гибкое применение механизмов цензуры в зависимости от категории нарушения, учёт возраста пользователя и его персональных настроек при отображении контента, а также предоставление администратору интерфейса для обработки апелляций.

Архитектура разрабатываемой системы, показанная на рисунке 1, строится на принципах органичного встраивания в существующую микросервисную экосистему приложения (Java, Spring Boot, Maven, PostgreSQL, MinIO). Для анализа изображений выбраны свёрточные нейронные сети EfficientNet-B0 и ResNet50_V2, для анализа текста – трансформерная модель BERT. Интеграция моделей в Java-сервис осуществляется через библиотеку Deep Java Library, позволяющую исполнять модели PyTorch и TensorFlow непосредственно в Java Virtual Machine, что обеспечивает единую среду разработки и сокращает задержки на межсервисное взаимодействие. Обработка изображений для цензурирования (размытие, пикселизация) выполняется с помощью OpenCV. Для обеспечения асинхронности используется брокер сообщений RabbitMQ. Данные о результатах классификации хранятся в базе данных PostgreSQL.

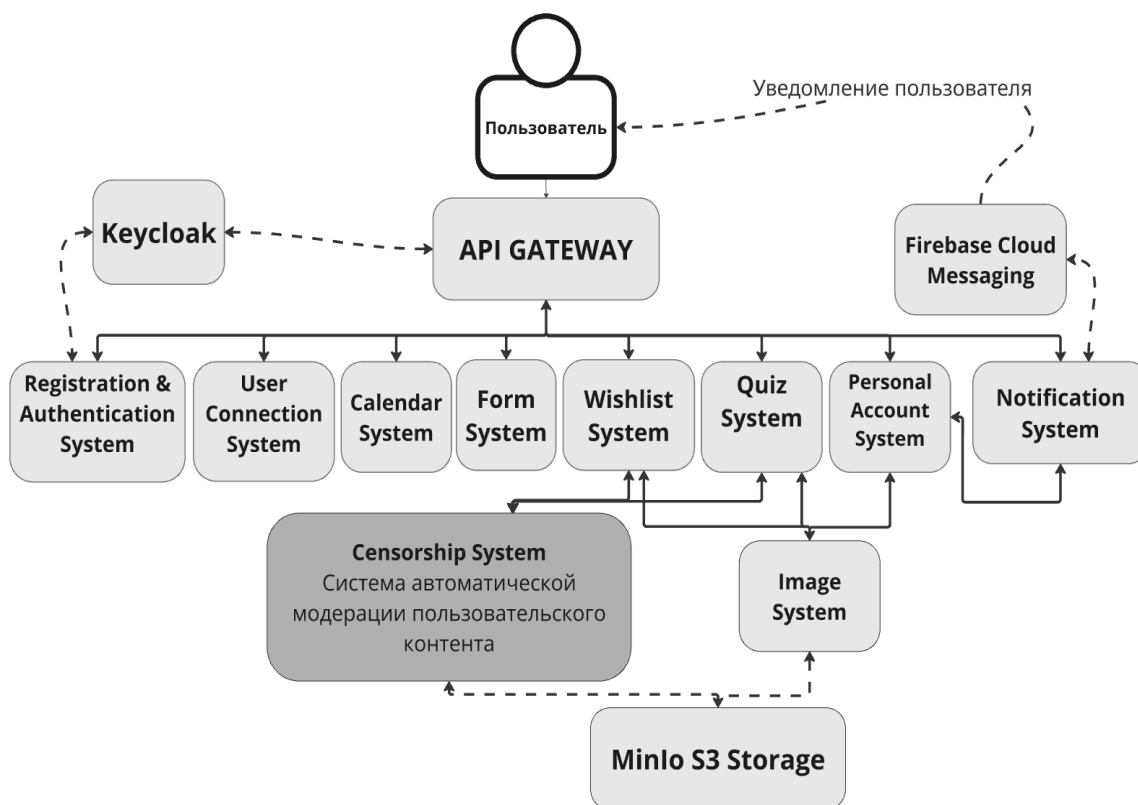


Рисунок 1 – Архитектура разрабатываемой системы

Проверка эффективности разработанного решения основана на оценке точности классификации моделей на тестовой выборке и результатах нагрузочного тестирования для определения максимальной пропускной способности системы и времени обработки запросов.

Разработанное программное обеспечение будет использовано в составе социального приложения для автоматизации процессов модерации пользовательского контента, что позволит сократить операционные расходы на ручную модерацию, обеспечить соответствие

требованиям законодательства и повысить безопасность цифровой среды для несовершеннолетних пользователей.

ЛИТЕРАТУРА

1. V. Shirsath, T. Wani, D. Bhakare, P. Lokhande, R. Chavda and V. Shah, "ChatGuard: A Profanity Classification Approach for Safer Online Conversations," 2023 1st DMIHER International Conference on Artificial Intelligence in Education and Industry 4.0 (IDICAIEI), Wardha, India, 2023, pp. 1-5, doi: 10.1109/IDICAIEI58380.2023.10406633. keywords: {Education;Oral communication;Machine learning;Censorship;Fourth Industrial Revolution;Monitoring;machine learning;profanity detection;online community;logistic regression;obscene and offensive words}.
2. Abdullah, M. S. N. bin, Karim, H. A., & AlDahoul, N. (2023). A Combination of Light Pre-trained Convolutional Neural Networks and Long Short-Term Memory for Real-Time Violence Detection in Videos. *International Journal of Technology*, 14(6), 1228–1236. <https://doi.org/10.14716/ijtech.v14i6.6655>
3. Sui Lyn, H., Mansor, S., AlDahoul, N., & Abdul Karim, H. (2020). Convolutional Neural Network-based Transfer Learning and Classification of Visual Contents for Film Censorship. *Journal of Engineering Technology and Applied Physics*, 2(2), 28–35. <https://doi.org/10.33093/jetap.2020.2.2.5>
4. Que, X., & Pan, Y. (2022). A Context-Aware Intelligent System to Assist User Profile Filtering using AI and Deep Learning (pp. 65–75). Academy and Industry Research Collaboration Center (AIRCC). <https://doi.org/10.5121/csit.2022.120105>

УДК 004.773

Р.Ю. Лысков (1 курс магистратуры),
В.Э. Шмаков, к.т.н, доцент

РАЗРАБОТКА МЕССЕНДЖЕРА С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ ДЛЯ МАЛОГО БИЗНЕСА

Современное общество невозможно представить без интернета. Цифровые технологии обеспечивают коммуникацию, хранение данных, финансовые операции и управление бизнес-процессами.

Однако, глобализация и стремительное развитие компьютерных технологий привело к возникновению нового вида правонарушений — киберпреступлений. В связи с этим, актуальной стала проблема обеспечения информационной безопасности. Анализ графика на рисунке 1 [1] показывает, как изменилось количество киберпреступлений, связанных с различными статьями уголовного кодекса (всего их 36) Российской Федерации.

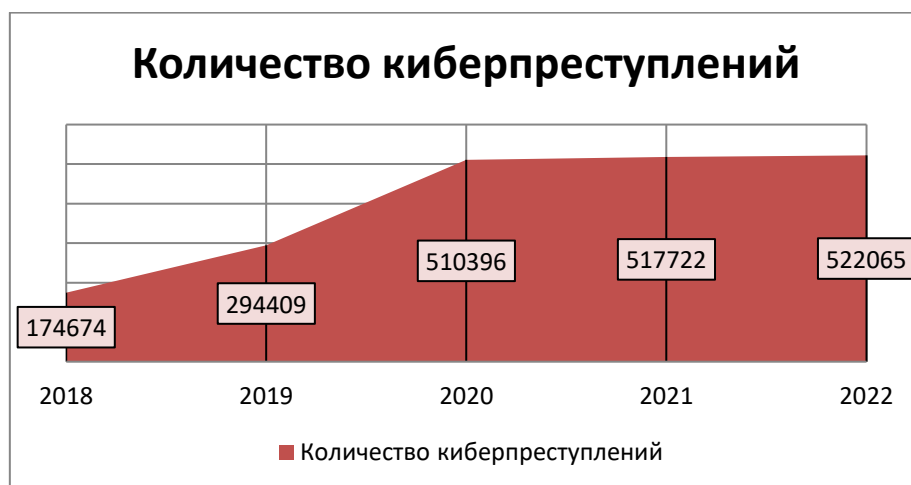


Рисунок 1 – Динамика нарушения различных статей УК РФ, связанных с киберпреступлениями

Число жертв разнообразных киберпреступлений растет с каждым годом и не собирается уменьшаться. Кроме физических лиц и крупных предприятий, жертвами становятся малый и средний бизнес.

Последние две группы особенно уязвимы, так как могут обладать рядом особенностей: отсутствие штатных ИБ-специалистов, активное использование публичных средств связи и облачных сервисов и т.д. Эти факторы сильно осложняют обеспечение специальных мер по противодействию угрозам информационной безопасности: фишинг, утечка данных, вредоносное ПО. [2, 3]

Кибератаки приводят к финансовым и репутационным потерям предприятия, вплоть до его закрытия.

Для защиты данных от злоумышленников крупные предприятия используют множество различных мер и средств. Одним из таких средств являются корпоративные мессенджеры, выступающие не просто средством общения, но и элементом системы информационной безопасности.

В таблице 1 [4-7] приведено сравнение существующих решений на рынке корпоративных мессенджеров.

Таблица 1 – Сравнение существующих корпоративных мессенджеров

Критерии	Slack	Microsoft Teams	Rocket.Chat	Mattermost
Доступность	Платный (оплата из РФ затруднена)	Платный (оплата из РФ затруднена)	Открытый исходный код	Открытый исходный код
Архитектура	Монолитная	Монолитная	Микросервисная	Микросервисная
Удобство развертывания	Зависит от инфраструктуры	Сложность развертывания в крупных компаниях	Требует навыков администрирования	Требует навыков администрирования
Возможность создания настраиваемого клиента	Ограниченные возможности	Возможности настройки ограничены инфраструктурой	Поддержка есть, но требует знаний разработки	Возможность создания через API и настройки
Возможности по обеспечению безопасности	Шифрование данных, MFA.	Шифрование, MFA, DLP.	LDAP/AD, 2FA, сквозное шифрование	AD/LDAP, шифрование, MFA

Анализ показывает, что для небольших компаний приведенные решения имеют ограничения, особенно с точки зрения доступности, стоимости, гибкости настройки и удобства внедрения. Slack и Microsoft Teams являются платными решениями. Кроме того, для российских компаний возможны сложности с оплатой и лицензированием, что ещё сильнее ограничивает их доступность. Rocket.Chat и Mattermost имеют открытый исходный код, однако требуют самостоятельного развертывания и администрирования и, как следствие, углубленных знаний в этих областях.

В качестве подхода к решению проблемы предлагается разработка корпоративного мессенджера с следующими особенностями:

- микросервисная архитектура;
- независимость от реализации клиентской части приложения средствами API;
- наличие графического интерфейса, облегчающего администрирование.

Данный подход позволит малому бизнесу обеспечить простую и гибкую настройку приложения и защиту данных без больших финансовых или временных затрат.

ЛИТЕРАТУРА

1. Киберпреступность в России: состояние и методология анализа / Т. М. Бормотова, А. Е. Ильин, Е. Б. Белянский, В. В. Макаров // Этносоциум и межнациональная культура. – 2023. – № 10(184). – С. 60-73. – EDN KERXTJ.

2. Малый бизнес: пять причин опасаться киберугроз. [Электронный ресурс] Режим доступа: <https://www.kaspersky.ru/blog/smb-five-reasons-to-protect-yourself/33386/> (дата обращения: 25.02.2026).
3. Тонких, А. С. Угрозы безопасности в облачных технологиях и методы их устранения / А. С. Тонких, Е. Ю. Авксентьева // Международный журнал гуманитарных и естественных наук. – 2024. – № 1-2(88). – С. 232-238. – DOI 10.24412/2500-1000-2024-1-2-232-238. – EDN CZLBFQ.
4. Slack [Электронный ресурс] Режим доступа: <https://slack.com/> (дата обращения: 25.02.2026).
5. Microsoft Teams [Электронный ресурс] Режим доступа: <https://www.microsoft.com/ru-ru/microsoft-teams/> (дата обращения: 25.02.2026).
6. Rocket.Chat [Электронный ресурс] Режим доступа: <https://www.rocket.chat/> (дата обращения: 25.02.2026).
7. Mattermost [Электронный ресурс] Режим доступа: <https://mattermost.com/> (дата обращения: 25.02.2026).

УДК 004.415.2

А.В. Месропян (4 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОНОМНОГО АУДИОРЕГИСТРАТОРА С ФУНКЦИЕЙ ПОДАВЛЕНИЯ ШУМОВ

В совершенно различных сферах деятельности в том или ином виде возникает необходимость взаимодействовать с клиентами. У банков, крупных и сравнительно небольших коммерческих компаний, а также в сфере обслуживания трудоустроено огромное количество людей, задачей которых является рекомендовать тот или иной продукт клиенту, помочь ему выбрать и приобрести подходящий продукт или услугу или решить какую-либо проблему с существующим продуктом. Оценка деятельности и качества работы сотрудников также является неотъемлемой частью работы многих компаний, необходимо оценивать кто из сотрудников, например, заслуживает поощрения или допускает грубые ошибки. Это позволяет влиять на эффективность работы компании в целом. На данный момент очень часто телефонные разговоры с банками или компаниями записываются именно для этих целей. Однако, в случае с личным взаимодействием, например, в офисе, собирать подобные данные оказывается более затруднительно.

Одним из способов решения данной проблемы является установка аудиорегистраторов – устройств для записи и хранения аудиоданных, однако несмотря на большой выбор устройств[2], существующие программные решения имеют большое количество недостатков, таких как низкое качество или скорость записи, недостаточный уровень программного шумоподавления или невозможность управления работой записывающих устройств удаленно.

Таким образом, целью данной работы является написание программного обеспечения для аудиорегистратора на базе одноплатного компьютера orange pi, поддерживающего двухпоточную запись аудио в двух и моноканальном режиме, который в дальнейшем сможет использоваться для прослушивания и обработки аудиозаписей, связанных с работой с клиентами. Также, данное устройство должно поддерживать работу в автономном режиме, постоянную запись, запись по команде, передаваемой удаленно, а также запись по расписанию. В качестве аппаратного обеспечения будет использоваться одноплатный компьютер orange pi [3]

Для достижения этой цели необходимо выполнить следующие задачи:

1. Поиск и выбор инструментов для записи аудиопотоков.
2. Выбор инструментов для шумоподавления.
3. Создание программного обеспечения для записи аудиопотоков с настройкой параметров записи, включая опцию удаления шумов.

4. Создание приложения для подавления шумов.
5. Создание серверного приложения для удаленного запуска аудиозаписи, изменения настроек и применения шумоподавления.

В качестве инструмента для записи аудио была выбрана библиотека для языка c++ portaudio[1]. Она поддерживает настройку частоты дискретизации, стерео и однопоточную запись, крайне быстрая библиотека для работы со звуком. Для удаления шумов из полученного аудио использовалась библиотека rnnoise[4], использующая рекуррентную нейросеть для обнаружения шумов в аудиофайле и удаляющая данные шумы. С их помощью было написано приложение на языке c++ включающее и выключающее запись аудиопотока в файл с параметрами, взятыми из файла конфигурации, а также применяющая шумоподавление при сохранении файлов в случае необходимости. В случае если шумоподавление применяется, записанное аудио также предварительно преобразуется в необходимый для работы rnnoise формат. В конечном итоге, на python был реализован rest-API сервер для включения и выключения записи, а также изменения значений конфигурации, включая настройку шумоподавления. Сервер поддерживает как единичное включение или выключение записи, так и настройку записи по расписанию. Примерная схема итоговой архитектуры приведена на рисунке 1.

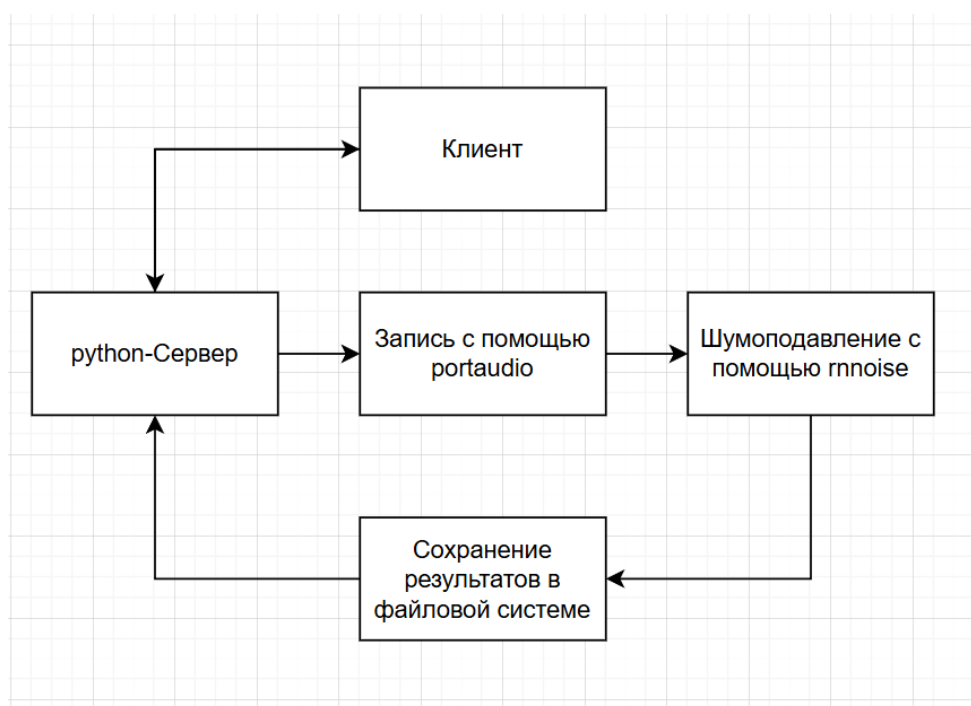


Рисунок 1 – Архитектура системы

В результате работы было реализовано программное обеспечение для аудиорегистратора на базе платформы orange pi, поддерживающее запись аудио в двух и моноканальном режиме, с возможностью удаления шумов, работа с которым осуществляется через python-сервер.

ЛИТЕРАТУРА

1. Official PortAudio Documentation Portal [Электронный ресурс] – URL: <https://www.portaudio.com/docs.html> - (дата обращения: 21.01.26)
2. STELBERRY M-1000 [Электронный ресурс] – URL: <https://stelberry.ru/products/M1000.html> - (дата обращения: 21.01.26)
3. Официальный ресурс Orange Pi [Электронный ресурс] – URL: <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-One.html> - (дата обращения - 21.01.26)
4. An interactive demo of xiph/rnnoise [Электронный ресурс] – URL: <https://jmvalin.ca/demo/rnnoise/> - (дата обращения – 21.01.26)

ИНТЕРПРЕТИРУЕМАЯ КЛАССИФИКАЦИЯ ДИАБЕТИЧЕСКОЙ РЕТИНОПАТИИ

Растущая заболеваемость сахарным диабетом и его осложнениями, в частности диабетической ретинопатии (ДР) - ведущей причины предотвратимой слепоты, стимулирует внедрение систем глубокого обучения для автоматической классификации [1, 2]. Несмотря на то, что точность современных алгоритмов сопоставима с экспертами-офтальмологами [3], их клиническое применение сдерживается проблемой “черного ящика” и недоверием врачей [1, 4]. Существующие методы объяснимого ИИ, в основном, дают семантически неопределенные карты внимания, поэтому перспективным направлением считается интеграция ХАИ с методами семантической сегментации [5].

Однако современные подходы к совместному обучению для сегментации и классификации [3] имеют ограничения: их объяснения часто слабо связаны с конкретными патологиями [6], а использование результатов сегментации как входных данных для классификатора может приводить к каскадным ошибкам из-за пропуска мелких структур. Поэтому создание прозрачных систем поддержки принятия решений требует новых архитектур, обеспечивающих семантически ясную и клинически релевантную интерпретируемость.

Таким образом, целью данной работы является построение интерпретируемой системы для классификации степени диабетической ретинопатии за счёт применения механизмов внимания и методов объяснимого ИИ.

Для достижения поставленной цели предложен последовательный каскадный конвейер обработки (рисунок 1). На первом этапе которого происходит сегментация сосудистой системы, результаты которой применяются в качестве анатомического контекста на следующем этапе - семантической сегментации патологий.

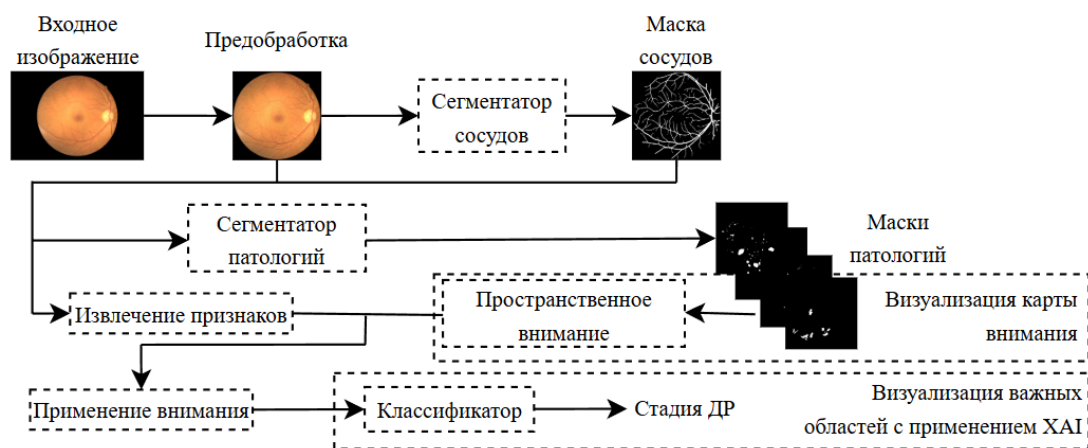


Рисунок 1 – Концептуальная схема

Одним из основных компонентов разрабатываемого подхода является то, что маски используются не как дополнительные каналы входа, а интегрируются в механизм внимания для модуляции извлеченных признаков.

Ключевая идея заключается в двухэтапной обработке информации от масок: сначала маски проходят через механизм канального внимания, который учится перевзвешивать важность каждой отдельной маски (например, кровоизлияний, микроаневризм и т.д.), затем применяется пространственное внимание - взвешенные маски объединяются с полученными проекциями признаков. После чего комбинация подаётся в небольшую свёрточную сеть, которая генерирует карту пространственного внимания. Эта карта поэлементно умножается на исходные признаки, выделяя пространственные области, значимые для классификации.

Таким образом, архитектура стремится использовать информацию из масок как для выделения релевантных каналов, так и для пространственной фокусировки.

На завершающем этапе модифицированные карты признаков подаются в классификатор, определяющий стадию диабетической ретинопатии. При этом интеграция методов объяснимого ИИ (ХАИ) [4], а также визуализация полученных масок сегментации и генерируемой механизмом внимания карты, наглядно демонстрируют, на какие пространственные области изображения и соответствующие патологии опиралась модель при принятии решения (рисунок 2).

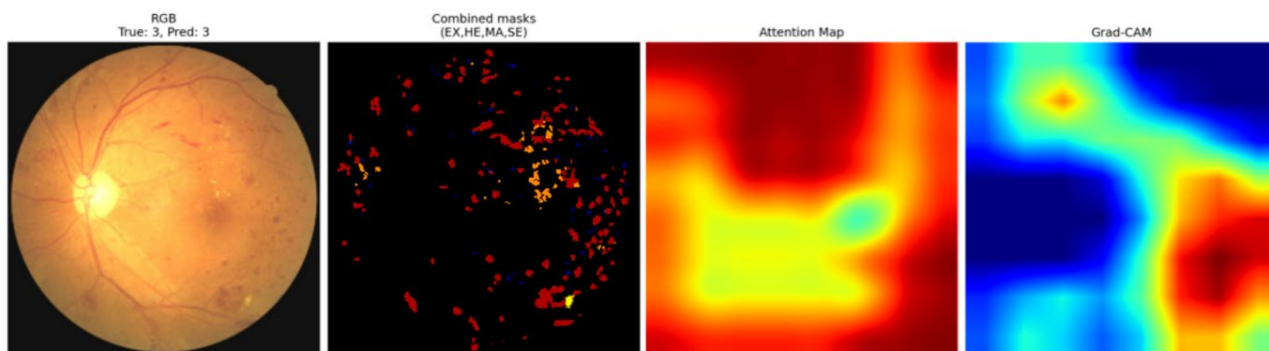


Рисунок 2 – Результаты

Рассматриваемый подход хотя и зависит от качества входных масок, как и при их конкатенации, однако показывает более устойчивые результаты, т.к. маски в данном случае выполняют направляющую функцию, позволяя модели избирательно усиливать информативные области признаков и подавлять шумовые или ошибочные участки.

Сложность и вычислительные затраты рассматриваемой системы обусловлены необходимостью в интерпретации предсказаний и своевременном выявлении ошибок, что особенно важно в сфере медицины [1].

Был разработан каскадный подход для классификации диабетической ретинопатии, основанный на использовании семантических масок сегментации для управления механизмом мягкого пространственного внимания. Предложенная архитектура обеспечивает клинически интерпретируемые объяснения решений модели.

ЛИТЕРАТУРА

1. Yang Z. et al. Classification of diabetic retinopathy: Past, present and future //Frontiers in endocrinology. – 2022. – Т. 13. – С. 1079217.
2. Guo X., Malykhina G. F. TransCNN: Fusion of Transformer and CNN for Detection of Lumbar Degenerative Spine Lesions //2025 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). – IEEE, 2025. – С. 783-788.
3. Haq N. U. et al. Computationally efficient deep learning models for diabetic retinopathy detection: a systematic literature review //Artificial Intelligence Review. – 2024. – Т. 57. – №. 11. – С. 309.
4. Макарова А. А., Малихина Г. Ф. Объяснимый искусственный интеллект в системах поддержки принятия решений, Всероссийская научно-практическая конференция с международным участием «ПРОМЫШЛЕННЫЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ» : сборник научных трудов, 2025 г., Санкт-Петербург, / Под общ. ред. Д. Г. Арсеньева; СПбПУ. с. 187-200.
5. Urooj M. et al. Single domain generalization in diabetic retinopathy: A neuro-symbolic learning approach //arXiv preprint arXiv:2509.02918. – 2025.
6. Said Z., Ben-Bouazza F. E., Mekour M. Towards Interpretable Diabetic Retinopathy Detection: Combining Multi-CNN Models with Grad-CAM //International Journal of Advanced Computer Science & Applications. – 2024. – Т. 15. – №. 10.

ПРОГРАММНЫЙ КОМПЛЕКС УПРАВЛЕНИЯ РОБОТИЗИРОВАННОЙ ЛИНИЕЙ
INDUSTRY 4.0 НА ПЛАТФОРМЕ 1С

Современная промышленность находится на этапе перехода к Четвёртой промышленной революции (Индустрия 4.0), характеризующейся массовым внедрением информационных технологий, автоматизацией производственных процессов и интеграцией искусственного интеллекта [4]. Однако подготовка квалифицированных кадров для работы в новых условиях сталкивается с серьёзными проблемами. Высшие учебные заведения часто не располагают эффективными и доступными ERP-тренажёрами производства. Решения зарубежных компаний, таких как Siemens, Festo и SAP, оказываются слишком дорогими и сложными для широкого внедрения в образовательный процесс.

Особую актуальность проблема приобретает в условиях импортозамещения. Уход компании SAP с российского рынка привёл к тому, что программное обеспечение, ранее использовавшееся для обучения, стало недоступным для университетов из-за санкций и высокой стоимости подписки (до 8000 евро в год). Это вызвало утрату актуальности учебных стендов, ориентированных на западное программное обеспечение, и создало потребность в отечественных аналогах.

Целью данной работы является разработка и создание программного комплекса для управления роботизированным макетом Fischertechnik Industry 4.0 на базе отечественной платформы «1С:Предприятие 8.5», способного заменить ушедшее с рынка решение SAP.

Для достижения цели были решены следующие задачи:

- проектирование многоуровневой архитектуры взаимодействия между платформой 1С, промежуточным API-сервером и физическими компонентами макета;
- разработка конфигурации в среде 1С, включающей документы для управления производственными циклами и регистры для мониторинга состояния оборудования;
- создание программного слоя на языке Python для управления контроллерами оборудования;
- апробация разработанного решения на публичных мероприятиях для оценки образовательного и демонстрационного потенциала.

Архитектура разработанной системы построена по трёхуровневому принципу, обеспечивающему гибкое взаимодействие между бизнес-логикой и физическим оборудованием.

Верхний уровень реализован в виде конфигурации на платформе «1С:Предприятие 8.5». В нём разработаны документы-оркестраторы, такие как «Производственный заказ» и «Заказ на сортировку», проводка которых инициирует работу физической линии. Для хранения и отображения актуального состояния склада и загруженности компонентов в реальном времени используются регистры сведений и дашборд на начальной странице конфигурации.

Промежуточный уровень представлен API-сервером, разработанным на языке Python с использованием фреймворка FastAPI. Сервер принимает HTTP/JSON-запросы от платформы 1С, валидирует их и транслирует команды нижнему уровню управления. Данный подход демонстрирует современные возможности платформы 1С по интеграции с IoT-устройствами через стандартные веб-протоколы.

Нижний уровень включает контроллеры оборудования. Каждый из пяти компонентов линии (кран, склад, покрасочная станция, конвейер, сортировщик) управляется индивидуальным контроллером с собственным программным кодом на Python. Для координации работы всех контроллеров реализована центральная управляющая программа, содержащая класс-обертку Factory. Этот класс агрегирует все устройства и обеспечивает синхронизацию их работы согласно производственному циклу. Весь сервер

контейнеризирован с помощью docker-compose для упрощения развёртывания и масштабирования.

Схема взаимодействия компонентов системы:

1С:Предприятие – HTTP/JSON – Python API Server – Factory Class – Контроллеры устройств – Fischertechnik

Апробация разработанного комплекса была проведена на двух крупных мероприятиях. Впервые стенд был продемонстрирован на финале всероссийского чемпионата «Профессионалы 2025» в Санкт-Петербурге, где студенты СПбПУ проводили мастер-классы по доработке конфигурации для управления макетом [1]. Участники, включая школьников и педагогов, решали задачи по настройке интеграции в Конфигураторе 1С, знакомясь с практиками командной разработки. Второй раз стенд был показан на XXVI Международной конференции «Новые информационные технологии в образовании» в 2026 году, где с проектом ознакомились руководство фирмы «1С».

Практическая значимость разработанного комплекса заключается в решении проблемы нехватки доступных инструментов для обучения, отмеченной в исследованиях [2, 3]. Стенд демонстрирует, что отечественная платформа 1С, широко используемая для автоматизации таких предприятий, как Ростех, Газпром нефть и РусГидро, способна управлять сложными роботизированными системами в концепции Индустрии 4.0. В отличие от дорогих и сложных зарубежных аналогов, данное решение доступно для вузов, мобильно и может служить основой для образовательных курсов по IoT и промышленной разработке на платформе 1С.

В ходе работы была достигнута основная цель: создано работоспособное программное обеспечение, замещающее функционал ушедшей SAP ERP в рамках учебного стенда. Комплекс подтвердил свою эффективность как инструмент для демонстрации интеграционных возможностей 1С и обучения студентов принципам работы цифрового производства.

Перспективы развития проекта включают замену аппаратной части Fischertechnik на отечественного производителя, например компанию Smarta, более глубокую интеграцию с аналитикой на платформе версии 8.5.4, а также интеграцию с полноценной конфигурацией 1С:ERP. Это позволит создать полностью масштабируемое и функциональное учебное решение для массовой подготовки инженерных кадров цифровой экономики.

ЛИТЕРАТУРА

1. Сотрудники Передовой инженерной школы СПбПУ «Цифровой инжиниринг» приняли активное участие во всероссийском чемпионате профессионального мастерства «Профессионалы 2025» // Передовые инженерные школы. – 2025. – URL: <https://pish.spbstu.ru/news/9299> (дата обращения: 16.03.2026).
2. Реализация концепции «Университет 4.0» в Сибирском федеральном округе // КиберЛенинка. – 2023. – URL: <https://cyberleninka.ru/article/n/realizatsiya-kontseptsii-universitet-4-0-v-sibirskom-federalnom-okruge> (дата обращения: 16.03.2026).
3. Problems and Prospects of Digitalization of Education for Industry 4.0 in Russian Federation // ResearchGate. – 2024. – URL: https://www.researchgate.net/publication/377557985_Problems_and_Prospects_of_Digitalization_of_Education_for_Industry_40_in_Russian_Federation (дата обращения: 16.03.2026).
4. Что такое индустрия 4.0 и что нужно о ней знать // РБК. – 2020. – URL: <https://trends.rbc.ru/trends/industry/5e740c5b9a79470c22dd13e7> (дата обращения: 16.03.2026).
5. Интеграция IoT с 1С: реальные кейсы и потенциальные риски // IT-Aurora. – 2024. – URL: <https://it-aurora.ru/news/integratsiya-iot-s-1s-realnye-keysy-i-potentsialnye-risk.html> (дата обращения: 16.03.2026).
6. Рынок ERP в России в 2024 г. может превысить 100 млрд рублей // CNews. – 2023. – URL: https://www.cnews.ru/news/line/2023-12-29_rynok_erp_v_rossii_v_2024_gmozhet (дата обращения: 16.03.2026).

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ ПРОГНОЗИРОВАНИЯ ГИПОГЛИКЕМИИ НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

Гипогликемия представляет собой критическое снижение уровня глюкозы в крови, сопровождающееся нарушением когнитивных функций, слабостью, вегетативными реакциями и, в тяжелых случаях, развитием комы [5]. В условиях непрерывного мониторинга глюкозы (Continuous Glucose Monitoring, CGM) становится возможным не только фиксировать текущее значение показателя, но и анализировать динамику изменения гликемии во времени, что создает предпосылки для краткосрочного прогнозирования риска гипогликемии [4,5].

Целью данной работы является разработка мобильного приложения промышленного уровня, обеспечивающего визуализацию данных CGM, интеграцию готовой модели машинного обучения для прогнозирования гипогликемии и формирование пользовательских уведомлений в реальном времени. Работа относится к секции С1 «Прикладная программная инженерия: продукты, сервисы и системы» и ориентирована на демонстрацию архитектурного обоснования и технологической реализации программного продукта.

Архитектура приложения построена с использованием фреймворка Flutter, обеспечивающего кроссплатформенную разработку для Android и iOS на единой кодовой базе [1]. В качестве языка реализации используется Dart. Выбор Flutter обусловлен высокой производительностью за счет компиляции в нативный код, наличием декларативной модели UI и развитой экосистемой пакетов для работы с потоками данных и визуализацией графиков.

С точки зрения программной архитектуры приложение реализовано по принципам Clean Architecture с разделением на три уровня:

- presentation layer (UI и управление состоянием);
- domain layer (бизнес-логика, сценарии использования, интерфейсы репозитория);
- data layer (работа с внешними источниками данных и моделью прогнозирования).

Для управления состоянием используется реактивный подход (например, BLoC или Riverpod), что обеспечивает предсказуемость потоков данных и тестируемость компонентов.

Получение данных от датчика CGM (например, FreeStyle Libre) осуществляется через промежуточный серверный слой. Датчик передает измерения в облачную платформу производителя, после чего backend-приложение (REST API) выполняет:

1. Аутентификацию пользователя.
2. Периодический опрос или прием webhook-событий с новыми измерениями.
3. Нормализацию и валидацию данных.
4. Передачу агрегированных данных мобильному приложению по HTTPS.

Мобильное приложение взаимодействует с сервером по REST API и получает данные в формате JSON. Для минимизации задержек реализован механизм кэширования и локального хранения последних измерений с использованием SQLite.

Интеграция готовой ML-модели реализуется двумя возможными способами. В прототипе выбран вариант серверной инференции: обученная модель (например, LSTM или градиентный бустинг), развернутая в виде микросервиса на Python с использованием TensorFlow или PyTorch, принимает на вход временное окно последних N измерений и возвращает прогноз уровня глюкозы через заданный горизонт времени (например, 60 минут) [4].

Архитектурно модель инкапсулируется в отдельном сервисе прогнозирования:

- вход: последовательность измерений и временные метки;
- обработка: формирование признаков, нормализация, запуск инференса;
- выход: прогнозируемое значение и вероятность гипогликемии.

Backend-уровень агрегирует данные CGM и результаты предсказания, после чего мобильное приложение отображает:

- текущее значение глюкозы;
- график динамики (временной ряд);
- прогнозируемую кривую;
- индикатор риска гипогликемии.

Визуализация реализована с использованием специализированных Flutter-пакетов для построения интерактивных графиков. Обновление данных происходит каждые 5 минут, что соответствует частоте измерений CGM.

Дополнительно реализована система push-уведомлений: при превышении порога вероятности гипогликемии сервер инициирует отправку уведомления через Firebase Cloud Messaging [6]. Таким образом достигается проактивная поддержка пользователя.

С точки зрения нефункциональных требований архитектура учитывает:

- масштабируемость (микросервисная модель backend);
- отказоустойчивость (логирование и обработка ошибок API);
- безопасность (HTTPS, токен-аутентификация, разграничение доступа);
- расширяемость (возможность замены ML-модели без изменения клиентской части).

Демонстрационная версия приложения включает пользовательский интерфейс с главным экраном мониторинга, экраном истории измерений и экраном настроек оповещений. Среднее время отклика при получении прогноза не превышает 300 мс при серверной инференции.

Таким образом, в работе представлена архитектура мобильной системы мониторинга гликемии, интегрирующая промышленный ML-компонент и обеспечивающая полный цикл обработки данных: от получения измерений с датчика до отображения прогноза и генерации предупреждений. Предложенное решение демонстрирует практическое применение современных технологий мобильной разработки и инженерных принципов построения масштабируемых программных систем.

ЛИТЕРАТУРА

1. Flutter Documentation. [Электронный ресурс]. Режим доступа: <https://flutter.dev/> (дата обращения: 27.02.2026). Франсуа Шолле. Глубокое обучение на Python. – 2025. – С. 203-225.
2. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. [Электронный ресурс]. Режим доступа: <https://www.tensorflow.org/> (дата обращения: 27.02.2026).
3. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // Proceedings of CVPR. – 2016.
4. Li K. et al. Blood Glucose Prediction with LSTM Networks // IEEE Access. – 2020.
5. ElSayed N. et al. Recent Advances in Hypoglycemia Prediction Using CGM Data // Sensors. – 2023.
6. Firebase Cloud Messaging Documentation. [Электронный ресурс]. Режим доступа: <https://firebase.google.com/docs/cloud-messaging> (дата обращения: 27.02.2026).

УДК 004.5

А.Р. Муратов (4 курс бакалавриата),
И.А. Шемякин, ст. преподаватель

РАЗРАБОТКА КАЧЕСТВЕННОГО КЛИЕНТСКОГО ИНТЕРФЕЙСА ДЛЯ СОЦИАЛЬНОЙ СЕТИ

В современном мире мгновенный обмен информацией прочно вошел в повседневную жизнь людей. И хотя на рынке социальных сетей есть свои явные лидеры, аудитория продолжает расти, и одновременно с этим формируется запрос на появление альтернативных площадок.

Чаще всего выбор пользователей строится, основываясь на их опыте взаимодействия с интерфейсом той или иной платформы. Многие крупные компании в результате развития своих платформ отошли от первоначальной простоты восприятия и идеи публикации коротких

сообщений. Это усложняет интерфейс, повышает порог входа для новых пользователей и создает когнитивную нагрузку, приводит к потере части аудитории, заинтересованной именно в базовой функции микроблогинга [1]. Таким образом, существует ниша для продукта, фокусирующегося на эргономике и простоте ядерного сценария.

В рамках данного проекта целью работы является разработка интерфейса для социальной сети, ориентированной на микроблоги. Фокус при проектировании и реализации интерфейса был взят на эргономичность, простоту и высокую отзывчивость интерфейса. Комплекс задач, решенных в ходе реализации проекта:

- Разработка макета приложения, соответствующего функциональным требованиям проекта.

- Настройка окружения для разработки, установка зависимостей и конфигурация сборки проекта.

- Верстка всех необходимых компонентов для составления полноценных страниц приложения.

- Настройка маршрутизации между страницами через единый центр навигации для реализации SPA архитектуры.

- Реализация условного рендеринга модальных окон, настройка навигации, верстка форм и базовая валидация.

- Поэтапное подключение к серверу, настройка эндпоинтов и обработка данных.

- Настройка тестового окружения и разработка тестов.

Ключевым аспектом разработки является построение модульной и масштабируемой архитектуры клиентского приложения. Предлагаемая архитектура базируется на разделении ответственности между несколькими логическими слоями.

Слой представления реализован с использованием библиотеки React [2]. Компонентный подход позволяет разбить сложный интерфейс на независимые, переиспользуемые и тестируемые компоненты. В проекте применяется методология атомарного дизайна, где интерфейс декомпозируется на атомы (кнопки, поля ввода, иконки), молекулы (поисковая строка, карточка поста) и организмы (шапка сайта, список постов, лента). Это напрямую соответствует принципам модульности и повторного использования в программной инженерии.

Слой управления состоянием является критически важным для социальной сети, где данные имеют сложную структуру (пользователи, посты, ленты, уведомления, сессия) и множество источников обновления. В проекте используется Redux Toolkit (RTK) – официальный, рекомендуемый подход к использованию Redux, который решает ключевые проблемы избыточности шаблонного кода и сложности настройки [3].

В рамках управления состоянием проводится четкое разделение на:

Серверное состояние – данные, получаемые с бэкенда. Для работы с ним используется RTK Query – встроенное решение для кэширования серверного состояния. RTK Query автоматически кэширует полученные данные (лента, посты, профили), управляет повторными запросами и инвалидацией кэша. Например, после создания нового поста происходит автоматическая инвалидация кэша ленты, что гарантирует актуальность отображаемых данных без написания избыточного кода.

Клиентское состояние – состояние, относящееся исключительно к интерфейсу: открыто/закрыто модальное окно, текущая выбранная вкладка, состояние форм, индикаторы загрузки. Это состояние управляется локально в компонентах или через Redux в зависимости от масштаба использования.

Слой маршрутизации обеспечивает навигацию между страницами приложения. Использование React Router позволяет реализовать Single Page Application (SPA), где переходы между страницами происходят без полной перезагрузки, что значительно улучшает пользовательский опыт и воспринимаемую производительность.

Слой стилизации реализован с использованием TailwindCSS – фреймворка, использующего утилитарный подход [5]. Данный подход позволяет стилизовать компоненты

непосредственно в разметке путем добавления готовых классов, устраняя необходимость переключения между файлами и ускоряя создание адаптивных интерфейсов.

В проекте применяются передовые паттерны и технологии разработки для обеспечения высокого качества пользовательского опыта.

Оптимистичные обновления – паттерн, при котором интерфейс обновляется мгновенно в ответ на действие пользователя, не дожидаясь ответа от сервера. В случае ошибки запроса производится откат изменений. Это создает ощущение «живого», отзывчивого интерфейса, что критически важно для удержания пользователя.

Кэширование данных является основой производительности. RTK Query автоматически кэширует ответы от API на основе ключей запросов. При повторном посещении страницы ленты данные отображаются мгновенно из кэша, а в фоне выполняется проверка их актуальности.

Ленивая загрузка применяется для маршрутов и крупных компонентов. Код для этих частей приложения загружается только в момент необходимости, что уменьшает размер первоначальной сборки и ускоряет загрузку стартовой страницы.

В работе спроектирована и реализована клиентская часть социальной сети для микроблогов. Предложенная модульная архитектура на основе React и Redux Toolkit обеспечивает масштабируемость и высокую производительность приложения. Применение современных подходов (оптимистичные обновления, кэширование, компонентный подход, строгая типизация) направлено на создание положительного пользовательского опыта, что является ключевым фактором успеха для социальной платформы.

ЛИТЕРАТУРА

1. Норман Д. Дизайн привычных вещей / Пер. с англ. – М. Вильямс, 2018. – 384 с.
2. React Documentation [Электронный ресурс]. – URL: <https://react.dev/> (дата обращения: 14.03.2026).
3. Redux Toolkit Documentation [Электронный ресурс]. – URL: <https://redux-toolkit.js.org/> (дата обращения: 14.03.2026).
4. TypeScript Documentation [Электронный ресурс]. – URL: <https://www.typescriptlang.org/docs/> (дата обращения: 14.03.2026).
5. TailwindCSS Documentation [Электронный ресурс]. – URL: <https://tailwindcss.com/docs> (дата обращения: 14.03.2026).

УДК 004.42

И.М. Немчинов (4 курс бакалавриата),
В.Э. Шмаков, к.т.н., доцент

РАЗРАБОТКА СЕРВИСА ПО АРЕНДЕ ВОДНОГО ТРАНСПОРТА

Рынок аренды водного транспорта относится к сфере услуг, поэтому качество клиентского опыта и оперативность обработки заявок напрямую влияют на конверсию в бронирование и повторные обращения.

На практике для данного рынка характерны следующие особенности: фрагментированные каналы продаж, ручная обработка заявок, недостаточная прозрачность для конечного пользователя и отсутствие инструментов автоматизации взаимодействия сотрудников для судовладельца.

В совокупности эти особенности приводят к повышенным операционным расходам, ошибкам в расписании и снижению качества клиентского опыта, что определяет актуальность разработки специализированной системы.

Таким образом, цель данной работы – разработать прототип backend составляющей сервиса по аренде водного транспорта, в котором будут оптимизированы бизнес-процессы и решены вышеописанные проблемы.

Для реализации данной работы необходимо решить следующие задачи:

1. Проанализировать существующие решения.
2. Формализовать требования.
3. Спроектировать архитектуру сервиса.
4. Реализовать прототип сервиса.
5. Тестирование прототипа.

На данный момент проект находится на стадии реализации прототипа. Для реализации была выбрана микросервисная архитектура [1, 2], в качестве основного языка программирования был выбран язык Go [3, 4], в качестве СУБД была выбрана PostgreSQL [5]

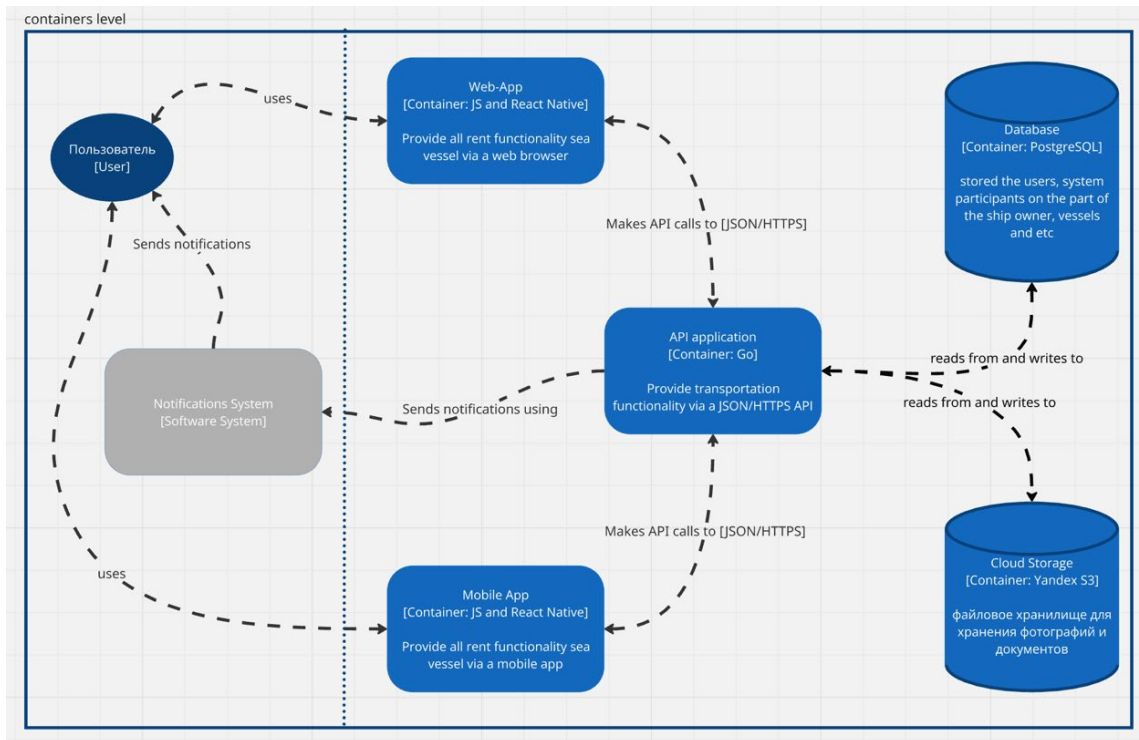


Рисунок 1 – Модель уровня контейнеров

Выбор микросервисной архитектуры (рисунок 1) обусловлен тем, что сервис объединяет несколько самостоятельных доменных пространств – пользователя, судно, расписание и бронирование. Изоляция этих доменов упрощает сопровождение системы и внедрение изменений.

Язык программирования Go был выбран, так как он хорошо подходит для разработки серверных приложений и микросервисов благодаря высокой производительности, удобно реализованной конкурентности и простоте разработки.

В дальнейшем планируется реализовать front-end часть сервиса и довести прототип до стабильно работающей версии.

ЛИТЕРАТУРА

1. James Lewis, Martin Fowler – Microservices [Электронный ресурс] <https://martinfowler.com/articles/microservices.html>
2. REST API Overview. 2023. [Электронный ресурс] <https://developer.mozilla.org/en-US/docs/Glossary/REST>
3. Golang programming language. [Электронный ресурс] <https://go.dev/>
4. The Go Programming Language - Alan A.A. Donovan and Brian Kernighan, 2015
5. PostgreSQL: Documentation [Электронный ресурс] <https://www.postgresql.org/docs/>

АНАЛИЗ ЛОГИСТИЧЕСКИХ ДАННЫХ ТРАНСПОРТНО-ЭКСПЕДИТОРСКОЙ КОМПАНИИ ДЛЯ АВТОМАТИЗАЦИИ РАСЧЁТА КОММЕРЧЕСКИХ ПРЕДЛОЖЕНИЙ

На сегодняшний день в транспортно-экспедиторской компании получение ответов на вопросы бизнеса-анализа, такие как: оценка рентабельности маршрутов, предоставление сводной информации по количеству перевозимого товара, выручке и затратам и др., является долгими и трудоемкими. В процессе анализа могут возникать ошибки ручной обработки исторических данных, расчеты могут быть сложными и не до конца прозрачными. Все перечисленные издержки приводят к формированию неточного коммерческого предложения. Поэтому актуальной является задача автоматизации проведения бизнес-анализа и построение коммерческих предложений с использованием программного средства [2, 3].

Целью работы является разработка автоматизированного решения, которое на основе структурированных данных позволяет оперативно формировать основу для коммерческого предложения, а также проводить оперативный и точный расчет необходимых экономических параметров. Такое решение позволит не только сократить трудоёмкость составления коммерческого предложения, но и повысить его актуальность за счёт учёта исторических данных по клиентам и успешным операциям [1, 4].

Для достижения цели были поставлены следующие ключевые задачи.

1. Подготовка данных из 1С (xlsx-файл) и очистка дублирующихся и некорректных строк.
2. Импорт подготовленных данных в Elasticsearch, создание индекса с корректными типами данных [3].
3. Визуализация ключевых показателей деятельности: число грузов, вес, объём, выручка, маржа, расходы, загруженность персонала и т.д. [3].
4. Создание интерактивных дашбордов и аналитических панелей в инструменте Kibana с фильтрацией по ключевым параметрам (датам, клиентам, странам и т.д.) [3].

Для автоматизации расчёта коммерческих предложений на основе логистических данных, экспортируемых из 1С «1АБ:Карго», была разработана гибкая архитектура, построенная на принципах потоковой обработки, визуальной аналитики и модульной архитектуры. Решение реализовано с использованием следующего технологического стека:

- Elasticsearch + Kibana (+ Vega) – хранение, поиск, расчет и визуализация данных [3];
- Python – обработка и загрузка данных, расчёт части параметров;
- Docker Compose – контейнеризация и развёртывание компонентов.

Архитектура состоит из трёх основных компонентов:

1. Elasticsearch – центральное хранилище данных, в котором индексируются нормализованные логистические данные [3].
2. Kibana – презентационный слой, предоставляющий интерфейс для построения аналитических дашбордов, графиков и таблиц [3].
3. Python-приложение – модули обработки данных, отвечающий за прием выгрузки в формате xlsx из 1С, очистку и нормализацию исходных данных, расчёт производных метрик, преобразование данных в формат NDJSON с bulk-загрузкой в Elasticsearch.

Такая архитектура обеспечивает минимальную задержку между выгрузкой данных и их доступностью в аналитике, простоту масштабирования, гибкость анализа – пользователи могут строить кастомные дашборды под бизнес-сценарии без участия разработчиков [1, 3].

На основе расчетов была реализована визуализация в виде дашбордов в Kibana. Создано 2 группы дашбордов с 57 визуализациями. На рисунке 1 и рисунке 2 представлены примеры визуализаций. Благодаря представленным визуализациям, система обеспечивает полный цикл работы с данными для заказчика: от обработки данных до расчета заданных параметров.

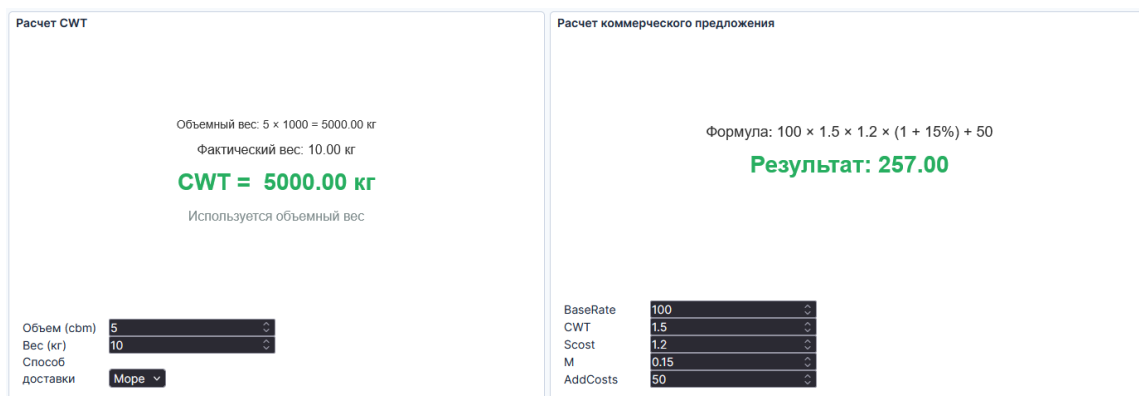


Рисунок 1 – Пример визуализаций, созданных с помощью Vega

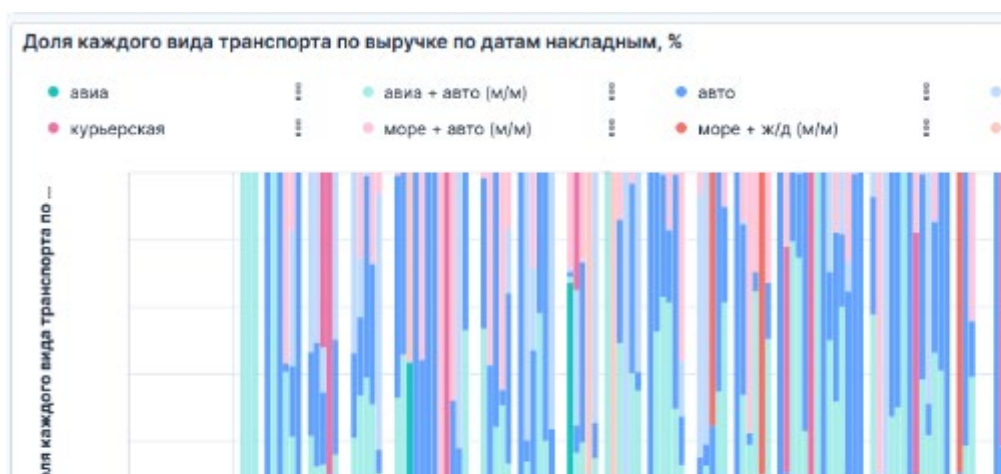


Рисунок 2 – Пример визуализации, созданной с помощью Lens

Полученная архитектура была упакована в Docker-контейнер для переносимости. Также были созданы bat- и sh-скрипты для быстрого развертывания. Итоговый образ ПО был передан заказчику в эксплуатацию.

ЛИТЕРАТУРА

1. Методы, алгоритмы и архитектуры распределенной обработки больших данных / И. В. Никифоров, О. А. Юсупова, Н. В. Воинов, А. Д. Ковалев. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2023. – 194 с. – ISBN 978-5-7422-8461-1. – EDN DFTKEX.
2. Никифоров, И. В. Курсовое проектирование по учебной дисциплине "Наука о данных и аналитика больших объемов информации" : Учебное пособие / И. В. Никифоров. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2017. – 62 с. – ISBN 978-5-7422-5638-0. – EDN XPRQXB.
3. Программные инструменты обработки и визуализации данных. Elasticsearch, Logstash, Kibana, Grafana, Prometheus / И. В. Никифоров, О. А. Юсупова, Н. В. Воинов [и др.]. – Санкт-Петербург : Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. – 140 с. – ISBN 978-5-7422-8075-0. – DOI 10.18720/SPBPU/2/id23-74. – EDN KPGDVS.
4. Big data processing system for analysis of GitHub events / N. Voinov, K. Rodriguez Garzon, I. Nikiforov, P. Drobintsev // Proceedings of 2019 22nd International Conference on Soft Computing and Measurements, SCM 2019: 22, St. Petersburg, 23–25 мая 2019 года. – St. Petersburg, 2019. – P. 187-190. – DOI 10.1109/SCM.2019.8903782. – EDN RDBGKV.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ФОРМИРОВАНИЯ БАЗЫ ДАННЫХ ПО СУДАМ-ПРОТОТИПАМ

Проектирование новых судов опирается на анализ уже построенных ранее судов. Их характеристики позволяют применять проверенные технологии и технические решения при разработке новых проектов. В связи с этим актуальной задачей становится формирование структурированной базы данных судов-прототипов.

Сегодня информация о судах часто разрознена и хранится на различных ресурсах в разных форматах. При этом значительная часть технической документации публикуется в виде PDF-файлов.

Для решения этой задачи был использован ETL-процесс (Extract, Transform, Load). На этапе Extract происходит извлечение информации из исходных документов. Далее на этапе Transform осуществляется очистка и преобразование полученных данных, а на этапе Load происходит загрузка обработанной информации в формируемую базу данных [1].

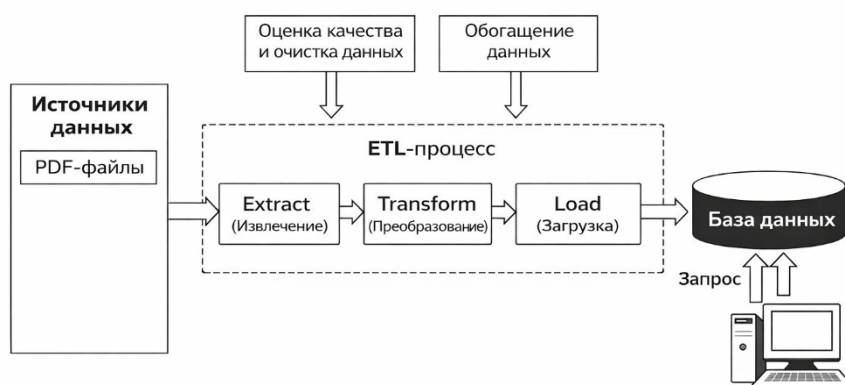


Рисунок 1 – Схема ETL-процесса

В данной работе была разработана программа на C++, предназначением которой является автоматическое извлечение текстовой информации из PDF-файлов, в которых содержится техническое описание и характеристики разных судов-прототипов. Полученные данные после этапов извлечения (Extract) и преобразования (Transform) сохраняются в виде CSV-файлов, что позволяет использовать полученные данные для формирования базы данных [2].

Одной из проблем при работе с PDF является появление лишних пробелов между символами, что приводит к разбиению слов на отдельные символы из-за чего появляется затруднение автоматизированной обработки. Чаще всего это происходит из-за специфической кодировки или использования нестандартных шрифтов в исходном документе [3].

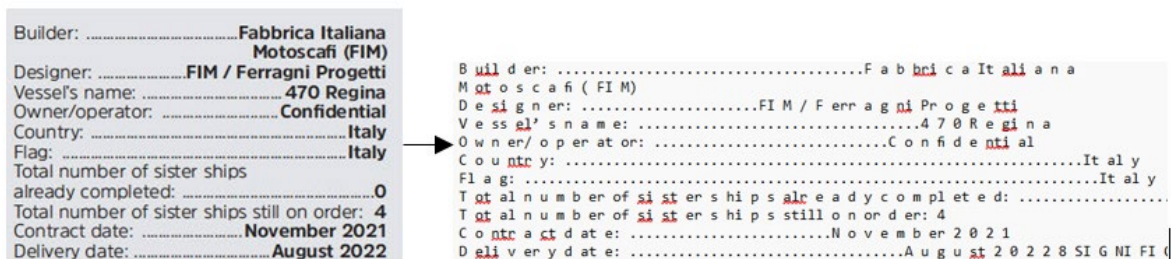


Рисунок 2 – Пример текста, извлеченного из PDF-файла с лишними пробелами

В разработанном приложении данная проблема была решена на этапе извлечения (Extract) текста. Алгоритм анализирует извлеченный текст на наличие большого количества пробелов и выполняет удаление этих пробелов, что позволяет получить корректную структуру слов.

Figure 3 shows two versions of a text block. The left version has many spaces between words, making it difficult to read. The right version has the spaces removed, resulting in a cleaner, more structured text. An arrow points from the left version to the right version, indicating the transformation.

Рисунок 3 – Результат обработки текста после удаления лишних пробелов

Кроме этой проблемы, при работе с PDF-файлами были выявлены и другие проблемы извлечения текста. Некоторые документы представляют собой сканированные изображения страниц, что приводит к невозможности извлечения данных без применения компьютерного зрения.

Также в отдельных документах есть проблема с дублированием символов, возникающая из-за наличия нескольких текстовых слоев, появляющихся после обработки системного распознавания текста.

Figure 4 shows two versions of a text block. The left version has some characters repeated, making it difficult to read. The right version has the duplicated characters removed, resulting in a cleaner, more structured text. An arrow points from the left version to the right version, indicating the transformation.

Рисунок 4 – Пример дублирования символов при извлечении текста

После этапа извлечения выполняется преобразование и очистка полученных данных. Итоговая информация загружается в СУБД PostgreSQL, что позволяет обеспечить удобство дальнейшей обработки и анализа данных.

FileNan	VesselName	Builder	Designer	OwnerOperator	Country	DeliveryDate	Length(m)	MaxSpeed(knots)	ImoNumber	Gross	SisterShip
SSS 2007	VB Jerez	Union Naval Valencia Ltd	Robert Allan	Sertosa (Grupo Boluda)	Spain	04.07.2007	23	12	237	8	
SSS 2007	Harry Evans	Cheoy Lee Shipyards Ltd	Robert Allan Ltd	Rio Tinto Aluminium	Australia	23.11.2007	30	12	399	2	
SSS 2008	Handy One	Great Lakes Shipyard	Jensen Maritime Consultants	Tugz International LLC	United States of America	11.04.2008	22	12	100	1	

Рисунок 5 – Результат выполнения извлечения

ЛИТЕРАТУРА

1. Цехановский, В. В. Управление данными : учебник / В. В. Цехановский, В. Д. Чертовской. — Санкт-Петербург: Лань, 2022. — 432 с. — ISBN 978-5-8114-1853-4. — Текст: электронный // Лань: электроннобиблиотечная система. — URL: <https://e.lanbook.com/book/212084> - страницы 105-106.
2. Кузнецов, С. Д. Основы баз данных : учебное пособие / С. Д. Кузнецов. — Москва: Интернет-университет информационных технологий, 2017. — 488 с. — Текст: электронный // Национальный открытый университет «ИНТУИТ». — URL: <https://www.intuit.ru/studies/courses/74/74/info>
3. Федотов, М. А. Методы обработки текстовой информации : учебное пособие / М. А. Федотов. — Москва: Физматлит, 2016. — 296 с.

РАЗРАБОТКА МЕТОДА ПЕРСОНАЛИЗАЦИИ ИНТЕРФЕЙСА МОБИЛЬНОГО
ПРИЛОЖЕНИЯ НА ОСНОВЕ ПОВЕДЕНИЯ ПОЛЬЗОВАТЕЛЯ

Виртуальные сенсорные элементы управления широко используются в мобильных играх и приложениях, в которых сенсорный ввод является основным способом взаимодействия с интерфейсом. Тем не менее единая статичная раскладка не способна учесть разнообразие размеров и соотношений сторон современных экранов, а также индивидуальные антропометрические особенности пользователей: длину и подвижность пальцев, привычный хват устройства, угол его наклона. По данным исследований, доля ошибочных касаний в ходе интенсивных игровых сессий достигает 20-30%, а у пользователей с нестандартными размерами рук превышает 40% [1]. Установлено также, что систематические смещения точки касания от целевой зоны в среднем составляют 4-8 мм и зависят от положения элемента на экране и ориентации устройства [2].

Существующие подходы к решению этой проблемы можно разделить на три группы. Первая – фиксированные пресеты раскладки, предлагающие 3-6 вариантов и не покрывающие полного разнообразия характеристик пользователей. Вторая – полностью ручная настройка каждого элемента, которая требует значительных усилий и нередко даёт неоптимальный результат. Третья – полуавтоматические эвристические системы, учитывающие лишь средние координаты касания и игнорирующие направление и величину систематических ошибок [3]. Ни один из перечисленных подходов не обеспечивает автоматической адаптации интерфейса без участия пользователя на основе анализа его реального поведения.

Целью настоящей работы является разработка метода персонализации интерфейса мобильного приложения, который автоматически корректирует расположение элементов управления в соответствии с индивидуальными моторными паттернами пользователя.

Для достижения поставленной цели необходимо решить задачи, описанные ниже.

1. Провести анализ существующих подходов к адаптации сенсорных интерфейсов мобильных приложений и выявить их ключевые ограничения.
2. Разработать модель сбора и хранения телеметрических данных о взаимодействии пользователя с элементами интерфейса.
3. Разработать алгоритм выявления индивидуальных паттернов ошибок касания с применением методов статистического анализа и машинного обучения.
4. Разработать алгоритм оптимизации параметров элементов управления с учётом выявленных паттернов и ограничений компоновки.
5. Реализовать предложенный метод и провести его экспериментальную оценку.

В основе предлагаемого метода лежит фоновый сбор телеметрии касаний. Для каждого взаимодействия пользователя с элементом управления фиксируются координаты точки касания, вектор смещения от центра целевой зоны, расстояние промаха и временная метка. Данные накапливаются на протяжении всего периода использования приложения, что позволяет отслеживать долгосрочные изменения поведения пользователя [4].

На этапе анализа для каждого элемента управления вычисляется медианный вектор систематического смещения и строится распределение расстояний промахов. Для выявления устойчивых пространственных областей ошибочных касаний применяются алгоритмы кластеризации на основе плотности [5]. Полученные характеристики используются для постановки задачи оптимизации нового положения элементов: новая позиция каждой кнопки или стика вычисляется так, чтобы минимизировать среднее расстояние промаха при соблюдении ограничений на взаимное расстояние между элементами и максимально допустимое смещение от исходной конфигурации [6].

Предложенный метод реализован в виде мобильного приложения на платформе Flutter с использованием языка Dart [7,8], среды разработки Android Studio [9,10] и протестирован на нескольких Android-устройствах с различными размерами и разрешениями экранов. По итогам испытаний точность попаданий по основным элементам управления возросла, среднее расстояние промаха сократилось, а оценки субъективного удобства по шкале System Usability Scale (SUS) повысились с диапазона «плохо» (26-50 баллов) до диапазона «хорошо» (71-84 балла). Подавляющее большинство участников предпочли продолжать работу с адаптированной раскладкой, а саму процедуру калибровки сочли ненавязчивой и органично вписанной в процесс работы приложения.

ЛИТЕРАТУРА

1. Zhang Y., Li H. Touch error patterns in mobile gaming: A large-scale field study // IEEE Trans. Games. – 2024. – Vol. 16. – No 3. – P. 512-523.
2. Henze N., Mayer S., Le H. V. et al. Touch accuracy in the wild: A large-scale field study // Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. – 2024. – Vol. 8. – No 4. – Article 162.
3. Henze N., Rukzio E., Boll S. 100,000,000 taps: analysis and improvement of touch performance in the large // Proc. 13th Int. Conf. Human Computer Interaction with Mobile Devices and Services (MobileHCI 2011). – Stockholm, 2011. – P. 133-142.
4. Leiva L. A., Martin-Albo D., Bailly G. et al. Touch event logging for mobile user modeling: Current practices and future directions // Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. – 2025. – Vol. 9. – No 1. – Article 32.
5. Ester M., Kriegel H.-P., Sander J., Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise // Proc. 2nd Int. Conf. KDD-96. – Portland, 1996. – P. 226-231.
6. Wierzbicki A., Dabrowski M. Constrained multi-objective optimization of mobile UI layouts // Proc. MobileHCI 2024. – Melbourne, 2024. – Article 27.
7. Свидетельство о государственной регистрации программы для ЭВМ № 2025613621 Российская Федерация. Программа для онлайн-тренажера для освоения сквозных бизнес-процессов предприятия : заявл. 28.01.2025 : опубл. 13.02.2025 / А. Н. Амбражей, В. А. Касилов, Е. Жарко [и др.] ; заявитель федеральное государственное автономное образовательное учреждение высшего образования “Санкт-Петербургский политехнический университет Петра Великого”. – EDN NATSGG.
8. Интеграция между Backend и MQTT брокером через Apache Kafka: оптимизация обмена данными и снижение нагрузки / А. Н. Тихонов, А. И. Пухальский, Д. Ф. Дробинцев, А. В. Гончаров // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 24–25 апреля 2024 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. – С. 220-222. – EDN WLPECC.
9. Тимошук, Е. В. Анализ данных профилей сотрудников в социальной сети linkedin / Е. В. Тимошук, И. В. Никифоров, Д. Ф. Дробинцев // Современные технологии в теории и практике программирования : сборник материалов конференции, Санкт-Петербург, 23 апреля 2020 года / Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2020. – С. 133-135. – EDN MXXIJH.
10. Базовая функциональность систем централизованного управления мобильными устройствами / А. В. Самочадин, Т. Н. Самочадина, Д. А. Тимофеев, С. О. Попов // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. – 2014. – № 4(200). – С. 17-28. – EDN SUFHYP.

СЕРВИС УПРАВЛЕНИЯ ОТСУТСТВИЯМИ В МИКРОСЕРВИСНОЙ КОРПОРАТИВНОЙ ПЛАТФОРМЕ

Автоматизация кадровых процессов, включая переход к электронному документообороту и централизованному хранению данных сотрудников, является актуальной задачей в условиях цифровизации предприятий [1]. Одним из ключевых процессов является управление отпусками и отсутствиями, которое включает фиксацию периодов отсутствия, учет нормативных требований и контроль остатка отпускных дней.

Существующие решения для автоматизации кадровых процессов нередко поставляются в составе крупных систем и требуют внедрения полного программного комплекса целиком, что ограничивает гибкость интеграции с уже работающей системой компании и усложняет адаптацию под ее внутренние регламенты. Дополнительные требования связаны с защитой персональных данных, информационной безопасностью и необходимостью импортозамещения [2]. В этих условиях возникает необходимость разработки специализированного сервиса управления отсутствиями, встроенного в корпоративную микросервисную платформу, обеспечивающего поэтапное замещение соответствующей функциональности ERP-решения и перенос соответствующей функциональности в распределенную микросервисную архитектуру.

Система управления отпусками и отсутствиями реализована в виде микросервисов, встроенных в корпоративную платформу [3]. Основным компонентом решения является сервис управления отсутствиями, который реализует бизнес-логику обработки заявок: прием и валидацию пользовательских данных, хранение заявок и отправку событий, инициирующих запуск и дальнейшее выполнение процесса обработки заявки в сервисе управления бизнес-процессами. В рамках выполнения бизнес-логики сервис управления отсутствиями взаимодействует с другими компонентами платформы, получая сведения о сотрудниках, организационной структуре и справочных данных, а также использует платформенные механизмы аутентификации и авторизации и сервисы электронного документооборота и отправки уведомлений. Обмен данными реализован через синхронные и асинхронные механизмы взаимодействия, при которых одни сервисы выступают источниками кадровых данных, а другие используют полученные данные в своей бизнес-логике. Общая схема взаимодействия сервиса управления отсутствиями с другими компонентами корпоративной платформы представлена на рисунке 1.

Процессная логика обработки заявок вынесена в отдельный сервис управления бизнес-процессами. Жизненный цикл заявки формализован в виде BPMN-диаграмм, которые определяют этапы обработки заявки и события, инициирующие переходы между состояниями процесса. Разделение бизнес-логики и процессной логики обеспечивает независимость развития компонентов: изменение маршрутов согласования выполняется на уровне BPMN-диаграмм и не требует трудоемкой переработки бизнес-логики сервиса управления отсутствиями, а изменение правил обработки данных не требует модификации сервиса управления процессами.

Микросервисы реализованы на языке Kotlin с использованием фреймворка Spring Boot. Для исполнения BPMN-моделей используется платформа управления бизнес-процессами Camunda, интеграция с которой осуществляется через Java API. В качестве системы хранения данных используется PostgreSQL. Выбор данного технологического стека обусловлен его зрелостью и применимостью для построения масштабируемых распределенных систем [4]. Использование данных технологий также обусловлено требованиями корпоративной платформы, в рамках которой основная часть сервисов реализована на Kotlin и Java.

Следование принятому технологическому стандарту обеспечивает единообразие архитектурных решений, упрощает интеграцию компонентов и снижает сложность сопровождения системы.

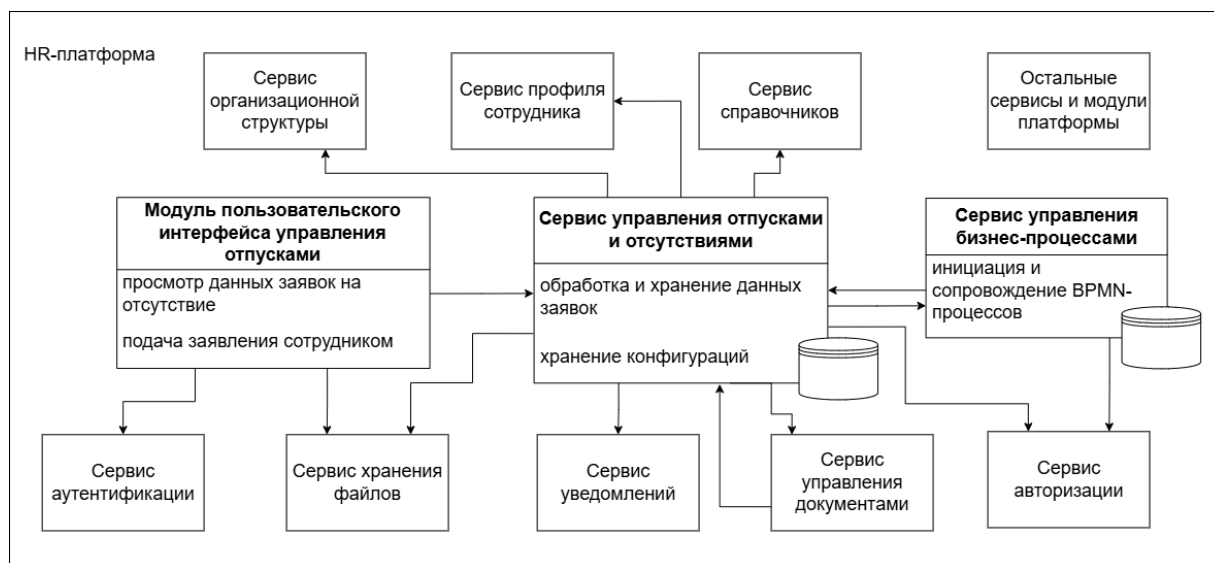


Рисунок 1 – Архитектура решения в корпоративной платформе

Сервис реализован и в рамках опытной эксплуатации демонстрирует практический подход к поэтапному замещению ERP-функциональности прикладными микросервисами. Декомпозиция монолитной логики, формализация процессов в BPMN и интеграция с сервисами кадровых данных создают основу для построения масштабируемой и устойчивой корпоративной платформы [5].

ЛИТЕРАТУРА

1. Оборин М. С. Технологии управления человеческими ресурсами в условиях цифровизации // Вестник Нижегородского университета им. Н. И. Лобачевского. Серия: Социальные науки. – 2023. – № 1 (69). – С. 33–41.
2. Карпова Н. Е., Бабинова А. А. Обеспечение безопасности персональных данных в информационной системе предприятия // Безопасность цифровых технологий. – 2024. – № 2 (113). – С. 55–68. – DOI 10.17212/2782-2230-2024-2-55-68.
3. Newman S. Building Microservices: Designing Fine-Grained Systems. – 2nd ed. – Sebastopol: O'Reilly Media, 2021. – 586 p.
4. Java is Distributed. [Электронный ресурс] Режим доступа: <https://www.javaspring.net/blog/java-is-distributed> (дата обращения: 01.03.2026)
5. Taibi D., Lenarduzzi V., Pahl C. Processes, motivations, and issues for migrating to microservices architectures // IEEE Cloud Computing. – 2017. – Vol. 4. – No. 5. – С. 22–32.

УДК 004.453

Д.А. Осокин (2 курс магистратуры),
Н.В. Воинов, к.т.н., доцент

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЯ ДЛЯ ПОЗИЦИОНИРОВАНИЯ ИОТ-УСТРОЙСТВ БЕЗ ИСПОЛЬЗОВАНИЯ GPS

Стремительная интеграция технологий Интернета вещей (IoT) и носимой электроники формирует глобальную экосистему подключенных устройств. Возникают новые технологические вызовы, среди которых центральное место занимает проблема точного позиционирования объектов внутри помещений или в условиях недоступности сигналов глобальных навигационных спутниковых систем [1]. Экспоненциальный рост числа радиоэлектронных средств существенно усложняет электромагнитную обстановку, приводя к

насыщению радиочастотного спектра и интерференции. В связи с этим разработка доступных программных решений для автоматизированного обнаружения, классификации и локализации радиосигналов с помощью мобильных устройств является актуальной научно-технической задачей, востребованной в промышленном и потребительском секторах.

Основной целью данной работы является разработка программно-алгоритмического обеспечения для пространственной локализации и анализа источников радиоизлучения с использованием мобильных вычислительных платформ.

Для достижения поставленной цели необходимо решение следующих задач:

1. Проведение сравнительного анализа методов определения координат.
2. Адаптация вычислительно емких алгоритмов локализации для работы на ресурсах мобильных устройств.
3. Программная реализация комплекса для обнаружения и классификации радиосигналов.

Решение задачи базируется на анализе трех основных классов алгоритмов позиционирования [2]: дальномерных, сигнатурных и спектральных. В основе дальномерных методов лежит оценка расстояния через измерение мощности принятого сигнала (RSSI) [3]. Перевод уровня мощности в метрическое расстояние базируется на модели распространения радиоволн, описываемой формулой Фрииса:

$$Pr = Pt * Gt * Gr * \left(\frac{\lambda}{4 * \pi * d} \right)^2 \quad (1)$$

где Pr – мощность на входе приемника, Pt – мощность передатчика, Gt и Gr – коэффициенты усиления антенн, λ – длина волны, d – расстояние между антеннами. Однако в условиях многолучевого распространения точность данного метода снижается.

В качестве альтернативы рассмотрен метод сценарного анализа (Fingerprinting), состоящий из этапов предварительного обучения (формирования базы данных радиоотпечатков) и позиционирования [4]. Текущий вектор сигналов сравнивается с эталонным с использованием детерминированных (kNN) или вероятностных подходов.

Для задач, требующих сверхвысокой точности оценки угла прихода сигнала (AoA), проанализирован алгоритм MUSIC (Multiple Signal Classification) [5]. Метод базируется на разложении пространства наблюдений на сигнальное и шумовое подпространства с построением функции псевдоспектра:

$$P_{music}(\theta) = \frac{1}{a(\theta)^H * E_n * E_n^H * a(\theta)} \quad (2)$$

где $a(\theta)$ – управляющий вектор антенной решетки, E_n – матрица шумового подпространства, H – оператор эрмитова сопряжения. Пики данной функции указывают на углы прихода сигналов [1].

Выбор итогового алгоритма позиционирования представляет собой компромисс между требуемой точностью и доступной вычислительной мощностью. Для программной реализации предложен гибридный подход [2,3], сочетающий энергоэффективные методы оценки RSSI с элементами спектрального анализа. В качестве целевой платформы выбрана операционная система Android. Архитектурная открытость платформы предоставляет доступ к низкоуровневым API радиомодулей (Wi-Fi, Bluetooth) для извлечения первичных характеристик сигнала без использования дополнительного аппаратного обеспечения. В качестве языка разработки использован Kotlin, чья строгая система типов и безопасность null-ссылок повышают надежность приложения в процессе непрерывного радиомониторинга.

В результате проведенного исследования предложена архитектура мобильного программного комплекса позиционирования. Интеграция алгоритмов оценки параметров сигналов и API операционной системы Android позволила разработать решение, трансформирующее стандартный смартфон в аналитический инструмент сканирования

радиоэфира. Разработанный подход демонстрирует возможность автоматизированной локализации источников радиоизлучения в сложной электромагнитной обстановке без обращения к службам GPS, оптимизируя потребление системных ресурсов мобильного устройства.

ЛИТЕРАТУРА

1. Chen M. et al. A new method for AOA estimation and TDOA/AOA joint localization of GNSS spoofer // Proceedings of the 2025 International Technical Meeting of The Institute of Navigation. – 2025. – С. 521-541.
2. Chung P. J., Viberg M., Yu J. DOA estimation methods and algorithms // Academic Press Library in Signal Processing. – Elsevier, 2014. – Vol. 3. – P. 599-650.
3. Vicente D. et al. Kalman filter for target tracking using coupled RSS and AoA measurements // 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC). – IEEE, 2017. – P. 2004-2008.
4. Sun D. et al. A BLE indoor positioning algorithm based on weighted fingerprint feature matching using AoA and RSSI // 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP). – IEEE, 2021. – P. 1-6.
5. Arshaghi A. Enhancing DOA Estimation: MUSIC Algorithm and Comparative Analysis of TOA, TDOA, and AOA Techniques // International Journal of Smart Electrical Engineering. – 2025. – Т. 14. – №. 1. – С. 25-31.

УДК 004.85

К.А. Овсянников (4 курс бакалавриата),
О.Г. Малеев, к.т.н., доцент

ПОСТРОЕНИЕ ПОВЕДЕНЧЕСКОГО ПОРТРЕТА КЛИЕНТА НА ОСНОВЕ ТРАНЗАКЦИОННЫХ ДАННЫХ ДЛЯ ЗАДАЧ ПЕРСОНАЛИЗАЦИИ В РОЗНИЧНОЙ СЕТИ

В современных условиях высокой конкуренции на рынке продуктового ритейла ключевым фактором успеха становится персонализация взаимодействия с покупателями [1]. Розничные сети накапливают огромные объёмы данных о транзакциях, предлагаются подходы по анализу потребителей с использованием алгоритмов видеоаналитики [2], однако значительная часть клиентов остаётся анонимной либо идентифицируется только по номеру карты лояльности, что ограничивает применение классических маркетинговых подходов [3], основанных на демографических характеристиках. Целью данной работы является разработка и апробация подхода к построению поведенческого портрета клиента на основе транзакционных данных, включающего сегментацию покупателей и восстановление их демографических характеристик (пол, возрастная группа) с помощью методов машинного обучения.

Исходными данными послужили анонимизированные записи о покупках (790 513 чеков) и информация о 10 000 клиентах одной из розничных сетей. Характерной особенностью набора данных является высокая доля пропусков [4] в персональной информации: фамилия заполнена лишь у 36,5 % клиентов, а поле «пол» в 80 % случаев содержит значение Unknown. Это обстоятельство подтверждает актуальность разработки методов, устойчивых к неполноте данных и способных восстанавливать недостающие характеристики на основе истории покупок.

На этапе предобработки выполнена агрегация транзакций до уровня клиента с формированием расширенного набора признаков. Помимо классических RFM-метрик [5][6] (давность последней покупки, частота, суммарные затраты) были вычислены:

- средний чек;
- разнообразие корзины (среднее количество уникальных товаров в чеке);
- коэффициент использования скидков;

- стабильность покупательской активности (стандартное отклонение интервалов между покупками);
- доли затрат по товарным категориям (на основе иерархии товаров);
- временные характеристики (предпочтительные дни недели, время суток);
- предпочтения по магазинам (доля чеков в каждом магазине).

На основе этих признаков построен 73-мерный вектор, описывающий поведенческий профиль каждого клиента. Для вычисления признаков разработан конвейер на языке Go, обеспечивающий высокую производительность и возможность масштабирования на большие объёмы данных. Результат сохраняется в файл «customer_vectors.csv», который служит основой для дальнейшего анализа.

Для сегментации клиентов применён алгоритм кластеризации k-means [7]. Оптимальное число кластеров ($k=5$) выбрано с помощью метода локтя и оценки силуэта. В результате выделены следующие сегменты, интерпретируемые с точки зрения покупательского поведения:

1. «Лояльные» - регулярные покупки, высокий средний чек, широкий ассортимент.
2. «Экономные» - часто используют скидки, небольшая сумма чека, ограниченный набор категорий.
3. «Активные выходного дня» - пик покупок приходится на субботу и воскресенье.
4. «Спящие» - давно не совершали покупок, низкая частота.
5. «VIP» - максимальные значения частоты, суммы и разнообразия.

Полученная сегментация позволяет точнее настраивать коммуникацию с каждой группой и формировать релевантные предложения.

Важной задачей является восстановление демографических характеристик анонимных клиентов. На основе поведенческих векторов обучены две модели классификации:

- классификатор пола (мужской / женский / неизвестно);
- классификатор возрастной группы (4 группы: до 25, 25–35, 36–50, старше 50).

В качестве алгоритма использован градиентный бустинг (CatBoost), продемонстрировавший наилучшее качество на валидационной выборке. Метрики качества модели пола составили: accuracy = 0,81, F1-score = 0,78. Для возрастных групп accuracy = 0,73, macro F1 = 0,69. Наиболее информативными признаками оказались частота покупок, доля затрат на определённые категории товаров (например, алкоголь, детские товары, косметика) и средний чек.

Полученные модели позволяют для анонимных клиентов (с пропущенными или недостоверными персональными данными) восстанавливать пол и возраст с приемлемой точностью, что даёт возможность включать их в программы персонализации, ранее доступные только для зарегистрированных пользователей.

Разработанный подход реализован в виде программного комплекса на языках Go и Python, интегрирующего этапы загрузки данных, расчёта признаков, кластеризации и обучения моделей. Комплекс может быть легко встроен в существующую аналитическую инфраструктуру розничной сети.

В дальнейшем планируется расширить набор признаков за счёт анализа последовательностей покупок (с использованием рекуррентных нейронных сетей), а также добавить прогнозирование оттока клиентов и рекомендательную систему на основе поведенческих профилей.

ЛИТЕРАТУРА

1. Котлер Ф. Маркетинг менеджмент. – СПб.: Питер, 2019.
2. Ицзя, Л. Применение нейронных сетей для оценки трафика магазина / Л. Ицзя, О. Г. Малеев // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 26–27 апреля 2023 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. – С. 54-55. – EDN WJRDNR.

3. Ho T., Nguyen S., Nguyen H. et al. "An Extended RFM Model for Customer Behaviour and Demographic Analysis in Retail Industry" // Business Systems Research Journal. — 2023. — Vol. 14, № 1. — P. 26-53. (Scopus, WoS)
4. Швецов Ф.Е. Микросегментация потребителей на основе машинного обучения: обзор методов и российская практика // Экономика и предпринимательство. — 2025. — № 6. — С. 236–240.
5. Tyagi U., Gupta N. Customer Segmentation Using Machine Learning // European Economic Letters. — 2025. — Vol. 15, No. 1. — P. 234–241.
6. Mazumder A. AI-Powered Customer Segmentation in Grocery Retail // International Journal of Computer Trends and Technology. — 2025. — Vol. 73, No. 2. — P. 45–52.
7. Ebadi Jalal M., Elmaghraby A. Analyzing the Dynamics of Customer Behavior: A New Perspective on Personalized Marketing through Counterfactual Analysis // Journal of Theoretical and Applied Electronic Commerce Research. — 2024. — Vol. 19, No. 3. — P. 1660-1681. — DOI: 10.3390/jtaer19030081. (Scopus, WoS)

УДК 004.453

К.С.Панкратова (4 курс бакалавриата),
А.О. Жаранова, ст. преподаватель

АВТОМАТИЗИРОВАННОЕ СРЕДСТВО ОРГАНИЗАЦИИ ПРОЦЕССА ВЯЗАНИЯ ИЗДЕЛИЙ С МОДУЛЕМ РАСПОЗНАВАНИЯ ПЕТЕЛЬ НА СХЕМАХ ПО ВЯЗАНИЮ КРЮЧКОМ

В последние годы наблюдается значительное возрождение интереса к рукоделию, в том числе и вязанию крючком. Число людей, регулярно занимающихся вязанием и смежными видами рукоделия, оценивается в миллионах по всему миру. Так, крупнейшая специализированная международная платформа Ravelry объединяет более 9 миллионов зарегистрированных пользователей [1].

В настоящее время на рынке представлены разнообразные приложения, предназначенные для отслеживания процесса вязания крючком. В рамках изучения обзора существующих решений в области организации процесса вязания изделий [2], а также данных из платформ для скачивания приложений выявлены недостатки, а именно, пользователи отмечают сложность интерфейса, отсутствие поддержки русского языка и невозможность создавать и управлять собственными проектами из-за внутренних ошибок системы, что снижает доступность приложения для пользователей. А также в некоторых приложениях отсутствует возможность регистрации для российских пользователей. Несмотря на некоторые высокие оценки, согласно пользовательским отзывам и личному опыту, выявлены следующие недостатки в виде плохо продуманного интерфейса счётчика рядов, отсутствие качественной поддержки русского языка. А также ряд пользователей отметили неэстетичный дизайн.

Также стоит отметить, что ИИ-помощники повсеместно внедряются в разные сферы жизни человека, например, они используются для решения задач распознавания образов и символов. Современные подходы, такие как глубокие нейронные сети, позволяют эффективно интерпретировать визуальные данные, включая схемы и изображения. В проекте планируется реализовать подобный модуль для работы со схемами вязания, который позволит пользователю тратить меньше времени на расшифровку всех обозначений. Особенно полезным данный инструмент станет для новичков, которые могут иметь недостаточно опыта в чтении схем.

Таким образом, целью данного проекта является разработка приложения для организации процесса вязания изделия с интегрированным интеллектуальным модулем обнаружения и классификации символов схем на примере вязания крючком.

Для решения одной из центральных проблем, а именно, разработка модуля для распознавания схем, выбрана модель YOLOv11, так как эта модель предоставляет высокую точность [3], необходимую для корректной классификации условных обозначений, и

отличается простотой внедрения в готовый продукт. YOLOv11 выигрывает за счет экосистемы UltraLytics, которая предоставляет унифицированный и простой интерфейс. Обучение и тонкая настройка модели требуют минимальных усилий. Таким образом, YOLO является более предпочтительным решением для прикладных систем, требующих высокой скорости работы и простоты внедрения, что очень важно в разрабатываемом приложении

При выборе модели также учтены ограничения, связанные со сбором и разметкой данных. Схемы вязания не имеют стандартных открытых датасетов, а сами символы петель характеризуются плотностью расположения на схемах, несколькими вариантами обозначения и разными поворотами символов на схеме. Это накладывает дополнительные требования к объёму и качеству обучающей выборки. Ultralytics для YOLOv11 предоставляет готовые инструменты для аугментации данных, автоматической проверки разметки и дообучения на небольших датасетах, что снижает трудоёмкость подготовки данных и делает модель более устойчивой при ограниченном объёме обучающей выборки [4].

Для разработки серверной части выбран Java. Он демонстрирует более высокую пропускную способность [5]. Java обладает типизацией [6], которая позволяет обнаруживать ошибки на этапе компиляции, повышает надёжность. Java также имеет возможность использования готовых моделей, например, ONNX Runtime имеет высокопроизводительный Java API [7], который идеально подходит для запуска оптимизированных моделей в Java-среде.

Архитектура backend-части приведена на рисунке 1.

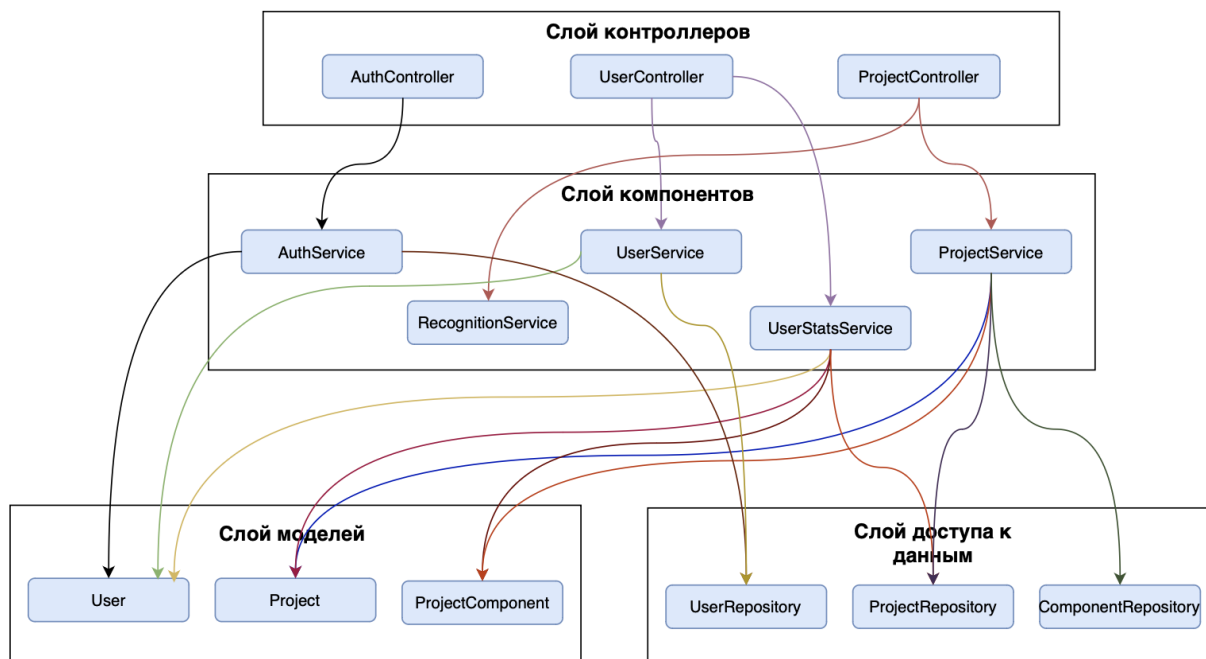


Рисунок 1 – Архитектура backend-части приложения

Система разделена на отдельные компоненты, отвечающие за разный функционал приложения: авторизация пользователей, вывод статистики пользователя, создание и работа над проектами по вязанию, просмотр своего профиля, а также компонент, содержащий различные инструменты для мастеров, например, перерасчет необходимого количества пряжи в зависимости от данных связанного образца.

Таким образом, реализованное средство автоматизирует процесс вязания изделий для мастеров и позволяет использовать интеллектуальный модуль для решения проблемы по чтению схем для вязания.

ЛИТЕРАТУРА

1. Ravelry. About us // Ravelry. [Электронный ресурс] Режим доступа: <https://www.ravelry.com/about> (дата обращения: 25.02.2026).

2. Review of the Best Knitting Software and Apps for Designing Patterns, Tracking Progress, and Connecting with the Knitting Community / YouTalent. [Электронный ресурс] Режим доступа: <https://blog.youtalent.com/review-best-knitting-software-apps-designing-patterns-tracking-progress-connecting-with-knitting-community>. Дата публикации: 21.09.2024 (дата обращения: 26.02.2026)
3. Jegham N. et al. Yolo evolution: A comprehensive benchmark and architectural review of yolov12, yolov11, and their previous versions //arXiv preprint arXiv:2411.00201. – 2024. (дата обращения: 04.01.2026)
4. Ultralytics. Ultralytics YOLO11. Overview // Ultralytics: сайт. URL: <https://docs.ultralytics.com/models/yolo11/> (дата обращения: 07.02.2026)
5. Habr. Python vs Java: кто быстрее и дешевле // Habr. [Электронный ресурс] Режим доступа: https://habr.com/ru/companies/ru_mts/articles/912730/#3 (дата обращения: 06.01.2026)
6. Java Documentation. Types // Oracle. Java Documentation. [Электронный ресурс] Режим доступа: <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html> (дата обращения: 07.01.2026)
7. ONNX Runtime. Get started with ORT for Java // ONNX Runtime. . [Электронный ресурс] Режим доступа: <https://onnxruntime.ai/docs/get-started/with-java.html> (дата обращения: 07.01.2026)

УДК 004.415

М.А. Потапова (4 курс бакалавриата),
С.К. Круглов, к.т.н., доцент

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ МОНИТОРИНГА И УПРАВЛЕНИЯ МИКРОКЛИМАТОМ КОМНАТНЫХ РАСТЕНИЙ

Современное развитие технологий интернета вещей (IoT), микроконтроллерных платформ и методов обработки данных позволяет создавать интеллектуальные системы автоматизации бытовых процессов. Одним из перспективных направлений является автоматизация ухода за комнатными растениями, требующая поддержания оптимальных параметров микроклимата, таких как влажность почвы, температура воздуха, освещённость и влажность воздуха [1].

Существующие решения можно условно разделить на три группы: коммерческие датчики мониторинга растений, комплексные системы автоматизированного полива и пользовательские решения на базе микроконтроллерных платформ. Коммерческие устройства ориентированы преимущественно на сбор данных и информирование пользователя, но не обеспечивают полноценного автоматического управления. Комплексные системы «умного сада» обладают расширенной функциональностью, однако отличаются высокой стоимостью и избыточностью для бытового применения. Пользовательские решения на базе микроконтроллерных платформ обеспечивают гибкость и низкую стоимость, однако чаще всего используют простые пороговые алгоритмы управления.

Таким образом, актуальной задачей является разработка доступной программно-аппаратной системы мониторинга и автоматизированного управления условиями выращивания растений с использованием сетевых технологий и методов интеллектуального анализа данных.

Целью работы является разработка прототипа распределённой программно-аппаратной системы мониторинга и автоматического управления микроклиматом комнатных растений на основе технологий IoT и методов машинного обучения.

Предлагаемая система имеет многоуровневую архитектуру и включает встроенный уровень, серверный уровень и клиентский уровень. Аппаратная подсистема реализована на базе микроконтроллера ESP32, подключённого к датчикам влажности почвы, температуры и влажности воздуха, а также освещённости. В качестве исполнительного устройства используется перистальтический насос, обеспечивающий автоматический полив растений.

Передача телеметрических данных осуществляется по протоколу MQTT. Данные с датчиков передаются на серверную часть системы, где сохраняются в базе данных и

используются для анализа состояния микроклимата. Серверная часть реализована на языке Python и выполняет функции приёма данных, хранения телеметрии, анализа параметров среды и формирования управляющих команд.

В отличие от традиционных систем, использующих фиксированные пороговые значения для активации полива, в разработанной системе применяется модель машинного обучения для прогнозирования необходимости полива растений. В качестве алгоритма классификации используется ансамблевый метод Random Forest [6]. Входными признаками модели являются температура воздуха, влажность воздуха, влажность почвы, освещённость и тип растения.

Экспериментальная апробация разработанной системы проводилась в программной среде с использованием эмуляции датчиков и базы данных SQLite. В процессе тестирования было сформировано более 700 записей телеметрических данных. Данные были разделены на обучающую и тестовую выборки в соотношении 80/20. По результатам эксперимента точность классификации модели составила около 98 %.

Анализ важности признаков показал, что наибольшее влияние на принятие решения оказывает влажность почвы, а также уровень освещённости и тип растения.

Разработанная система демонстрирует возможность интеграции микроконтроллерных устройств, сетевых протоколов передачи данных и методов машинного обучения для создания интеллектуальных IoT-решений. Предложенный подход обеспечивает гибкость архитектуры системы и возможность её дальнейшего масштабирования.

ЛИТЕРАТУРА

1. Tang P., Liang Q., Li H., Pang Y. Application of Internet-of-Things Wireless Communication Technology in Agricultural Irrigation Management: A Review // Sustainability. – 2024. – Vol. 16. – No. 9.
2. Abdelmoneim A. A., Kimaita H. N., Al Kalaany C. M., Derardja B., Dragonetti G., Khadra R. IoT Sensing for Advanced Irrigation Management: A Systematic Review of Trends, Challenges and Future Prospects // Sensors. – 2025.
3. Akshatha Y., Poornima A. S. IoT Enabled WSN and Machine Learning Techniques to Surveillance the Smart Irrigation System // International Journal of Intelligent Systems and Applications in Engineering. – 2023.
4. Aisyah N., Ulhaq E. D., Dharmawan A., Purbakawaca R. Design of an IoT-Based Smart Irrigation System Using Soil Moisture Sensors for Water Efficiency // Journal of Physics: Conference Series. – 2025.
5. Ardiansah I., Bafdal N., Bono A., Suryadi E. An Overview of IoT-Based Intelligent Irrigation Systems for Greenhouse: Recent Trends and Challenges // Journal of Applied Engineering Science. – 2022.
6. Breiman L. Random Forests // Machine Learning. – 2001.

УДК 004.057.4

А.Ф. Скрипкарь (4 курс бакалавриата),
И.В. Шошмина, к.т.н., доцент

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЕРСОНАЛИЗИРОВАННОГО ПОДБОРА ТРЕНИРОВОЧНЫХ ПРОГРАММ НА ОСНОВЕ АНАЛИЗА ПАРАМЕТРОВ ТЕЛА ПОЛЬЗОВАТЕЛЯ

В современном мире всё большее число людей стремится к ведению здорового образа жизни и регулярным физическим нагрузкам. Вместе с тем подбор оптимальной тренировочной программы представляет значительную сложность без привлечения профессионального тренера, поскольку требует учёта индивидуальных физических параметров пользователя, его целей и доступных условий для занятий. Существующие приложения для фитнеса, такие как Nike Training Club, FitBod и Freeletics, предлагают обширные библиотеки упражнений, однако не обеспечивают персонализацию на основе анализа физических параметров тела и не учитывают комплексно условия тренировки [1,2],

Целью настоящей работы является разработка мобильного приложения, обеспечивающего автоматический подбор персонализированных тренировочных программ на основе анализа параметров тела пользователя и его предпочтений относительно условий проведения тренировок.

Для достижения поставленной цели необходимо решение следующих задач.

1. Анализ существующих решений в области персонализированных фитнес-приложений и выявление их ключевых ограничений.
2. Разработка модели классификации типа телосложения пользователя на основе антропометрических параметров (рост, вес, индекс массы тела, тип фигуры).
3. Проектирование алгоритма подбора тренировочных упражнений с учётом физических параметров пользователя и условий проведения занятий (дома, на улице, в спортивном зале).
4. Реализация мобильного приложения с пользовательским интерфейсом для ввода параметров, просмотра рекомендованных программ и отслеживания прогресса.
5. Тестирование прототипа и оценка качества рекомендаций.

Предлагаемое решение представляет собой мобильное приложение, архитектура которого включает три основных компонента. Первый компонент — модуль сбора и анализа параметров пользователя — обеспечивает ввод антропометрических данных и на их основе выполняет классификацию типа телосложения по шкале соматотипов (экторморф, мезоморф, эндоморф) с применением алгоритмов машинного обучения [3,4,6]. Второй компонент — рекомендательный модуль — осуществляет подбор упражнений из структурированной базы данных с учётом выявленного соматотипа, заявленных целей (снижение веса, набор мышечной массы, поддержание физической формы) и выбранных условий тренировки. Третий компонент — пользовательский интерфейс — предоставляет удобные экраны для настройки профиля, просмотра персонального плана занятий и фиксации результатов.

Ключевым отличием разрабатываемого приложения от существующих аналогов является комплексный учёт параметров тела пользователя при формировании тренировочного плана, а не только его субъективных предпочтений. Помимо этого, реализована возможность адаптации программы к различным условиям тренировки, что повышает практическую применимость решения для широкой аудитории пользователей.

В рамках текущего этапа разработана концепция системы и спроектирована её архитектура. Прототип приложения реализован с использованием фреймворка React Native [5], обеспечивающего кроссплатформенную совместимость с операционными системами iOS и Android [5]. Серверная часть построена на основе Python (FastAPI), модель классификации соматотипа обучена на публично доступных датасетах антропометрических измерений [3]. На следующих этапах планируется расширение базы упражнений, интеграция механизма адаптивного обучения на основе обратной связи пользователя, а также проведение пользовательского тестирования.

ЛИТЕРАТУРА

1. Буй Тхи Тху Хьонг, Кузнецов А. А. Персонализированные рекомендательные системы в фитнес-приложениях: обзор подходов // Информационные технологии и системы. – 2022. – № 3. – С. 45–52.
2. Soomro S., Bhutta M. N. M., Khan Z., Tahir M. A. Smart fitness tracker with AI-based workout recommendation // IEEE Access. – 2023. – Vol. 11. – P. 12341–12355. – DOI 10.1109/ACCESS.2023.3241567.
3. Комолов Н. А., Петров И. С. Методы классификации соматотипов человека на основе антропометрических данных с применением алгоритмов машинного обучения // Научно-технический вестник информационных технологий, механики и оптики. – 2023. – Т. 23. – № 2. – С. 310–318.
4. Sharma A., Kumar R. Body composition analysis using deep learning for fitness applications // Journal of Health Informatics. – 2024. – Vol. 16. – № 1. – P. 88–101.

5. Гордеева Е. В., Смирнова О. В. Разработка мобильных приложений для здоровья: архитектурные подходы и практики // Современные технологии в теории и практике программирования: сборник материалов конференции, Санкт-Петербург, 2023 г. – СПб.: ПОЛИТЕХ-ПРЕСС, 2023. – С. 112–115.
6. Howard A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications // arXiv:1704.04861. – 2017.

УДК 004.4

Н.А. Рожнев, Г.Р. Родимов (3 курс бакалавриата),
Е.В. Комарова, ст. преподаватель

АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ И ГРАНИЦЫ ПРИМЕНИМОСТИ ПРОТОКОЛА WEBSOCKET В СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

Современный мир невозможно представить без онлайн-сервисов: мессенджеры, чаты, онлайн игры, биржи и совместные редакторы – все это неотъемлемая часть нашей жизни. Началом развития real-time систем стал протокол WebSocket, который обеспечил постоянное двустороннее соединение (full-duplex) в реальном времени [2]. До появления WebSocket такое соединение было невозможно и имитировалось такими технологиями как: Polling – клиент через фиксированные интервалы отправляет HTTP-запросы, чтобы проверить появились ли новые данные, а сервер отвечает вне зависимости от изменений состояния системы и Long Polling – развитая версия Polling, где сервер не формирует немедленный ответ, а удерживает соединение открытым до появления данных или наступления тайм-аута. Потребность в развитии появилась после того, как веб-страницы стали не просто статичными элементами, а полноценными интерактивными сервисами. Именно поэтому в декабре 2011 года появился стандарт RFC 6455, “The WebSocket Protocol”, который используется и по сей день в создании полномасштабных веб платформ [1]. WebSocket – это протокол прикладного уровня, который в отличие от HTTP-модели “запрос-ответ” устанавливает длительное соединение, в котором обе стороны могут бесперебойно обмениваться пакетами. Соединение начинается с HTTP-запроса, с помощью которого соединение переключается на протокол WebSocket, а конкретно, посылая пакет - Upgrade:WebSocket. Если сервер поддерживает данный протокол, он отвечает статусом 101 Switching Protocols, тем самым подтверждая переход. После перехода соединение становится полнодуплексным и остаётся открытым, но дальнейшая коммуникация происходит в формате WebSocket-кадров. WebSocket для передачи данных использует протокол транспортного уровня TCP.

Спецификация протокола WebSocket содержит управляющие кадры ping, pong, close. Ping и pong используются как сигналы активности, чтобы соединение не было закрыто по тайм-ауту. Фрейм close используется для корректного завершения сессии. Для защиты прозрачных прокси-серверов от отравления хеша, данные клиента побитово шифруются алгоритмом XOR. Однако у соединения существует проблема обратного давления (backpressure), которая требует контроля в практической эксплуатации: так как стандарт не имеет встроенного уровня управления окном передачи данных, то в ситуации, когда отправка пакетов происходит быстрее, чем ее успевает обработать сервер, TCP-буферы неконтролируемо растут, что ведет к утечкам памяти. Для этого разработчикам необходимо вручную вводить лимиты очередей. Кроме того, в случаях горизонтального масштабирования приходится использовать внешних брокеров (Redis, RabbitMQ, и прочие), так как каждое соединение монополюно удерживает дескриптор [4].

Что касается безопасности и сетевой инфраструктуры – эти задачи становятся отдельным пунктом, необходимым для понимания работы WebSocket соединения. Спецификация не регламентирует механизмов авторизации, поэтому аутентификация происходит на этапе рукопожатия или в первом переданном фрейме. Браузеры не применяют политику Same-Origin к веб-сокетами и не отправляют предварительные CORS запросы, обязывая бэкенд

валидировать заголовок Origin для защиты от CSWSH – межсайтового перехвата [3]. Традиционные балансировщики L7 требуют дополнительной настройки тайм-аутов для маршрутизации долгоживущих сессий. Кроме того, усложняется деплой: обновление узла требует graceful shutdown – алгоритма мягкого отключения для тысяч активных соединений с последующим плавным переподключением клиентов.

Благодаря технологии соединения WebSocket, был реализован архитектурный фундамент современных real-time систем, который до сих пор является стандартом двустороннего соединения с огромной экосистемой – на его основе появились брокеры сообщений, серверные библиотеки (Socket.io, ws, Netty), облачные шлюзы, элементы мониторинга. Несмотря на то, что в наши дни уже существуют альтернативные решения, направленные на устранение недостатков WebSocket, они не могут заменить его, а лишь закрывают узконаправленные сценарии. SSE упрощает однонаправленную трансляцию там, где двусторонний канал избыточен, однако не поддерживает двунаправленность. WebTransport предлагает мультиплексированные потоки поверх QUIC для экстремальных требований к задержкам (онлайн-игры, VR, стриминг в 4K), однако все еще формирует свою экосистему и жестко привязан к HTTP/3. Таким образом, WebSocket с 2011 года по сей день является незаменимым стандартом двунаправленного соединения.

Таблица 1 – Сравнение технологий обмена данными в системах реального времени

Технология	Суть работы	Задержка	Нагрузка на сервер	Идеальный сценарий использования	Недостатки
HTTP Long Polling	Удержание сервером HTTP-запроса до появления новых данных, затем отправка ответа и повторное подключение	Средняя / высокая	Высокая: постоянная переустановка TCP-соединений, повторный разбор заголовков	Поддержка устаревших систем, где закрыты порты или недоступны новые протоколы	Большой оверхед на заголовки, лишний сетевой трафик
WebSocket	Единый постоянный двусторонний канал поверх TCP после HTTP-рукопожатия	Низкая	Средняя: удерживаются файловые дескрипторы, требуется контроль буферов	Симметричный полнодуплексный обмен: чаты, мультиплеерные игры, биржи, редакторы	Backpressure, head-of-line blocking, необходимость внешней инфраструктуры для масштабирования
Server-Sent Events (SSE)	Однонаправленная потоковая передача текстовых данных поверх HTTP	Низкая	Низкая / средняя: работает в рамках стандартного HTTP-соединения	Передача данных только от бэкенда к фронтенду: котировки, ленты новостей, уведомления	Передача данных только в одну сторону: сервер → клиент
WebTransport	Мультиплексированные потоки и датаграммы поверх протокола QUIC (UDP)	Минимальная	Оптимизированная: снижена нагрузка за счёт отсутствия блокировок TCP	Интенсивный обмен данными с экстремально низкими задержками в нестабильных сетях	Жёсткая привязка к HTTP/3; экосистема инфраструктуры пока находится в стадии формирования

ЛИТЕРАТУРА

1. WebSocket Protocol (RFC 6455) [Электронный ресурс] Режим доступа: <https://datatracker.ietf.org/doc/html/rfc6455>
2. Mozilla Developer Network. WebSocket API [Электронный ресурс] Режим доступа: https://developer.mozilla.org/ru/docs/Web/API/WebSockets_API
3. WHATWG WebSockets Standard [Электронный ресурс] Режим доступа: <https://websockets.spec.whatwg.org/>
4. Ably. WebSockets and real-time communication explained [Электронный ресурс] Режим доступа: <https://ably.com/topic/websockets>

УДК 004.42

С.Ю. Рышкова (2 курс магистратуры),
О.Г. Малеев, к.т.н., доцент

ИНТЕЛЛЕКТУАЛЬНЫЙ МОДУЛЬ ПЛАНИРОВАНИЯ ПИТАНИЯ НА ОСНОВЕ АНАЛИЗА ПРОДУКТОВЫХ ЗАПАСОВ И ПРЕДПОЧТЕНИЙ ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

Масштаб продовольственных потерь приобретает характер глобального вызова. По данным Продовольственной и сельскохозяйственной организации ООН около 13,2% продовольствия теряется в период от сбора урожая до розницы, а ещё 19% выбрасывается на этапе реализации и в домохозяйствах. Парадокс в том, что на фоне миллиардов тонн выброшенной продукции около 733 млн человек в мире продолжают страдать от голода [1].

Данная тенденция прослеживается и в России: исследования в четырёх крупных городах показывают, что среднестатистическая семья выбрасывает 82,4 кг продуктов в год [2]. В масштабах страны домохозяйства генерируют не менее 4,37 млн тонн отходов ежегодно, и наибольшие потери (24–26%) фиксируются именно для скоропортящихся продуктов.

Одним из способов снижения объёма пищевых отходов является использование цифровых систем планирования питания, позволяющих формировать рацион с учётом имеющихся продуктов. Мобильные приложения могут автоматически анализировать продуктовые запасы пользователя и рекомендовать рецепты, позволяющие максимально эффективно использовать доступные ингредиенты.

В рамках работы разрабатывается интеллектуальный модуль рекомендаций рецептов и планирования питания, предназначенный для мобильного приложения. Модуль выполняет анализ имеющихся продуктов, формирует список подходящих рецептов и генерирует план питания с учётом предпочтений пользователя.

Мобильное приложение реализуется на языке программирования Dart [3] с использованием фреймворка Flutter [4], позволяющего разрабатывать кроссплатформенные мобильные приложения для Android и iOS. Хранение данных осуществляется с использованием облачной платформы Firebase [5], обеспечивающей синхронизацию пользовательских данных и доступ к базе рецептов.

Архитектура программного модуля реализована на основе принципов Clean Architecture [6]. В соответствии с данным подходом система разделена на три логических слоя (рисунок 1).

Слой представления реализует пользовательский интерфейс мобильного приложения. На данном уровне расположены экраны рекомендаций рецептов, генерации рецептов и формирования плана питания. Доменный слой содержит бизнес-логику интеллектуального модуля. В данном слое реализованы алгоритмы анализа ингредиентов, формирования рекомендаций рецептов, генерации рецептов и построения плана питания.

Предлагаемый модуль реализует алгоритм сопоставления ингредиентов с рецептами, хранящимися в базе данных Firebase. На первом этапе анализируются продуктовые запасы пользователя на основе данных о продуктах, добавляемых в систему при сканировании чеков и штрих-кодов, после чего формируется множество доступных ингредиентов. Далее

происходит фильтрация рецептов с учётом пользовательских предпочтений и ограничений, которые задаются пользователем в интерфейсе приложения (например, исключённые ингредиенты или тип питания). Затем рассчитывается степень соответствия рецепта имеющимся продуктам, на основе которой формируются рекомендации по рациону. Для повышения корректности сопоставления используется этап нормализации ингредиентов, при котором различные текстовые формы одного продукта приводятся к единому представлению.

Дополнительно реализован механизм генерации рецептов на основе доступных ингредиентов. Генерация выполняется с использованием языковой модели платформы Hugging Face [7], которая формирует текст рецепта и последовательность шагов приготовления блюда.

Также модуль выполняет формирование плана питания, при котором из набора рекомендованных рецептов составляется рацион на день или неделю с учётом предпочтений пользователя и максимального использования доступных ингредиентов. Построение плана питания на заданный период осуществляется с использованием алгоритма на основе рекуррентной нейронной сети с архитектурой долгой краткосрочной памяти (LSTM) [8].

Слой данных обеспечивает взаимодействие системы с внешними источниками данных. В данном слое реализованы репозитории рецептов и пользовательских предпочтений, выполняющие доступ к базе данных и внешним сервисам.

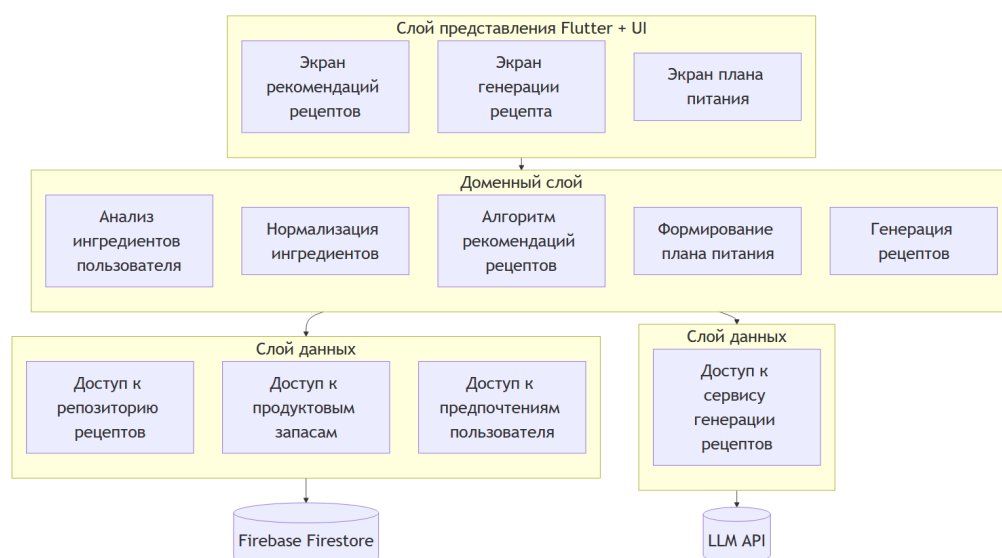


Рисунок 1 – Архитектурный шаблон системы

Базовая версия интеллектуального модуля включает функциональность рекомендаций рецептов, их генерации и планирования питания. На данный момент реализованы механизм нормализации наименований продуктов, алгоритм ранжирования рецептов по коэффициенту покрытия доступных продуктов, модули генерации рецептов и формирования рациона.

Тестирование планируется проводить на устройствах Android. Будут проверены пользовательские сценарии: получение рекомендаций рецептов на основе текущих продуктов, генерация рецепта и построение плана питания на заданный период. По результатам тестирования предполагается оптимизация алгоритмов и подготовка к сборке под iOS.

ЛИТЕРАТУРА

1. FAO. Statistical Yearbook 2025: World Food and Agriculture. – Rome: Food and Agriculture Organization of the United Nations, 2025. – DOI: 10.4060/cd4313en.
2. Бондарчук О. Н., Ульрих Е. В., Федоров Д. Е., Соснина О. Б. Количественная и качественная оценка пищевых отходов, образующихся в домохозяйствах // Инновации и продовольственная безопасность. – 2024. – № 4(46). – С. 9–16.
3. Dart programming language | Dart. [Электронный ресурс]. Режим доступа: <https://dart.dev/> (дата обращения: 01.03.2026).

4. Flutter – Build apps for any screen. [Электронный ресурс]. Режим доступа: <https://flutter.dev/> (дата обращения: 01.03.2026).
5. Firebase | Google’s Mobile and Web App Development Platform. [Электронный ресурс]. Режим доступа: <https://firebase.google.com/> (дата обращения: 05.03.2026).
6. Atuncar Yataco C. P., Lopez Salinas M. A., Herrera Quispe J. A. Mobile Application Based on Clean Architecture for Sustainable Tourism Recommendation Using BERT // International Journal of Advanced Computer Science and Applications (IJACSA). – 2025. – Vol. 16, No. 11. – P. 1032–1039. – DOI: 10.14569/IJACSA.2025.0161199.
7. Hugging Face – The AI community building the future. [Электронный ресурс]. Режим доступа: <https://huggingface.co/> (дата обращения: 07.03.2026).
8. Чигасова М. А. Прогнозирование котировок акций с использованием нейронной сети / М. А. Чигасова, О. Г. Малеев // Современные технологии в теории и практике программирования : Сборник материалов конференции, Санкт-Петербург, 26 апреля 2022 года. – СПб., 2022. – С. 180–182. – EDN GRFEDM.

УДК 004.453

А.К. Селиверстова (4 курс бакалавриата),
К.А. Туральчук, к.т.н., доцент

РАЗРАБОТКА МОДУЛЯ ДЛЯ ПРОВЕРКИ УЧЕБНЫХ ЗНАНИЙ НА ОСНОВЕ LLM

Массовая цифровизация образования и рост рынка EdTech усиливают спрос на автоматизированные системы контроля знаний. По оценкам отраслевых источников, мировой рынок генеративного ИИ в образовании и образовательных технологий в целом демонстрирует устойчивый рост: ResearchAndMarkets оценивает масштаб в 0,36 млрд долл. в 2024 г., 0,53 млрд долл. в 2025 г. и 2,24 млрд долл. к 2029 г. с CAGR 43,7% [1]. Российский рынок образовательных технологий по оценкам BusinessStat, в 2024 вырос на 21% и составил 149 млрд руб. [2]. В совокупности эти факторы усиливают внедрение генеративных ИИ решений в образовании [3].

Традиционное ручное составление вопросов и использование шаблонных тестов плохо масштабируется и требует значительных трудозатрат. В этих условиях большие языковые модели (LLM) открывают возможности для автоматической генерации вопросов [4], в том числе с опорой на таксономию Блума [5], однако их применение ограничивается рисками галлюцинаций и логической несогласованности. Исходя из этого, перспективным решением становится использование подхода Retrieval-Augmented Generation (RAG), обеспечивающего привязку генерации к материалам курса и повышение качества контроля знаний.

Целью работы является разработка модуля для проверки учебных знаний на основе LLM. Для достижения этой цели необходимо решение следующих задач:

1. Анализ предметной области: исследование современных методов контроля знаний и выявление их ограничений, а также анализ возможностей LLM для их автоматизации.
2. Обзор существующих решений для генерации вопросов и оценки знаний.
3. Выбор инструментальных средств для задач генерации вопросов и проверки знаний.
4. Проектирование архитектуры системы.
5. Программная реализация диалоговой системы, включающей загрузку практической работы студента, формирование контекста, генерацию вопросов, ориентированных на практические знания студента.
6. Тестирование разработанной системы.

С учетом поставленных задач и сформулированных требований была разработана архитектура системы, обеспечивающая диалоговый формат контроля знаний на основе LLM и RAG. Модуль ориентирован на проверку понимания практических аспектов дисциплины, основываясь на материалах курса и загружаемой работе студента.

Разрабатываемая система включает пользовательский плагин для Moodle, серверное приложение, RAG-подсистему, а также компоненты хранения данных. Плагин выступает точкой взаимодействия пользователя с системой: через него преподаватель загружает материалы курса и получает результаты, а студент загружает работу и отвечает на сформированные вопросы.

Серверное приложение, реализованное на FastAPI [6], отвечает за обработку запросов от клиентского модуля, управление пользовательскими сессиями, генерацию вопросов, оценку ответов и координацию обращения к RAG-подсистеме и внешней языковой модели.

Для повышения точности генерации и снижения вероятности галлюцинаций в архитектуру включена RAG-подсистема. Ее назначение состоит в поиске релевантных фрагментов в материалах курса и работе студента с последующим включением найденного контекста в запрос к языковой модели.

В качестве внешнего LLM-провайдера выступает YandexGPT, выбранный с учётом доступности, хорошей поддержки русского языка и возможности интеграции через API. Языковая модель используется как для генерации вопросов, так и для оценки ответов студента, а также формирования обратной связи и определения дальнейшего сценария диалога.

Архитектура системы адаптивного контроля знаний представлена на рисунке 1.

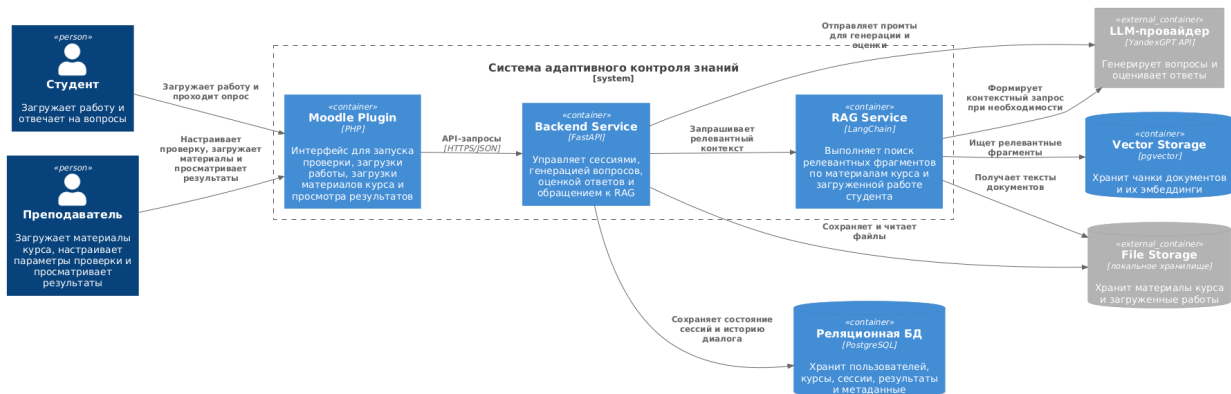


Рисунок 1 – Архитектура системы адаптивного контроля знаний

Результатом реализации является прототип системы, обеспечивающий гибкий диалоговый формат контроля знаний, при котором вопросы формируются с учетом материалов учебного курса и работы студента. Включение RAG-подсистемы позволяет повысить релевантность и содержательную точность генерации, снизить вероятность фактических ошибок и обеспечить тесную привязку проверки к целям конкретной дисциплины.

При этом разрабатываемая система не рассматривается как средство полной автоматизации контроля знаний и замены преподавателя. Ее назначение состоит в поддержке преподавателя на этапе первичной проверки и предварительного анализа работы студента. Такой подход позволяет сократить рутинную нагрузку, сохраняя за преподавателем принятие итогового оценочного решения.

ЛИТЕРАТУРА

1. Research and Markets. Generative AI in Edtech Market Report 2025. — 2025. — URL: <https://www.researchandmarkets.com/reports/6035186/generative-ai-in-edtech-market-report> (дата обращения: 18.10.2025).
2. РБК Маркетинг. Анализ рынка образовательных технологий (EdTech) в России: итоги 2024 года и тенденции 2025 года. — 2025. — URL: <https://marketing.rbc.ru/articles/15521/> (дата обращения: 18.10.2025).
3. EdTechs. Главные тренды EdTech-рынка в 2024 году: исследование от СберОбразования и Smart Ranking. — URL: <https://edtechs.ru/analitika-i-intervyu/glavnye-trendy-edtech-rynka-v-2024-godu-issledovanie-ot-sberobrazovaniya-i-smart-ranking/> (дата обращения: 22.10.2025).

4. НИУ ВШЭ. Новые подходы к оцениванию: искусственный интеллект как драйвер изменений в образовании. — 2025. — URL: <https://publications.hse.ru/books/1085560416> (дата обращения: 21.10.2025).
5. Scaria N., Chenna S. D., Subramani D. Automated Educational Question Generation at Different Bloom's Skill Levels using Large Language Models: Strategies and Evaluation // Lecture Notes in Computer Science. — 2024. — URL: https://link.springer.com/chapter/10.1007/978-3-031-64299-9_12 (дата обращения: 28.10.2025).
6. JetBrains PyCharm Team. Django, Flask, FastAPI: Which Python Backend Framework to Choose. — 2025. — URL: <https://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi/> (дата обращения: 09.11.2025).

УДК 004.514

М.А. Сидоркина (4 курс бакалавриата),
А.М. Хахина, д.т.н., профессор

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ С АДАПТИВНЫМ ИНТЕРФЕЙСОМ ДЛЯ ЧТЕНИЯ, ОРИЕНТИРОВАННОГО НА ПОЛЬЗОВАТЕЛЕЙ С СИНДРОМОМ ДЕФИЦИТА ВНИМАНИЯ И ГИПЕРАКТИВНОСТИ

Современные цифровые образовательные и информационные платформы преимущественно представляют контент в текстовом формате, что требует от пользователя длительного поддержания концентрации внимания. Для людей с синдромом дефицита внимания и гиперактивности данный процесс сопровождается значительными трудностями, связанными с высокой отвлекаемостью, сниженной устойчивостью внимания и ограничениями рабочей памяти [1, 2].

Актуальность проблемы обусловлена масштабом распространенности синдрома дефицита внимания и гиперактивности и растущей ролью цифровых платформ в образовании и информационном поиске. Традиционные веб-интерфейсы, как правило, статичны и не учитывают динамику когнитивных процессов [3, 4]. Перспективным направлением является использование технологий трекинга взгляда, позволяющих в реальном времени анализировать визуальное внимание пользователя и на основе этого изменять параметры отображения контента [5].

Целью данной работы является разработка веб-приложения для чтения с динамической адаптацией интерфейса на основе трекинга взгляда, ориентированного на пользователей с СДВГ, и экспериментально оценить его эффективность.

Для достижения поставленной цели необходимо решение следующих *задач*:

1. Исследование особенностей восприятия и поведения пользователей с синдромом дефицита внимания и гиперактивности в веб-среде.
2. Поиск и анализ существующих подходов, инструментов и технологий для поддержки пользователей с особенностями восприятия и нарушениями концентрации внимания.
3. Разработка архитектуры веб-приложения, которое обеспечивает среду для чтения, реализует отслеживание взгляда пользователя и предоставляет обратную связь пользователю в реальном времени.
4. Проведение сравнительного анализа и обоснование выбора инструментальных средств для реализации прототипа веб-приложения.
5. Разработка прототипа веб-приложения, включающего пользовательский интерфейс для чтения, систему трекинга взгляда, механизм детекции снижения внимания и набор визуальных адаптаций.
6. Проведение тестирования разработанного прототипа и оценка его эффективности в условиях реального использования.

В рамках работы проведен обзор существующих технологических средств для трекинга взгляда, включая аппаратные и программные решения. В качестве технологии отслеживания

взгляда была выбрана библиотека WebGazer.js, работающая непосредственно в браузере и не требующая установки специализированного оборудования. Внедрение технологии в приложение реализовано через пакет `@webgazer-ts/core`. Пакет `@webgazer-ts/core` предоставляет типизированную обертку над WebGazer.js, он обеспечивает тот же уровень точности и производительности, но при этом устраняет необходимость ручного объявления типов [6].

Архитектура приложения представлена на рисунке 1. Система построена по принципу разделения ответственности: модуль трекинга взгляда инкапсулирует работу с браузерным API и камерой, модуль процессора производит математическую интерпретацию движений взгляда. Связь между слоями организована через GazeService – это единственная точка входа для пользовательского интерфейса.

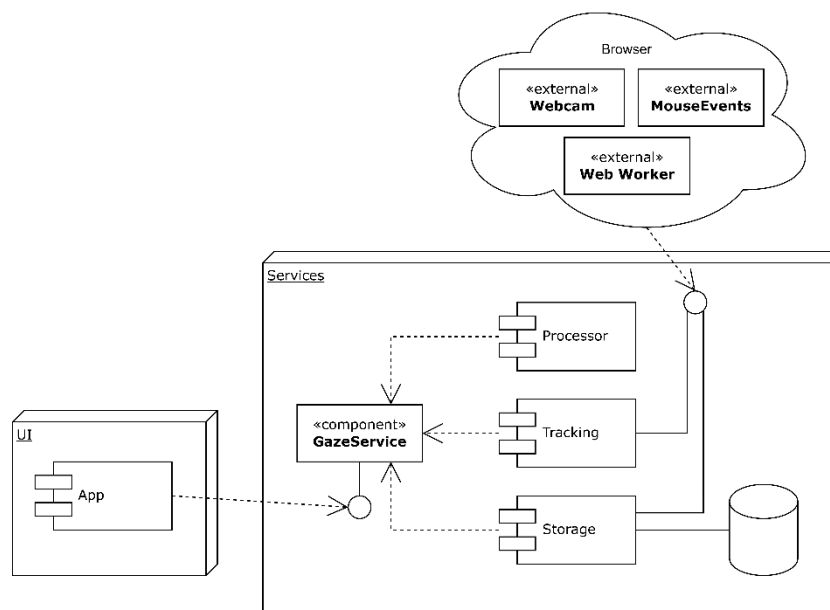


Рисунок 1 – Архитектура приложения

Динамические адаптации в контексте разрабатываемой системы представляют из себя визуальные эффекты, возникающие в ответ на различные типы движений взгляда. Условия отрисовки визуальных эффектов формализованы в виде правил. При проектировании учитывались требования к инклюзивности в соответствии со стандартом WCAG [7]: минимальное количество эффектов, их плавность, возможность полного их выключения и другие.

Клиентское приложение *реализовано* на языке TypeScript. Выбор обусловлен необходимостью обеспечения надежности и масштабируемости приложения. Статическая типизация упрощает сопровождение проекта, повышает читаемость кода и позволяет снизить количество ошибок при интеграции и настройке трекинга взгляда. Для разработки пользовательского интерфейса использовалась библиотека React. React позволяет создать интерактивную и высоко производительную среду для чтения, что особенно важно при реализации адаптивных элементов интерфейса.

ЛИТЕРАТУРА

1. Gabarron E., Denecke K., Lopez-Campos G. Evaluating the evidence: a systematic review of reviews of the effectiveness and safety of digital interventions for ADHD // BMC Psychiatry. – 2025. – Vol. 25. – No 1. – P. 414. – DOI 10.1186/s12888-025-06825-0. – PMID 40264083. – PMCID PMC12016436.
2. Le Cunff A.-L., Martis B.-L., Glover C., Ahmed E., Ford R., Giampietro V., Dommett E. J. Cognitive load and neurodiversity in online education: a preliminary framework for educational research and policy // Frontiers in Education. – 2025. – Vol. 9. – Art. 1437673. – DOI 10.3389/educ.2024.1437673.

3. Heumader P., Miesenberger K., Murillo-Morales T. Adaptive User Interfaces for People with Cognitive Disabilities within the Easy Reading Framework // Computers Helping People with Special Needs. – 2020. – Vol. 12377. – P. 53-60. – DOI 10.1007/978-3-030-58805-2_7. – PMCID PMC7479802.
4. Carrera-Rivera A., Larrinaga F., Lasa G., et al. AdaptUI: A Framework for the development of Adaptive User Interfaces in Smart Product-Service Systems // User Modeling and User-Adapted Interaction. – 2024. – Vol. 34. – P. 1929-1980. – DOI 10.1007/s11257-024-09414-0.
5. Tran T., Lee H.-N., Park J.H. Discovering Accessible Data Visualizations for People with ADHD // Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24). – New York: ACM, 2024. – Art. 64. – P. 1-19. – DOI 10.1145/3613904.3642112.
6. WebGazer. [Электронный ресурс] Режим доступа: <https://github.com/brownhci/WebGazer> (дата обр. 17.03.2026).
7. World Wide Web Consortium (W3C). Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation 06 May 2025. [Электронный ресурс] Режим доступа: <https://www.w3.org/TR/WCAG21> (дата обр. 17.03.2026).

УДК 004.453

К.А. Скорина (2 курс магистратуры),
С.А. Молодяков, д.т.н., профессор

РАЗРАБОТКА РЕДАКТОРА МЕДИАФАЙЛОВ С ИНТЕГРИРОВАННЫМ NLP-МОДУЛЕМ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОГО ВЗАИМОДЕЙСТВИЯ

Современные инструменты видеомонтажа предоставляют широкие функциональные возможности, однако их освоение требует значительных временных затрат, а интерфейс зачастую перегружен элементами управления. Пользователь, не обладающий профессиональными компетенциями, сталкивается с трудностями при попытке выполнить типовые операции: обрезка, сжатие, изменение разрешения. Существующие решения с элементами искусственного интеллекта ограничены узким кругом задач и не предлагают унифицированного диалогового взаимодействия. Целью настоящей работы является разработка архитектуры и создание прототипа редактора медиафайлов, оснащенного NLP-модулем, способным интерпретировать запросы на естественном языке, генерировать исполняемые команды FFmpeg[1], предоставлять контекстные инструкции по работе с интерфейсом и автономно выполнять операции редактирования.

В данной работе рассматривается фундаментальный вопрос: как спроектировать интеллектуального ассистента, который объединяет взаимодействие на естественном языке с функциями редактирования медиа, учитывая при этом пользователей с разным уровнем экспертизы и различными предпочтениями во взаимодействии? Мы предлагаем архитектуру мультимодального ИИ-помощника, работающую в трёх взаимодополняющих режимах:

1. Режим ассистента: предоставляет пошаговые руководства с визуальной подсветкой элементов интерфейса, позволяя начинающим пользователям осваивать рабочие процессы редактирования через объяснения на естественном языке.

2. Автоматический режим: непосредственно выполняет операции редактирования на основе требований пользователя, давая опытным пользователям возможность обходить ручную навигацию и выполнять задачи с помощью разговорных команд.

3. Режим эксперта FFmpeg: генерирует исполняемые инструкции командной строки FFmpeg для продвинутых пользователей, которым требуется точный контроль или возможности пакетной обработки.

Основные вклады данной работы включают:

– Единую архитектуру, объединяющую понимание естественного языка, управление диалогом и планирование действий для приложений редактирования медиа.

– Модульную систему классификации намерений и заполнения слотов, обученную на предметно-ориентированных сценариях редактирования.

- Осведомлённый о контексте менеджер диалога, поддерживающий состояние сеанса при взаимодействии.
- Расширяемый планировщик действий, преобразующий запросы пользователя в исполняемые операции через API редактирования медиа.
- Комплексную оценку, демонстрирующую значительное повышение эффективности выполнения задач и удовлетворённости пользователей.

Рисунок 1 иллюстрирует общую архитектуру системы. На этой схеме показана базовая архитектура OpenShot Cloud API[2]. Сервер принимает команды для редактирования видео с помощью HTTP REST API и хранит данные проекта в локальной базе данных PostgreSQL[3]. Когда видео нужно экспортировать или обработать, сервер отправляет эти задачи доступному воркеру. Воркер обрабатывает каждый видеопроjekt (по одному за раз), создавая финальное видео. После завершения работы воркер может отправить данные на URL вебхука, загрузить видео в облачное хранилище или загрузить видеофайл вручную. Все серверы и рабочие узлы обмениваются файлами по протоколу HTTP для доступа к исходным ресурсам (например, видео, аудио, файлам изображений) и для обмена готовым видео. Всю архитектуру можно запустить на одном экземпляре виртуальной машины или, при желании, разделить рабочие узлы (что позволит автоматически масштабировать пул рабочих узлов). К поддерживаемым платформам развёртывания системы относятся AWS, Azure и Google Cloud[4].

Разрабатываемый подход реализован в видеоредакторе OpenShot[5] в качестве расширения для приложения.

Подход был реализован на языке Python[6] - языке разработки OpenShot.

Один из способов абстрагироваться от аппаратной части объектной системы хранения упаковать программные сервисы в контейнеры Docker[7]. Уменьшив зависимость между программным обеспечением и платформой, мы упростим разработку, тестирование и позволим развёртывать её на практически любом наборе аппаратных ресурсов.

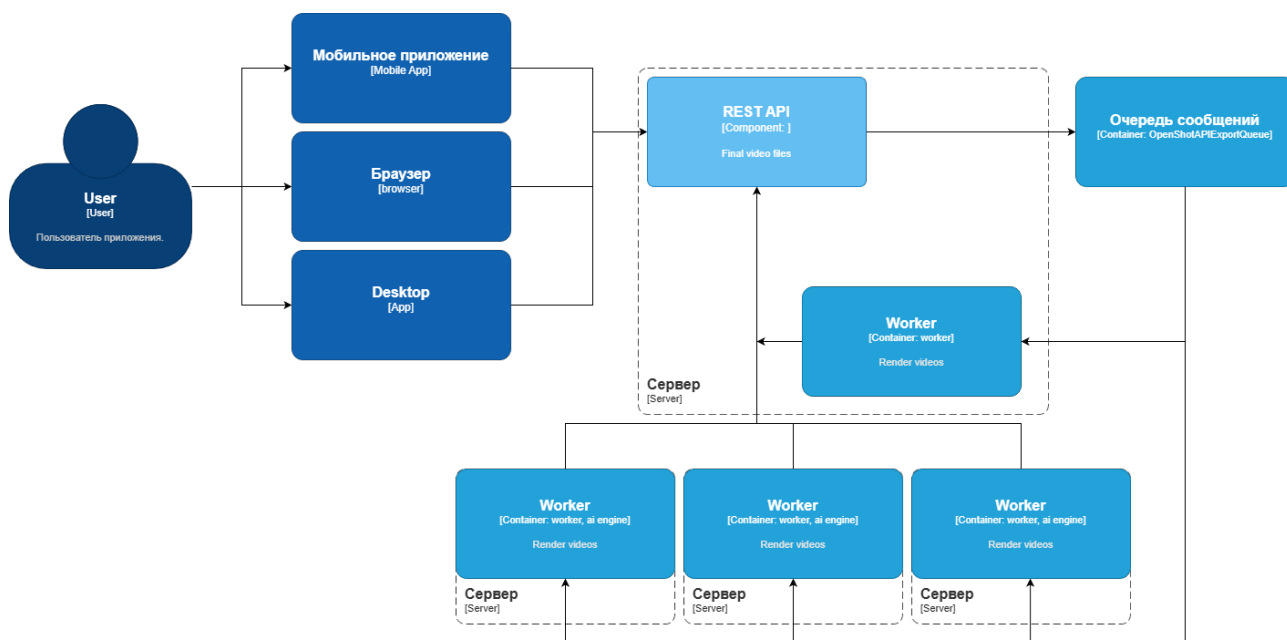


Рисунок 1 – Архитектура системы

ЛИТЕРАТУРА

1. Молодяков С. А. Применение функций FFmpeg в мультимедийных приложениях (100 примеров на Python). СПб. : ПОЛИТЕХ-ПРЕСС. – 2023. – 514 с. DOI 10.18720/SPBPU/2/i23-44
2. OpenShot Cloud API. [Электронный ресурс] - <https://cloud.openshot.org/doc/index.html>
3. PostgreSQL [Электронный ресурс] - <https://www.postgresql.org/docs/>
4. Google Cloud [Электронный ресурс] - <https://cloud.google.com/>

5. Developers: Openshot [Электронный ресурс] - <https://www.openshot.org/static/files/user-guide/developers.html/>
6. Python Documentation. [Электронный ресурс] - <https://docs.python.org/3/index.html>
7. James Turnbull. “The Docker Book” Version v17.03.0 – 2017. – 400pp.

УДК 004.4

В.А. Спирчина (2 курс, магистратура),
П.Д. Дробинцев, к.т.н., доцент

РАЗРАБОТКА IOS-СУПЕРАППА ДЛЯ ИНТЕГРАЦИИ МЕДИА-СЕРВИСОВ

Выход на новый рынок с уже сложившейся экосистемой медиа-продуктов ставит перед компанией задачу быстрого привлечения и удержания максимально широкой аудитории. В этих условиях суперапп рассматривается как инструмент для объединения различных развлекательных сервисов в одном мобильном приложении: музыка, фильмы, сериалы и игры становятся доступны через единую точку входа с общей авторизацией и подпиской. Это позволяет агрегировать аудиторию разных ниш и наращивать общее число активных пользователей.

Целью данной работы является разработка iOS-супераппа с модульной архитектурой, позволяющей интегрировать независимые миниприложения (далее миниаппы) в виде внешних SDK.

Суперапп выступает хост-платформой для внешних медиа-сервисов [2]. Итоговая архитектура супераппа имеет модульную структуру. Хост часть включает в себя:

- модуль авторизации и управления пользовательской сессией;
- модуль навигации (таб-бар, роутинг по разделам и обработка диплинков);
- модуль управления подписками и проверкой прав доступа;
- модуль конфигурации и фича-флагов (включение/отключение миниаппов, экспериментальные функции);
- модуль аналитики и логгирования;
- модуль управления аудиофокусом, координирующий воспроизведение звука между миниаппами и системными событиями;
- модуль-контейнер, отвечающий за загрузку, инициализацию и отображение внешних миниаппов.

Миниаппы поставляются в виде внешних SDK (в формате Swift Package). Каждое миниприложение реализует единый контракт взаимодействия с супераппом, определенный в виде протоколов:

- метод инициализации с контекстом (пользовательский токен, язык интерфейса, информация о подписке);
- предоставление корневого экрана для встраивания в навигацию супераппа (UIViewController или SwiftUI View);
- декларация требований к подписке и необходимым разрешениям;
- интерфейс для отправки событий (например, для аналитики или запроса аудиофокуса) в ядро супераппа.

Суперапп не знает внутреннего устройства конкретного миниаппа и работает только через объявленный протокол-контракт. Это позволяет подключать, обновлять и при необходимости отключать отдельные миниаппы без изменения кода ядра, ограничиваясь управлением конфигурацией.

В качестве основного языка программирования используется Swift [3]. Для описания пользовательского интерфейса применяется SwiftUI [4]. Внутри модулей используется архитектурный шаблон Model–View–ViewModel (MVVM) [1], позволяющий разделить бизнес-логику и представление и упростить модульное тестирование.

Для организации модульности и подключения внешних SDK-миниаппов использован Swift Package Manager, что позволяет изолировать зависимости, управлять версиями модулей и интегрировать их в общий проект по единым правилам. Асинхронные операции (загрузка конфигурации, проверка подписки, ленивая подгрузка миниаппов) выполняются с использованием Grand Central Dispatch и механизма `async/await`.

Таким образом, разработанная модульная архитектура iOS-супераппа демонстрирует промышленный подход к созданию платформы для интеграции независимых медиа-сервисов. Она позволяет одновременно решать бизнес-задачу по наращиванию аудитории на новом рынке и инженерную задачу по масштабируемому развитию сложного программного продукта.

ЛИТЕРАТУРА

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2020. – 448 с.
2. Unleashing the Power of Super App Architecture [Электронный ресурс]. – Режим доступа: <https://virtualspirit.me/insights/133/unleashing-the-power-of-super-app-architecture> (дата обращения: 09.03.2026).
3. The Swift Programming Language (Swift 5.9) [Электронный ресурс] // Apple Developer Documentation. – Режим доступа: <https://developer.apple.com/swift/> (дата обращения: 09.03.2026).
4. SwiftUI [Электронный ресурс] // Apple Developer. – Режим доступа: <https://developer.apple.com/xcode/swiftui/> (дата обращения: 09.03.2026).
5. Newman S. Building Microservices. Designing Fine-Grained Systems. – Beijing; Boston: O'Reilly Media, 2021. – 428 p.
6. Organizing Your Code with Frameworks and Packages [Электронный ресурс] // Apple Developer Documentation. – Режим доступа: https://developer.apple.com/documentation/swift_packages (дата обращения: 09.03.2026).

УДК 004.453

В.А. Спиридонов (4 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

РАЗРАБОТКА МОДУЛЯ ERP-СИСТЕМЫ НА ЯЗЫКЕ SCALA С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА GLOBAL

В условиях современной экономики и политики импортозамещения перед крупными промышленными предприятиями России остро встает вопрос перехода на отечественное программное обеспечение. Уход с рынка таких гигантов, как SAP и Oracle, создает потребность в развитии функциональности локальных ERP-платформ. Однако классические учетные системы зачастую не обладают встроенными механизмами для сложной прогнозной аналитики, потоковой обработки данных и гибкой микросервисной архитектуры, что затрудняет цифровую трансформацию предприятий и требует разработки дополнительных высокотехнологичных модулей [2].

Таким образом, целью данной работы является разработка модуля для российской ERP-системы Global с использованием языка Scala, который обеспечит обработку больших данных, гибкую интеграцию и отказоустойчивость.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Исследовать существующие ERP-решения (SAP, Oracle, 1C), включая анализ их архитектуры, функциональности и выявление преимуществ и недостатков для российского рынка.
2. Провести обоснование выбора языка программирования и платформы разработки путем сравнительного анализа технологических стеков.
3. Спроектировать архитектуру модуля, обеспечивающую его интеграцию с ядром Global ERP через REST API и событийную шину.

4. Реализовать core-модуль на Scala с использованием фреймворка Global.

В ходе работы были проанализированы существующие на рынке решения, такие как SAP ERP [4], Oracle Fusion Cloud ERP [1] и 1C:ERP [3], а также проведен сравнительный анализ их функциональности. Рассмотрение флагманских решений показало, что, хотя SAP и Oracle предлагают мощную аналитику (особенно на платформе HANA и с встроенным AI), их стоимость и сложность внедрения остаются критическими недостатками для многих организаций в текущих реалиях. В то же время платформа 1C:ERP, доминирующая на локальном рынке, испытывает сложности с производительностью при обработке действительно больших массивов данных и реализации сложных производственных алгоритмов.

С учетом выявленных рыночных ниш и технологических вызовов был сформирован список требований к разрабатываемому модулю. Во-первых, архитектура модуля должна быть гибридной: стабильное ядро ERP дополняется микросервисами для аналитики. Во-вторых, критически важным является обеспечение отказоустойчивости и производительности при работе с большими данными в реальном времени, что невозможно без современного технологического стека.

После выявления основных требований был осуществлен выбор технологий для реализации разрабатываемого модуля.

В качестве основного языка программирования был выбран язык Scala. Сравнительный анализ с традиционным для корпоративной среды языком Java показал, что, несмотря на более высокий порог вхождения, выразительность Scala и его мощная система типов позволяют писать более надежный и лаконичный код для сложной бизнес-логики. Кроме того, экосистема Scala является лидирующей в области обработки данных (Apache Spark, Akka Streams), что напрямую соответствует задаче разработки аналитических модулей.

Для разработки бизнес-логики и интеграции с базовой платформой был выбран Global Framework [5]. Сравнение с альтернативным подходом — разработкой модуля «с нуля» — выявило стратегическое преимущество использования готовой платформы. Global Framework предоставляет готовые механизмы безопасности, документооборота и интеграции, что позволяет сфокусироваться на создании уникальной бизнес-ценности модуля, а не на написании инфраструктурного кода. Это решение также полностью соответствует требованиям импортозамещения и гарантирует совместимость с ядром системы Global ERP.

Функционал интеграции и высоконагруженных вычислений планируется реализовать с использованием библиотек экосистемы Scala (например, Akka для построения отказоустойчивых систем), что в сочетании с платформой Global обеспечит создание промышленного решения.

На третьем этапе работы было выполнено проектирование архитектуры модуля. Для обеспечения бесшовной интеграции с ядром Global ERP предложена гибридная схема взаимодействия, включающая два уровня: синхронный (REST API) для оперативного обмена данными в режиме запрос-ответ и асинхронный (событийная шина) для гарантированной доставки сообщений между сервисами и обеспечения отказоустойчивости при пиковых нагрузках. Такой подход позволяет изолировать разрабатываемый модуль от монолитного ядра, обеспечивая независимое развертывание и масштабирование аналитических компонентов.

Четвертым этапом стала реализация core-модуля на языке Scala с использованием фреймворка Global. Базовый модуль включает в себя модели предметной области, сервисы валидации входных данных и слой доступа к данным. Благодаря использованию функциональных подходов удалось добиться предсказуемого поведения системы и упростить тестирование ключевых бизнес-операций. Готовые компоненты Global Framework (аутентификация, аудит, журналирование) были интегрированы в core-модуль без дополнительных трудозатрат.

Таким образом, в ходе выполнения работы решены поставленные задачи: проведен анализ существующих ERP-решений, обоснован выбор технологического стека

спроектирована гибридная архитектура интеграции с ядром Global ERP через REST API и событийную шину, а также реализован core-модуль на Scala с использованием готовых компонентов платформы Global. Предложенный подход позволяет создавать надежные, масштабируемые и соответствующие требованиям импортозамещения решения, расширяющие функциональность отечественных ERP-систем.

ЛИТЕРАТУРА

1. Уэйнер, Р. Oracle Cloud Infrastructure: полное руководство. — М.: ДМК Пресс, 2023. — 420 с.
2. Архипов, М. В. Системы управления ресурсами предприятия (ERP): теория и практика внедрения. — М.: ИНФРА-М, 2020. — 316 с.
3. Официальный портал 1С для разработчиков [Электронный ресурс]. — Режим доступа: <https://v8.1c.ru/erp/> (Дата обращения: 15.03.2026).
4. SAP Help Portal: документация по SAP S/4HANA [Электронный ресурс]. — Режим доступа: <https://www.sap.com/central-asia-caucasus/products/erp/what-is-sap-erp.html> (Дата обращения: 15.03.2026).
5. Сайт системы Global ERP [Электронный ресурс]. — Режим доступа: <https://global-system.ru/> (Дата обращения: 15.03.2026).

УДК 004.421.5

К.А. Стародубцева (4 курс бакалавриата),
О.Г. Малеев, к.т.н., доцент

РАЗРАБОТКА АРХИТЕКТУРЫ ИНТЕГРИРУЕМОГО ЯДРА AI-ПОМОЩНИКА ДЛЯ ФИТНЕС-ТРЕНИРОВОК С КОНТРОЛЕМ ТЕХНИКИ ВЫПОЛНЕНИЯ УПРАЖНЕНИЙ

В последние годы наблюдается стремительный рост популярности цифровых фитнес-сервисов: по данным Research and Markets, объем мирового рынка цифрового фитнеса к 2028 году достигнет 250 млрд долларов [1]. Однако параллельно фиксируется увеличение числа травм, связанных с неправильной техникой выполнения упражнений, особенно среди начинающих, занимающихся без контроля профессионального тренера. Данное противоречие формирует актуальную научно-практическую задачу создания доступных и эффективных инструментов, способных обеспечить безопасность тренировок за счет автоматизированного контроля движений и адаптации программ под индивидуальные особенности пользователя.

Существующие на рынке решения можно разделить на две категории: приложения-планировщики (Fitbod, JEFIT), генерирующие персонализированные программы на основе целей и истории тренировок, и AI-тренеры с компьютерным зрением (Zing Coach, Freeletics), анализирующие технику выполнения в реальном времени. Несмотря на достаточно успешное развитие подходов в видеоаналитике [2], ключевым недостатком современных систем является отсутствие замкнутого цикла адаптации, при котором биомеханические ошибки, выявленные компьютерным зрением, напрямую влияли бы на корректировку тренировочной программы [3].

Целью работы является разработка архитектурного ядра интеллектуальной системы-помощника для фитнес-тренировок, интегрирующего функции контроля техники выполнения упражнений и адаптивного формирования тренировочных программ на основе объективных данных.

Для достижения поставленной цели решаются следующие задачи:

1. Анализ существующих решений в области AI-фитнес-приложений и выявление их ограничений.
2. Исследование и формализация эталонных биомеханических параметров для базового набора упражнений (приседания, выпады, отжимания, тяги, планка).
3. Обоснование выбора технологического стека для создания производительного и интегрируемого ядра.

4. Разработка модульной архитектуры системы, включающей модули компьютерного зрения, логики анализа, генератора программ и интеграции с физиологическими данными.

Предлагаемая *архитектура* базируется на использовании фреймворка MediaPipe Pose [4] для трекинга ключевых точек тела и вычисления биомеханических параметров (углы в суставах, амплитуда, скорость, стабильность) в реальном времени. Для классификации качества выполнения упражнений предусмотрено применение кастомных моделей машинного обучения, развертываемых с помощью TensorFlow Lite. Модуль логики принятия решений реализуется на языке Python с использованием фреймворка FastAPI, что обеспечивает гибкость интеграции через RESTful/WebSocket API. Генератор тренировочных программ использует гибридный подход, сочетающий детерминированные правила (Rule-Based System) и модели машинного обучения для долгосрочной адаптации плана. Интеграция с физиологическими данными (пульс) осуществляется через абстрактный интерфейс поверх Google Fit API.

Ключевой особенностью разрабатываемой системы является реализация замкнутого цикла адаптации: информация о типичных биомеханических ошибках, выявленных модулем компьютерного зрения, передается в генератор программ и учитывается при формировании последующих тренировок (например, включение корректирующих упражнений или снижение рабочего веса). Данный подход позволяет преодолеть основной недостаток существующих решений – разобщенность функций планирования и контроля.

В результате выполнения работы будут получены следующие результаты:

1. Формализованная модель данных и эталонные параметры для набора базовых упражнений.
2. Обоснованная модульная архитектура интегрируемого ядра AI-фитнес-помощника.
3. Прототип системы, реализующий ключевые функции анализа техники и адаптивного планирования.
4. Четко определенные программные интерфейсы (API) для интеграции ядра с внешними приложениями.

Практическая значимость работы заключается в создании универсального интегрируемого ядра, которое может быть использовано разработчиками фитнес-приложений, фитнес-клубами, онлайн-платформами и реабилитационными центрами для добавления функционала AI-тренера в свои продукты без необходимости проведения собственных глубоких исследований в области компьютерного зрения и биомеханики.

ЛИТЕРАТУРА

1. Research and Markets. Digital Fitness Market – Growth, Trends, COVID-19 Impact, and Forecasts (2023–2028). – 2023. – 145 p.
2. Ицзя, Л. Применение нейронных сетей для оценки трафика магазина / Л. Ицзя, О. Г. Малеев // Современные технологии в теории и практике программирования: Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 26–27 апреля 2023 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. – С. 54-55. – EDN WJRDNR.
3. Иванов А.В., Петров С.К. Применение методов компьютерного зрения для анализа физических упражнений // Искусственный интеллект и принятие решений. – 2021. – № 3. – С. 45–58.
4. Lugaresi C., Tang J., Nash H. et al. MediaPipe: A Framework for Building Perception Pipelines // arXiv preprint arXiv:1906.08172. – 2019.
5. Смирнова О.А. Тренды цифровизации фитнес-индустрии: от онлайн-тренировок к AI-тренерам // Цифровая трансформация. – 2022. – № 4. – С. 22–30.
6. Зацюрский В.М., Аруин А.С., Селуянов В.Н. Биомеханика двигательной деятельности: Учебник для вузов. – М.: Спорт, 2021. – 320 с.

ИИ-ПОМОЩНИК ДЛЯ ОТЕЛЬЕРОВ И СПЕЦИАЛИСТОВ КЛИЕНТСКОГО ОТДЕЛА

Платформа TravelLine, предназначенная для автоматизации работы отелей, санаториев и хостелов, является сложной в эксплуатации. Это приводит к большой нагрузке на службу сопровождения, которая отвечает на большое количество вопросов. С ростом обращений ситуация становится критичной и требует внедрения инновационных решений для автоматизации взаимодействия с клиентами. Одним из наиболее перспективных направлений является использование вопросно-ответных систем (QA-систем). Такие информационные системы способны принимать вопросы и отвечать на них на естественном языке.

Интеграция существующих систем (например, Alliants Experience Platform или Edward от Edwardian Hotels) [1, 2] в платформу TravelLine является не только финансово затратной, но и трудно реализуемой с точки зрения кастомизации. Поэтому требуется разработка специализированных средств.

Таким образом, целью работы является разработка вопросно-ответной системы в виде ИИ-помощника для работы с платформой OmniDesk – централизованной системой обработки всех обращений в поддержку – и личного кабинета пользователя (отельера).

Внедрение ИИ-помощника позволит отельерам быстро получать ответы на вопросы о работе TravelLine, консультации по настройкам и процессам. Это не только повысит удобство использования платформы, но и сократит время ожидания помощи. Для специалистов поддержки и клиентского отдела, ИИ-помощник станет инструментом повышения эффективности работы за счет автоматизации рутинных запросов.

Для достижения поставленной цели решаются следующие задачи: проведение обзора существующих решений для ИИ-помощников и оценка их применимости с учетом имеющихся ограничений; разработка архитектуры системы; выполнение программной реализации системы; тестирование работы системы и анализ показателей её эффективности.

Разрабатываемая система (представлена на рисунке 1 и обведена рамкой) представляет собой интеллектуального помощника для ответов на вопросы, интегрированного с TravelLine.

Архитектура системы включает клиентскую часть, отвечающую за визуализацию чата и взаимодействие с пользователем и реализованную на React с TypeScript [3], и серверную часть, разработанную на C# ASP.NET Core [4].

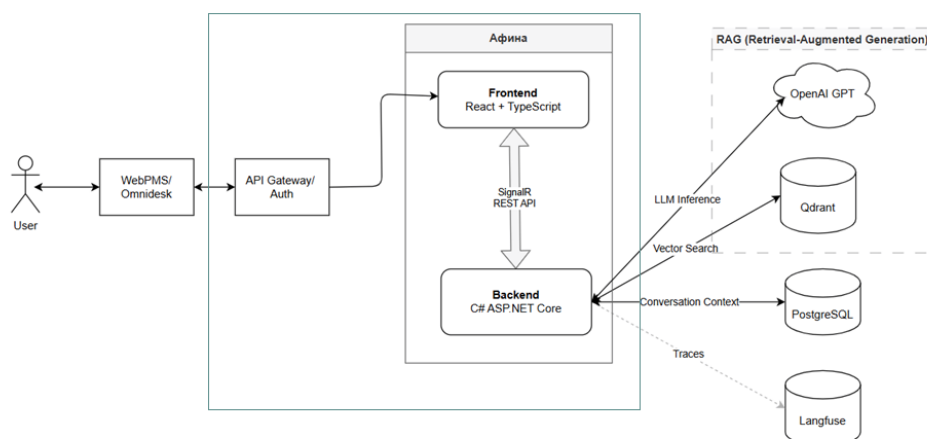


Рисунок 1 – Базовая схема архитектуры

Для достижения высокой точности и достоверности ответов используется RAG-модуль [5], интегрированный с большой языковой моделью (LLM) OpenAI GPT. Хранение данных обеспечивается несколькими базами данных: векторной базой Qdrant, реляционной

PostgreSQL. Трафик данных маршрутизируется через API Gateway, а безопасность доступа гарантируется системой аутентификации.

При разработке системы применяется модульная архитектура, обеспечивающая масштабируемость продукта. Такое разделение не только упрощает замену отдельных частей без влияния на другие, но и открывает возможности для дальнейшего расширения системы.

ЛИТЕРАТУРА

1. Alliants. Enabling Exceptional Experiences [Электронный ресурс] // Alliants.com. – URL: <https://www.alliants.com/> (дата обращения: 09.11.2025).
2. С днём рождения, Эдвард! Виртуальный хозяин Edwardian Hotels отмечает первый год [Электронный ресурс] // HotelSpeak. – URL: <https://www.hotelspeak.com/2017/06/edwardian-hotels-edward-one-year-service/> (дата обращения: 09.11.2025).
3. React. TypeScript в React [Электронный ресурс] // Официальная документация React. – URL: <https://react.dev/learn/typescript> (дата обращения: 09.03.2026).
4. Telerik. ASP.NET Core basics: organizing projects & architectural patterns [Электронный ресурс] / Telerik Blogs. – 2024. – URL: <https://www.telerik.com/blogs/asp-net-core-basics-organizing-projects-architectural-patterns> (дата обращения: 09.03.2026).
5. Yandex Cloud. RAG: учим искусственный интеллект работать с новыми данными [Электронный ресурс] // Yandex Cloud. – URL: <https://yandex.cloud/ru/blog/posts/2025/05/retrieval-augmented-generation-basics> (дата обращения: 09.03.2026).

УДК 004.89

В.А. Сусликов (3 курс бакалавриата),
О.В. Александрова, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ СТРАТИФИКАЦИИ РИСКОВ ОСЛОЖНЕНИЙ САХАРНОГО ДИАБЕТА НА ОСНОВЕ КЛИНИКО-ЛАБОРАТОРНЫХ ДАННЫХ

Диабетическая ретинопатия и нефропатия — это тяжелые микрососудистые осложнения сахарного диабета, приводящие соответственно к потере зрения и развитию хронической почечной недостаточности [3]. Ретинопатия характеризуется поражением сосудов сетчатки, что может привести к отслойке сетчатки и слепоте. Нефропатия проявляется поражением клубочкового аппарата почек, прогрессирующим снижением скорости клубочковой фильтрации и развитием терминальной почечной недостаточности [3]. Своевременное выявление пациентов с высоким риском развития этих осложнений позволяет назначить превентивную терапию и замедлить их прогрессирование. Для оценки риска используются клинико-лабораторные показатели: уровень гликированного гемоглобина (HbA1c), отражающий долгосрочный гликемический контроль, показатели липидного обмена, артериальное давление, альбуминурия и другие факторы.

В работе используются предоставленные Государственным Педиатрическим Медицинским Университетом г. Санкт-Петербурга обезличенные данные пациентов с сахарным диабетом. Данные представляют собой выгрузки из медицинских информационных систем в формате CSV. Каждый файл содержит демографические данные (пол, дата рождения), антропометрические показатели (рост, вес, индекс массы тела), результаты лабораторных исследований в динамике (HbA1c, креатинин, скорость клубочковой фильтрации, микроальбуминурия, липидный профиль), клинические измерения (артериальное давление), а также отметки о наличии осложнений (ретинопатия, нефропатия) с датами их диагностики. Лабораторные показатели представлены множеством записей за разные даты, однако интервалы между измерениями неравномерны и варьируются от нескольких недель до нескольких лет. Также присутствуют пропуски — некоторые показатели измерялись не во все визиты и не у всех пациентов.

Исходные данные имеют сложную многоуровневую структуру. Фактически, по каждому пациенту есть статические характеристики (пол, дата рождения) и множество временных рядов (динамика HbA1c, креатинина, давления и т.д.). Количество измерений у разных пациентов существенно варьируется. Для использования в моделях машинного обучения необходимо преобразовать эти данные в формат «один пациент — одна строка» [1]. Для этого из временных рядов извлекаются агрегированные признаки: средние значения, стандартные отклонения (характеристика вариабельности), минимальные и максимальные значения, коэффициенты линейных трендов (характеризуют динамику ухудшения или улучшения показателей). При расчете трендов необходимо учитывать неравномерность интервалов между измерениями и общую длительность наблюдения за пациентом, чтобы признаки были сопоставимы между разными пациентами. К полученным признакам добавляются статические характеристики, целевой переменной выступает наличие или отсутствие осложнения (ретинопатии или нефропатии) на момент последнего наблюдения.

После подготовки данных проводится обучение ML-моделей на большей части данных, оставшаяся часть используется для валидации. При этом необходимо обеспечить, чтобы данные конкретного пациента целиком находились либо только в тренировочном наборе, либо только в валидационном. Это критически важно, поскольку у одного пациента может быть множество записей за разные даты, и если они попадут в разные выборки, модель может запомнить индивидуальные паттерны конкретного пациента и при валидации показать завышенное качество, которое не будет достигаться на новых пациентах.

Для решения задачи бинарной классификации могут использоваться различные методы: логистическая регрессия (интерпретируемая базовая модель, позволяющая оценить влияние каждого признака), случайный лес (учитывает нелинейные взаимодействия и устойчив к переобучению), градиентный бустинг (XGBoost, LightGBM), а также нейросетевые подходы. Выбор конкретных методов определяется компромиссом между точностью предсказания и интерпретируемостью модели, что особенно важно в медицинских приложениях, где врачу необходимо понимать, на основании каких факторов модель сделала прогноз [4].

В задачах бинарной классификации используются следующие метрики качества. Accuracy (точность) — доля правильных ответов модели. Precision (точность положительных предсказаний) — доля истинно положительных среди всех примеров, которые модель отнесла к положительному классу. Recall (полнота) — доля истинно положительных среди всех реально положительных примеров, показывает способность модели выявлять пациентов с осложнениями. F1-мера — гармоническое среднее precision и recall, полезна при дисбалансе классов. AUC-ROC — интегральная характеристика, показывающая способность модели разделять классы (0.5 — случайное угадывание, 1.0 — идеальное разделение) [2].

ЛИТЕРАТУРА

1. Уткин Л.В. Машинное обучение. Лекции по курсу, СПбПУ, 2015 [Электронный ресурс] Режим доступа: <https://levutkin.github.io/teaching/machine-learning> (дата обращения: 27.02.2026)
2. Франсуа Шолле. Глубокое обучение на Python. – 2025. – С. 203-225.
3. International Diabetes Federation. IDF Diabetes Atlas, 10th edn. Brussels, Belgium: 2021. [Электронный ресурс] Режим доступа: <https://www.diabetesatlas.org>.
4. Development and Validation of a Machine Learning Model to Predict Near-Term Risk of Iatrogenic Hypoglycemia in Hospitalized Patients. JAMA Network Open | Diabetes and Endocrinology. -- 2021.
5. Data-based modeling for hypoglycemia prediction: Importance, trends, and implications for clinical practice. [Электронный ресурс] Режим доступа: <https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2023.1044059/full>.

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОБМЕНА СООБЩЕНИЯМИ И ОНЛАЙН-ВСТРЕЧ

Коммуникационные платформы являются критической инфраструктурой для распределённых команд: через них обеспечиваются оперативные обсуждения, обмен файлами, проведение созвонов и планирование встреч. На практике организации сталкиваются с зависимостью от внешних облачных сервисов и рисками несоответствия требованиям хранения данных и регуляторике, недостаточной интеграцией коммуникаций с внутренними сервисами, а также сложностью эксплуатации гибридных сетевых сценариев (локальная сеть, VPN, удалённый доступ). Рост требований к управляемости и безопасности (централизованное хранение истории, аудит действий, разграничение прав, резервное копирование) делает актуальной разработку специализированных корпоративных решений [1-3].

Цель работы – обосновать и спроектировать программный продукт Aulynk (О-Линк) как корпоративную платформу для обмена сообщениями и проведения онлайн-встреч, ориентированную на сценарии, близкие к Microsoft Teams: личные чаты, командные комнаты, запланированные встречи по ссылке и серверное хранение истории коммуникаций.

Для достижения цели сформулированы следующие задачи:

1. Сформировать функциональные и нефункциональные требования к платформе (диалоги, комнаты, присутствие, уведомления, поиск, вложения).
2. Описать сценарии онлайн-встреч: мгновенный звонок из чата и подключение по ссылке на запланированную встречу.
3. Спроектировать серверное хранение сообщений и метаданных (пользователи, комнаты, участники, сообщения, вложения, события).
4. Обосновать выбор протоколов обмена событиями в реальном времени (WebSocket [4]) и технологий аудио/видеосвязи (WebRTC [5]).
5. Определить требования к информационной безопасности (аутентификация, RBAC, шифрование транспорта, аудит).
6. Предложить целевую архитектуру системы (клиент, сервер приложений, база данных, медиа-контур) и план развития прототипа

На рисунок 1 представлена укрупнённая архитектура Aulynk (О-Линк), включая текущий P2P-вариант звонков и планируемое расширение через SFU.

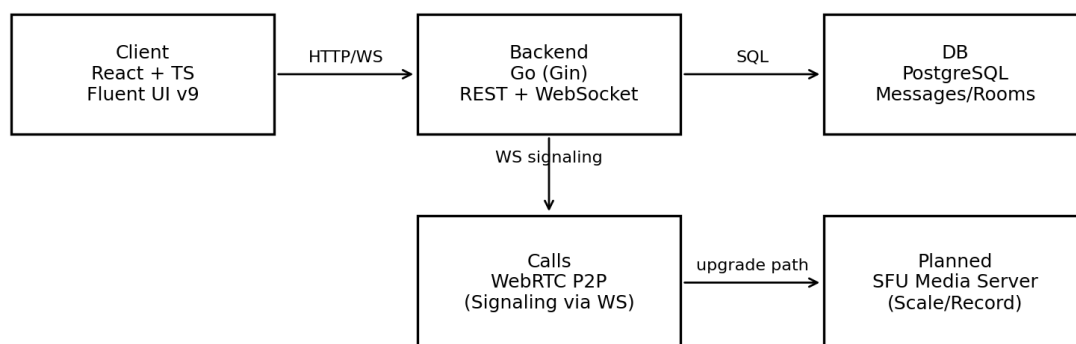


Рисунок 1 – Архитектура Aulynk (О-Линк)

Aulynk проектируется на основе трёхзвенной архитектуры: клиентские приложения (web/desktop), сервер приложений и хранилище данных, с выделением медиа-контура для звонков. Доставка событий в реальном времени (сообщения, статусы присутствия,

системные события) реализуется через постоянное двустороннее соединение WebSocket, тогда как операции управления и получения истории оформляются в виде REST/JSON API. Модуль онлайн-встреч строится на WebRTC, а сигнализация (обмен SDP/ICE, управление состоянием участников) выполняется сервером по WebSocket/HTTP.

В качестве технологического стека предлагаются: Go [6] для серверной части (высокая производительность и удобство конкурентной обработки соединений), PostgreSQL [7] для надёжного хранения истории и метаданных, а также React + TypeScript и Fluent UI React v9 [8,9] для построения клиентского интерфейса. Для эксплуатации и качества ПО предусматриваются TLS, JWT/сессионные токены, RBAC-модель ролей, журналирование критических действий, а также CI-проверки (линтеры и тесты) для серверной и клиентской частей.

Результатом выполненной работы является сформированная спецификация требований и целевая архитектура продукта, а также обоснование выбора протоколов и технологий для реализации прототипа. Критериями оценки демонстрационной версии прототипа определены: задержка доставки сообщений и событий, устойчивость соединений WebSocket при сетевых переключениях, корректность сохранения и извлечения истории из базы данных, а также выполнение базовых требований безопасности (аутентификация, контроль доступа, шифрование транспорта).

Таким образом, разработка Aulynk (О-Линк) обоснована потребностью организаций в управляемой корпоративной среде коммуникаций с контролем над данными и возможностью интеграции с внутренними сервисами. Предложенная архитектура и выбранный стек обеспечивают поэтапное расширение функциональности – от чата и комнат к звонкам и планированию встреч.

ЛИТЕРАТУРА

1. Королев, Д. О. Исследование эффективности применения бинарных нейронных сетей при детектировании объекта на изображении / Д. О. Королев, О. Г. Малеев // Информатика и ее применения. – 2023. – Т. 17, № 3. – С. 88-92. – DOI 10.14357/19922264230312. – EDN TOCVAL.
2. Чигасова, М. А. Прогнозирование котировок акций с использованием нейронной сети / М. А. Чигасова, О. Г. Малеев // Современные технологии в теории и практике программирования : Сборник материалов конференции, Санкт-Петербург, 26 апреля 2022 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2022. – С. 180-182. – EDN GRFEDM.
3. Ларионов, В. С. Эффективность распараллеливания метода на основе роя частиц при оптимизации обучения нейронных сетей / В. С. Ларионов, О. Г. Малеев // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2022. – № 7. – С. 71-77. – DOI 10.37882/2223-2966.2022.07.18. – EDN EIZWER.
4. RFC 6455. The WebSocket Protocol. IETF, 2011. [Электронный ресурс] Режим доступа: <https://datatracker.ietf.org/doc/html/rfc6455> (дата обращения: 08.03.2026).
5. W3C. WebRTC: Real-Time Communication in Browsers (WebRTC 1.0). [Электронный ресурс] Режим доступа: <https://www.w3.org/TR/webrtc/> (дата обращения: 08.03.2026).
6. The Go Programming Language Documentation. [Электронный ресурс] Режим доступа: <https://go.dev/doc/> (дата обращения: 08.03.2026).
7. PostgreSQL Documentation. [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/> (дата обращения: 08.03.2026).
8. React Documentation. [Электронный ресурс] Режим доступа: <https://react.dev/learn> (дата обращения: 08.03.2026).
9. Fluent UI React v9 Documentation. [Электронный ресурс] Режим доступа: <https://storybook.fluentui.dev/react/?path=/docs/concepts-introduction--docs> (дата обращения: 08.03.2026).

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ ПРОДАЖ ИГРОВЫХ ПРЕДМЕТОВ И
УЧЁТА КАПИТАЛА НА БАЗЕ ТОРГОВОЙ ПЛОЩАДКИ MARKET CS:GO

В работе рассмотрена проблема автоматизации процессов приобретения и реализации игровых предметов в игре Counter Strike 2 на торговой площадке Market CS:GO [5] с одновременным ведением учета движения денежных средств. Основные трудности, возникающие при работе через официальную торговую площадку Steam, связаны с высокой комиссией в размере 13 процентов, необходимостью постоянного ручного мониторинга рыночной ситуации для поддержания конкурентоспособных цен, отсутствием удобного интерфейса для управления продажами, недостаточной прозрачностью учета капитала в режиме реального времени, а также ограничениями на вывод средств с торговой площадки. Указанные факторы приводят к снижению прибыли реселлеров, росту операционных расходов и замедлению оборачиваемости капитала [6].

Для решения поставленной задачи проведен детальный анализ предметной области, сформулированы функциональные и нефункциональные требования к программному комплексу [3]. В качестве технологического решения использована платформа .NET 9 с языком программирования C#, асинхронная архитектура для обработки API-запросов в реальном времени, интеграция с Telegram Bot API для управления системой и Google Sheets для совместного финансового учета, Enterprise Architect для визуального моделирования и проектирования системы [1]. На основе данного решения система была полностью реализована, развернута на отдельном физическом сервере и в настоящее время успешно функционирует в реальных условиях [2]. Управление торговым ботом осуществляется через удобный интерфейс Telegram-бота, и система активно используется командой реселлеров в повседневной торговой деятельности.

Система представляет собой независимое программное решение, обеспечивающее автоматическое выставление игровых предметов на продажу, их повторное размещение в случае отмены обмена [5], динамическое обновление цен с учетом текущего состояния рынка (стакана ордеров) и заранее установленных лимитных цен. Важными функциями являются централизованный сбор баланса на главный аккаунт, а также ведение детального учета движения средств в доступных денежных средствах, замороженном балансе и стоимости находящихся в обороте игровых предметов. Интерфейс на базе Telegram позволяет осуществлять полный контроль над всеми процессами даже с мобильного устройства. Разработанное решение соответствует заявленным показателям качества: время реакции системы на изменение цены конкурента не превышает 10 секунд, архитектура поддерживает одновременную работу до 100 торговых сессий без потери производительности, а коэффициент доступности составляет 99,9 процента [2].

Область применения системы охватывает обширный рынок реселлеров игровых предметов в Counter Strike 2 [6] и позволяет эффективно использовать ее как индивидуальными специалистами, так и командами из нескольких человек (реселлеры, бухгалтер, директор). Разработанное решение открывает возможность коммерциализации программного продукта по модели подписки, а также может применяться для автоматизации перепродажи цифровых активов на других торговых площадках. Технологическая новизна заключается в комплексном объединении высокоскоростного торгового бота с системой реального времени финансовой аналитики и многопользовательского доступа через мобильный интерфейс, что ранее не имело широкого распространения в данной предметной области.

Полученные результаты подтверждают эффективность выбранного подхода к проектированию и объектно-ориентированному моделированию [3]. Внедрение системы позволило существенно сократить объем ручного труда, полностью исключить риск

убыточных продаж за счет жестких лимитных цен, повысить скорость оборачиваемости капитала и обеспечить прозрачный финансовый контроль для всех участников процесса. Разработанная архитектура обладает высоким потенциалом масштабирования и может служить основой для дальнейшей интеграции с другими торговыми площадками цифровых активов.

ЛИТЕРАТУРА

1. Ньюмен С. Создание микросервисов. СПб: Питер, 2022. 432 с.
2. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. СПб: Питер, 2020. 640 Фаулер М. UML.
3. Основы. Краткое руководство по стандартному языку объектного моделирования. СПб. Символ-Плюс, 2018. 192 с.
4. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер, 2005. 352 с.
5. Документация API торговой площадки Market CSGO [Электронный ресурс]. URL: <https://market.csgo.com/api>
6. Steam Web API Documentation [Электронный ресурс]. URL: <https://steamcommunity.com/dev>

УДК 004.8

Е.Д. Титова (2 курс бакалавриата),
С.В. Григорьев, к. ф.-м. н., доцент,
А.А. Чистякова, стажёр

СРАВНЕНИЕ НЕЙРОННЫХ СЕТЕЙ, УПРАВЛЯЮЩИХ СИМВОЛЬНОЙ МАШИНОЙ

С ростом сложности архитектуры разрабатываемых программ растет и необходимость в их тестировании. Одним из способов, применяющихся для создания тестового покрытия, обнаружения ошибок и анализа безопасности, является символьное исполнение [1].

Символьное исполнение — это техника статического анализа, исследующая поведение программы не через перебор конкретных значений, а путем анализа всех возможных путей выполнения. Переменные хранятся в виде символьных выражений, что позволяет инструменту изучать различные сценарии ветвления. На каждом шаге анализа конфигурация символьной машины может быть представлена в виде графа: метод, который нужно протестировать, разбивается на базовые блоки, и по ним строится граф потока управления. К нему добавляются вершины, описывающие возможные состояния символьного исполнения. Перемещение этих вершин по графу и обеспечивает исследование путей выполнения программы.

Таким образом, задача анализа сценариев поведения программы сводится к обходу графа. Традиционные подходы к выбору пути при обходе полагаются на эвристики. Однако с ростом сложности программы количество возможных состояний также растет, что делает традиционные методы неэффективными и порождает задачу оптимизации обхода графа. Для этих целей можно использовать методы машинного обучения, в частности графовые нейронные сети [3–5].

Существует инфраструктура PySymGym [2], предназначенная для обучения AI-агентов навигации в инструментах символьного исполнения с целью решения проблемы взрыва путей. Агент должен выбирать наиболее перспективные пути для исследования, чтобы достичь максимального покрытия кода за минимальное время с минимальным количеством генерируемых тестов. Задачу выбора состояний можно формализовать разными способами. Первый подход предполагает бинарную классификацию: одни состояния бесперспективны, другие же ведут к хорошему результату. Второй подход рассматривает каждое состояние с точки зрения вероятности привести к успеху; задача агента — научиться максимально точно оценивать эту меру, даже если разница между состояниями не всегда бинарна.

Целью данной работы является сравнение лучших моделей, обученных двумя указанными подходами.

Для достижения поставленной цели необходимо решить следующие задачи.

1. Обучить агента по двум сценариям.
2. Сравнить метрики тестов, сгенерированных символьной машиной под управлением каждой из обученных моделей.

На архитектурном уровне задачу формализации выбора состояний можно решить, используя разные функции потерь при обучении агента. В данной работе первый способ обучения был реализован с использованием функции потерь BCEWithLogitsLoss, второй – с использованием KLDivLoss из фреймворка PyTorch.

На рисунке 1 представлено сравнение тестового покрытия, достигнутого символьной машиной под управлением моделей, обученных разными методами.

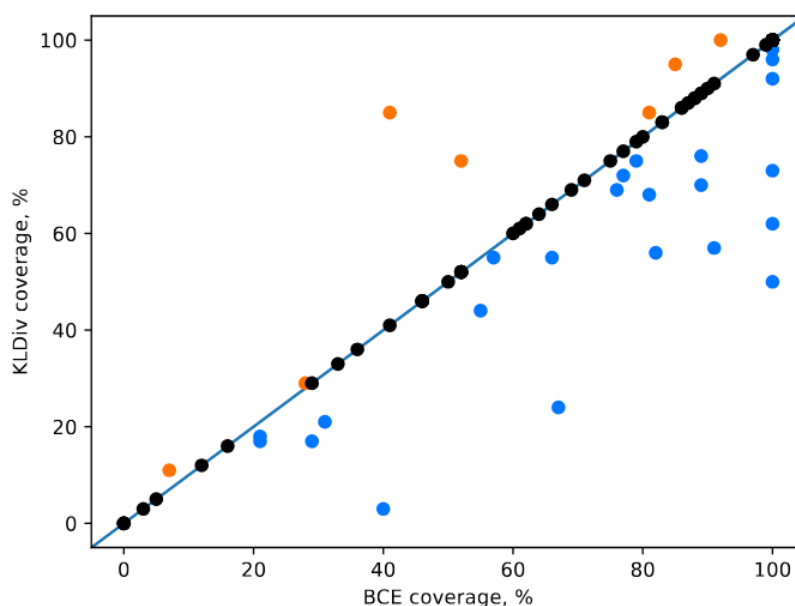


Рисунок 1 – Сравнение процента тестового покрытия для тестов, сгенерированных символьной машиной под управлением моделей, обученных разными методами

Каждая точка на графике соответствует отдельному методу, для которого были сгенерированы тесты. Координаты точки — это процент тестового покрытия, достигнутого символьной машиной под управлением агентов, обученных с использованием соответствующих функций потерь. Из графика видно, что подход с жестким разделением всех состояний на два класса является более перспективным.

ЛИТЕРАТУРА

1. Kasper Luckow, Rody Kersten and Corina Pasareanu. Complexity vulnerability analysis using symbolic execution // Software Testing, Verification and Reliability. — 2020. -02. — Vol. 30.
2. А. А. Чистякова. Создание графовой нейронной сети, предсказывающей пути при символьном исполнении. Выпускная квалификационная работа. Санкт-Петербургский государственный университет.
3. Learning to explore path for symbolic execution / Jingxuan He, Gishor Sivanrupan, Petar Tsankov, Martin Vechev //
4. Path Exploration Strategy for Symbolic Execution based on Multi-strategy Active Learning / Jingxuan He, Dalin Zhang, Dongqing Zhu et al. // Proceedings of the 15th Asia-Pacific Symposium on Internetwork 24. — New York, NY, USA : Association for Computing Machinery, 2024. — P. 165-168. URL: <https://dl.acm.org/doi/10.1145/3671016.3671403>
5. Jie Wu, Chengyu Zhang, Geguang Pu. Reinforcement Learning Guided Symbolic Execution // 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER) / IEEE. — 2020. — P. 662-663.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ АНАЛИЗА ПИЩЕВЫХ ПРИВЫЧЕК ПО ПОКУПАЕМЫМ ПРОДУКТАМ И ПО СОКРАЩЕНИЮ ИХ ОТХОДОВ

Проблема пищевых отходов имеет глобальный масштаб и остаётся актуальной для России. Согласно исследованиям, в торговых сетях ежегодно образуется около 17,23 млн тонн пищевых отходов, тогда как в домохозяйствах — порядка 4,37 млн тонн [1].

Существенная доля отходов формируется непосредственно в домохозяйствах. В среднем на одну семью приходится около 82,4 кг пищевых отходов в год, что составляет примерно 29,9 кг на человека [1].

Одной из причин образования пищевых отходов является отсутствие системного контроля приобретённых продуктов, включая учёт покупок и отслеживание сроков годности. Нерациональное планирование закупок и хранения продуктов приводит к увеличению объёма выбрасываемой пищи и неэффективному использованию продовольственных ресурсов.

Таким образом, целью работы является разработка мобильного приложения, обеспечивающего автоматизированный учёт приобретаемых продуктов за счёт сканирования чеков, штрих-кодов и распознавания продукта по фото, анализ пищевых привычек на основе анализа покупаемых продуктов и сокращение пищевых отходов за счёт организации контроля сроков годности и уведомлений, а также интеграцию с носимыми устройствами.

Мобильное приложение будет реализовано на языке Dart [2], с использованием кроссплатформенного фреймворка Flutter [3]. Для управления состоянием применяется паттерн Bloc/Cubit [4], разделяющий интерфейс и бизнес-логику. Серверная часть построена на платформе Firebase [5]. Распознавание текста с чеков осуществляется с помощью ML Kit [6]. Для дообучения нейронной сети предполагается использовать метод распараллеливания вычислений роя частиц (PSO) [7].

Архитектура мобильного приложения построена на основе принципов Clean Architecture [8] и ориентирована на модульность, расширяемость и изоляцию бизнес-логики от внешних зависимостей. Структура системы включает следующие слои: представления, доменный и данных (рисунок 1).

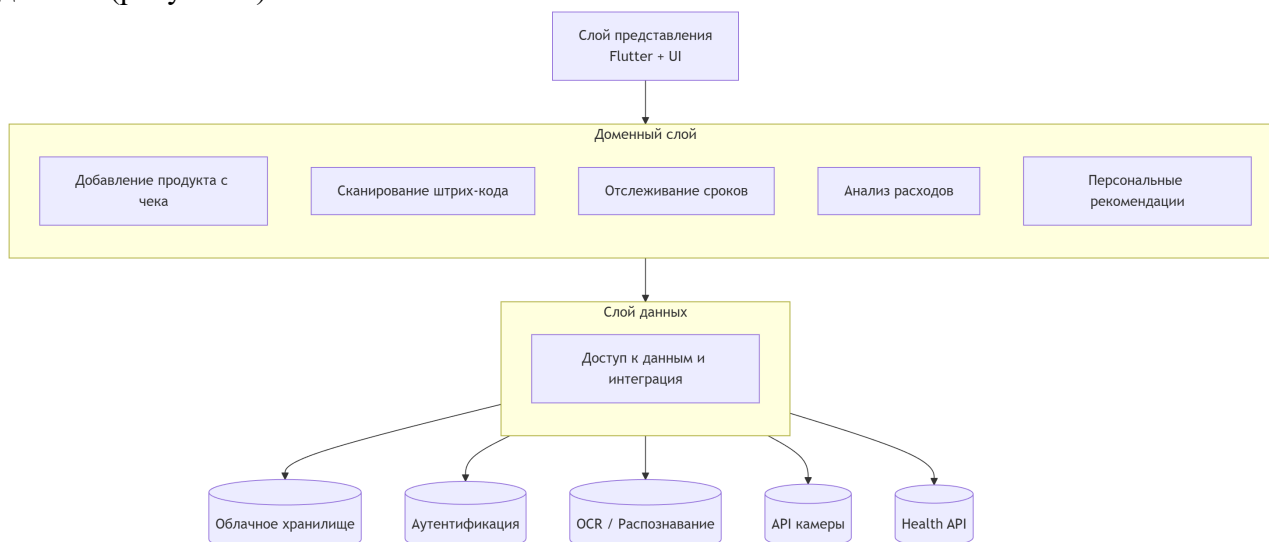


Рисунок 1 – Архитектурный шаблон системы

Слой представления обеспечивает взаимодействие пользователя с системой: добавление информации о приобретённых продуктах посредством сканирования чеков, штрих-кодов или распознавания по изображению, просмотр списка продуктов, получение уведомлений о приближении срока годности, а также отображение статистики покупок.

Доменный слой выполняет обработку основной бизнес-логики приложения. Здесь реализованы сценарии анализа данных о приобретённых продуктах, контроль сроков годности, формирование уведомлений пользователю и построение рекомендаций по рациональному потреблению.

Слой данных обеспечивает получение и сохранение информации, необходимой для работы приложения. Он взаимодействует с облачным хранилищем для хранения данных о продуктах и покупках, выполняет аутентификацию пользователей, обращается к сервисам распознавания текста чеков и сканирования штрих-кодов, а также обрабатывает данные, поступающие из внешних источников, включая сервисы и устройства экосистемы цифрового здоровья.

Первая версия мобильного приложения включает следующие возможности: регистрация и аутентификация пользователя, ведение списка продуктов, добавление информации о приобретённых товарах, контроль сроков годности продуктов, формирование уведомлений о приближении срока их истечения, а также отображение базовой статистики потребления и структуры покупок.

На текущем этапе разработки реализованы основные функциональные модули системы, включая личный кабинет пользователя, отображение списка продуктов и механизм добавления нового продукта.

Тестирование приложения на первом этапе планируется проводить на устройствах с операционной системой Android, для которой будет выпущена первая версия приложения. В процессе тестирования будут проверены основные пользовательские сценарии: регистрация и вход в систему, добавление и редактирование продуктов, отображение списка продуктов, а также корректность сохранения и отображения данных. В дальнейшем планируется расширение тестирования и поддержка платформы iOS.

ЛИТЕРАТУРА

1. Бондарчук О. Н. Торговые сети и их потребители с позиции образования пищевых отходов // Агропромышленные технологии Центральной России. – 2025. – № 1(35). – С. 28–37.
2. Dart programming language | Dart. [Электронный ресурс]. Режим доступа: <https://dart.dev/> (дата обращения: 06.03.2026).
3. Yousef A. M. A., Bubaker L. A. H. Evaluating the Flutter Framework in Academic Education: A Study of Student Experience and Challenges in Mobile Application Development // African Journal of Advanced Pure and Applied Sciences. – 2025. – Vol. 4, No 3. – P. 575–582. – DOI 10.65418/ajapas.v4i3.1471.
4. Boukhary S., Colmenares E. A Clean Approach to Flutter Development through the Flutter Clean Architecture Package // 2019 International Conference on Computational Science and Computational Intelligence (CSCI): Las Vegas, NV, USA, 5–7 Dec. 2019. – IEEE, 2019. – P. 1115–1120. – DOI: 10.1109/CSCI49370.2019.00211.
5. Saraf P. R., Jadhao S. M., Wanjari S. J., Kolwate S. G., Patil A. D. A Review on Firebase (Backend as a Service) for Mobile Application Development // International Journal for Research in Applied Science & Engineering Technology. – 2022. – Vol. 10, No 1. – P. 967–971. – DOI 10.22214/ijraset.2022.39958.
6. ML Kit | Google for Developers. [Электронный ресурс]. Режим доступа: <https://developers.google.com/ml-kit?hl=ru> (дата обращения: 09.03.2026).
7. Ларионов, В. С. Эффективность распараллеливания метода на основе роя частиц при оптимизации обучения нейронных сетей / В. С. Ларионов, О. Г. Малеев // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2022. – № 7. – С. 71–77. – DOI 10.37882/2223-2966.2022.07.18. – EDN EIZWEP.
8. Nagib Sabbag Filho. Separation of Responsibilities in Practice: Comparison between N-Tier Architecture and Clean Architecture // Leaders.Tec.Br. – 2025. – Vol. 2, No 14. – P. 1–8. – DOI 10.5281/zenodo.15164674.

ГИБРИДНАЯ ДВУХУРОВНЕВАЯ СИСТЕМА ПОДБОРА ПАРТНЕРОВ ДЛЯ ДЕЙТИНГ-ПРИЛОЖЕНИЯ С АСТРОЛОГИЧЕСКИМ УКЛОНОМ

В условиях растущей конкуренции на рынке дейтинг-платформ качество релевантного подбора потенциальных партнеров приобретает стратегическое значение. Пользователи нишевых приложений ожидают персонализированного подхода, учитывающего их специфические особенности и предпочтения. Это обуславливает повышенный интерес к совершенствованию рекомендательных механизмов [4]. Особую значимость представляет собой задача разработки гибридной рекомендательной системы для астрологического приложения знакомств, интегрирующей данные о поведении пользователя и показатели его астрологической совместимости с другими кандидатами.

Актуальность работы обусловлена ограниченностью универсальных подходов к подбору партнеров: методы, опирающиеся только на анкетные данные или поведенческие факторы, не учитывают астрологическую составляющую, значимую для целевой аудитории подобных приложений. Это снижает релевантность рекомендаций и может негативно сказываться на вовлеченности и лояльности пользователей. В связи с этим перспективной представляется разработка гибридной рекомендательной системы, объединяющей астрологические признаки с поведенческими паттернами для повышения точности подбора [1].

Целью работы является проектирование рекомендательной системы для дейтинг-приложения с астрологическим уклоном, обеспечивающей персонализированный подбор партнеров на основе сочетания астрологических и поведенческих факторов. Предлагаемая система предназначена для формирования индивидуальных подборок потенциальных кандидатов с учетом их астрологической совместимости, истории взаимодействий пользователя и характеристик просматриваемых анкет. В общем виде ее функционирование включает сбор и обработку пользовательских данных, выделение значимых признаков, влияющих на успешность взаимодействий, и последующее ранжирование кандидатов для выбора наиболее подходящих рекомендаций [4].

Работа рекомендательной системы предполагает несколько последовательных этапов. На первом этапе выполняется астрологический скоринг, в рамках которого натальная карта пользователя представляется в виде вектора числовых признаков — углов положения Солнца, Луны, Венеры, Марса и других планет. Для каждой пары пользователей рассчитывается индекс синастрии на основе угловых расстояний между ключевыми планетами: гармоничные аспекты (трин 120° , секстиль 60°) получают положительный вес, напряженные (квадратура 90° , оппозиция 180°) — отрицательный. Результатом становится астрологический рейтинг совместимости в диапазоне от -1 до 1, используемый для первичной фильтрации кандидатов [1]. Пользователю не показываются анкеты с рейтингом ниже установленного порога, что гарантирует соответствие хотя бы минимальным астрологическим критериям.

На втором этапе применяется поведенческий скоринг на основе алгоритма ALS-WR (Alternating Least Squares with Weighted Regularization). Строится разреженная матрица взаимодействий пользователей, где в ячейках фиксируются факты лайков, дизлайков, сообщений и просмотров анкет. Алгоритм раскладывает эту матрицу на эмбединги пользователей, выявляя скрытые паттерны предпочтений [2]: например, если пользователю А нравятся анкеты с указанием интереса к йоге, а у пользователя Б этот интерес присутствует, то система определит высокую вероятность взаимного интереса даже при отсутствии прямой истории взаимодействий. На выходе формируется показатель поведенческой близости — число от 0 до 1, предсказывающее вероятность лайка [2].

Финальный этап системы реализует гибридизацию полученных рейтингов методом стекинга [5]. Для каждой пары кандидатов формируется вектор признаков, включающий

астрологический рейтинг, поведенческую близость, разницу в возрасте, количество общих интересов и географическую близость. На этих векторах обучается алгоритм градиентного бустинга XGBoost [3], целевой переменной которого выступает реальный факт взаимодействия — поставил ли пользователь лайк. Модель автоматически определяет важность астрологических и поведенческих признаков, адаптируясь к индивидуальным особенностям аудитории [5]: для одних пользователей астрология может играть второстепенную роль, для других — являться критическим фактором выбора. Такой подход позволяет не просто комбинировать два рейтинга, а строить сложную функцию принятия решения, максимально точно предсказывающую реальные предпочтения пользователей [3].

ЛИТЕРАТУРА

1. Burke R. Hybrid Recommender Systems: Survey and Experiments // User Modeling and User-Adapted Interaction. — 2002. — Vol. 12. — No. 4. — P. 331-370.
2. Zhou Y., Wilkinson D., Schreiber R., Pan R. Large-Scale Parallel Collaborative Filtering for the Netflix Prize // Algorithmic Applications in Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings. — 2008. — P. 337-348.
3. Covington P., Adams J., Sargin E. Deep Neural Networks for YouTube Recommendations // Proceedings of the 10th ACM Conference on Recommender Systems. — 2016. — P. 191-198.
4. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. — New York: Springer, 2011. — 842 p.
5. Artemenko O., Pasichnyk V., Kunanets N., Shunevych K. Using sentiment text analysis of user reviews in social media for e-tourism mobile recommender systems // Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2020). Volume I: Main Conference. — CEUR Workshop Proceedings, Vol. 2604. — 2020. — P. 259-271.

УДК 004.453

Ц. Цай, С.Е. Юшков (2 курс магистратуры),
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ РЕЗУЛЬТАТОВ РЕШЕНИЯ МЕТОДОМ АНАЛИЗА ИЕРАРХИЙ ПРИ РАЗЛИЧНЫХ ПРАВИЛАХ КОМБИНИРОВАНИЯ СВИДЕТЕЛЬСТВ

В сложных условиях, таких как многокритериальное групповое принятие решений и управление ИТ-проектами, решения часто зависят от свидетельств из нескольких источников. Однако существующие единые методы объединения сталкиваются с двумя основными проблемами: во-первых, достоверность или важность различных участников часто неодинакова; во-вторых, классическое правило Демпстера может давать контринтуитивные результаты при высокой степени конфликта между источниками свидетельств.

Целью данной работы является создание, реализация и тестирование адаптивного Python-фреймворка для объединения мульти-источниковых свидетельств. Этот фреймворк сначала выполняет «смягчение» (дисконтирование) исходных свидетельств на основе важности участников по Бейнону [1], а затем автоматически переключается между правилом Демпстера [2] и правилом Ягера [3] в зависимости от уровня конфликта между свидетельствами.

Основной алгоритм предлагаемого подхода состоит в следующих шагах:

1. Ввод: Система принимает ВРА $(m_1, \dots, m_N)$ от N участников и их соответствующие веса важности $(w_1, \dots, w_N)$.

2. Предварительная обработка (дисконтирование): применяется алгоритм Бейнона для вычисления относительного коэффициента дисконтирования $w_k^* = w_k / \max(W)$. Для ВРА

каждого участника выполняется корректировка: $m'_k(A) = w_k^* \cdot m_k(A)$ (для всех $A \subset \Omega, A \neq \Omega$), $m'_k(\Omega) = 1 - \sum m'_k(A)$ (оставшаяся масса присваивается неопределённости) [1].

3. Обнаружение конфликта: вычисляется коэффициент конфликта K между скорректированными свидетельствами m'_1 и m'_2 (и последующими): $K = \sum_{B \cap C = \emptyset} m'_1(B)m'_2(C)$.

4. Адаптивное объединение: задаётся пороговое значение T .

ЕСЛИ $K < T$: вызывается функция DempsterCombine, которая усиливает определённость за счёт нормализации.

ИНАЧЕ: вызывается функция YagerCombine, которая преобразует конфликт в неопределённость, избегая ошибочной сходимости.

Архитектура разработанной системы представлена на рисунке 1.

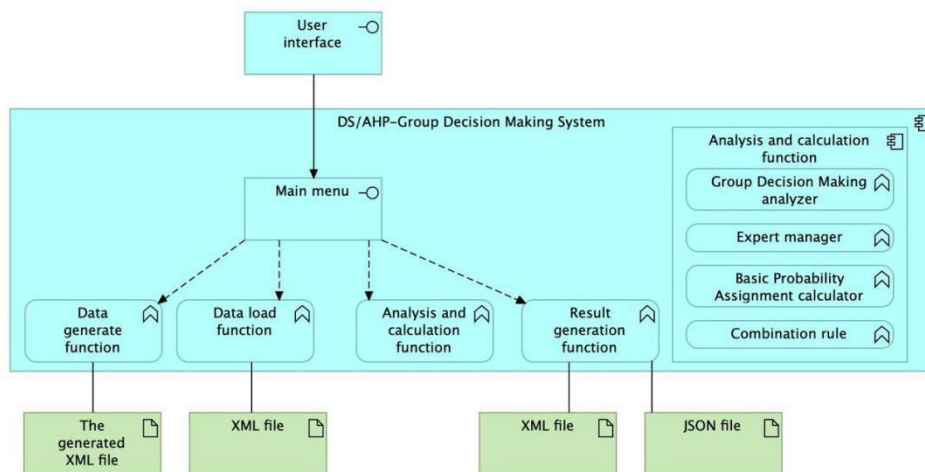


Рисунок 1 – Архитектура системы

Для проведения экспериментов был разработан генератор входных данных в формате XML, который позволяет создавать тестовые наборы данных с заданным количеством альтернатив, критериев и экспертов. Генератор позволяет задавать различные распределения весов экспертов (равномерное, убывающее, случайное) и создаёт реалистичные парные сравнения альтернатив по каждому критерию для каждого эксперта. Это позволяет моделировать различные сценарии группового принятия решений с различным уровнем согласованности мнений экспертов. Для валидации предложенного адаптивного подхода было проведено 10 экспериментов с различными конфигурациями. Каждый эксперимент повторялся 10 раз для статистической значимости. Результаты повторных экспериментов оказались абсолютно идентичными. Результаты сравнивались по трём правилам комбинирования. Все входные данные и результаты представлены в репозитории [4].

Рассмотрим практическую задачу по выбору рекомендованных учебников для освоения содержания курса из статьи [5]. Входной файл для этого датасета находится в репозитории под названием data_from_article. Результат данного эксперимента представлен в таблице 1, где под латинскими прописными буквами подразумевается название альтернативы, а в скобках приведена итоговая оценка альтернативы. Итоговая оценка альтернативы рассчитывается следующим образом: $\gamma \cdot \text{Bel}(\{A_i\}) + (1 - \gamma) \cdot \text{Pl}(\{A_i\})$.

Таблица 1 – Результаты эксперимента

Ранг/ Правило	Демпстера	Ягера	Адаптивное
1	G (0.208)	G (0.186)	A (0.179)
2	E (0.185)	F (0.181)	F (0.171)
3	A (0.179)	E (0.148)	G (0.122)
4	F (0.173)	A (0.144)	H (0.089)
5	J (0.060)	J (0.114)	J (0.083)

В данном примере адаптивное правило выбрало вариант А в качестве оптимального, тогда как оба классических правила выбрали вариант G. Этот результат объясняется тем, что

оценки учебного материала G разными экспертами имели высокий уровень конфликта, особенно между экспертами с разной степенью важности. Адаптивный алгоритм обнаружил этот конфликт, преобразовал конфликтную часть в неопределённость, что привело к снижению рейтинга G. Хотя вариант A не получил наивысшей оценки у какого-либо одного эксперта, он показал более согласованные результаты в оценках нескольких экспертов с низким уровнем конфликта, поэтому после адаптивного объединения накопил более высокую степень доверия. По аналогичной причине вариант H вошёл в топ-5, заменив вариант E.

Артефакты эксперимента и скрипты размещены в открытом репозитории [4] и показывают. Результаты показывают, что разработанное адаптивное правило склонно выбирать учебные материалы с более высокой степенью консенсуса и меньшим конфликтом.

ЛИТЕРАТУРА

1. Beynon M., Curry B., Morgan P. The Dempster–Shafer theory of evidence: an alternative approach to multicriteria decision modelling // Omega. – 2000. – Vol. 28, No 1. – P. 37–50.
2. Shafer G. A mathematical theory of evidence. – 2020.
3. Kisworo K. Fmadm: Yager Model in Fuzzy Decision Making // Jurnal Tekno Kompak. – 2018. – Vol. 12, No 1. – P. 1–4.
4. Cai C., Iushkov S., Parkhomenko V. A. DS-AHP [Электронный ресурс] // GitHub. – URL: <https://github.com/YuxSer/DS-AHP> (дата обращения: 25.01.2026).
5. Beynon M. J. A method of aggregation in DS/AHP for group decision-making with the non-equivalent importance of individuals in the group // Computers & Operations Research. – 2005. – Vol. 32, No. 7. – P. 1881–1896.

УДК 004.453

М.Е. Царев (2 курс магистратура),
В.А. Пархоменко, ст. преподаватель

ИЗУЧЕНИЕ ВЛИЯНИЯ ВЫБОРА ФУНКЦИИ ДЕФАССИФИКАЦИИ НЕЧЁТКИХ ТРЕУГОЛЬНЫХ ЧИСЕЛ НА ВЫЯВЛЕНИЕ ПРЕДВЗЯТЫХ ЭКСПЕРТОВ

В задачах группового принятия решений широко применяются методы, основанные на экспертных оценках. Одной из ключевых проблем таких методов является наличие предвзятых экспертов, систематически завышающих или занижающих оценки альтернатив. Игнорирование предвзятости может приводить к искажению итоговых результатов и снижению качества принимаемых решений. Для выявления предвзятых экспертов применяются статистические подходы, в частности методы, основанные на анализе субъективных интервалов средних оценок экспертов [1]. При использовании нечётких чисел при оценке альтернатив возникает необходимость их дефазсификации, то есть преобразования нечёткого значения в чёткое (точное). В данной работе показано, что выбор функции дефазсификации существенно влияет на результаты детектирования предвзятых экспертов.

Целью работы является исследование влияния выбора функции дефазсификации так называемых нечётких треугольных чисел [2] на выявление предвзятых экспертов методом субъективных интервалов. Для достижения поставленной цели решены следующие задачи: 1) реализация базового алгоритма выявления предвзятых экспертов на основе субъективных интервалов средних оценок; 2) разработка нескольких способов дефазсификации треугольных нечётких чисел; 3) проведение вычислительных экспериментов на тестовых данных с различной степенью предвзятости экспертов; 4) анализ чувствительности алгоритма к параметру неопределённости и выбору функции дефазсификации.

Каждая оценка эксперта моделируется треугольным нечётким числом вида $(x - \delta, x, x + \delta)$, где δ задаёт уровень неопределённости оценки. В работе используется метод выявления предвзятых экспертов на основе интервалов средних значений оценок [1]. Для каждого эксперта вычисляется среднее значение оценок и стандартное отклонение, после чего строится

интервал при заданном уровне значимости. Эксперт считается предвзятым, если его интервал пересекается с интервалами других экспертов меньше установленного числа раз. Мы установили минимально необходимое количество пересечений равное 1. Это означает, что если интервал эксперта не будет иметь пересечения ни с одним другим интервалом, он будет считаться предвзятым. В наших наборах данных предвзятым называется тот эксперт, все значения которого либо лучше (позитивная предвзятость), либо хуже (негативная предвзятость) по каждому отдельному критерию.

Эксперименты проводились на наборе альтернатив с различной степенью предвзятости одного из экспертов. Следуя статье [2], мы создали набор данных, который включает в себя: 4 эксперта, которые выставили оценки, 4 критерия и веса критериев и экспертов. Всего создано 50 альтернатив, представляющие из себя матрицу $M \times N$ оценок данной альтернативы, эксперт n давал оценку данной альтернативе по критерию m . Для простоты эксперимента веса критериев и экспертов были одинаковыми во всех альтернативах. Кроме того, в каждой альтернативе есть эксперт, который указывал завышенные или заниженные оценки по всем критериям.

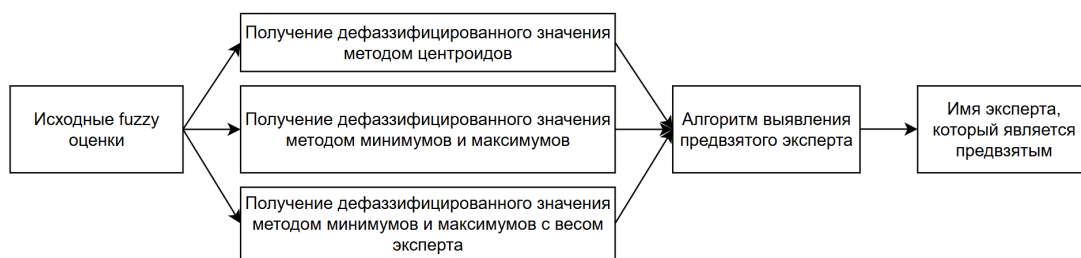


Рисунок 1 – Алгоритм проведения эксперимента

В работе рассмотрены следующие способы дефаззификации.

1. Центроидная дефаззификация. Точное значение вычисляется как взвешенное среднее левого, центрального и правого значений треугольного числа.

2. Метод минимумов и максимумов. Каждая оценка заменяется интервалом $(x - \delta, x + \delta)$, что приводит к расширению выборки оценок эксперта. Такой подход повышает чувствительность алгоритма к слабой предвзятости и позволяет выявлять экспертов с умеренным систематическим смещением оценок при определённом диапазоне δ .

3. Метод минимумов и максимумов с учётом веса эксперта. Каждая оценка заменяется интервалом $(x - 2\delta * (1 - w), x + 2\delta * (1 - w))$. Ширина интервала зависит от веса эксперта в группе w . Для экспертов с меньшим весом неопределённость увеличивается сильнее, с большим весом – слабее.

В таблице 1 представлены результаты работы описанного алгоритма по выявлению предвзятых экспертов на нашем наборе данных, состоящем из 50 альтернатив. Значение в ячейке таблицы указывает на то, сколько раз разработанный алгоритм выбрал i -го эксперта в качестве предвзятого. F1 мера рассчитана следующим образом: за верные ответы принято правильное определение предвзятого эксперта (см. ранее), за неверные ответы принимается не определение или неверное определение предвзятого эксперта. В нижней строке указано истинное количество случаев предвзятости для каждого эксперта.

Таблица 1 – Результат обнаружения предвзятого эксперта различными алгоритмами

Методы дефаззификации	e1	e2	e3	e4	f1-score
Центроидная дефаззификация	0	0	0	1	0.039
Метод min и max	3	0	4	5	0.387
Метод min и max с весом эксперта	3	0	2	5	0.333
Истинное значение (из 50 альтернатив)	13	10	10	17	-

На синтетическом наборе данных центроидная дефаззификация показала самый слабый результат, всего одно верное определение предвзятого эксперта. В то время как методы, основанные на расширении интервалов оценок, повышают чувствительность к предвзятости. Так метод *min* и *max* корректно определил 12 предвзятых экспертов. Модификация метода *min* и *max* с весом эксперта показала более устойчивое значение, которое допускает больший диапазон δ , но ценой чуть худшей точности – 10. Всего набор данных включает 50 альтернатив с предвзятыми экспертами. Артефакты экспериментов и разработанная программа размещены в открытом репозитории [4].

ЛИТЕРАТУРА

1. Rabiee M., Aslani B., Rezaei J. A decision support system for detecting and handling biased decision-makers in multi criteria group decision-making problems //Expert Systems with Applications. – 2021. – Т. 171. – С. 114597.
2. Ma J., Lu J., Zhang G. Decider: a fuzzy multi-criteria group decision support system //Knowledge-Based Systems. – 2010. – Т. 23. – №. 1. – С. 23-31.
3. Ratandhara H.M., Kumar M. An α -cut intervals based Fuzzy Best–Worst Method for Multi-Criteria Decision-Making // Applied Soft Computing. – 2024. – Vol. 159. – Art. 111625.
4. Tsarev M., Parkhomenko. V. The-study-of-defazzification-methods-to- identify-expert-bias-using-confidence-intervals // Github URL: Доступ: <https://github.com/neens16191-student/The-study-of-defazzification-methods-to-identify-expert-bias-using-confidence-intervals> (дата обращения: 09.01.2026).

УДК 004.4

А.А. Чемодуров (4 курс бакалавриата),
Б.М. Медведев, к.т.н., доцент

РАЗРАБОТКА ОФФЛАЙН-РЕЖИМА ДЛЯ МОБИЛЬНОГО И ВЕБ-ПРИЛОЖЕНИЯ ОРГАНИЗАЦИИ СОВМЕСТНОГО ДОСУГА

Несмотря на развитие телекоммуникационных систем, пользователи часто сталкиваются с нестабильным доступом или отсутствием интернета, что делает приложения, полностью зависящие от сети, неработоспособными [1]. В связи с этим концепция *offline-first* архитектуры, предполагающая приоритетную работу с локальными данными и фоновую синхронизацию с сервером, становится ключевым направлением для повышения устойчивости и качества пользовательского опыта [2]. Эмпирические исследования, приведенные в работе Zhao et al. [3], подтверждают высокий потенциал кэширования и предзагрузки (*prefetching*) для снижения сетевых задержек в мобильных приложениях. Выбор стратегии кэширования (клиентское, серверное, на уровне приложения) напрямую влияет на производительность и требует учёта специфики данных [4, 5]. Использование фреймворка Flutter, компилирующего код для различных платформ, позволяет унифицировано реализовать логику оффлайн-режима для Android- и Web-версий приложения [6].

Целью работы является разработка и внедрение оффлайн-режима в кросс-платформенное приложение организации совместного досуга для обеспечения его работоспособности при отсутствии или нестабильности сетевого соединения.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать требования к программным средствам оффлайн-режима.
2. Спроектировать архитектуру приложения, обеспечивающую изоляцию логики работы с данными от пользовательского интерфейса.
3. Разработать программные средства локального хранения и кэширования данных, получаемых с сервера, а также синхронизации данных между локальным кэшем и сервером при восстановлении сетевого доступа.
4. Провести тестирование разработанных программных средств.

Для приложения были разработаны следующие требования к оффлайн-режиму: приложение должно обеспечивать загрузку в локальный кэш данных с сервера при доступности сети, а также доступ к основным данным, хранящимся на сервере, даже в случае недоступности сервера за счет заранее кэшированных данных. В текущей версии приложение напрямую обращается к серверу.

В разрабатываемой версии, архитектура которой показана на рисунке 1, вводится промежуточный слой – репозиторий, который инкапсулирует источник данных. При запросе данных репозиторий сначала обращается к локальному хранилищу (кэшу). Если данные в кэше отсутствуют или устарели, а сеть доступна, выполняется запрос к серверу. Полученные данные сохраняются в кэш и возвращаются пользовательскому интерфейсу. При отсутствии сети данные отображаются исключительно из кэша.

Для реализации локального хранилища выбрана связка:

- Hive – легковесное NoSQL-хранилище для кэширования структурированных данных (профиль пользователя, текстовые данные из списка желаний пользователя).
- CachedNetworkImage – специализированная библиотека для эффективного кэширования изображений, позволяющая отображать их при отсутствии сети без дополнительной реализации логики загрузки.

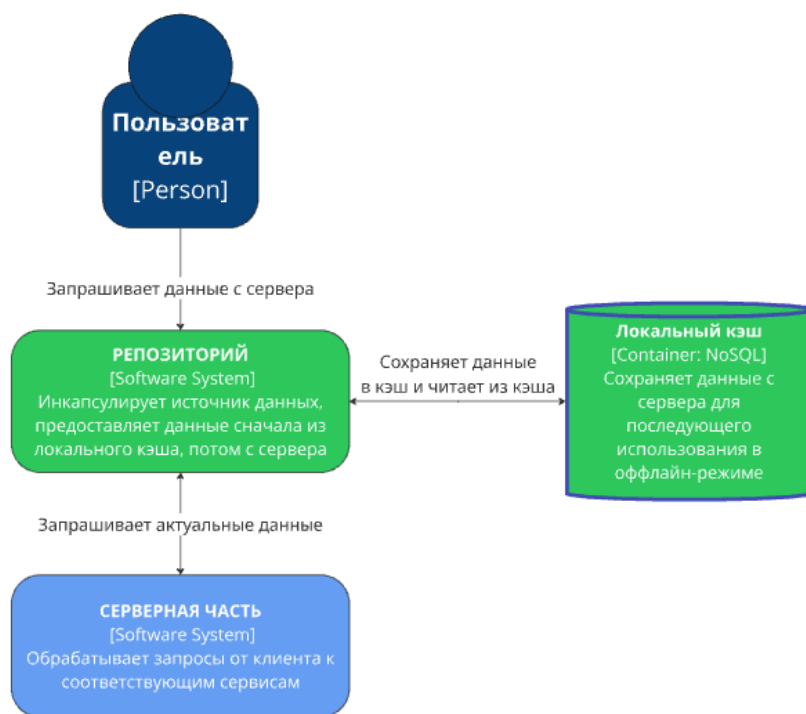


Рисунок 1 – Архитектура приложения

Разработанные программные средства, состоящие из слоя репозитория и менеджера кэширования, обеспечивают устойчивую работу кросс-платформенного приложения при отсутствии сетевого соединения. Тестирование разработанных программных средств содержит проверки работоспособности приложения в условиях отсутствия сетевого соединения, а также измерение времени загрузки данных различных сервисов. Внедрение оффлайн-режима позволит сократить время загрузки данных при повторных посещениях и исключить потерю функциональности приложения в условиях нестабильной сети. Разработанное ПО планируется использовать в составе существующего приложения, что повысит его конкурентоспособность и удобство для пользователей.

На следующих этапах работы планируется реализовать фоновую синхронизацию и разрешение конфликтов данных при обновлении информации на сервере и в кэше.

ЛИТЕРАТУРА

1. Piastou, M. Evaluating the Efficiency of Caching Strategies in Reducing Application Latency. *Journal of Science & Technology*, 2023, vol. 4, no. 6, pp. 83-98.
2. Сначалаофлайн // appmaster.io [Электронный ресурс] — Режим доступа: <https://appmaster.io/ru/glossary/snachala-oflain>
3. Zhao Y., Wat P., Laser M.S., Medvidović N. Empirically assessing opportunities for prefetching and caching in mobile apps. *In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18)*, 2018, pp. 554–564.
4. Телегин В.А. Влияние стратегий кэширования данных на пользовательское взаимодействие в мобильных приложениях. *Инновационная наука*, 2023, no. 11-2, pp. 45-51.
5. Zulfa M., Hartanto R., Permanasari A. Caching strategy for Web application – a systematic literature review. *International Journal of Web Information Systems*, 2020, vol. 16, pp. 545-569.
6. OfflineFirst в мобильных приложениях. Кэширование // habr.com [Электронный ресурс] — Режим доступа: <https://habr.com/ru/companies/friflex/articles/902060/>

УДК 004.8

А.Л. Халиуллин, Н.А. Фомин (4 курс бакалавриата),
Е.А. Павлов, ст. преподаватель

КОНТЕКСТНО-ЗАВИСИМАЯ РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА КОНТЕНТА ДЛЯ АСТРОЛОГИЧЕСКОГО ПРИЛОЖЕНИЯ

В современных цифровых сервисах важную роль играет персонализация пользовательского опыта. Особенно это актуально для приложений, в которых пользователь регулярно взаимодействует с большим объемом информации и ожидает получения релевантного контента. В условиях роста количества цифровых материалов возрастает значимость рекомендательных систем, позволяющих отбирать наиболее подходящие публикации для конкретного пользователя [1].

Актуальность работы обусловлена тем, что универсальные механизмы показа материалов, не учитывающие особенности конкретного пользователя и контекст его взаимодействия с системой, не обеспечивают достаточного уровня персонализации. В результате снижается релевантность рекомендуемого контента, что может отрицательно влиять на вовлеченность пользователей и качество взаимодействия с приложением. В связи с этим представляет интерес разработка контекстно-зависимой рекомендательной системы, способной учитывать индивидуальные характеристики пользователя, свойства контента и параметры текущего сценария использования приложения [2].

Целью работы является проектирование рекомендательной системы контента для астрологического приложения, обеспечивающей персонализированный подбор публикаций с учетом пользовательских, контентных и контекстных факторов. Предлагаемый подход опирается на идеи современных рекомендательных систем, в которых используются как признаки самих объектов, так и информация о поведении пользователя [1], а также гибридные методы формирования рекомендаций [3].

Работа рекомендательной системы предполагает несколько последовательных этапов. На первом этапе собираются данные о пользователе, включая дату, время и место рождения, астрологические признаки, выбранные интересы, историю просмотров, свайпов, сохранений, дочитываний и переходов к полному материалу. Далее анализируются признаки самого контента, среди которых тема публикации, теги, формат, длина текста, популярность, новизна, астрологическая направленность, а также наличие комментария и его тональность. Дополнительно учитываются контекстные параметры, такие как время входа в приложение, часть суток, текущая активность пользователя и его последние действия в рамках сессии [2]. На основе совокупности этих данных формируется оценка релевантности каждой карточки контента. Пользователю в ленте показывается карточка, состоящая из основного блока, в

котором создатель контента привлекает внимание к материалу, и отдельного мини-блока снизу, содержащего комментарий другого пользователя. Такой комментарий может быть как положительным, так и отрицательным и выступает дополнительным фактором, влияющим на решение о переходе к просмотру. Для рекомендательной системы этот мини-блок представляет собой дополнительный параметр, позволяющий выявлять закономерности между способом подачи материала и реакцией пользователя. Особое значение имеет также время просмотра контента, поскольку оно позволяет оценить глубину взаимодействия: быстрый выход из материала может интерпретироваться как слабый интерес, тогда как длительный просмотр, дочитывание или сохранение являются положительными сигналами [4]. Кроме того, система может адаптировать рекомендации в зависимости от времени суток, поскольку в разное время пользователь может быть склонен к восприятию разных типов контента, например более легких и развлекательных материалов в вечерние часы [2].

Разрабатываемый алгоритм направлен на повышение релевантности контента, предлагаемого пользователю, а также на улучшение общего качества взаимодействия с приложением. В качестве основы рассматривается модель, в которой рекомендации формируются на базе пользовательской активности, истории взаимодействия с публикациями, характеристик контента и контекстных факторов, влияющих на актуальность материала для конкретного пользователя [1; 2]. Для учета неявной обратной связи, выраженной через просмотры, свайпы и длительность взаимодействия, могут использоваться подходы, основанные на анализе *implicit feedback* [4].

Таким образом, разработка контекстно-зависимой рекомендательной системы для астрологического приложения представляет собой актуальную задачу прикладной программной инженерии и может служить основой для дальнейшего развития контекстно-зависимых механизмов персонализации контента в пользовательских сервисах [1; 2].

ЛИТЕРАТУРА

1. Ricci F., Rokach L., Shapira B. *Recommender Systems Handbook*. - New York: Springer, 2011.
2. Aggarwal C. C. *Recommender Systems: The Textbook*. - Cham: Springer, 2016.
3. Adomavicius G., Tuzhilin A. Context-Aware Recommender Systems // *AI Magazine*. - 2011. - Vol. 32. - No. 3. - P. 67-80.
4. Zhang S., Yao L., Sun A., Tay Y. Deep Learning Based Recommender System: A Survey and New Perspectives // *ACM Computing Surveys*. - 2019. - Vol. 52. - No. 1. - P. 1-38.
5. Bobadilla J., Ortega F., Hernando A., Gutiérrez A. Recommender Systems Survey // *Knowledge-Based Systems*. - 2013. - Vol. 46. - P. 109-132.

УДК 004.89:378.1

Д.А. Чубаров (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ РЕПЕТИТОРОВ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

В современном образовании наблюдается устойчивый рост спроса на услуги репетиторов: по данным аналитических агентств, объём мирового рынка частного обучения достиг 10,42 млрд долларов и прогнозируется его увеличение до 23,73 млрд долларов к 2030 году [1]. В России также фиксируется значительный рост числа частных преподавателей, работающих без специализированных инструментов автоматизации. Репетиторы вынуждены вручную вести учёт занятий, отслеживать успеваемость учеников и планировать расписание, что снижает эффективность их работы и затрудняет объективный анализ прогресса обучаемых.

Анализ существующих программных решений показывает, что универсальные CRM-системы (Bitrix24, AmoCRM) не учитывают специфику образовательного процесса, а системы

управления обучением (LMS) ориентированы на групповое обучение и избыточны для индивидуального репетитора. Специализированные сервисы для репетиторов не предоставляют функций интеллектуального анализа данных и недоступны в русскоязычном сегменте рынка [2]. Таким образом, существует потребность в разработке специализированной платформы, сочетающей функции управления занятиями с методами машинного обучения.

Применение методов машинного обучения в образовательных системах (Educational Data Mining, EDM) является активно развивающейся областью исследований [3]. Прогнозирование успеваемости учащихся с применением алгоритмов классификации и регрессии демонстрирует высокую эффективность для раннего выявления трудностей в обучении и персонализации образовательного процесса [4, 5]. Интеллектуальные обучающие системы на основе ML способны адаптировать содержание и темп обучения к индивидуальным потребностям каждого ученика [6].

Целью данной работы является разработка платформы для репетиторов с использованием методов машинного обучения. Для достижения поставленной цели необходимо решение следующих задач:

1. Провести анализ рынка и существующих программных решений для автоматизации работы репетиторов, выявить их достоинства и недостатки.
2. Сформулировать функциональные и нефункциональные требования к разрабатываемой платформе.
3. Спроектировать архитектуру системы, структуру базы данных и REST API.
4. Реализовать серверную и клиентскую части приложения с использованием FastAPI и Vue.js.
5. Разработать модуль машинного обучения для прогнозирования оценок, классификации учеников по уровню риска и анализа слабых тем.
6. Провести тестирование разработанной платформы и оценить качество ML-моделей.

Разработанное приложение реализует клиент-серверную архитектуру. Серверная часть построена на основе фреймворка FastAPI (Python) и обеспечивает RESTful API для управления данными об учениках, занятиях и оценках. Клиентская часть разработана с использованием Vue.js и предоставляет интерактивный интерфейс с поддержкой календарного планирования занятий, журнала оценок и аналитических дашбордов. Хранение данных осуществляется в реляционной СУБД PostgreSQL, развёртывание системы выполнено средствами Docker и Docker Compose.

Ключевой особенностью платформы является модуль машинного обучения, реализованный с использованием библиотеки scikit-learn. Модуль включает три компонента: прогнозирование следующей оценки ученика на основе алгоритма линейной регрессии по истории оценок; классификацию учеников по группам риска (низкий, средний, высокий) на основе среднего балла и тренда успеваемости; анализ успеваемости по темам с выявлением слабых и сильных областей [5]. Результаты анализа отображаются на странице аналитики с визуализацией посредством библиотеки ECharts.

В результате работы создан функциональный прототип платформы для репетиторов, включающий модули управления учениками, планирования занятий с поддержкой повторяющихся событий, ведения журнала оценок и интеллектуального анализа успеваемости. Разработанная платформа демонстрирует практическую применимость методов EDM для автоматизации и повышения эффективности индивидуального обучения.

ЛИТЕРАТУРА

1. Global Private Tutoring Market Report 2024–2030. – Grand View Research, 2024. – URL: <https://www.grandviewresearch.com/industry-analysis/private-tutoring-market> (дата обращения: 10.03.2026).
2. Baker R. S., Inventado P. S. Educational Data Mining and Learning Analytics // Learning Analytics: From Research to Practice. – New York: Springer, 2014. – P. 61–75. – DOI: 10.1007/978-1-4614-3305-7_4.

3. Huang A. Y. Q., Lu O. H. T. Artificial intelligence in intelligent tutoring systems toward sustainable education: a systematic review // Smart Learning Environments. – 2023. – Vol. 10. – Art. 41. – DOI: 10.1186/s40561-023-00260-y.
4. Jedidi Y., Ibriz A., Benslimane M. et al. Predicting students' academic performance and modeling using data mining techniques // Mathematical Modeling and Computing. – 2024. – Vol. 11, No. 3. – P. 814–825.
5. Chytas K. et al. Predicting Student Performance and Enhancing Learning Outcomes: A Data-Driven Approach Using Educational Data Mining Techniques // Computers. – 2025. – Vol. 14, No. 3. – Art. 83. – DOI: 10.3390/computers14030083.
6. Alam A., Mohanty A. Predicting Students' Performance Employing Educational Data Mining Techniques, Machine Learning, and Learning Analytics // Communications in Computer and Information Science. – 2023. – Vol. 1893. – DOI: 10.1007/978-3-031-43140-1_15.

УДК 004.588

М.А. Чурина (4 курс бакалавриата),
И.В. Зайцев, ст. преподаватель

РАЗРАБОТКА ВНЕШНЕГО МОДУЛЯ ТЕСТИРОВАНИЯ ДЛЯ LMS MOODLE НА ОСНОВЕ СТАНДАРТА LTI 1.3

Сейчас системы управления учебными курсами все активнее применяются в различных образовательных учреждениях. Они помогают создать среду взаимодействия преподавателей и студентов, предоставляя возможности удобного контроля и самоконтроля выполнения учебных заданий с более широким охватом как по количеству контролируемых заданий, так и по количеству студентов (по сравнению с выборочным ручным контролем, который тоже необходим для определения доверия результатам конкретных студентов в условиях доступности нейросетей). Одной из наиболее распространённых LMS (learning management system - система управления обучением) является Moodle. Это бесплатное программное обеспечение с открытым исходным кодом. Оно используется в школах, университетах и других организациях для дистанционного обучения и организации образовательных курсов. [6]

LMS Moodle имеет встроенные средства создания различных заданий (Quiz activity) с автоматической проверкой успешности выполнения, а также возможность подключения внешних модулей тестирования с регистрацией результатов в существующих в основной системе аккаунтах студентов, что позволяет добавить новые возможности. В частности, в технических дисциплинах встречаются задания, в которых ответом являются векторы, матрицы, таблицы (например, при обучении языку SQL), а также большие расчетные задания, включающие в себя несколько этапов вычислений (с промежуточными результатами) на основе одного и того же набора исходных данных. Таких типов заданий в стандартной системе Moodle не предусмотрено.

Таким образом, становится актуальной задача разработки расширяемых инструментов тестирования, интегрируемых с LMS без изменения её внутренней архитектуры.

Целью работы является разработка внешнего модуля тестирования, интегрируемого с LMS Moodle по стандарту Learning Tools Interoperability (LTI) 1.3. [1] Этот внешний модуль тестирования может быть совместим не только с Moodle, но и с любой LMS, поддерживающей стандарт LTI 1.3.

Для достижения поставленной цели были решены следующие задачи:

- Проанализированы спецификации стандарта LTI 1.3 и сервисов LTI Advantage, исследованы возможности интеграции внешних инструментов с Moodle.
- Разработана архитектура внешнего модуля тестирования и архитектура интеграции с LMS Moodle.
- Реализована аутентификация пользователя через Moodle и механизм передачи оценок в журнал Moodle через Assignment and Grade Services (AGS). [2]

- Отображение тестовых заданий в отдельном веб-приложении.
- Обеспечено сохранение пользовательских данных при нестабильном соединении.

На схеме приведена архитектура внешнего модуля тестирования (рисунок 1). Для интеграции использован стандарт LTI 1.3 с OAuth 2.0 и OpenID Connect для проверки доступа. Оценки передаются через AGS, записывая результаты тестов в LMS.

Серверная часть на Node.js с библиотекой Itijs поддерживает OAuth, Deep Linking, Names and Roles Provisioning и AGS. Это решение было выбрано из-за безопасности, поддержки авторизации и удобства REST API.

Клиентская часть на React и TypeScript позволила создать расширяемый современный компонентный интерфейс.

IndexedDB обеспечивает сохранение ответов при обрыве соединения, позволяет избежать потери прогресса. [3] [5]

PostgreSQL используется для хранения данных о тестах и результатах.

Для стабильности запуска всей системы используется Docker.

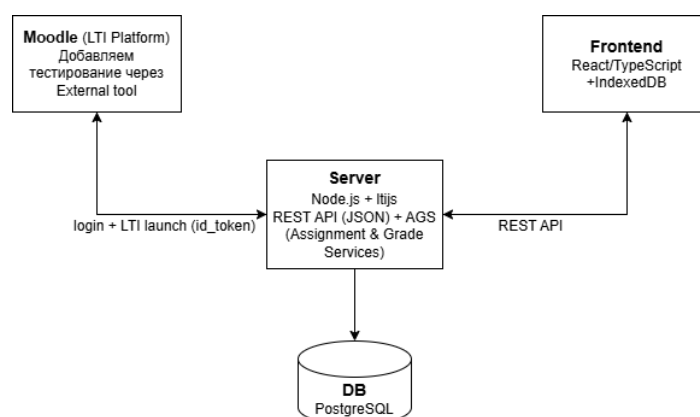


Рисунок 1 – Архитектура внешнего модуля тестирования

В отличие от встроенного инструмента Moodle Quiz, разработанный модуль:

- поддерживает различные форматы заданий (математические задачи с ответами в виде векторов и матриц, задачи по базам данных с ответами в виде файлов, содержащих таблицы, задания, состоящие из нескольких последовательных шагов);
- поддерживает работу при отсутствии соединения;
- обеспечивает независимое от LMS Moodle развитие дополнительных функций при необходимости.

Новизна работы заключается в реализации механизма офлайн-сохранения ответов, что позволяет проходить тесты даже при нестабильном соединении без потери ответов. Реализована поддержка новых типов заданий в рамках LTI-интеграции без модификации ядра самой LMS.

Разработанный модуль может быть использован в образовательных организациях для внедрения в учебные курсы новых типов заданий с автоматической проверкой ответов.

ЛИТЕРАТУРА

1. Learning Tools Interoperability (LTI)® Core Specification v1.3 [Электронный ресурс] Режим доступа: <https://www.imsglobal.org/spec/lti/v1p3> (дата обращения: 02.03.2026)
2. Learning Tools Interoperability Assignment and Grade Services Specification. [Электронный ресурс] Режим доступа: <https://www.imsglobal.org/spec/lti-ags/v2p0> (дата обращения: 02.03.2026)
3. MDN Web Docs. IndexedDB API. [Электронный ресурс] Режим доступа: <https://developer.mozilla.org> (дата обращения: 02.03.2026)
4. Moodle-LTI integration of ASPA using modern web technologies [Электронный ресурс] Режим доступа: <https://lutpub.lut.fi/handle/10024/167285> (дата обращения: 02.03.2026)

5. Evaluating IndexedDB Performance on Web Browsers [Электронный ресурс] Режим доступа: https://www.researchgate.net/publication/320614525_Evaluating_IndexedDB_Performance_on_Web_Browsers (дата обращения: 02.03.2026)
6. A systematic review of online examinations: A pedagogical innovation for scalable authentication and integrity [Электронный ресурс] Режим доступа: <https://www.sciencedirect.com/science/article/pii/S0360131520302220> (дата обращения: 02.03.2026)
7. Learning Tools Interoperability: AI-Driven Integration, Security Innovations, and Future Trends in Digital Education [Электронный ресурс] Режим доступа: <https://www.ijssat.org/papers/2025/1/2276.pdf> (дата обращения: 02.03.2026)

УДК 004.89

А.С. Шлякова (2 курс магистратуры),
А.Д. Ковалев, к.т.н., доцент

ПРОГРАММНЫЙ СЕРВИС FEW-SHOT ОБУЧЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННОГО ПОСТРОЕНИЯ ВЕРОЯТНОСТНОГО ПСИХОЛОГИЧЕСКОГО ПРОФИЛЯ ПОЛЬЗОВАТЕЛЯ ПО ТЕКСТАМ ВКОНТАКТЕ

Исследования в области киберпсихологии демонстрируют устойчивые корреляции между речевыми паттернами пользователя и его личностными характеристиками [1, 2]. По лексическому составу текстов (постов, комментариев, статусов) возможно предсказывать выраженность интроверсии/экстраверсии, демонстративности, тревожности, дистимности и экзальтированности [3, 4]. Существующие подходы достигают F1-меры до 0.75 [5], но требуют минимум 500–1000 размеченных текстов для адаптации под новый конструкт, что делает оперативное внедрение новых шкал невозможным.

Few-shot обучение позволяет адаптировать модель к новой задаче на основе 5–20 размеченных примеров. Методы мета-обучения, такие как Model-Agnostic Meta-Learning (MAML), находят исходное состояние параметров, оптимальное для быстрой донастройки. Интеграция few-shot подхода с API ВКонтакте обеспечивает гибкую систему для HR-специалистов, психологов и исследователей. Целью данной работы является снижение трудоемкости адаптации моделей психологического профилирования за счет применения few-shot обучения, строящего вероятностный профиль пользователя по пяти показателям текстового цифрового следа ВКонтакте.

Базой предлагаемого подхода является гипотеза о манифестации психологических конструктов в устойчивых лексико-семантических паттернах. На основе интеграции психолингвистических словарей и embeddings-моделей была разработана формализованная система лингвистических маркеров для пяти целевых показателей (таблица 1).

Таблица 1 – Лингвистические маркеры психологических показателей

Показатель	Лексические маркеры	Примеры слов-маркеров
Интроверсия/Экстраверсия	Местоимения 1 л., глаголы действия/состояния	«я», «мне» vs «мы», «команда»
Демонстративность	Оценочная лексика, превосходная степень	«лучший», «идеально», «топ»
Тревожность	Лексика негативной оценки, семантика угрозы	«страх», «боюсь», «тревога»
Дистимность	Лексика сниженного настроения	«грусть», «устал», «жаль»
Экзальтированность	Интенсификаторы, восклицания	«восхитительно», «ах», «обожаю»

Архитектура сервиса реализует двухконтурный принцип. Первый контур — предобучение базовой модели RuBERT-tiny (облегчённая версия BERT для русского языка)

на крупном корпусе текстов в парадигме мета-обучения MAML. Внешний цикл MAML оптимизирует исходные параметры для быстрой донастройки под новые задачи. Второй контур — адаптация под новый психологический профиль на 5–20 размеченных примерах с усилением внимания к лингвистическим маркерам (таблица 1). Внутренний цикл MAML (до 10 шагов градиентного спуска) обновляет параметры классификатора.

На рисунке 1 представлена диаграмма компонентов в нотации UML 2.5, отражающая взаимодействие VK API, модуля сбора и очистки текстов, модуля лингвистических маркеров, мета-модели RuBERT-tiny+MAML и классификатора. Стрелками показаны информационные потоки: запрос к VK API, передача текстов, извлечение маркеров и embeddings, обмен embeddings и градиентами во внутреннем цикле.

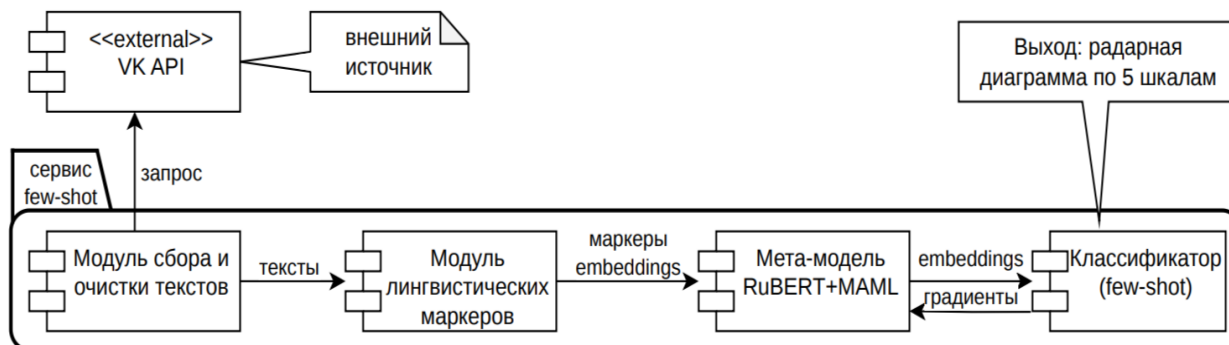


Рисунок 1 – Диаграмма компонентов

Программная реализация предполагается на Python с использованием PyTorch/higher (MAML), Transformers (RuBERT-tiny), VK API, FastAPI и Plotly. Оценка эффективности будет проводиться по метрикам F1 и AUC-ROC на выборках 5, 10 и 20 текстов. Базовые методы сравнения: zero-shot (без донастройки) и fine-tuning full (на 500+ текстах). Дополнительно планируется A/B-тестирование для оценки вклада маркерных масок внимания. Ожидается достижение $F1 \geq 0.72$ на 20 текстах, что сопоставимо с качеством при обучении на 300–500 текстах, и сокращение времени внедрения новых шкал с 2–3 недель до 30–40 минут.

ЛИТЕРАТУРА

1. Ивашко К. С., Изосимова С. А., Пигуз В. Н., Ковалева К. В. Компьютерная лингвистика в психологии: ключ к пониманию языка и поведения человека // Язык и текст. – 2025. – Т. 12, № 2. – С. 140–162. – URL: https://psyjournals.ru/journals/langt/archive/2025_n2/Ivashko_Izosimova_et_al.
2. Sheinov V.P. Social media addiction and personality: A review of research // RUDN Journal of Psychology and Pedagogics. – 2021. – Vol. 18, No. 3. – P. 607–630. – DOI 10.22363/2313-1683-2021-18-3-607-630. – URL: <http://dx.doi.org/10.22363/2313-1683-2021-18-3-607-630>.
3. Жданова А. Н. Онтологический подход к предсказанию психологических характеристик пользователей соцсетей на основе графов знаний // Искусственный интеллект и принятие решений. – 2025. – № 3. – С. 40–59. – DOI 10.14357/20718594250304.
4. Mudrova K. R., Zhdanova A. N. Predicting individual behavior reactions with social media digital footprints // Вестник молодых учёных и специалистов Самарского университета. – 2023. – № 2 (23). – С. 59–66.
5. Хлобыстова А. О., Абрамов М. В., Столярова В. Ф. Исследование тенденций взаимосвязи между профориентационными предпочтениями пользователей и их цифровыми следами в социальной сети // Научно-технический вестник информационных технологий, механики и оптики. – 2023. – Т. 23, № 3. – С. 564–574. – DOI 10.17586/2226-1494-2023-23-3-564-574.

ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ КОМПЬЮТЕРНОГО ЗРЕНИЯ В ПРИЛОЖЕНИИ ДЛЯ АВТОМАТИЗАЦИИ ОБРАБОТКИ ДЕТАЛЕЙ НА ЛАЗЕРНЫХ СТАНКАХ

Подготовка управляющих программ для лазерных станков с ЧПУ остается процессом, требующим ручной генерации G-code и высокой квалификации оператора, особенно при обработке сложных геометрий и крупногабаритных деталей.

Зачастую лазерные станки с гальванометрической системой навигации после установки или по мере использования допускают ошибки позиционирования, возникающие вследствие нелинейности отклонения галво-зеркал и оптических искажений (абберации, неравномерность фокусировки). Настройка и установка зеркал также производится оператором, что приводит к дополнительным смещениям. Процесс коррекции ошибок системы наведения называется калибрацией галво-поля [2] [6]. Наиболее распространенные методы калибровки носят итерационный характер и требуют непосредственного взаимодействия с оператором [4]. Блок-схема процесса ручной калибровки показана на рисунке 1.

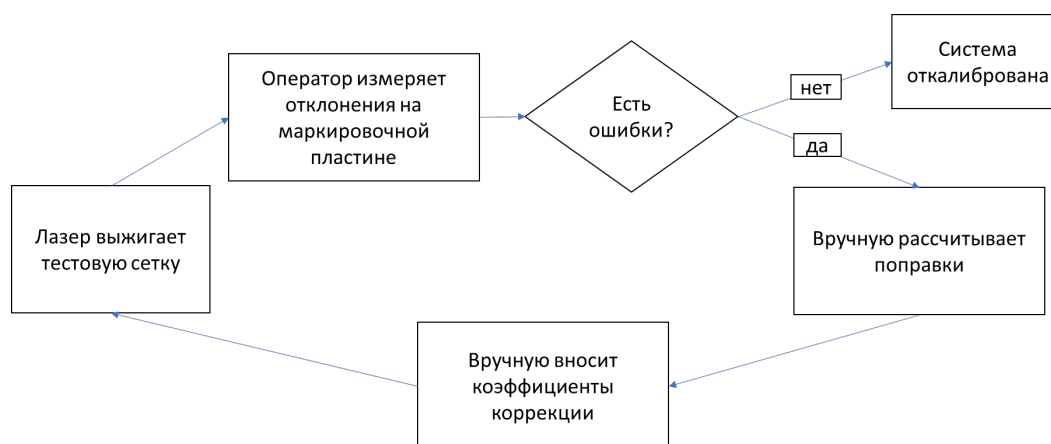


Рисунок 1 – Процесс ручной калибровки галво-зеркал

Целью работы является снижение влияния человеческого фактора и сокращение времени технологической подготовки лазерной обработки деталей за счет внедрения методов компьютерного зрения для автоматической калибровки и компенсации ошибок позиционирования гальванометрической системы, а также формирования управляющих программ.

Проведен анализ и сравнительная оценка методов ститчинга изображений (image stitching) [1] – процесса совмещения множества перекрывающихся кадров для получения карты рабочей зоны. В результате был выбран алгоритм MIST [3], обеспечивающий устойчивое и точное совмещение изображений с учетом геометрических искажений и неоднородности освещения. Основным преимуществом подхода является учет модели перемещения координатного стола и минимизация накопленных ошибок совмещения. Ститчинг используется для получения целостного представления о геометрии обрабатываемой области, а также для формирования корректных входных данных для дальнейших вычислений.

Модуль Galvo Field Correction [2] был реализован с использованием двух алгоритмов, что позволило провести сравнительную оценку различных подходов к калибровке. Первый основывался на сеточном паттерне: лазер выжигает параллельные и перпендикулярные

прямые, далее проводится сканирование рабочей зоны и поиск точек пересечения с использованием библиотеки OpenCV [5]. Данная техника характеризуется чувствительностью к условиям освещения, выбору размера сетки, а также искажениям изображения, что ограничивает ее применение в условиях нестабильной съемки. Второй метод основан на использовании кругового калибровочного паттерна [4] и включает несколько вычислительных этапов, таких как предобработка изображений, преобразование Хафа и методы оптимизации для минимизации ошибок.

В результате применения предложенных подходов точность коррекции [6] без взаимодействия с оператором при контролируемых условиях составляет порядка 10 мкм. Такие значения являются достаточными для большинства задач, а при необходимости уменьшения ошибок на порядок можно провести дополнительную итерацию уже при помощи специалиста.

Как итог, интеграция методов компьютерного зрения и коррекции галво-поля делает возможным переход к полуавтоматизированным или автономным системам подготовки обработки, снижая зависимость от квалификации оператора и повышая стабильность использования на всем рабочем поле.

ЛИТЕРАТУРА

1. Lyu W., Zhou Z., Chen L., et al. A survey on image and video stitching // *Virtual Reality & Intelligent Hardware*. – 2019. – Т. 1, № 1. – С. 55–83. – DOI: 10.3724/SP.J.2096-5796.2018.0008.
2. Zhang T., Liang Y., Wang H., et al. Field distortion correction in galvanometric scanning system by interpolation with symmetric polynomials // *Precision Engineering*. – 2023. – Т. 83. – С. 134–141. – DOI: 10.1016/j.precisioneng.2023.06.001.
3. Chalfoun J., Majurski M., Blattner T., et al. MIST: Accurate and Scalable Microscopy Image Stitching Tool with Stage Modeling and Error Minimization // *Scientific Reports*. – 2017. – Т. 7. – Ст. 4988. – DOI: 10.1038/s41598-017-04567-y.
4. Yeung H., Lane B. M., Donmez M. A., Moylan S. In-situ calibration of laser/galvo scanning system using dimensional reference artefacts // *CIRP Annals – Manufacturing Technology*. – 2020. – Т. 69, № 1. – С. 441–444. – DOI: 10.1016/j.cirp.2020.03.016.
5. Bradski G., Kaehler A. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. – Sebastopol: O'Reilly Media, 2016. – 1028 p.
6. Putzer M., Zweifel M., Kneubühler F., et al. Self-optimizing method and software for calibration and mapping of a laser system for laser machining // *Optics and Lasers in Engineering*. – 2024. – Т. 178. – Ст. 108245. – DOI: 10.1016/j.optlaseng.2024.108245.

УДК 004.822

В.П. Шемякина (1 курс магистратуры),
Т.С. Горавнева, к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ФОРМИРОВАНИЯ ПОРЯДКА ПОСЕЩЕНИЯ ВРАЧЕЙ В МЕДИЦИНСКОМ ЦЕНТРЕ

Целью работы является разработка приложения для поддержки процесса обслуживания пациентов в медицинском центре университета, позволяющего организовать электронную запись пациентов на профосмотр или диспансеризацию, а также отслеживать правильный порядок посещения врачей, учитывая время прихода.

Вся информация о сотрудниках медицинского центра и пациентах, а также об их продвижении в очереди, в данном проекте хранится в базе данных.

Разработанное приложение удовлетворяет следующим требованиям:

Требования для системы администрирования:

- Наличие регистрации пользователей;
- Наличие аутентификации и авторизации пользователей;

- Наличие возможности восстановления пароля;
- Возможность просмотра личных данных о пользователях.

Требования для системы записи на профосмотр или диспансеризацию:

- Возможность изменения и добавления личных данных о пользователях (только для уполномоченных лиц);
- Возможность просмотра, изменения и добавления данных об устройстве медицинского центра (кабинеты, этажи, связь кабинетов с врачами) (только для уполномоченных лиц);
- Возможность просмотра, изменения и добавления данных о профосмотрах или диспансеризации пациентов (только для уполномоченных лиц);
- Возможность отслеживания процесса продвижения пациентов по свободным врачам, начиная со времени прихода и учитывая время осмотра (выбранного случайным образом в некоторых пределах);
- Формирование наглядной формы отображения процесса продвижения пациентов по свободным врачам с выводом полного времени на профосмотр.

Для разработки приложения были рассмотрены современные технологии, которые могут быть использованы в процессе разработки. Для этого было произведено сравнение основных операционных систем и инструментальных сред программирования с позиции их пригодности для разработки приложений.

В результате были выбраны следующие средства разработки:

- Язык программирования Python;
- Django 5.1;
- PostgreSQL 17 и SQLite3;
- Язык программирования JavaScript;
- HTML и CSS;
- Веб-сервер Nginx [1].

При разработке, отладке и тестировании программного комплекса использовалась среда разработки Visual Studio Code 1.97.

Фреймворк Django [2] помогает быстро создавать надежные и масштабируемые веб-приложения, предоставляя также множество встроенных функций. PostgreSQL является одной из самых надежных, мощных и популярных реляционных баз данных, к тому же ее коммерческая версия сертифицирована ФСТЭК РФ, в то время как SQLite идеально подходит для разработки и тестирования и довольно проста в использовании.

В целом выбранный стек технологий обеспечивает баланс между простотой разработки, производительностью, масштабируемостью приложения. Эти инструменты хорошо интегрируются друг с другом, что позволяет быстро разрабатывать качественные веб-приложения.

Алгоритм работы приложения:

1. Создание талона администратором.

- Администратор создает новый талон с помощью административной панели Django.
- При сохранении талона автоматически через API в таблицу приемов добавляются все приемы, указанные при создании талонов с пометкой «незавершенный».

2. Автоматическое распределение приемов.

- Задача Celery 1 раз в несколько секунд проверяет базу данных на наличие незавершенных приемов и назначает прием в очередь по алгоритму распределения.

3. Реактивное обновление интерфейса.

Django Channels (WebSocket) обеспечивает мгновенное обновление данных на странице очереди у всех клиентов:

- При изменении статуса очереди сервер отправляет событие через WebSocket.
- JavaScript динамически обновляет интерфейс, отображая очередь в режиме реального времени без перезагрузки страницы.

При переходе по домену `smtu-med.ru` или при переходе по IP-адресу сервера, которые нужно ввести в адресной строке браузера, открывается главная страница приложения (рисунок 1).

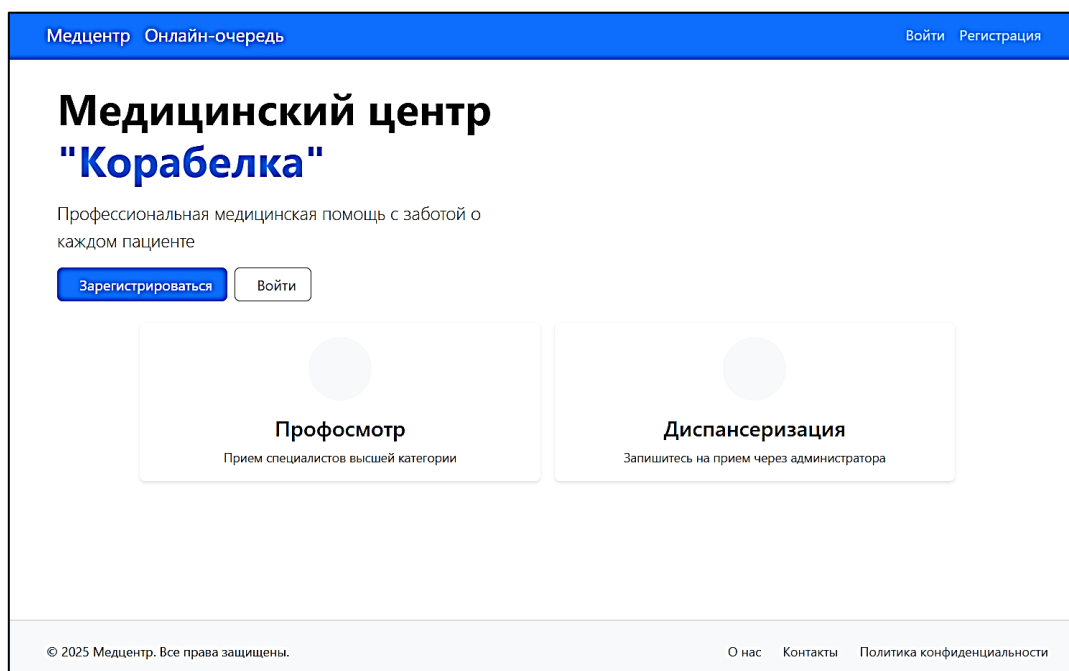


Рисунок 1 – Начальный экран приложения

Архитектура реализованного приложения разработана с учетом возможного масштабирования системы, в которое могут входить как переход с единого фронтенда и бэкенда на их разделение (с помощью разработанного API), так и добавление функциональности для пользователей, например, возможность самостоятельно записываться на прием.

ЛИТЕРАТУРА

1. Как купить домен и использовать доменное имя для размещения сайта или своих сервисов через NGINX/ [URL]: <https://habr.com/ru/articles/716002/> (дата обращения: 22.05.2025).
2. The web framework for perfectionists with deadlines | Django. [URL]: <https://www.djangoproject.com/> (дата обращения: 10.04.2025).

УДК 004.8

А.А. Шматков (4 курс бакалавриата),
Е.В. Комарова, ст. преподаватель

РАЗРАБОТКА ФРЕЙМВОРКА ДЛЯ ЦИФРОВОЙ ОБРАБОТКИ МУЗЫКИ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Современная музыкальная индустрия и аудио обработка сталкиваются с вызовом: существующие решения либо предлагают ограниченный набор ручных инструментов (цифровые звуковые станции), либо являются специализированными и сложными для интеграции в единый конвейер обработки. Задачи разделения аудио на компоненты (стемы), изменения темпа и тона, а также интеллектуальной постобработки часто решаются разрозненными инструментами, что замедляет рабочий процесс и требует от пользователя глубоких технических знаний. Применение нейросетевых моделей, таких как Demucs, открывает новые возможности для качественного анализа и модификации музыки, однако их интеграция в удобное прикладное решение остается нетривиальной задачей [5].

Таким образом, целью данной работы является разработка и реализация модульного фреймворка для цифровой обработки музыки, который объединяет нейросетевые методы разделения на стемы, алгоритмы цифровой обработки сигналов (DSP) для последующего улучшения звука и интуитивно понятный пользовательский интерфейс, обеспечивая полностью локальную и эффективную работу с аудиоматериалом.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Выполнить анализ существующих подходов к нейросетевому разделению аудио на источники и семантическим преобразованиям (изменение темпа и тона), выделив их ограничения при практическом применении.
2. Спроектировать модульную архитектуру фреймворка, разделяющую ядро обработки и пользовательский интерфейс.
3. Реализовать программные модули для загрузки/сохранения аудио, нейросетевой сепарации на стемы, семантических преобразований и расчета метрик качества разделения.
4. Разработать алгоритм адаптивной постобработки разделенных стемов, включающий автоматическое выделение ведущего и бэк-вокалов и частотно-зависимое шумоподавление.
5. Интегрировать реализованные модули в веб-интерфейс на базе Gradio для демонстрации работы фреймворка и интерактивной оценки качества обработки пользователем.

Архитектура предложенного решения построена по двухуровневому принципу. Ядро отвечает за всю логику обработки: модуль ввод/вывода унифицирует загрузку аудио через библиотеки Librosa и SoundFile в Python [1]; модуль разделения источников инкапсулирует вызов предобученной модели из библиотеки Demucs [2]. Ключевой инновацией разработанного фреймворка является встроенный алгоритм постобработки. После получения базовых стемов выполняется их адаптивная обработка. Вокал дополнительно разделяется на ведущий и бэк-вокал с помощью авторского метода на основе звуковой декомпозиции. Далее для каждого типа стема применяется цепочка эффектов: спектральное шумоподавление и частотно-зависимый усилитель. Например, для ведущего вокала выполняется усиление средних частот и адаптивное смягчение резких шипящих согласных, а для ударных – более строгое шумоподавление для подавления затухающих призывов. Это позволяет добиться чистоты звука, сравнимой с работой звукорежиссера.

Важной частью работы является реализация нейросетевого разделения источников звука и последующей цифровой обработки в единой системе. На первом этапе аудиосигнал подается в модель Demucs, представляющей собой сверточно-рекуррентный кодировщик. Модель напрямую обрабатывает временной сигнал, поэтому на выходе каждая ветвь предсказывает его для одного из целевых источников, после чего исходный микс представляется как сумма предсказанных стемов. Входной аудиосигнал автоматически приводится к частоте дискретизации модели и формату: пакет, каналы, фрагменты. Разделение выполняется пакетно-ориентированно с перекрытием сегментов, что позволяет обрабатывать длинные треки, экономно используя память. Полученные стемы сохраняются в виде словаря «источник» - «массив фрагментов» и далее передаются в модуль постобработки. Такой подход позволяет изолировать модуль от остального кода, упрощает отладку и облегчает замену модели при необходимости. Для вокала дополнительно выполняется разделение на ведущий и бэк-вокалы, это достигается преобразованием каналов и наложением фильтров для очищения звука. На втором этапе каждый стем проходит через адаптивное шумоподавление на основе оценки зашумленности. На третьем этапе используется частотный усилитель для вокала: средние и низкие частоты усиливаются, а выбивающиеся шипящие согласные занижаются. Таким образом, фреймворк реализует связный конвейер: нейросетевое разделение источников звука, расщепление вокала на ведущий/бэк, адаптивная постобработка.

Для оценки эффективности разработанного фреймворка был проведен эксперимент. В качестве тестового набора использовалось 10 аудиотреков различных жанров. Основное внимание уделялось сравнению качества разделения и постобработки с базовым выводом модели Demucs. Оценка проводилась как по объективным метрикам (SDR, SAR) [7], так и

путем субъективного прослушивания. В таблице 1 приведены обобщенные результаты для вокальных стемов:

Таблица 1 – Звуковые метрики вокала

Конфигурация	SDR, дБ	SAR, дБ
Demucs	6.8	7.5
Demucs + постобработка	8.1	8.9

Таким образом, предложенный алгоритм постобработки повышает SDR для вокальной партии в среднем на 1.3 дБ и улучшает показатель SAR примерно на 1.4 дБ за счет подавления артефактов разделения и фонового «мусора» в паузах. Субъективная оценка подтвердила более естественное звучание ведущего вокала и уменьшение навязчивых призывов в бэк-вокале.

Перспективы развития фреймворка связаны как с расширением набора используемых моделей, так и с углублением постобработки. Во-первых, возможна интеграция специализированных нейросетей-усилителей для вокала и инструментов, реализованных в виде PyTorch-модулей поверх уже разделенных стемов. Во-вторых, представляется перспективным добавить интерактивную визуализацию (графики метрик SDR по трекам) и механизмы тонкой настройки параметров постобработки пользователем. Наконец, модульная архитектура позволяет в дальнейшем встроить поддержку альтернативных моделей разделения и реализацию автоматического подбора параметров обработки под жанр и характер входного материала, что расширит область применения разработанного фреймворка.

ЛИТЕРАТУРА

1. Python documentation [Электронный ресурс] – URL: <https://docs.python.org/3/> (дата обращения: 10.02.2025).
2. Demucs Library Documentation [Электронный ресурс] – URL: <https://github.com/facebookresearch/demucs> (дата обращения: 10.02.2025).
3. Gradio: Build Machine Learning Web Apps [Электронный ресурс]. – URL: <https://www.gradio.app/> (дата обращения: 10.02.2025).
4. Pyrubberband: Time-stretch and pitch-shift audio [Электронный ресурс] – URL: <https://pypi.org/project/pyrubberband/> (дата обращения: 11.02.2025).
5. Steinmetz C., Uhle C., Everardo F., Mitcheltree C., McElveen J, Jot M., Wichern G. – Audio Signal Processing in the Artificial Intelligence Era, 2025 [Электронный ресурс] – URL: <https://www.merl.com/publications/docs/TR2025-116.pdf> (дата обращения: 17.02.2025).
6. Haibin W., Xuanjun C., Yi-Cheng L., Kai-wei C., Ho-Lam C., Alexander H. L., Hung-yi L. – Towards audio language modeling – an overview, 2024 [Электронный ресурс] – URL: <https://arxiv.org/html/2402.13236v1> (дата обращения: 17.02.2025).
7. A Comparative Analysis of Deep Learning Based Music Source Separation, 2022 [Электронный ресурс] – URL: <https://www.mdpi.com/2076-3417/12/15/7563> (дата обращения: 18.02.2025).

УДК 004.85

Н.А. Щукин (4 курс бакалавриата),
К.А. Туральчук, к.т.н., доцент

РАЗРАБОТКА ТОРГОВОГО БОТА ДЛЯ АЛГОРИТМИЧЕСКОЙ ТОРГОВЛИ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

Современные финансовые рынки характеризуются высокой скоростью изменения котировок, возросшей ролью алгоритмических стратегий и существенной долей автоматизированных сделок в общем объеме торгов. В таких условиях ручное принятие торговых решений становится недостаточно эффективным: возрастает влияние человеческого фактора, увеличивается вероятность ошибок и запаздываний при исполнении сделок. Одним

из подходов к снижению этих рисков и повышению устойчивости торговых стратегий является использование торговых ботов, реализующих алгоритмическую торговлю с применением методов машинного обучения [6].

Целью данной работы является разработка системы для алгоритмической торговли на финансовых рынках с использованием алгоритмов машинного обучения для генерации торговых сигналов.

Для достижения поставленной цели предполагается решение следующих основных задач.

1. Выполнить анализ существующих подходов к построению алгоритмических торговых систем и применению методов машинного обучения к прогнозированию финансовых временных рядов и принятию торговых решений [4].
2. Спроектировать архитектуру торгового бота, включающую модули сбора и предобработки рыночных данных, модуль генерации сигналов на основе моделей машинного обучения, модуль управления позицией и рисками, а также модуль интеграции с биржевым API для исполнения сделок.
3. Реализовать прототип системы, обеспечивающий автоматизированный сбор исторических и потоковых котировок, проведение бэктестинга стратегий и симуляцию исполнения сделок с учётом торговых комиссий.
4. Исследовать применение моделей машинного обучения, в том числе рекуррентных нейронных сетей типа LSTM для краткосрочного прогнозирования цен [1], градиентного бустинга для классификации торговых состояний рынка и алгоритмов глубокого обучения с подкреплением для адаптивного управления позицией [2].
5. Провести экспериментальное сравнение разработанных стратегий с базовыми алгоритмическими подходами по метрикам доходности, риск-профиля и устойчивости на различных рыночных режимах [4].

Разрабатываемый торговый бот имеет модульную архитектуру. В качестве входных данных используются рыночные котировки (цены «открытия», «закрытия», максимальные и минимальные цены, объёмы), а также производные признаки: технические индикаторы, статистические характеристики временных окон, признаки волатильности и ликвидности. Модуль предобработки данных отвечает за очистку, агрегацию и нормализацию временных рядов, а также за формирование обучающих выборок с заданным горизонтом прогнозирования.

Для прогнозирования направленных изменений цен и вероятности наступления торговых событий предполагается использовать модели глубокого обучения. Рекуррентные сети типа LSTM выбираются благодаря их способности учитывать долговременные зависимости во временных рядах и устойчивости к шуму при достаточном объёме данных [5]. На выходе LSTM-модели формируется либо вероятностная оценка направления движения цены, либо непрерывный прогноз изменения цены, который далее преобразуется в торговый сигнал с учётом заданной пороговой логики и ограничений по риску.

Для моделирования дискретного процесса принятия торговых решений и адаптации стратегии под изменяющиеся рыночные условия планируется использование алгоритмов обучения с подкреплением [2]. Агент получает вознаграждение, зависящее от финансового результата и соблюдения ограничений по максимальной просадке и волатильности, и обучается вырабатывать политику управления позицией (объём входа в сделку, размещение стоп-лосс и тейк-профит ордеров, правила выхода из позиции). На этапе реализации предполагается применение варианта глубокого Q-обучения с доработками для учёта транзакционных издержек и проскальзываний [4].

Система управления рисками интегрирована в контур исполнения сделок и включает ограничения по размеру позиции относительно капитала, дневным и недельным лимитам потерь, а также фильтры по ликвидности и волатильности инструмента [3]. Архитектура торгового бота предусматривает возможность как полностью автоматического режима

исполнения, так и полуавтоматического сценария, при котором решения модели подтверждаются пользователем через интерфейс мониторинга.

Для оценки качества разрабатываемого решения предполагается проведение серии экспериментов на исторических данных по нескольким ликвидным инструментам.

Ожидается, что использование сочетания методов глубокого обучения для прогнозирования и обучения с подкреплением для управления позицией позволит повысить соотношение доходности к риску по сравнению с простыми техническими стратегиями, а также обеспечить адаптацию торгового бота к изменяющимся рыночным условиям.

ЛИТЕРАТУРА

1. Fischer T., Krauss C. Deep learning with long short-term memory networks for financial market predictions // *European Journal of Operational Research*. – 2018. – Vol. 270. – P. 654–669.
2. Zhang Z., Zohren S., Roberts S. Deep reinforcement learning for trading // *The Journal of Financial Data Science*. – 2020. – Vol. 2. – No. 2. – P. 25–40.
3. Dixon M., Halperin I., Bilokon P. *Machine Learning in Finance: From Theory to Practice*. – Cham: Springer, 2020. – 613 p.
4. Liu H., Wang X., Li Y. Deep reinforcement learning in algorithmic trading: A review // *Expert Systems with Applications*. – 2023. – Vol. 224. – 119989.
5. Цховребов М. Р., Мураев А. Г. Разработка архитектуры автономной торговой системы с использованием LSTM для прогнозирования цен фьючерсов // *Актуальные исследования*. – 2025. – № 25 (260). – С. 45–52.
6. Иванов А. В., Петров Д. С. Использование методов машинного обучения в задачах алгоритмической торговли на фондовом рынке // *Автоматизация и современные технологии*. – 2021. – № 7. – С. 32–40.

УДК 004.622

Д.С. Шушин (2 курс магистратуры),
П.Д. Дробинцев, к.т.н., доцент

РАЗРАБОТКА АНАЛИТИЧЕСКИХ ВИТРИН ДЛЯ АГРЕГАЦИИ И АНАЛИЗА НЕУНИФИЦИРОВАННЫХ ПРОИЗВОДСТВЕННЫХ ДАННЫХ В НЕФТЕГАЗОВОМ СЕКТОРЕ

Ежедневная деятельность инженеров нефтегазовых предприятий требует оперативного выявления рисков, контроля работы оборудования и оценки состояния добывающих скважин. Со стороны операторов данных значимая информация о производственных процессах фиксируется в виде суточных, еженедельных и ежемесячных отчетов, имеющих как стандартизированные параметры, так и ненормированные текстовые комментарии [1]. Высокая частота поступления таких отчетов и большие объемы накопленной информации в сочетании с неструктурированным характером отдельных записей делают невозможным полное ручное извлечение полезных сведений. Эта проблема приводит к задержкам при принятии решений, выявлении недостатков и определении областей улучшения процесса производства [2].

Помимо текстовых комментариев, производственные данные характеризуются высокой неоднородностью форматов и структур: они представлены в реляционных и нереляционных базах, текстовых файлах, устаревших проприетарных форматах. Дополнительную сложность создают семантические различия в отчетах: в обозначениях, единицах измерения и методиках расчета однотипных параметров, что затрудняет автоматическое сопоставление информации из разных источников и расчет общих производственных показателей. В связи с этим применение онтологий, таксономий и формальных моделей знаний становится важным требованием при построении связного аналитического пространства [3]. Наличие данного требования определяет актуальность разработки специализированной системы аналитических витрин, адаптированной к работе с неунифицированными данными нефтегазовой отрасли.

Целью работы является разработка программной системы автоматизированного построения аналитических витрин, адаптированной для работы с неоднородными источниками информации в нефтегазовой отрасли, позволяющей повысить эффективность анализа производственных данных. Для достижения поставленной цели решены следующие задачи:

1. Проведен анализ особенностей неунифицированных данных и существующих подходов к их обработке.
2. Сформулированы функциональные и нефункциональные требования к системе.
3. Разработана архитектура и реализованы ключевые модули загрузки и валидации данных, а также формирования и визуализации витрин.
4. Проведено тестирование, подтверждающее корректность работы предложенного решения.

В основу архитектуры системы заложены принципы технологической открытости и модульности, обеспечивающие гибкость, независимость от проприетарных решений и возможность адаптации системы под специфические задачи нефтегазовой отрасли. В качестве СУБД используется PostgreSQL [4] с расширением PostGIS [5]. Такая связка позволяет хранить как иерархию производственных объектов, так и их пространственную привязку, необходимую для геоаналитики. Серверная логика реализована на Python, веб-интерфейс построен с использованием фреймворка Dash [6], обеспечивающего создание интерактивных дашбордов с высокой степенью настраиваемости. Архитектура развертывания (рисунок 1) включает выделенные серверы базы данных, авторизации и приложений, что гарантирует масштабируемость и отказоустойчивость. Используемая структура позволяет независимо масштабировать компоненты при росте нагрузки и упрощает развертывание в различных средах.

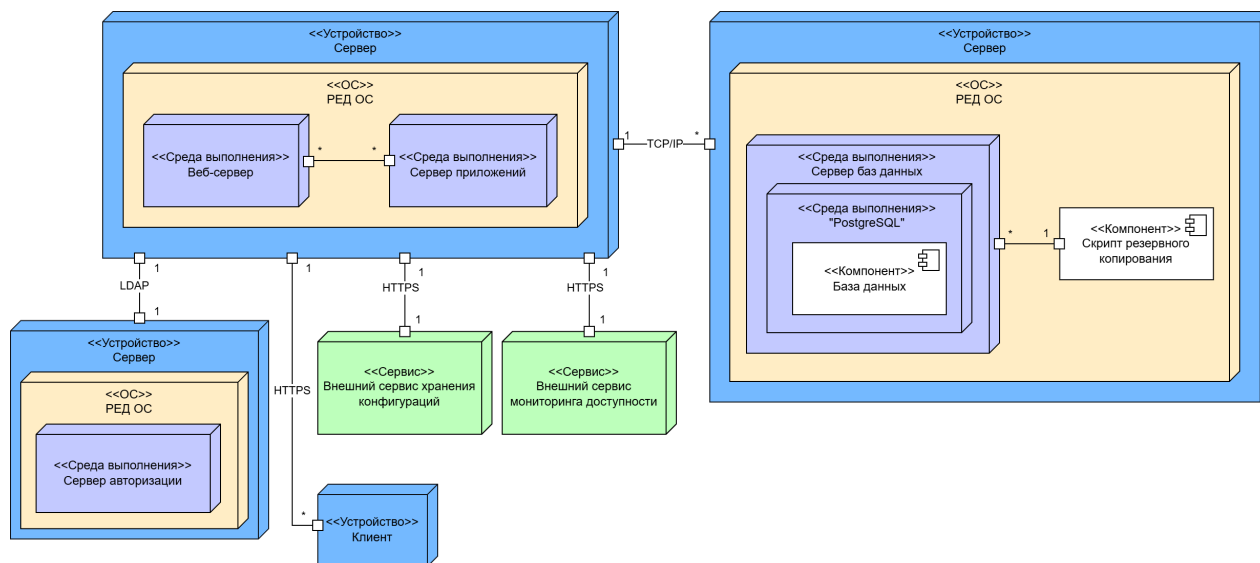


Рисунок 1 – Диаграмма развертывания системы

Загрузка информации в систему выполняется через веб-интерфейс путем отправки CSV-файлов, структура которых соответствует predetermined шаблонам. На этапе загрузки проводится автоматическая валидация: контроль структуры, типов данных и ссылочной целостности со справочниками. Корректные данные проходят многоступенчатое согласование профильными специалистами, после чего попадают в аналитическое хранилище. Для обеспечения производительности запросов используется слой материализованных витрин, где данные агрегированы по временным периодам и иерархии производственных объектов.

Разработанный интерфейс визуализации обеспечивает представление производственных показателей в виде интерактивных дашбордов, включающих временные ряды, картографические слои и табличные данные. Пользовательские фильтры позволяют

динамически изменять уровень детализации и выбирать рассматриваемые объекты и периоды без необходимости написания запросов.

Результатом работы является система аналитических витрин, ориентированная на консолидацию и анализ неунифицированных производственных данных нефтегазовой отрасли. Предложенные архитектурные решения и реализованные модули обеспечивают автоматизацию процессов загрузки, валидации и агрегации данных, позволяющую сократить время анализа и повысить обоснованность управленческих решений. Дальнейшее развитие системы может включать расширение круга источников данных и внедрение методов предиктивной аналитики.

ЛИТЕРАТУРА

1. Lee M. X., Marlot M. Information Retrieval from Oil and Gas Unstructured Data with Contextualized Framework // EAGE Digitalization Conference and Exhibition. – 2023. – Vol. 2023. – P. 1-5. – DOI 10.3997/2214-4609.202332039
2. Пономарёв К. К. Методология реализации цифровой трансформации в компаниях нефтегазового сектора // Инновации и инвестиции. – 2023. – №10. – С. 363-369. – DOI 10.24412/2307-180X-2023-10-363-369
3. Foidl H., Golendukhina V., Ramler R., Felderer M. Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers // Journal of Systems and Software. – 2024. – Vol. 207. – DOI 10.1016/j.jss.2023.111855
4. PostgreSQL: The world's most advanced open source database. [Электронный ресурс] Режим доступа: <https://www.postgresql.org> (дата обращения: 22.02.2026)
5. PostGIS. [Электронный ресурс] Режим доступа: <https://postgis.net> (дата обращения: 10.03.2026)
6. Dash: PyPI [Электронный ресурс] Режим доступа: <https://pypi.org/project/dash/> (дата обращения: 11.03.2026)

УДК 681.3.06

А.И. Щеглов (4 курс бакалавриата),
В.А. Семенова-Тян-Шанская, к.т.н., доцент

РАЗРАБОТКА СПРАВОЧНИКА ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ ПО СУДОСТРОЕНИЮ НА ОСНОВЕ НОРМАЛИЗАЦИИ РАЗНОРОДНЫХ ДАННЫХ

В условиях цифровой трансформации судостроения ключевым требованием является обеспечение однозначности и структурированности данных. Однако анализ текущего состояния нормативно-справочной информации (НСИ) в области классификации судов выявил серьезные системные противоречия. Открытые отраслевые регистры — Российский морской регистр судоходства (РМРС) и Регистровая книга Российского классификационного общества (РКО) содержат данные низкого качества. Вариативность формулировок, неконтролируемое использование сокращений, орфографические ошибки и нарушение целостности данных делают невозможной их автоматизированную обработку и интеграцию в современные информационные системы [1, 2].

Для решения этой проблемы ключевым этапом работы стала разработка алгоритма нормализации данных для автоматизированной обработки неструктурированных текстовых записей из Регистровой книги РКО. Разработанный алгоритм, основанный на подходах компьютерной лингвистики к извлечению информации из текстов [3, 4], включает: предобработку текста [5], лексический анализ на основе специализированных словарей и финальное структурирование с морфологическим согласованием [6].

На входе алгоритм получает «сырую» строку, содержащую проблемы, характерные для данных РКО, а на выходе формирует структурированную и нормализованную запись. Для практической реализации алгоритма и визуализации результатов было разработано приложение с использованием веб-технологий (HTML, CSS, JavaScript), обеспечивающее

удобный графический интерфейс, в то время как логика обработки данных реализована на языке программирования Python. Пример обработки представлен в таблице 1.

Таблица 1 – Пример нормализации данных

Исходные данные	Обработанные данные
Баржа. Бункерная баржа	Баржа Бункерная
Несамоходная бункерная баржа	Баржа Бункерная неСамоходная
Грузопассажирский дебаркадер	Дебаркадер Грузопассажирский
Грузопассажирский дебаркадер, швартовка судов	Дебаркадер Грузопассажирский Швартовка судов
Несамоходный земснаряд	Земснаряд неСамоходный
Дноуглубительный н/с земснаряд	Земснаряд неСамоходный Дноуглубительный
Пассажирское судно. Пассажирский водомётный катер	Катер Водометный Пассажирский
Н/с дизель-электрический плавкран	Кран Дизель-электрический Плавучий неСамоходный
Несамоходный дизельэлектрический плавкран	Кран Дизель-электрический Плавучий неСамоходный
Вспомогательное судно. Теплоход разъездной	Теплоход Вспомогательный Разъездной

Результатом применения алгоритма стал очищенный перечень уникальных типов судов. На его основе была построена иерархическая структура, обеспечивающая логическую организацию терминов по принципу "от общего к частному". Фрагмент полученной иерархии представлен на рисунке 1.

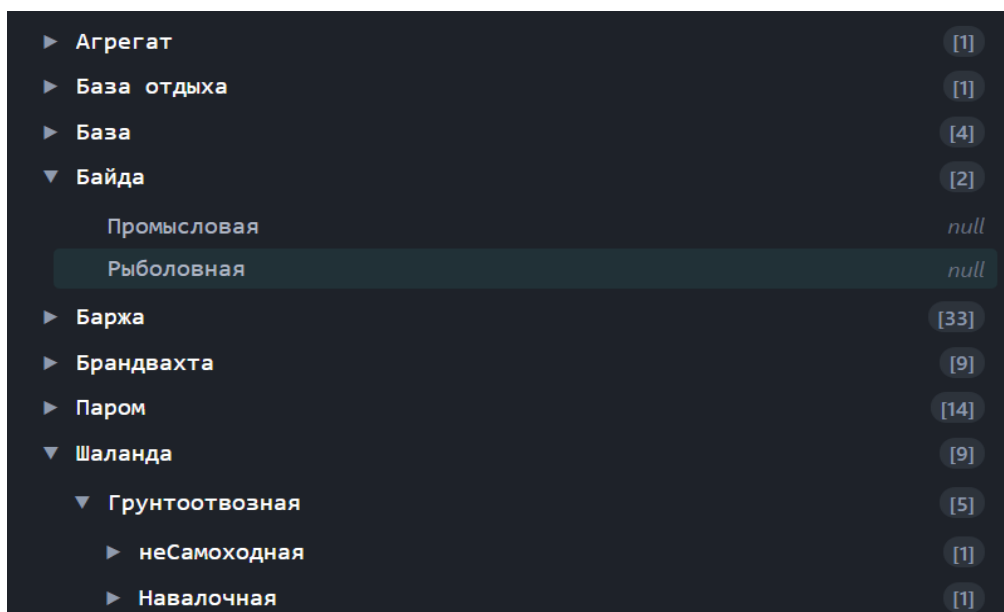


Рисунок 1 – Фрагмент иерархической структуры типов и назначений судов

Разработанная структура решает проблему терминологической неоднозначности, устраняет дублирование и создает основу для внедрения цифровых моделей в судостроительной отрасли. В дальнейшем планируется расширение классификатора и его интеграция в системы управления данными.

ЛИТЕРАТУРА

1. Тянянский С.С. Модель согласования данных при интеграции информационных систем // Радиоэлектроника и информатика. 2006. №2. – С. 85-86.
2. Султанова С.Ш., Муталлиева К.И. Информационная база и способы её организации // Academy. 2016. №9 (12). – 7 с.

3. Клышинский Э.С., Кочеткова Н.А. Метод извлечения технических терминов с использованием меры странности // Новые информационные технологии в автоматизированных системах. 2014. №17. – 366 с.
4. Автоматическая обработка текстов на естественном языке и анализ данных : учеб. пособие / Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. — М.: Изд-во НИУ ВШЭ, 2017. — 269 с.
5. Акжолов Р.К., Верига А.В. ПРЕДОБРАБОТКА ТЕКСТА ДЛЯ РЕШЕНИЯ ЗАДАЧ NLP // Вестник науки. 2020. №3 (24). – С. 66-67.
6. Федюченко Л.Г., Виноградова Е.В. Сравнительно-сопоставительный категориальный анализ терминосистем «Судостроение» и “Shipbuilding” // Вестник Пермского университета. Российская и зарубежная филология. 2024. №2. – С. 62-65.

УДК 004.4

Ян Кай (2 курс магистратуры),
Б.М. Медведев, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ АНАЛИЗА И ПРОГНОЗИРОВАНИЯ ЦЕН НА СЕЛЬСКОХОЗЯЙСТВЕННУЮ ПРОДУКЦИЮ НА ОСНОВЕ МОДЕЛИ SARIMA

Анализ и прогнозирование цен на сельскохозяйственную продукцию является важной задачей для аграрного сектора и продовольственного рынка. Колебания цен оказывают значительное влияние на экономические решения производителей, торговых организаций и государственных структур. В условиях роста объёмов данных и цифровизации аграрных рынков возрастает необходимость разработки информационных систем, позволяющих автоматически анализировать исторические данные и формировать прогнозы цен [1].

Исследования показывают, что методы анализа временных рядов широко применяются для прогнозирования цен на сельскохозяйственную продукцию и позволяют выявлять закономерности изменения ценовых показателей [2]. В работе для прогнозирования используется модель сезонной авторегрессии и скользящего среднего SARIMA, позволяющая учитывать сезонные колебания цен на сельскохозяйственную продукцию [3]. Модель строится на основе анализа временных рядов исторических цен, а её параметры определяются с использованием статистического анализа автокорреляционной и частичной автокорреляционной функций. Подобные методы временных рядов успешно применяются при анализе динамики цен на сельскохозяйственную продукцию [4].

Несмотря на значительное количество исследований, посвящённых применению моделей временных рядов для прогнозирования цен на сельскохозяйственную продукцию, многие работы сосредоточены преимущественно на анализе методов прогнозирования и оценке точности моделей, тогда как вопросы интеграции алгоритмов прогнозирования в практические информационные системы остаются недостаточно проработанными. В связи с этим целью данной работы является разработка подхода к анализу и прогнозированию цен на сельскохозяйственную продукцию, объединяющего методы временных рядов и программные средства обработки данных в рамках единой информационной системы. Для достижения поставленной цели решаются задачи организации получения и предварительной обработки данных о ценах, построения модели прогнозирования на основе SARIMA, а также разработки программной архитектуры системы, обеспечивающей хранение данных, выполнение прогнозирования и визуализацию результатов.

Для практической реализации метода разработана информационная система анализа и прогнозирования цен. Архитектура системы, приведенная на рисунке 1, включает уровень представления, серверный уровень, модуль аналитики данных и модуль хранения данных. Модуль аналитики обеспечивает получение и предварительную обработку данных, прогнозирование на основе модели SARIMA и формирование результатов. Обработка данных реализована на языке Python с использованием библиотеки pandas, а прогнозирование — с

использованием библиотеки statsmodels. Хранение данных и результатов прогнозирования осуществляется в базе данных MySQL. Серверная часть системы построена на основе фреймворка FastAPI и предоставляет REST API для взаимодействия с клиентским приложением. Пользовательский интерфейс реализован как веб-приложение на базе Vue3 с использованием библиотеки ECharts для визуализации цен и результатов прогнозирования. Развертывание системы осуществляется в контейнерной среде Docker с поддержкой мониторинга и журналирования [5].

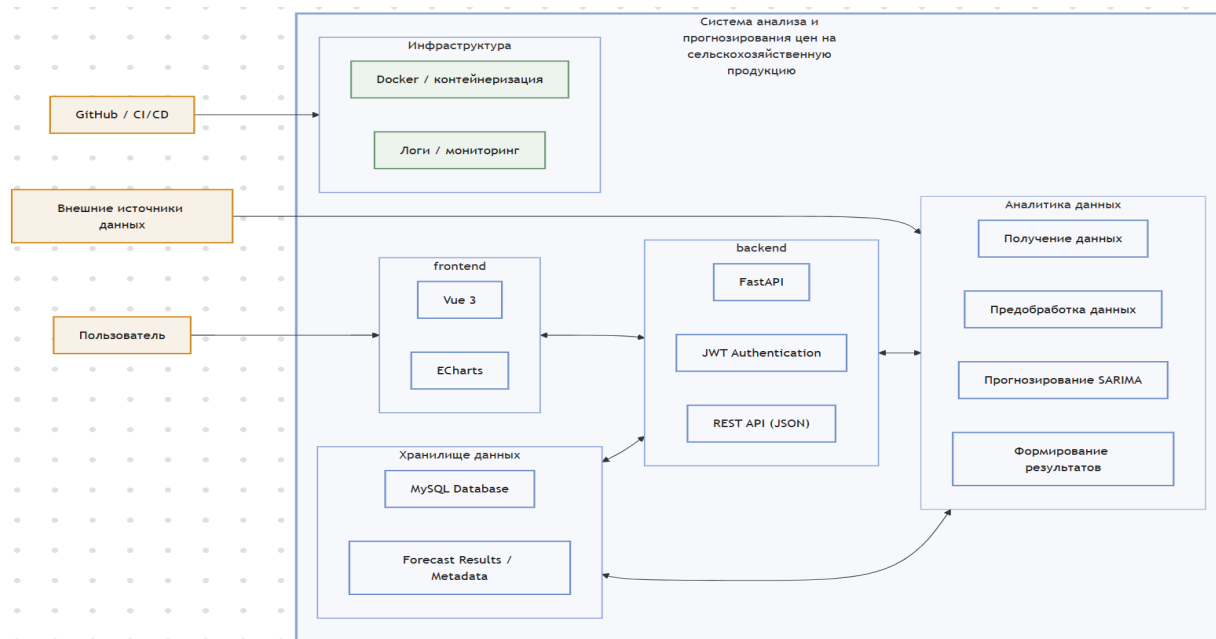


Рисунок 1 – Архитектура системы

Разработанная система обеспечивает автоматизированный анализ исторических данных и построение краткосрочных прогнозов цен на горизонте до 12 месяцев с ожидаемой средней погрешностью прогнозирования около 5–20 %, а также визуализацию результатов через веб-интерфейс. Подобные информационные системы могут использоваться для мониторинга ценовой динамики и поддержки аналитических инструментов аграрного рынка [6]. Развитие методов прогнозирования и информационных систем анализа данных является важным направлением исследований в области аграрной экономики и анализа рынков [7].

ЛИТЕРАТУРА

1. Sun F., Meng X., Zhang Y., Wang Y., Jiang H., Liu P. Agricultural product price forecasting methods: a review // Agriculture. 2023. Vol. 13, № 9. Art. 1671. DOI: 10.3390/agriculture13091671.
2. Zhao C. A review of agricultural product price forecasting research // BIO Web of Conferences. 2024. Vol. 142. Art. 01002. DOI: 10.1051/bioconf/202414201002.
3. OECD, FAO. OECD-FAO Agricultural Outlook 2025–2034. Paris: OECD Publishing, 2024.
4. 邵华英. 基于 ARIMA 模型的粮食价格波动分析 // 社会科学前沿. 2023. T. 12, № 5. С. 2470–2478. DOI: 10.12677/ass.2023.125334.
5. 王章琴, 蓝尉太, 张淑亭, 刘芋良, 贾礼平. 基于 ARIMA 模型的蔬菜价格预测研究 // 统计学与应用. 2025. T. 14, № 7. С. 221–231. DOI: 10.12677/sa.2025.147199.
6. 周晓娅, 何松芝. 基于 Python 技术对农产品价格预测及系统开发 // 人工智能与机器人研究. 2024. T. 13, № 3. С. 662–672. DOI: 10.12677/airr.2024.133067.
7. Zhang N., An Q., Zhang S., Ma H. Price prediction for fresh agricultural products based on a boosting ensemble algorithm // Mathematics. 2025. Vol. 13. Art. 71. DOI: 10.3390/math13010071.

ИЕРАРХИЧЕСКАЯ СИСТЕМАТИЗАЦИЯ ГЕНЕРАЦИИ ЦЕЛЕОРИЕНТИРОВАННЫХ ПРОЦЕДУРНЫХ АНИМАЦИЙ

Современные способы создания анимаций для трёхмерных (3D) персонажей включают в себя анимационные клипы, сопоставлением движения [1] и процедурной анимацией, основанной на обратной кинематике (ОК). Эти методы эффективны, общеприняты и повсеместны, но весьма трудоемки и ресурсоемки. Для их реализации применяется интеграция процедурных анимаций в существующий набор для адаптации к конкретным ситуациям. Подобные процедурные подходы к анимации активно исследуются и дорабатываются, привнося так же больше авторства при помощи параметризации процедур [2]. В современных реалиях не смотря на широкое применение процедурных анимаций как независимых анимаций, так и дополнений к традиционным анимационным клипам все ещё отсутствует общий подход для их реализации, в следствии чего при необходимости добавления в видеоигру или другой продукт процедурных анимаций разработчику приходится реализовывать их «с нуля».

В настоящей работе исследуются возможность систематизации процедурных анимаций, разделение анимации на разные иерархические уровни для лучшего структурирования и модульности анимации и возможностей генерации процедурных анимаций с помощью комбинации атомарных объектов ответственных за анимацию на разных уровнях иерархии.

Для передвижения персонажей широко используется метод скелетной анимации, довольно сложный и трудоемкий [3]. Так как существует множество вариантов реализации скелета, а также невероятное множество дополнительных модификаторов как для отдельных костей, так и для скелета, изменять или создавать новый механизм или систему для движения отдельных вершин модели представляется излишним. Вместо этого, типичные системы преобразования модели для анимации следует интегрировать в качестве абстракции. Лучшим подходом для поддержки множества реализаций и модификаций управления скелета, в частности обратной кинематики, включая множества реализации ОК, будет система, в которой будет производиться модификация координат, не привязанных к конкретным костям или другим объектам, напрямую связанных с моделью [4]. Вместо этого, будет произведена модификация трёхмерных координат, а также поворота и размера (полной матрицы трансформации) абстрактного объекта. Назовём такой объект исходной точкой. В большинстве ситуаций исходная точка требует только пространственные координаты, определяющие её положение в пространстве, но в отличии от типичного математического объекта “точка” также имеет вектор поворота и размера, которые необходимы для полного контроля над процедурной анимацией. Такой объект, при помощи интерфейса, будет модифицировать положения точки в соответствии с процедурной анимацией, к которому уже конкретная реализация преобразования модели, будь она основана на методе скелетной анимации или нет, будет привязана, наследуя преобразование объекта. Таким образом достигается гибкость и модульности системы, позволяя при помощи подобной системы анимации двигать отдельные вершины, кости и другие объекты.

Для полной системы процедурной анимации необходимо определить сами процедуры. Для этого разделим анимацию на отдельные уровни следующим образом: движение точки к цели, система процедур, система генерации и система взаимодействия.

Движение и динамика точки к цели (Система движения) являются основой движения исходных точек к точкам-целям. Для этой системы достаточно простой линейной интерполяции между координатами исходной точки и точки цели. Так же эту систему можно заменить на более подходящую под нужды аниматора как для всех анимаций, так и для отдельных атомарных движений. Для достижения лучшего качества и параметризации

рекомендуется использовать следующую динамику второго порядка (second order dynamics) [5] т. к. в связи со вторым законом Ньютона большинство систем реального мира являются системами второго порядка.

Для системы способной на создание, редактирования и адаптации к окружению движений и динамик необходимо определить интерфейс, меняющий позицию и преобразование точек-целей, через который система движения будет получать цель для движения. Для этого достаточно определить единственный абстрактный метод, который будет отвечать за обновление точки-цели каждый шаг анимации.

Для адаптивности и разнообразия набора анимаций следует включить возможность редактирования параметров динамики, методов, цепочек методов и наборов методов. Для этого определим интерфейсы замещения и правил для замещения отдельных параметров, методов и цепочек методов, а также их комбинаций и групп. Так как анимация является творческим и субъективным предметом исследования, конкретные реализации правил замещения будут оставлены для пользователя системы, за исключением возможного набора подобных правил для часто используемых анимаций.

Система взаимодействия является сугубо программным набором интерфейсов для систем, представленных выше. Стоит отметить, что паттерн программирования инъекция зависимостей представляется оптимальным выбором для реализации.

Процедурные анимации широко используются и систематизация этого подхода к анимированию отличное дополнение к набору инструментов разных аниматоров, в особенности для разработчиков видеоигр. Систематизация процесса создания процедур для таких анимаций является обоснованным. Иерархическая структура позволяет разделить процесс создания анимации на отдельные структуры и системы упрощая редактирование и сам процесс создания.

Представленная система отлично расширяет спектр возможностей при создании анимаций, особенно в тех случаях, когда реактивность на окружение играет важную роль в анимации. Процедурные анимации хоть и не являются заменой традиционным, все же представляет собой большое пространство для креативных идей и качественных продуктов в современном рынке видеоигр, который растет с каждым годом.

ЛИТЕРАТУРА

1. Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. Learned Motion Matching. ACM Trans. Graph. 39, 4, Article 1 (July 2020), 13 pages. <https://doi.org/10.1145/3386569.3392440>, 2020. P. 1-2.
2. Yuke Fu. Exploration of a Hybrid Procedural Animation Framework. Thesis. Department of Design Sciences. Sweden. 2025. P. 10.
3. David Rosen. Animation Bootcamp: An Indie Approach to Procedural Animation. Presentation. GDC. 2014. T. 1:40.
4. Xiao Xian. Skeletal Character Animation from Examples. Nanyang Technological University. Thesis. 2009. P. 50.
5. t3ssel8r. Giving Personality to Procedural Animations using Math. YouTube. — 2022. <https://youtu.be/KPoeNZZ6H4s>

УДК 004.453

Н.З. Айналияев (4 курс бакалавриата),
Е.В. Комарова, ст. преподаватель

РАЗВЕРТЫВАНИЕ SPA-ПРИЛОЖЕНИЙ В ОБЪЕКТНОМ ХРАНИЛИЩЕ КАК АЛЬТЕРНАТИВА КЛАССИЧЕСКОМУ ВЕБ-СЕРВЕРУ В DEVOPS-ПЛАТФОРМЕ

В современной практике разработки DevOps-платформы часто включают клиентские Single Page Applications (SPA), которые традиционно разворачиваются в виде Docker-контейнеров с веб-сервером Nginx внутри Kubernetes [1]. Однако SPA после завершения этапа сборки представляет собой статический набор файлов (HTML, JS, CSS), не требующий серверной логики на стороне фронтенда. Использование полноценного пода Kubernetes для раздачи статики является архитектурно избыточным решением: оно увеличивает нагрузку на вычислительные ресурсы кластера и усложняет конфигурацию инфраструктуры [1]. В связи с этим в данной работе исследована возможность перехода на S3-совместимые объектные хранилища как на более эффективную альтернативу в составе DevOps-платформы.

Традиционные методы разворачивания требуют поддержки Docker-образов, управления конфигурациями Ingress и постоянного выделения вычислительных ресурсов для обслуживания статического контента. Объектное хранилище (S3) функционирует по модели “оплата за использование”, исключая расходы на простаивающие инстансы. S3-хранилища обеспечивают доступность до 99,99% и горизонтальную масштабируемость без дополнительной настройки [3], а нативная интеграция с CDN позволяет доставлять контент из ближайших к пользователю точек присутствия [1, 4]. Сравнение подходов приведено в таблице 1.

Таблица 1– Сравнение подходов разворачивания SPA-приложений.

Критерий	Nginx (Docker)	Kubernetes Pod	Объектное хранилище
Операционные расходы	Высокие	Высокие	Низкие (трафик и хранение)
Сложность инфраструктуры	Средняя	Высокая	Низкая
Масштабируемость	Ручная / НРА	Автоматическая	Автоматическая
Управление версиями	Теги образов	Стратегии деплоя K8s	Метаданные объектов
Поддержка CDN	Доп. настройка	Доп. настройка	Нативная

Для автоматизации доставки SPA используется пайплайн непрерывной интеграции и доставки, интегрированный с объектным хранилищем и CDN (рисунок 1). На этапе CI выполняются сборка приложения и автоматические проверки, после чего формируются статические артефакты с хеш-суммами в именах файлов (cache busting), что позволяет эффективно кэшировать ресурсы в браузере и CDN [3]. На этапе доставки артефакты загружаются в объектное хранилище в отдельный префикс версии (например, releases/<commit>), что обеспечивает возможность хранения нескольких релизов и быстрого отката. После предварительного разворачивания и базовой проверки работоспособности выполняется переключение активной версии приложения посредством обновления index.html, который

публикуется последним для обеспечения согласованности загружаемых ресурсов. Завершающим шагом является инвалидация кэша CDN для index.html, обеспечивающая оперативную доставку новой версии пользователям [3]. Общая архитектура хостинга SPA в объектном хранилище и CDN показана на рисунке 2.

По итогам работы получены следующие результаты:

1. **Снижение операционных расходов.** Отказ от Kubernetes-пода устраняет постоянные затраты. Стоимость хранения и трафика ниже содержания пода в кластере [1].
2. **Ускорение FCP.** CDN дает задержку до 50 мс против 150–300 мс при раздаче из одного региона для удаленных пользователей [3, 5].
3. **Упрощение инфраструктуры.** Исключены Docker-образ веб-сервера, Deployment, Service и Ingress – меньше манифестов и точек отказа.
4. **Переносимость.** S3-совместимый API позволяет без изменений пайплайна использовать Yandex Object Storage, AWS S3 и другие S3-сервисы [2].

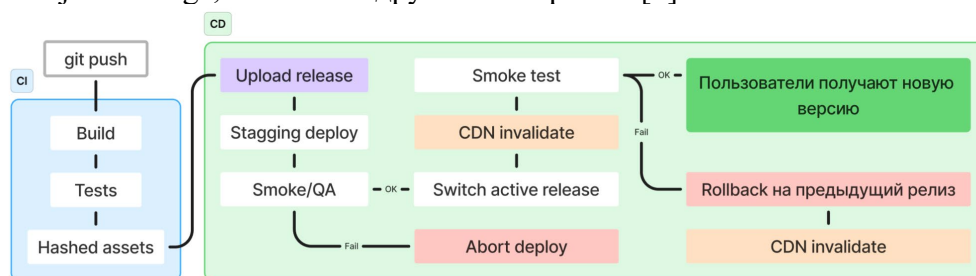


Рисунок 1 – Пайплайн доставки SPA в объектное хранилище

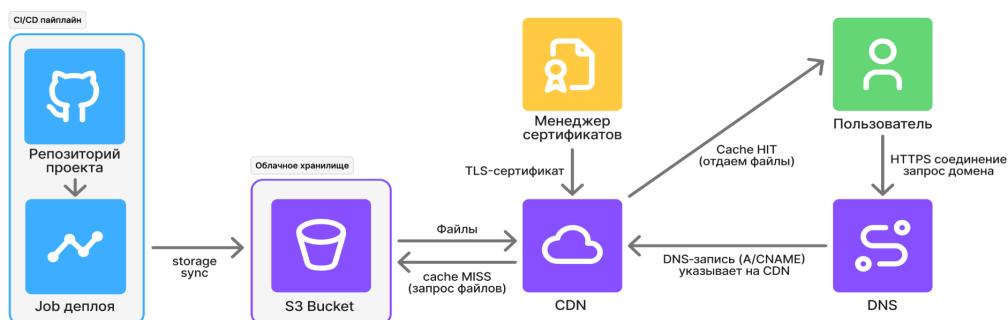


Рисунок 2 – Архитектура хостинга SPA на S3 и CDN.

Использование объектного хранилища для хостинга SPA-приложений является архитектурно обоснованным решением для DevOps-платформ. Оно позволяет значительно снизить операционные расходы и сложность сопровождения, сохраняя при этом высокую доступность и производительность, недостижимую классическими методами без существенных затрат на инфраструктуру. Стандартизация на S3-совместимом API обеспечивает независимость от конкретного облачного провайдера и воспроизводимость процесса разворачивания.

ЛИТЕРАТУРА

1. Singh R. Automating Static Website Deployment with GitHub Actions and AWS S3 // DEV Community. – 2025 – URL: <https://dev.to/aws-builders/automating-static-website-deployment-with-github-actions-and-aws-s3-12ad> (дата обращения: 08.12.2025).
2. Yandex Cloud. Статический сайт в Yandex Object Storage // Документация Yandex Cloud. — 2025 — URL: <https://yandex.cloud/ru/docs/storage/tutorials/static/>
3. Dhandala N. How to Deploy a React App to AWS S3 and CloudFront / N. Dhandala // OneUptime Blog. – 2026 – URL: <https://oneuptime.com/blog/post/2026-02-12-deploy-react-app-to-aws-s3-and-cloudfront/view> (дата обращения: 07.02.2026).
4. Savvaleukhin S. Как развернуть React приложение с помощью AWS S3 и CloudFront // Хабр. – 2022 – URL: <https://habr.com/ru/articles/693528/>

5. Руссин Ч. Автоматическое развертывание Vue.js-приложений // Tproger. — 2019. — URL: <https://tproger.ru/translations/vue-auto-deploy> (дата обращения: 16.02.2026).

УДК 004.4

А.Ф. Арбаева (4 курс, бакалавриат),
Т.В. Леонтьева, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО КАРКАСА ДЛЯ ПОСТРОЕНИЯ СЕРВЕРНО-УПРАВЛЯЕМОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА НА ЯЗЫКЕ DART И ОЦЕНКА ЕГО ВЛИЯНИЯ НА СКОРОСТЬ ВНЕДРЕНИЯ ФУНКЦИОНАЛЬНОСТИ, СОПРОВОЖДАЕМОСТЬ И ПРОИЗВОДИТЕЛЬНОСТЬ КЛИЕНТСКИХ ПРИЛОЖЕНИЙ

Современные мобильные приложения характеризуются высокой динамичностью пользовательского интерфейса, обусловленной необходимостью регулярного обновления промо-экранов, сценариев онбординга, витринных представлений, форм ввода и элементов, используемых в А/В-экспериментах. В рамках традиционной модели, при которой интерфейс статически встроен в клиентский бинарный пакет, любое существенное изменение UI требует выпуска новой версии приложения, прохождения процедуры модерации в магазинах приложений и последующего распространения обновления среди пользователей. Даже при относительно высокой скорости проверки совокупное время вывода изменений в эксплуатацию ограничено последовательностью этапов «разработка — сборка — публикация — доставка — обновление на устройстве пользователя». Так, по данным Apple, в среднем 90% отправок проходят ревью менее чем за 24 часа. Вместе с тем, политики магазинов приложений накладывают прямые ограничения на попытки обхода процедуры ревью посредством загрузки и исполнения кода после публикации: в частности, правило 2.5.2 [1] запрещает скачивание, установку и выполнение кода, который вводит новую либо изменяет существующую функциональность приложения. Следовательно, при необходимости ускорения вывода UI-изменений без нарушения требований платформ дистрибуции возникает потребность в подходе, допускающем изменение описания интерфейса и данных без поставки нового исполняемого кода на устройство. Подход Server-Driven UI удовлетворяет данному требованию при условии, что он реализован как интерпретация декларативного формата на основе заранее определённого белого списка компонентов.

Анализ существующих решений показывает, что концепция Server-Driven UI уже получила распространение в промышленных мобильных продуктах, где она используется для ускорения вывода функциональности, унификации интерфейсных сценариев между платформами и поддержки экспериментов. Подобные подходы описываются в инженерных практиках крупных компаний, а также представлены в виде open-source-решений, таких как DivKit, Adaptive Cards и json_dynamic_widget [2–4]. Однако для экосистемы Dart и особенно Flutter сохраняется потребность в фреймворке, который одновременно обеспечивает строгую типизацию UI-контрактов, безопасную интерпретацию конфигураций, поддержку обратной совместимости клиентов, кэширование, наблюдаемость и возможность воспроизводимой оценки влияния на производительность. В отличие от частных библиотек рендеринга, такой фреймворк должен рассматриваться как целостная инфраструктура, включающая формат описания экранов, клиентский рантайм, серверный слой композиции и средства измерения эксплуатационных характеристик.

Таким образом, целью статьи является описание архитектурных и программных решений, положенных в основу разработанного фреймворка Server-Driven UI для клиентских приложений на Dart/Flutter [5], а также анализ его влияния на скорость внедрения функциональности, сопровождаемость и производительность клиентских приложений.

В работе в качестве архитектурного решения выбрана существующая модель разделения серверной и клиентской ответственности (рисунок 1), применяемая в системах с

декларативным описанием интерфейса. На серверной стороне располагаются компоновщик пользовательского интерфейса, реестр схем и версий, механизмы сегментации и A/B-экспериментов, а также источники предметных данных. Клиентское приложение получает по HTTP/REST декларативное описание экрана в формате JSON, после чего выполняет его валидацию по схеме, декодирование во внутреннее представление и рендеринг в заранее определённый набор Flutter-виджетов. В клиентской части выделяются слои транспорта, кэширования, валидации, декодирования, рендеринга, диспетчеризации событий и телеметрии. Для повышения устойчивости предусматриваются ETag-валидация, локальный кэш, fallback-механизм на статический экран и политика обработки неизвестных компонентов без аварийного завершения приложения. Безопасность обеспечивается запретом на интерпретацию произвольного кода, использованием белого списка компонентов и дополнительным контролем целостности конфигурации.

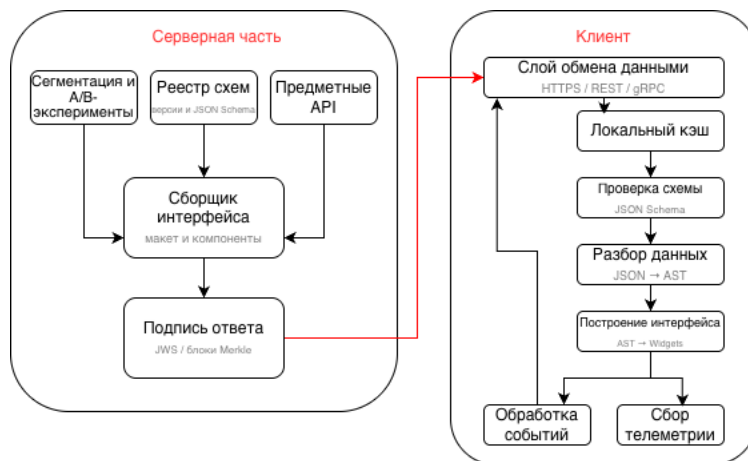


Рисунок 1 – Архитектура системы

Разработка велась в виде прототипного фреймворка на языке Dart с использованием Flutter для клиентской части. В качестве формата описания интерфейса используется JSON, что обеспечивает простоту доставки, интеграцию со стандартными веб-протоколами и возможность формальной проверки структуры данных. В прототипе выделены ядро модели UI, модуль парсинга и валидации, слой транспортного взаимодействия и библиотека рендеринга, преобразующая дерево декларативных узлов в набор Flutter-виджетов. Для снижения влияния на UI-поток тяжёлые операции разбора конфигураций могут выноситься в отдельные isolate-процессы, а оценка производительности выполняется средствами Flutter DevTools и integration_test. Сопоставление baseline-варианта со статическим UI и варианта с SDUI позволяет оценить выигрыш в скорости вывода UI-изменений и зафиксировать накладные расходы, связанные с сетевой доставкой, парсингом и построением интерфейсного дерева.

ЛИТЕРАТУРА

1. Политика магазина приложений (AppStore). [Электронный ресурс] Режим доступа: <https://developer.apple.com/app-store/review/guidelines/#software-requirements> (Дата обращения: 01.03.2026)
2. DivKit documentation. [Электронный ресурс] Режим доступа: <https://divkit.tech/docs/ru/> (Дата обращения: 01.03.2026)
3. Adaptive Cards documentation. [Электронный ресурс] Режим доступа: <https://learn.microsoft.com/en-us/adaptive-cards/> (Дата обращения: 01.03.2026)
4. json_dynamic_widget documentation. [Электронный ресурс] Режим доступа: https://pub.dev/documentation/json_dynamic_widget/latest/ (Дата обращения: 01.03.2026)
5. Flutter documentation. [Электронный ресурс] Режим доступа: <https://docs.flutter.dev/> (Дата обращения: 01.03.2026)

РАЗРАБОТКА СИСТЕМЫ УДАЛЁННОЙ НАСТРОЙКИ И УПРАВЛЕНИЯ
ФУНКЦИОНАЛЬНЫМИ ВОЗМОЖНОСТЯМИ ДЛЯ ПРИЛОЖЕНИЙ НА ЯЗЫКЕ DART

Удалённая конфигурация и features flags в современных приложениях решают две тесно связанные задачи: динамически изменять параметры или поведение системы без обязательного выпуска новой версии клиента; снижать риск вывода новой функциональности за счет постепенного включения и быстрого отката при деградациях. Практика особенно значима на мобильных платформах из-за неоднородности устройств, переменного качества сети и задержек обновления приложений. Это находит отражение в рекомендациях по использованию мобильных SDK в промышленных платформах feature management [1].

Анализ существующих решений показывает, что Firebase Remote Config [2], представляющий собой облачный сервис удалённой настройки параметров приложения, удобен для параметризации клиента; LaunchDarkly [3], являющийся промышленной платформой управления функциональными флагами, предоставляет развитые механизмы таргетинга и управляемых раскаток; Unleash [4], реализующий open-source-подход к управлению функциональными флагами, предлагает удалённую оценку флагов; OpenFeature [5], выступающий в качестве открытого стандарта интеграции таких механизмов, задаёт унифицированную модель их использования. Однако у приложений на Dart, в частности на фреймворке Flutter, сохраняется потребность в архитектуре, которая одновременно обеспечивает переносимость, офлайн-устойчивость и минимизацию утечки логики правил на клиент.

Таким образом, целью статьи является описание архитектуры системы удалённой конфигурации и управления функциональными флагами, ориентированной на использование в приложениях на Dart/Flutter, а также рассмотрение её поддержки поэтапного внедрения изменений, ограниченного развертывания, А/В-экспериментов, аудита и наблюдаемости. Для достижения этой цели необходимо решение задач, перечисленных ниже.

1. Выполнить обзор существующих промышленных решений и стандартов (Firebase Remote Config, LaunchDarkly, Unleash, OpenFeature) и их применимости к Dart/Flutter.
2. Сформулировать требования и ограничения для клиентского приложения (офлайн-кэш, безопасная “активация” обновлений, использование контекста, наблюдаемость).
3. Спроектировать архитектуру (control plane, data plane, версии, откаты, аудит, edge-кэш).
4. Описать API и формат конфигураций или флагов (типизация, причины выдачи, совместимость с OpenFeature, OFREP).
5. Описать реализацию клиента на Dart с примерами кода и примером JSON-конфигурации.
6. Определить стратегии rollout, канареек иди экспериментов и требования к событиям экспозиций и метрикам.
7. Проработать безопасность и управление доступом (RBAC, токены, аудит, запрет на секреты в клиентском конфиге).

В работе в качестве архитектурного решения выбрана существующая модель (рисунок 1): разделение control plane и data plane. В control plane располагаются административный интерфейс, API управления, хранилище флагов, сегментов и экспериментов, механизмы RBAC и журнал аудита. В data plane находятся evaluation API, edge/cache-слой и контур приёма телеметрии. Клиентское приложение через SDK передаёт контекст оценки и получает уже вычисленный бандл значений флагов, что позволяет не раскрывать клиенту правила сегментации и логику rollout. Для уменьшения нагрузки и повышения устойчивости предусмотрены кэширование, ETag-валидация и возможность edge-

развёртывания. Отдельным решением является фиксация версий снимков конфигурации, что позволяет выполнять воспроизводимые откаты.

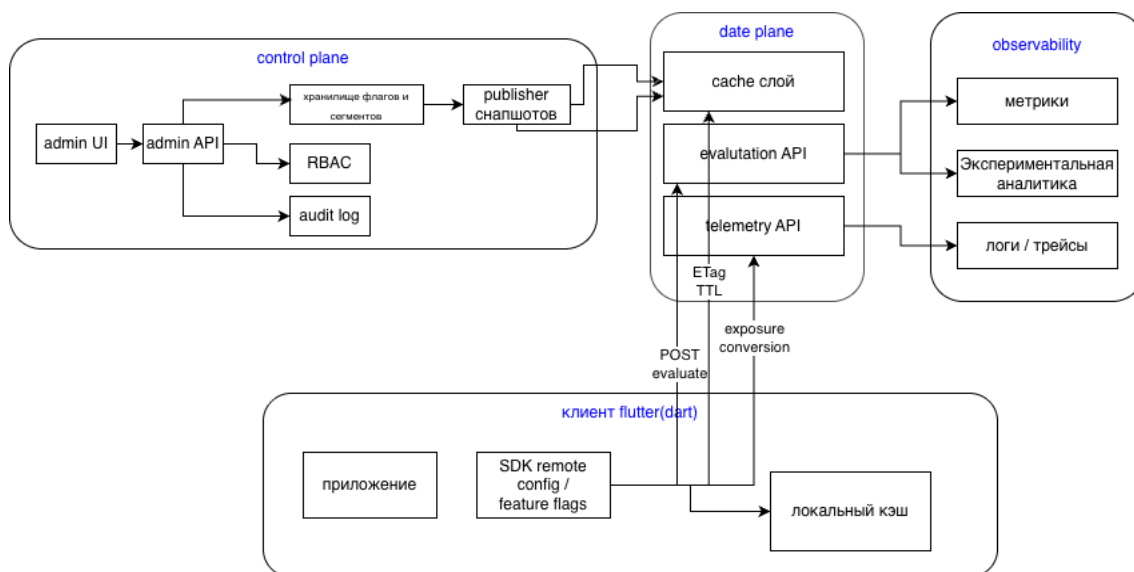


Рисунок 1 – Архитектура системы

Разработка велась в программной системе удалённой конфигурации и feature flags для приложений на языке Dart с использованием Flutter для клиентского SDK. Обмен данными с серверной частью выполняется по HTTP/REST, представление конфигурации — в формате JSON. Для поддержки работы клиента в условиях нестабильной сети используются локальное кэширование, ETag-валидация и TTL-стратегии обновления, а сбор телеметрии позволяет отслеживать применение конфигурации и состояние раскатки функциональности.

ЛИТЕРАТУРА

1. Понимание удаленной конфигурации в реальном времени. [Электронный ресурс] Режим доступа: <https://firebase.google.com/docs/remote-config?hl=ru> (Дата обращения: 10.03.2026)
2. Firebase Remote Config documentation. [Электронный ресурс] Режим доступа: <https://firebase.google.com/docs/remote-config?hl=ru> (Дата обращения: 10.03.2026)
3. LaunchDarkly documentation. [Электронный ресурс] Режим доступа: <https://launchdarkly.com/docs/home> (Дата обращения: 10.03.2026)
4. Unleash documentation. [Электронный ресурс] Режим доступа: <https://docs.getunleash.io/> (Дата обращения: 10.03.2026)
5. OpenFeature documentation. [Электронный ресурс] Режим доступа: <https://openfeature.dev/docs/reference/intro/> (Дата обращения: 10.03.2026)
6. Flutter documentation. [Электронный ресурс] Режим доступа: <https://docs.flutter.dev/> (Дата обращения: 10.03.2026)

УДК 004.056.5

Р.Э. Ахмедов (2 курс магистратуры),
Е.А. Павлов, ст. преподаватель

РАЗРАБОТКА ВЫСОКОИНТЕРАКТИВНОГО HONEYPOT С ИСПОЛЬЗОВАНИЕМ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ ДЛЯ ЗАЩИТЫ ВЕБ- ПРИЛОЖЕНИЙ

В условиях стремительного роста количества кибератак возрастает необходимость разработки инструментов, позволяющих не только обнаруживать попытки вторжения, но и анализировать поведение злоумышленников. Одним из наиболее эффективных средств исследования атак являются honeypot-системы — специализированные программные или аппаратные ресурсы, имитирующие уязвимые сервисы с целью привлечения

злоумышленников и фиксации их действий[1]. Полученные данные позволяют исследовать тактики атакующих, выявлять новые уязвимости и совершенствовать системы информационной безопасности.

В настоящее время существует большое количество программных решений для реализации honeypot-инфраструктуры. Среди наиболее распространённых можно выделить такие системы, как и Honeyd[2], Cowrie[3] и Dionaea[4]. Honeyd представляет собой SSH- и Telnet-honeypot, предназначенный для регистрации действий злоумышленников после успешной аутентификации. Dionaea ориентирован на обнаружение вредоносного программного обеспечения и позволяет эмулировать различные сетевые сервисы для фиксации попыток эксплуатации уязвимостей. Honeyd является универсальной системой эмуляции сетевых узлов, позволяющей имитировать работу различных операционных систем и сетевых сервисов.

Несмотря на широкое распространение данных решений, большинство существующих honeypot-систем обладают рядом ограничений. В частности, их поведение определяется статическими конфигурациями, что не позволяет динамически адаптировать систему к действиям атакующего[2]. Это снижает эффективность взаимодействия с злоумышленником и ограничивает возможности получения дополнительной информации о сценариях атак.

Одним из перспективных направлений развития honeypot-технологий является применение методов машинного обучения. В частности, методы обучения с подкреплением (Reinforcement Learning) позволяют создавать системы, способные формировать стратегию поведения на основе накопленного опыта взаимодействия с окружающей средой[5]. Использование данного подхода позволяет адаптировать поведение honeypot-системы в зависимости от действий атакующего и тем самым повышать эффективность анализа кибератак.

Целью данной работы является разработка прототипа интеллектуальной honeypot-системы, использующей методы Reinforcement Learning для адаптивного взаимодействия с атакующим и повышения эффективности сбора информации о кибератаках.

Рассмотрим архитектуру разрабатываемой системы на рисунке 1.

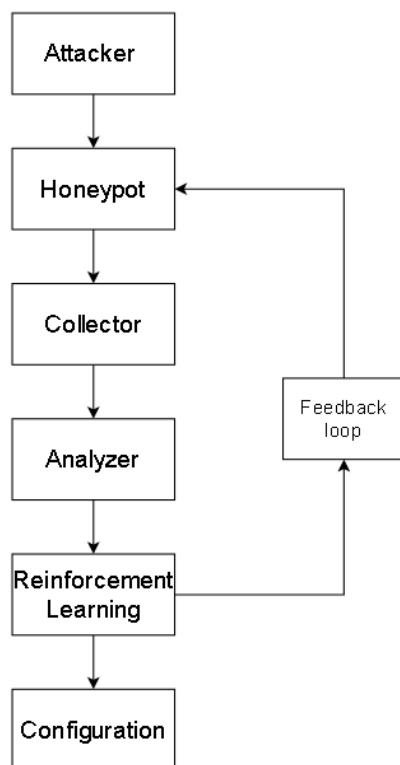


Рисунок 1 – Архитектура адаптивной Honeypot-системы с использованием Reinforcement Learning

Представленная архитектура включает несколько основных компонентов. Honeypot-модуль обеспечивает эмуляцию сетевых сервисов и принимает входящие соединения от потенциальных злоумышленников. Модуль сбора событий регистрирует сетевые соединения, команды атакующего и попытки эксплуатации уязвимостей. Полученные данные передаются в модуль анализа поведения, который формирует характеристики атак и состояние среды для алгоритма Reinforcement Learning. На основе этих данных RL-агент принимает решения о дальнейшем поведении системы, включая изменение конфигурации сервисов, эмуляцию дополнительных уязвимостей или завершение взаимодействия.

Стек технологий, используемый для реализации прототипа системы, включает язык программирования Python для реализации модулей анализа и алгоритмов обучения, серверную платформу Node.js с фреймворком Express для обработки сетевых взаимодействий, систему управления базами данных MySQL для хранения журналов событий и параметров обучения, а также библиотеку Stable-Baselines3 для реализации алгоритмов Reinforcement Learning[6]. Выбор технологий осуществлялся с использованием системы поддержки принятия решений, основанной на методе взвешенной суммы критериев, что позволило обосновать оптимальный набор инструментов разработки.

Разрабатываемая архитектура позволяет создать адаптивную honeypot-систему, способную изменять своё поведение в зависимости от действий злоумышленника. В отличие от традиционных решений предложенный подход обеспечивает более гибкое взаимодействие с атакующим и позволяет получать более детальную информацию о применяемых методах атак.

На текущем этапе разработан прототип системы, позволяющий разворачивать honeypot-сервисы, фиксировать действия атакующих и использовать алгоритмы Reinforcement Learning для выбора стратегии взаимодействия. В дальнейшем планируется проведение экспериментального исследования эффективности разработанной системы в условиях реальных сетевых атак, а также расширение набора эмулируемых сетевых сервисов.

ЛИТЕРАТУРА

1. Spitzner L. Honeypots: Tracking Hackers. — Boston: Addison-Wesley, 2003.
2. Provos N., Holz T. Virtual Honeypots: From Botnet Tracking to Intrusion Detection. — Boston: Addison-Wesley Professional, 2007.
3. Cowrie Honeypot. URL: <https://github.com/cowrie/cowrie>
4. Dionaea Honeypot. URL: <https://github.com/DinoTools/dionaea>
5. Sutton R., Barto A. Reinforcement Learning: An Introduction. — MIT Press, 2018.
6. Stable-Baselines3 Documentation. URL: <https://stable-baselines3.readthedocs.io>

УДК 004.453

Е.А. Бабин (4 курс бакалавриата),
К.А. Туральчук, к.т.н., доцент

РАЗРАБОТКА NO-CODE КОНСТРУКТОРА КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ С RUNTIME-ЗАГРУЗКОЙ КОНТЕНА НА БАЗЕ UNITY

Современные музеи и выставочные пространства активно внедряют интерактивные технологии: сенсорные киоски, проекционные инсталляции и игровые механики [1, 3, 4]. Однако создание такого программного обеспечения (ПО) обычно требует привлечения профессиональных разработчиков, что делает процесс дорогостоящим и медленным. Сотрудники музея (искусствоведы, кураторы), обладающие предметными знаниями, не имеют возможности самостоятельно влиять на цифровой контент без знания языков программирования [1, 6].

Существующие решения (например, универсальные CMS или конструкторы) либо слишком сложны для неподготовленного пользователя, либо не учитывают специфику мультимедийного и игрового взаимодействия, характерного для современных экспозиций [7]. Таким образом, актуальной является задача создания специализированного инструмента, позволяющего museum-специалистам визуально собирать интерактивные приложения для сенсорных панелей, работающие на различном оборудовании.

Целью работы является разработка кроссплатформенного инструментального средства «Museum Experience Builder», которое позволяет сотрудникам музеев без навыков программирования создавать интерактивные приложения для сенсорных панелей.

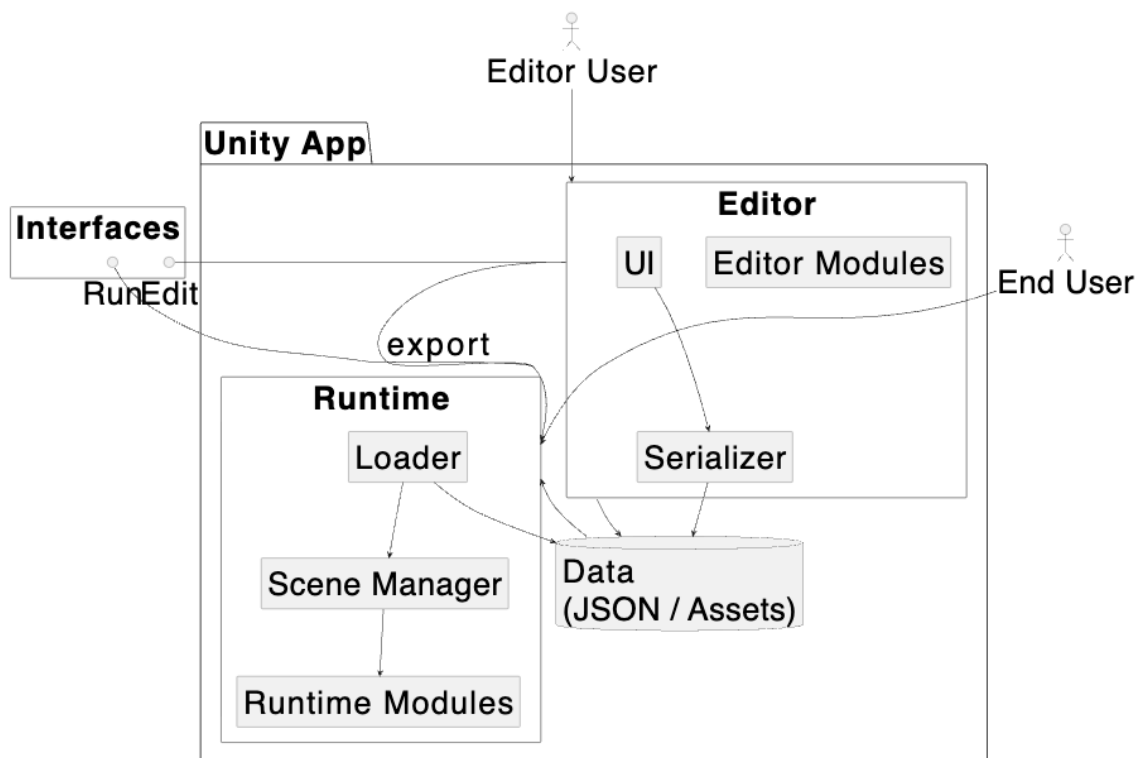


Рисунок 1 – Архитектура системы «Museum Experience Builder»

Ключевой особенностью предложенной архитектуры является разделение приложения на два логических режима, работающих поверх единого ядра (рисунок 1). Такой подход позволяет избежать дублирования кода при работе с файловой системой и модулями.

Режим редактора реализован только для платформы Windows. Он предоставляет графический интерфейс для управления проектом. При создании нового проекта генерируется иерархическая структура папок, включающая директории assets/ (для медиафайлов) и modules/ (для JSON-конфигураций конкретных экранов). Редактор работает напрямую с файловой системой, что обеспечивает прозрачность данных: пользователь может скопировать папку проекта на флешку и перенести на другое устройство [5].

Режим просмотра (Runtime) предназначен для исполнения проектов на целевых устройствах (Windows и Android). Приложение сканирует указанную пользователем папку, загружает project.json и scenes.json, после чего динамически создает сцены и инициализирует модули в соответствии с конфигурацией. Критически важным является то, что приложение-плеер не требует доступа в интернет и работает полностью автономно после копирования проекта [2].

В основе функциональной гибкости лежит модульная архитектура. Каждый функциональный блок (меню, медиа-слайд, 3D-просмотрщик, игра) реализован в виде независимого модуля. Для обеспечения расширяемости были определены два базовых интерфейса:

1. IEditorModule: отвечает за отрисовку интерфейса настройки модуля в редакторе и сериализацию данных модуля в JSON.

2. **IRuntimeModule**: отвечает за создание игрового объекта (**GameObject**) в рантайме и его инициализацию загруженными данными.

Для ускорения работы пользователя введено понятие «шаблонов модулей». Например, модуль «Медиа-слайд» может быть представлен в виде шаблонов «Текст+Изображение», «Галерея» или «Сравнение до/после». Каждый шаблон содержит предустановленные настройки компоновки элементов интерфейса, что снижает порог входа для пользователя.

Практическая значимость работы заключается в сокращении времени и стоимости разработки музейного ПО, а также в передаче контроля над контентом непосредственно кураторам экспозиций. Внедрение данного инструмента позволит музеям быстро обновлять экспозиции и создавать уникальные пользовательские сценарии.

ЛИТЕРАТУРА

1. Wang B., Li G. Optimization of Interactive Experience of Multimedia Technology in Exhibition Design of Art Museums // *Journal of Electrical Systems*. – 2024. – Т. 20, № 10s. – С. 7336–7348.
2. User experience model and design strategies for virtual reality-based cultural heritage exhibition / Q. Gong, N. Zou, W. Yang [et al.] // *Virtual Reality*. – 2024. – Т. 28. – DOI: 10.1007/s10055-024-00942-z.
3. Survey on 3D Visualization of Museum, its Use Cases, Implementation of Blockchain and the Research Ahead / S. Aruna, P. P. Mishra, A. Bansal [et al.] // *International Journal for Research in Applied Science and Engineering Technology*. – 2023. – DOI: 10.22214/ijraset.2023.50040.
4. Bayer T., Baez Santamaria S., Vossen P. A modular architecture for creating multimodal agents [Электронный ресурс] // *arXiv*. – 2022. – arXiv:2205.12456. – Режим доступа: <https://arxiv.org/abs/2205.12456> (дата обращения: 17.03.2026).
5. Jiang T., He Y., Chang H., Li J., Gou L. Exploring Museum Display Space Design Methodology Using Unity 3d and Vr Technology // *SSRN Electronic Journal*. – 2024. – Режим доступа: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4070735 (дата обращения: 17.03.2026).
6. Khor C. Y., Mubin S. A. AR Mobile Application for Enhancing National Museum Heritage Visualization // *International Journal of Software Engineering and Computer Systems*. – 2024. – Режим доступа: <https://journal.ump.edu.my/ijsecs/article/view/10584> (дата обращения: 17.03.2026).
7. Kellar M., Watters C., Shepherd M. An information architecture to support dynamic composition of interactive lessons and reuse of learning objects // *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. – 2004. – Режим доступа: <https://web.cs.dal.ca/~shepherd/pubs/lom.pdf> (дата обращения: 17.03.2026).

УДК 004.42

Д.Е. Булатов (2 курс магистратуры),
Т.В. Леонтьева, к.т.н., доцент

РАЗРАБОТКА СЕРВЕРНОГО ПРИЛОЖЕНИЯ СИСТЕМЫ «УМНЫЙ ДОМ» С ИНТЕЛЛЕКТУАЛЬНОЙ ПОДСИСТЕМОЙ КЛАССИФИКАЦИИ СОБЫТИЙ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

В настоящий момент технологии искусственного интеллекта массово внедряются в повседневную жизнь. В области применения систем «умный дом» есть также множество задач, решение которых связано с машинным обучением и позволит адаптироваться к различным условиям и работать в автономном режиме.

На сегодняшний день решений для систем «умного дома», полноценно поддерживающих работу с моделями машинного обучения нет. Кроме того, в большинстве случаев в роле серверного приложения выступает облачная инфраструктура производителя, что приводит к проблемам надежности, отказоустойчивости и безопасности. Но существуют и локальные серверные приложения, например Home Assistant[1]. Это мощный инструмент с открытым исходным кодом, который может быть настроен на работу с большим количеством

оборудования. Однако он и другие схожие решения не поддерживают использование интеллектуальных систем. Вся автоматизация в таких приложениях сводится к выполнению детерминированных правил.

Для достижения возможности применения предсказательных моделей в области «умного дома», можно использовать два варианта: разработка дополнительного ПО для решений с открытым исходным кодом, либо разработка решения, внутри которого будет интегрирован модуль, поддерживающий работу с моделями машинного обучения. Первый вариант является наиболее простым, но имеет ряд недостатков. В первую очередь речь идет о скорости работы такого решения. Рассмотренные в ходе исследования приложения с открытым исходным кодом имеют такую архитектуру, которая не позволит встроить в него модуль для работы с моделями машинного обучения. Данные требовалось бы постоянно сохранять во временном хранилище, а отдельному программному обеспечению в постоянном режиме отслеживать их и делать предсказания, а затем возвращать ответ так же через временное хранилище[2]. Такой подход влечет за собой большие временные и ресурсные затраты, а так же сложную интеграцию с API стороннего продукта. Поэтому было принято решение разработать отдельное модульное приложение для контроля и управления системой «умный дом», внутри которого будет поддерживаться работа с моделями.

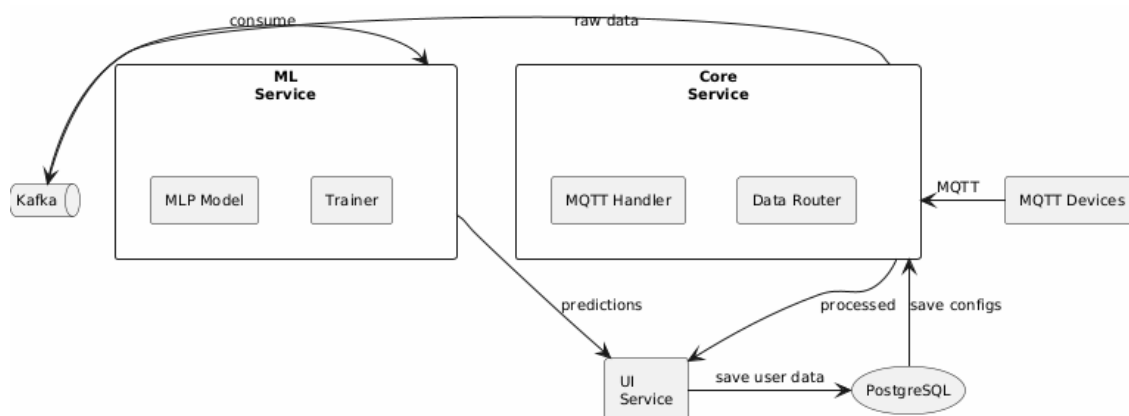


Рисунок 1 – Сервисы приложения

Программное обеспечение разбито на несколько модулей. Модуль mqtt является вспомогательными для работы системы. Он принимает сообщения от конечных устройств «умного дома» и передает их в ядро серверного приложения, которое в свою очередь выполняет полную обработку этих сообщений и формирует ответ или сохраняет в базу данных пришедшие данные после их обработки. В базовом виде, как это реализовано и в существующих решениях, модуль ядра приложения является транспортным узлом между связкой устройство - устройство или пользователь - устройство. В нем происходит анализ откуда и кому необходимо передать команду или данные при помощи сохраненных в базе данных конфигураций. Кроме того, в базе данных хранятся созданные пользователем сценарии работы, определенные в виде детерминированных правил[3]. Модуль предсказаний является отличительной особенностью разрабатываемого решения.

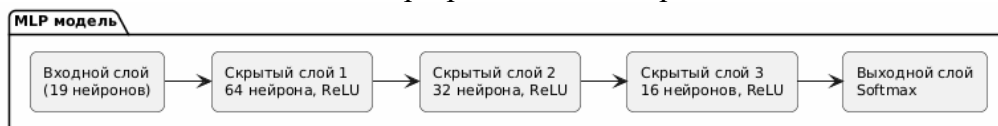


Рисунок 2 – Архитектура модели прогнозирования

Предсказательная модель работает с временными рядами[4]. На вход она получает временной ряд с показаниями датчиков и событий, дополняет их до непрерывных промежутков, извлекает признаки и по ним делает предсказание, получая метку текущего события и действия на него. Разработанная модель показала результат 98% точности на

тестовой выборке, взятой как 10% от реальных данных, аналогичных тем, на которых производилось обучение. При тестировании системы после интеграции модели в приложение, система корректно определяла события с регулярной повторяемостью, такие как уход из дома, возвращение домой, стандартную и аномальную активность.

Разработанное программное решение позволило расширить функционал «умного дома», добавив в него возможность автоматического определения события и реагирования на них так, как среагировал бы пользователь. Для дальнейшего расширения возможностей системы необходимо разработать модели, ориентированные на работу сразу с несколькими источниками входных данных. Такая доработка позволит расширить число определяемых классов событий и повысит точность предсказаний.

ЛИТЕРАТУРА

1. Документация Home Assistant. [Электронный ресурс] Режим доступа: <https://www.home-assistant.io/docs/>
2. Smith J., Brown A. A Serverless Architecture for Machine Learning-Based Smart Home Applications // Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP). 2022. P. 112-119
3. Булатов Д.Е., Коликова Т.В. Разработка архитектуры системы умный дом // Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 24–25 апреля 2024 года. – Санкт-Петербург : Изд-во Политехн. ун-та, 2024. – С. 23-25.
4. Сидорова Н.П., Алексеев В.В. Сравнительный анализ нейросетевых моделей прогнозирования временных рядов в задачах интернета вещей // International Journal of Open Information Technologies. – 2023. – Т. 11, № 5. – С. 32-41

УДК 004.45

Д.А. Баландин (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент

РАЗРАБОТКА И АВТОМАТИЗАЦИЯ РАЗВЕРТЫВАНИЯ ОТКАЗОУСТОЙЧИВОГО КЛАСТЕРА DECKHOUSE

Современные корпоративные информационные системы широко используют контейнерные технологии, обеспечивающие масштабируемость, гибкость управления ресурсами и отказоустойчивость. В качестве базовой платформы оркестрации применяется Kubernetes, предоставляющий механизмы управления жизненным циклом приложений и распределения нагрузки [1].

Для промышленной эксплуатации Kubernetes-кластеров используются платформенные решения, расширяющие функциональные возможности базовой системы. Одним из таких решений является Deckhouse Kubernetes Platform, обеспечивающая автоматизацию управления кластером, включая мониторинг, логирование и средства обеспечения безопасности [2].

В ряде корпоративных инфраструктур платформа применяется в конфигурации с одним узлом, что упрощает развертывание, однако приводит к формированию единой точки отказа и ограничивает возможности масштабирования [3].

Целью работы является снижение трудоемкости развертывания с несколькими узлами кластера Deckhouse в облачной инфраструктуре, за счёт применения подхода автоматизации, обеспечивающего отказоустойчивость и масштабируемость

Для достижения поставленной цели решены следующие задачи:

- проведение анализа современных инструментов автоматизации развертывания кластерных решений;
- сравнительный анализ платформ управления контейнерами;

- разработка подхода автоматизированного развертывания;
- практическая реализация инструментов автоматизации;
- экспериментальная оценка полученных результатов.

В ходе анализа установлено, что архитектура с одним узлом не соответствует требованиям промышленной эксплуатации, поскольку наличие единственного узла формирует единую точку отказа и снижает надёжность системы [3]. Исходная архитектура платформы Deckhouse представлена на рисунке 1.

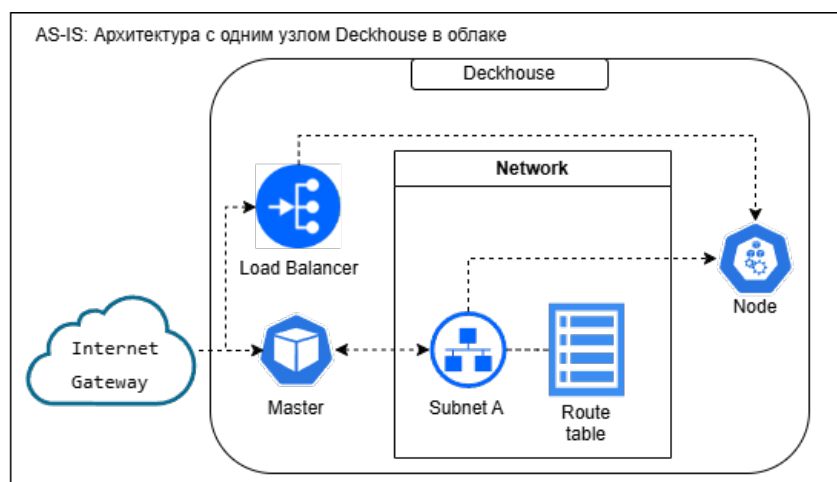


Рисунок 1 – Архитектура платформы Deckhouse с одним узлом (AS-IS)

В качестве решения предложена архитектура с несколькими узлами, включающая три управляющих узла (master), что соответствует рекомендациям по обеспечению отказоустойчивости Kubernetes-кластеров и позволяет реализовать кворум управляющей плоскости [4]. Целевая архитектура представлена на рисунке 2.

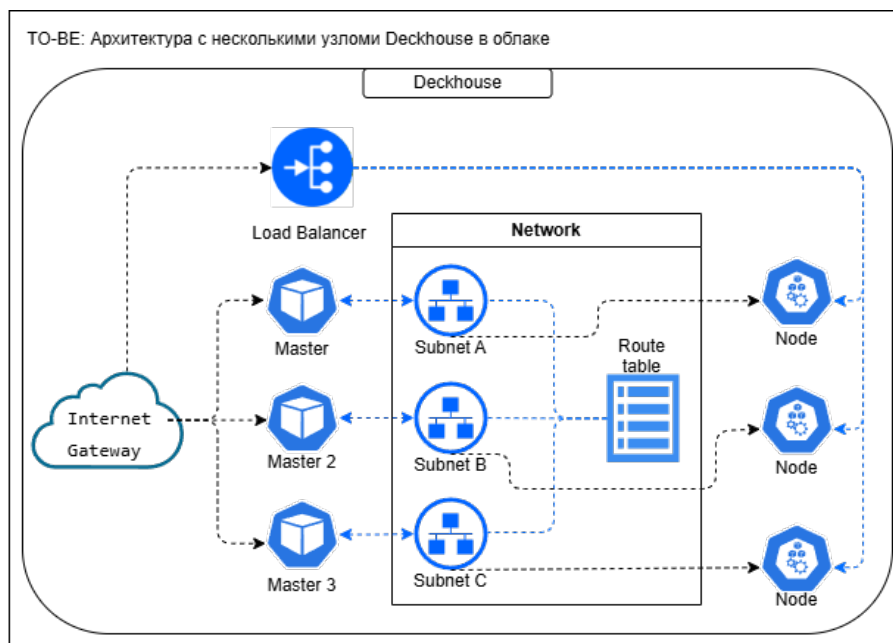


Рисунок 2 – Архитектура платформы Deckhouse с несколькими узлами (TO-BE)

В рамках практической реализации была модифицирована Ansible-роль установки Deckhouse для поддержки режима High Availability, а также реализована конфигурация балансировщика нагрузки на базе Nginx, обеспечивающего устойчивый доступ к управляющей плоскости кластера.

Внесённые изменения в конфигурационные файлы обеспечили корректную работу платформы Deckhouse в архитектуре с несколькими узлами.

Экспериментальная оценка показала, что переход к архитектуре с несколькими узлами позволил устранить единые точки отказа, повысить отказоустойчивость системы и обеспечить горизонтальное масштабирование. Применение инструментов автоматизации позволило сократить трудоёмкость развертывания на 30–40% и снизить эксплуатационные риски за счёт повышения воспроизводимости инфраструктуры.

В работе разработан и реализован подход автоматизации развертывания кластера Deckhouse с несколькими узлами. Предложенное решение обеспечивает отказоустойчивость, масштабируемость и снижение трудоёмкости развертывания, что соответствует требованиям промышленной эксплуатации корпоративных информационных систем.

ЛИТЕРАТУРА

1. Shemyakinskaya A., Nikiforov I. Disk Space Management Automation with CSI and Kubernetes // Proceedings of Seventh International Congress on Information and Communication Technology. – Springer, 2022. – P. 171–179. Deckhouse Kubernetes Platform: официальная документация. – URL: <https://deckhouse.io> (дата обращения: 09.10.2025).
2. Kubernetes Documentation. – URL: <https://kubernetes.io/docs> (дата обращения: 17.09.2025).
3. Никифоров И. В., Юсупова О. А., Воинов Н. В., Ковалев А. Д. Методы, алгоритмы и архитектуры распределенной обработки больших данных. – СПб.: СПбПУ, 2023. – 194 с.
4. Zhang Q., Chen M., Li L. Cloud Computing: State-of-the-Art and Research Challenges // Journal of Internet Services and Applications. – 2018. – Т. 9, № 1.
5. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment // Linux Journal. - 2014. - № 239.ч

УДК 004.8

Е.С. Булыкина (2 курс магистратуры),
А.В. Щукин, к.т.н., доцент

ИССЛЕДОВАНИЕ СПОСОБНОСТИ LLM К ВОССТАНОВЛЕНИЮ ЭЛЛИПСИСОВ

Развитие больших языковых моделей (Large Language Models, далее – LLM) открывает новые возможности в обработке текстов на естественном языке. Однако такое синтаксическое явление как эллипсис вызывает трудности в системах автоматического анализа текстов в силу того, что эллипсис широко распространен именно в устной речи и в художественной литературе в качестве средства выразительности.

Целью работы является экспериментальное исследование способности LLM восстанавливать предложения, содержащие эллипсис. Данный эксперимент является продолжением экспериментов, отраженных в работе [1]. Интерес к исследованию восприятия эллипсисов LLM связан с необходимостью разработки автоматической системы генерации тестовых заданий для тестирования когнитивных способностей [2].

В монографии [3] Падучева определяет эллипсис как конструкцию, в которой отсутствует элемент, но его можно восстановить. Это восстановление осуществляется с помощью обращения к более широкому контексту. В качестве тестового материала в данном исследовании используется набор из 100 предложений с эллипсисами: 26 предложений из сборников по планиметрии и 74 предложения из художественной и публицистической литературы. Предложения собраны вручную, поскольку ещё не существует наборов данных с такими текстами на русском языке.

Для проведения эксперимента выбраны наиболее популярные модели с поддержкой русского языка. Ключевым критерием для отбора моделей является наличие бесплатного доступа через API, поскольку модели с большим количеством параметров требуют также больших вычислительных ресурсов устройства, на котором проводится эксперимент. LLM сравниваются в таблице 1 по следующим критериям: необходимость VPN, количество

параметров модели, ограничение количества токенов, MMLU (Massive Multitask Language Understanding) — показатель качества знаний модели в разных областях.

Таблица 1 – Сравнение LLM, участвующих в эксперименте

Критерий	GigaChat 2 MAX	YandexGPT 5 lite	Gemini 3 Flash	GPT-OSS
Количество параметров	Не указано	8b	Не указано	120b
Необходимость VPN	Нет	Нет	Да	Да
Ограничения при использовании API	50 000 токенов на год	10 000 000 токенов в месяц	1 000 000 токенов в минуту	8 000 токенов в минуту
	-	-	-	1000 запросов в день
MMLU	0.86	0.758	0.812	0.9

Для доступа к моделям GigaChat, YandexGPT, Gemini использовался API, предоставленный разработчиками – Сбер, Яндекс, Google. Для доступа к моделям GPT OSS использовался сервис Groq.

Для проведения эксперимента составлен следующий запрос: «Эллипсис – это пропуск в тексте языковой единицы, которая может быть восстановлена из контекста. При глагольном эллипсисе пропускается повторное вхождение в предложении глагола или глагольной группы, в составе которой глагол и зависимые от него слова. Например, в предложении: “В равнобедренный прямоугольный треугольник ABC с прямым углом при вершине B вписан прямоугольник MNKB так, что две его стороны MB и KB лежат на катетах, а вершина N — на гипотенузе AC” — можно восстановить повторное вхождение слова “лежит” для формирования полного предложения из эллиптического. Тогда получим: “В равнобедренный прямоугольный треугольник ABC с прямым углом при вершине B вписан прямоугольник MNKB так, что две его стороны MB и KB лежат на катетах, а вершина N лежит на гипотенузе AC.” В представленных ниже предложениях восстанови слова на месте тире, опираясь на предыдущую часть предложения, как представлено в примере выше. В предложениях могут быть пропущены не только глагольные группы, но и существительные. Верни полностью восстановленные варианты каждого предложения». Далее в запросе приводятся 100 предложений, которые были использованы в экспериментах по генерации эллипсисов [4].

В таблице 2 приведена статистика работы моделей на 100 предложениях, в ячейках указано количество предложений, в которых наблюдается указанный признак после обработки моделью. Введены следующие сокращения: ГГр – глагольная группа, ИГр – именная группа.

Таблица 2 – Результаты экспериментов по генерации эллипсисов LLM

	YandexGPT	GigaChat MAX	Gemini 3 Flask	GPT-OSS
Эллипсис ГГр не восстановлен	3	12	1	7
ГГр восстановлена верно	84	70	98	71
ГГр восстановлена не полностью	7	7	-	3
ГГр в неверной форме/не тот глагол	6	11	1	19
Восстановлено подлежащее	4	2	15	3
Верно восстановлены ИГр	9	4	22	4
Неверное восстановлены ИГр	-	2	3	-

Рассмотрим пример, в котором все модели испытали трудности. Например, предложение: «Старшей из его дочерей не минуло еще двадцати лет, младшей – едва тринадцать». Согласно правилам восстановления глагольного эллипсиса, должна быть восстановлена полностью часть «не минуло», однако с точки зрения логики в данном предложении подразумевается, что младшей только исполнилось тринадцать, поэтому стоит восстановить глагол без связки «не». Логически верно восстановила эллипсис модель от Сбера, подставив «исполнилось», однако это не соответствует правилам исключения повторов при формировании эллипсиса. Ещё один текст, в котором модели не справились с

восстановлением именно глагольного эллипсиса содержал в качестве действия деепричастие. Модели

Выводы. В результате проведенного эксперимента выявлены существенные различия в эффективности обработки эллиптических конструкций различными языковыми моделями. Самой эффективной оказалась модель Gemini, которая пыталась максимально полно восстановить предложения, что иногда отрицательно сказывалось на смысле текста. Модели GigaChat и GPT показывают примерно одинаковый уровень, однако GigaChat гораздо чаще указывала, что эллипсиса в предложении нет, в то время как GPT чаще пыталась восстановить эллипсис, но использовала либо не тот глагол, либо глагол в неверной форме. В целом эксперимент демонстрирует перспективность использования LLM в задаче восстановления эллипсисов, однако также показывает необходимость контроля полученных результатов с помощью дополнительных методов.

ЛИТЕРАТУРА

1. E. Bulykina, X. Naidenova, V. Parkhomenko and T. Martirova, "Research on the perception of ellipses by natural and artificial intelligence," 2025 7th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Lipetsk, Russian Federation, 2025, pp. 280-285, doi: 10.1109/SUMMA68668.2025.11302331.
2. Разработка компьютерной системы тестирования когнитивных способностей респондентов на основе предложений с эллипсисами / К. Найденова [и др.] // Электронные библиотеки. — 2023. — Т. 26, № 3. — С. 340—364. — DOI 10.26907/1562-5419-2023-26-3-340-364.
3. Падучева Е. О семантике синтаксиса. — URSS, 2019. — 300 с.
4. X. A. Naidenova and E. S. Bulykina, Stimulus material of one hundred elliptical sentences in russian with the results of sentence processing using gigachat, yandexgpt and gemini language models, Zenodo, Sep. 2025. doi: 10.5281/zenodo.17200173. [Online]. Available: <https://doi.org/10.5281/zenodo.17200173>.

УДК 004.4

А.А. Вяткин (2 курс магистратуры),
Н.В. Воинов, к.т.н., доцент

ПОВЫШЕНИЕ МАСШТАБИРУЕМОСТИ CI/CD-ИНФРАСТРУКТУРЫ ПУТЕМ КОНТЕЙНЕРИЗАЦИИ GITLAB RUNNER В KUBERNETES

Современная CI/CD-инфраструктура должна обеспечивать быстрое масштабирование вычислительных ресурсов, предсказуемое время запуска заданий и контролируемую изоляцию сетевых взаимодействий. В текущей корпоративной архитектуре компании GitLab Runner разворачивается на виртуальных машинах с Ubuntu, что затрудняет эластичное наращивание мощности при пиковых нагрузках, усложняет унификацию политик доступа и повышает стоимость сопровождения. В качестве целевой платформы рассматривается Kubernetes, в котором для каждого задания может создаваться отдельный pod, что делает вычислительные ресурсы более управляемыми и изолированными.

Целью работы является повышение масштабируемости CI/CD-инфраструктуры за счет миграции GitLab Runner из VM-окружения в Kubernetes [1-3] с сохранением требований безопасности и наблюдаемости. Для достижения цели решаются следующие задачи: сравнение вариантов исполнения заданий, проектирование схемы сетевого разграничения доступа, проработка безопасного доступа к API Yandex Cloud, а также анализ ограничений, возникающих при запуске сборочных сценариев в непривилегированных pod'ах.

Сравнительное тестирование показало, что Kubernetes является более предпочтительной платформой по сравнению с serverless-подходом для задач длительной и массовой сборки. При запуске группы из 40 параллельных пайплайнов в Kubernetes получены более стабильные значения времени выполнения и времени ожидания в очереди, тогда как результаты serverless-

исполнения оказались существенно более вариативными. Дополнительными факторами выбора стали отсутствие жесткого ограничения на длительность задания, более высокая устойчивость к аварийному завершению пустых и прикладных пайплайнов, развитая наблюдаемость, а также возможность тонкой калибровки системы через параметры `concurrent`, `Horizontal Pod Autoscaler` и `Cluster Autoscaler`.

Предлагаемая схема работы основана на использовании `Kubernetes executor` для `GitLab Runner` и профилировании сетевых прав на уровне `pod`. Управляющий `runner` получает задание из `GitLab`, определяет профиль доступа на основании проекта или репозитория и запускает `pod` для выполнения `job` с соответствующим набором меток. Эти метки используются для автоматического применения политик `CiliumNetworkPolicy` [4,5]. Такой подход позволяет перейти от статической сетевой модели к политико-ориентированному разграничению доступа, в котором разрешения задаются не IP-адресами, а характеристиками рабочей нагрузки. Для межпространственных и внешних взаимодействий рассматривается применение правил по именам сервисов и `FQDN`, а также использование `cluster-wide` политик как базового механизма `deny-by-default`.

Отдельным направлением работы является безопасный доступ `pod`'ов к `API Yandex Cloud` [6,7]. Для этого рассматривается федерация сервисных аккаунтов `Identity and Access Management`, позволяющая получать IAM-токены без хранения долговременных ключей. Данный подход снижает риск компрометации учетных данных и упрощает управление доступом для CI/CD-сценариев, которым требуется обращение к облачным сервисам из кластера `Managed Service for Kubernetes`.

В ходе миграции выявлены и проанализированы технические ограничения. Наиболее существенными стали монтирование репозитория в непривилегированном `pod` без прямого доступа к `/dev/fuse`, необходимость безопасной сборки контейнерных образов в `rootless`-режиме, а также деградация времени выполнения сборки в чистом окружении. В качестве практического решения для сборки образов выбран `BuildKit`, а для работы с `FUSE` предложена модификация ОСИ-спецификации и конфигурации `containerd` на выделенных узлах CI/CD. Такой подход позволяет сохранить `privileged: false` для `pod`'ов и одновременно обеспечить требуемый функционал монтирования.

Пилотная эксплуатация на данных одного проекта показала разнонаправленный эффект. Среднее время нахождения задания в очереди сократилось с 1,5 минуты до менее одной секунды, что подтверждает выигрыш по масштабируемости и скорости планирования. При этом среднее суммарное время выполнения пайплайна увеличилось с 7,3 до 13,6 минуты. Анализ показал, что основной причиной роста является пересборка большого количества бинарных артефактов в чистом окружении и недостаточная эффективность распределенного кэша. Следовательно, миграция `runner`'ов в `Kubernetes` уже обеспечивает резкое сокращение очередей, однако для полного повышения производительности требуется дополнительная оптимизация кэширования и сетевой связанности со вспомогательными сервисами сборки.

Таким образом, контейнеризация `GitLab Runner` и перенос исполнения `job`'ов в `Kubernetes` позволяют построить более масштабируемую и управляемую CI/CD-инфраструктуру по сравнению с моделью на виртуальных машинах. Наиболее перспективным направлением дальнейшей работы является завершение шаблонов `CiliumNetworkPolicy` для различных классов проектов, стабилизация сценария монтирования репозитория на новых узлах и оптимизация кэша, влияющего на длительность сборки. Практическая значимость работы заключается в том, что предложенный подход может быть использован как основа для промышленной миграции CI/CD-платформы в среду `Kubernetes` с учетом требований безопасности, наблюдаемости и эластичности.

ЛИТЕРАТУРА

1. `Kubernetes executor` // `GitLab Docs`. [Электронный ресурс]. Режим доступа: <https://docs.gitlab.com/runner/executors/kubernetes/> (дата обращения: 17.03.2026).
2. `GitLab Runner Helm chart` // `GitLab Docs`. [Электронный ресурс]. Режим доступа: <https://docs.gitlab.com/runner/install/kubernetes/> (дата обращения: 17.03.2026).

3. Network Policies // Kubernetes Documentation. [Электронный ресурс]. Режим доступа: <https://kubernetes.io/docs/concepts/services-networking/network-policies/> (дата обращения: 17.03.2026).
4. Network Policy. CiliumNetworkPolicy // Cilium Documentation. [Электронный ресурс]. Режим доступа: <https://docs.cilium.io/en/stable/network/kubernetes/policy/> (дата обращения: 17.03.2026).
5. Locking Down External Access with DNS-Based Policies // Cilium Documentation. [Электронный ресурс]. Режим доступа: <https://docs.cilium.io/en/stable/security/dns/> (дата обращения: 17.03.2026).
6. Bristot D., Gritti F., Fontana G. et al. The eBPF Runtime in the Linux Kernel // arXiv. 2024. [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/2410.00026> (дата обращения: 17.03.2026).
7. Accessing the Yandex Cloud API from a Managed Service for Kubernetes cluster using a workload identity federation in Identity and Access Management // Yandex Cloud Documentation. [Электронный ресурс]. Режим доступа: <https://yandex.cloud/en/docs/tutorials/security/wlif-managed-k8s-integration> (дата обращения: 17.03.2026).

УДК 004.65

И.Р. Гайнетдинов (4 курс бакалавриата),
О.А. Юсупова, ст. преподаватель

ДЕКЛАРАТИВНЫЙ ФРЕЙМВОРК ДЛЯ ГЕНЕРАЦИИ ETL-ПАЙПЛАЙНОВ НА ОСНОВЕ YAML-КОНФИГУРАЦИЙ

Разработка ETL-пайплайнов остаётся трудоёмким процессом: инженеры данных вынуждены вручную программировать однотипную логику переноса и трансформации данных, включая сложные паттерны медленно меняющихся измерений (SCD) [1]. Концепция Data Mesh [2] усиливает потребность в стандартизированных, версионизируемых интерфейсах описания пайплайнов, тогда как существующие инструменты либо предлагают визуальный low-code подход, плохо совместимый со стандартными практиками разработки [3], либо требуют глубокого программирования, что создаёт высокий порог вхождения. Таким образом, существует явный разрыв между декларативным описанием бизнес-логики и её императивной технической реализацией, что снижает эффективность команд Data Engineering [4].

Цель работы — разработать декларативный фреймворк, позволяющий описывать логику ETL-процессов, включая сложные паттерны загрузки (SCD Type 1, SCD Type 2, SCD Type 3, SCD Type 4, инкрементальная загрузка), в виде человекочитаемых YAML-конфигураций с автоматической генерацией готового к выполнению кода для Apache Airflow [5]. Для достижения цели решались следующие задачи:

- анализ существующих инструментов и подходов к построению ETL-пайплайнов (Apache Airflow [5], Apache NiFi, dbt [6], Singer.io, облачные сервисы);
- проектирование YAML-схемы и архитектуры кодогенератора;
- реализация прототипа с поддержкой ключевых паттернов трансформации данных;
- экспериментальная оценка эффективности по сравнению с ручной разработкой.

Технологический стек фреймворка выбран исходя из принципа доминирования в экосистеме Data Engineering [4]. Основным языком реализации является Python: его выбор обусловлен нативной интеграцией с Apache Airflow [5] и богатством специализированных библиотек. Формат YAML используется для описания пайплайнов как практический стандарт Infrastructure as Code (по аналогии с Kubernetes, Ansible, Docker Compose). Библиотека PyYAML обеспечивает парсинг конфигураций, Pydantic — строгую валидацию схем, Jinja2 — шаблонизацию и генерацию итогового кода. Архитектурно проект реализован как модульный монолит (парсер YAML, валидатор схемы, генераторы для разных бэкендов, CLI); для обеспечения качества применяются pytest и hypothesis.

В результате работы разработан прототип фреймворка, позволяющий описывать ETL-пайплайны декларативно через YAML и автоматически генерировать исполняемые DAG-

файлы Apache Airflow [5]. Реализована поддержка паттернов SCD Type 1, SCD Type 2, SCD Type 3 и SCD Type 4 [1], инкрементальной загрузки данных. Сравнительный эксперимент показал сокращение времени разработки типового пайплайна и снижение объёма кода, требующего написания инженером данных вручную [4]. Обеспечено 100% покрытие unit-тестами для критически важных модулей управления версиями и инкрементальными обновлениями.

Разработанный фреймворк подтвердил состоятельность декларативного подхода к описанию ETL-процессов: поставленная цель — снижение трудоёмкости разработки пайплайнов через автоматическую генерацию кода из YAML-конфигураций — достигнута. Результаты работы применимы в промышленной data-инженерии и могут быть расширены для поддержки дополнительных оркестраторов и паттернов обработки данных.

ЛИТЕРАТУРА

1. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. – 3rd ed. – Indianapolis: Wiley, 2013. – 600 p.
2. Deghani Z. Data Mesh: Delivering Data-Driven Value at Scale. – O'Reilly Media, 2022. – 350 p.
3. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – O'Reilly Media, 2017. – 590 p.
4. Reis J., Housley M. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. – O'Reilly Media, 2022. – 414 p.
5. Apache Software Foundation. Apache Airflow Documentation. [Электронный ресурс] Режим доступа: <https://airflow.apache.org/docs/> (дата обращения: 01.12.2025)
6. dbt Labs. dbt Documentation. [Электронный ресурс] Режим доступа: <https://docs.getdbt.com> (дата обращения: 01.12.2025)

УДК 004.738.5

Р.Р. Ганиуллин (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

ОПТИМИЗАЦИЯ КОНФИГУРАЦИИ POSTGRESQL НА ОСНОВЕ АЛГОРИТМА SMAC

Современные системы сталкиваются с проблемой обработки и хранения большого объема данных, в таких системах база данных часто оказывается узким местом. Производительность базы данных значительно зависит от ее настройки и для более эффективной ее работы производят подстройку под определенный профиль нагрузки. Подстройка под профиль или, как ее еще называют тюнинг требует глубоких знаний базы, значительных усилий по экспериментам и тестированию, что делает этот процесс долгим, дорогим и недоступным большинству разработчиков и администраторов [1].

Один из способов решения данной проблемы - автоматизация процесса тюнинга.

Целью данной работы является разработка автоматизированного подхода к настройке СУБД PostgreSQL с использованием алгоритма SMAC на основе целевых метрик, позволяющего снизить трудоёмкость работы администратора и повысить производительность системы [6].

Для достижения этой цели необходимо решение задач, перечисленных ниже.

– Анализ существующих подходов к автоматизированной настройке СУБД и методов гиперпараметрической оптимизации.

– Проектирование архитектуры системы поддержки принятия решений по оптимизации конфигурации.

– Программная реализация модуля оптимизации на языке Python с использованием библиотек Optuna [2] и SMAC [4].

– Проведение экспериментальных исследований и оценка качества формируемых рекомендаций.

Спроектированная архитектура приложения предполагает несколько компонентов:

1. **Агент** – скрипт, находящийся на той же машине, что и база данных. Предназначен для изменения файла конфигурации, перезагрузки, ожидания запуска СУБД.

2. **Prometheus** – хранилище и сборщик метрик.

3. **Оптимизатор** – программа, содержащая основную логику оптимизации, предлагает новую конфигурацию, читает метрики из Prometheus. Алгоритм, используемый для оптимизации – SMAC.

4. **Генератор нагрузки.**

5. Непосредственно СУБД.

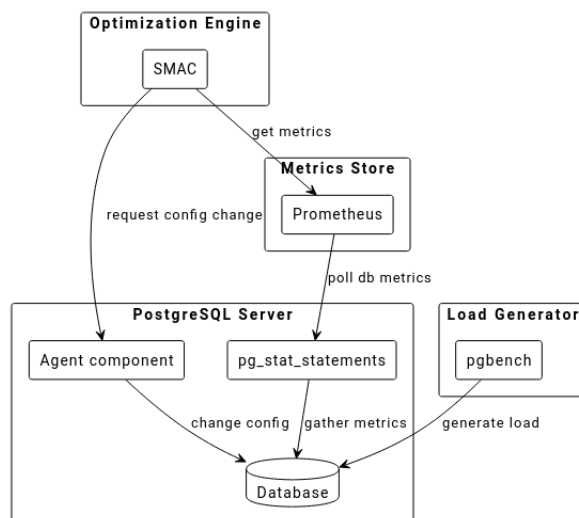


Рисунок 1 – Архитектура системы

Ожидаемый прирост производительности 10-15% по сравнению с конфигурацией, полученной путем применения стандартных рекомендаций и эвристик.

ЛИТЕРАТУРА

1. Б.А. Новиков Г.Р. Домбровская. Книга Настройка приложений баз данных. — БХВ- Петербург, 2006.
2. Optuna: A hyperparameter optimization framework [Электронный ресурс]. — Режим доступа: <https://optuna.org>, свободный. — Загл. с экрана. — Яз. англ. — Дата обращения: 21.01.2026.
3. Bergstra, J. Algorithms for Hyper-Parameter Optimization / J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl // Advances in Neural Information Processing Systems 24 (NIPS 2011). — 2011. — P. 2546–2554. — URL: <https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf> (дата обращения: 21.01.2026).
4. Lindauer, M. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization / M. Lindauer, K. Eggensperger, M. Feurer [et al.] // Journal of Machine Learning Research. — 2022. — Vol. 23, № 54. — P. 1–9. — URL: <https://doi.org/10.48550/arXiv.2109.09831> (дата обращения: 21.01.2026).
5. Van Aken, D. Automatic Database Management System Tuning Through Machine Learning / D. Van Aken, A. Pavlo, G. J. Gordon, B. Zhang // Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17). — Chicago, IL, USA, 2017. — P. 1235–1248. — DOI: 10.1145/3035918.3064029. — URL: <https://www.cs.cmu.edu/~ggordon/van-aken-et-al-parameters.pdf> (дата обращения: 21.01.2026).
6. Liu, Y. Automatic Configuration Tuning on Cloud Database: A Survey / Y. Liu, Z. Chen, X. Wang [et al.] // arXiv preprint arXiv:2404.06043. — 2024. — URL: <https://arxiv.org/pdf/2404.06043> (дата обращения: 21.01.2026).

ИССЛЕДОВАНИЕ И ВЕРИФИКАЦИЯ МЕХАНИЗМОВ ОПТИМИЗАЦИИ
ХРАНЕНИЯ СТРУКТУРЫ ДАННЫХ JSONB В СУБД POSTGRESQL 18

СУБД PostgreSQL является одной из самых востребованных баз данных, и широко используется в различных крупных проектах и корпорациях. Postgres – проект с открытым исходным кодом, имеющий долгую историю. Но развитие технологий делает некоторые ключевые механизмы, используемые в Postgres, устаревшими [1]. Одним из таких механизмов является способ хранения больших данных – TOAST (The Oversized-Attribute Storage Technique, Методика хранения сверхбольших атрибутов) [2].

Среди множества баз данных Postgres имеет ряд преимуществ, одним из которых является наличие jsonb – оптимизированного json, использование которого позволяет значительно ускорить работу базы данных [3]. Но, когда размер jsonb становится больше порога в 2 КБ, он отправляется в TOAST-таблицу. Механизм хранения сложных структур данных в ней неоптимален, что приводит к значительному падению производительности [4].

Существуют решения, реализующие алгоритмы оптимального хранения больших структур данных, которые называются toaster [5]. Toaster для jsonb был разработан для Postgres 16, но не был принят ключевыми разработчиками по ряду причин.

Таким образом, целью данной работы является исследование и адаптация данного алгоритма для актуальной версии Postgres – 18-ой.

Для достижения поставленной цели необходимо решение следующих задач:

1. Проанализировать jsonb toaster для Postgres 14.
2. Выделить ключевые алгоритмы, оптимизирующие хранение.
3. Реализовать данные алгоритмы для Postgres 18.
4. Провести анализ эффективности, верифицировать результаты.

Разработанное решение реализует программу для оптимизированного хранения jsonb в Postgres 18. Данный алгоритм использует inline хранение, оптимизацию построения объектов jsonb, при которой ключи и значения хранятся в отсортированном порядке, что делает доступ более быстрым. Также реализованы такие оптимизации, как механизм частичной декомпрессии, частичного извлечения toast значения.

Алгоритм реализован на языке C, при использовании библиотек, задействованных для реализации toaster jsonb для Postgres 14, при использовании компилятора gcc, на ОС Linux.

В результате работы создан алгоритм, реализующий оптимизированное хранение jsonb в Postgres 18. Результаты проверены при помощи механизма pgbench с использованием SQL-скриптов для имитации интенсивных нагрузок, а также при использовании различных наборов данных.

ЛИТЕРАТУРА

1. PostgreSQL Global Development Group. PostgreSQL 18 Documentation. Chapter 73. TOAST. — 2024. — URL: <https://postgrespro.ru/docs/postgresql/current/storage-toast>.
2. Рогов Е. В. PostgreSQL изнутри. — М.: ДМК Пресс, 2022.
3. Бартунов О. С., Сигаев Ф. А. JSONB: история создания и развитие бинарного типа данных в PostgreSQL // Материалы конференции HighLoad++. — 2014.
4. PostgreSQL Global Development Group. PostgreSQL Source Code: src/backend/utils/adt/jsonb.c. — 2024. URL: <https://github.com/postgres/postgres/blob/master/src/backend/utils/adt/jsonb.c>.
5. Иванов И., Панюков В. Как мы сделали свой TOAST для JSONB // Хабр. — 2023. — URL: <https://habr.com/ru/companies/postgrespro/articles/710104/>.

РАЗРАБОТКА ПРОГРАММНОГО ИНСТРУМЕНТА ДЛЯ МОНИТОРИНГА СОБЛЮДЕНИЯ
СОГЛАШЕНИЙ ОБ УРОВНЕ СЕРВИСА В КОРПОРАТИВНЫХ ПРОЦЕССАХ

Современные IT-компании функционируют в условиях высокой динамики бизнес-процессов и распределенной архитектуры информационных систем. Ключевым инструментом формализации обязательств перед заказчиками выступают соглашения об уровне сервиса (SLA), фиксирующие нормативные сроки обработки инцидентов, выполнения запросов и реализации изменений. По мере масштабирования деятельности компании, сопровождающегося ростом числа клиентов и проектов, объем данных, подлежащих контролю, существенно возрастает. Ручной анализ выполнения SLA становится трудоемкой задачей, требующей значительных временных ресурсов и повышающей риск ошибок, связанных с человеческим фактором.

Существующие платформы управления задачами и ITSM-системы (Jira, ServiceNow, YouTrack) предоставляют базовые механизмы отслеживания сроков, однако их функционал ограничен типовыми сценариями и требует значительной кастомизации для учета специфических условий конкретных соглашений, мультипроектной структуры и контекстных факторов, влияющих на фактическое время выполнения задач.

Совокупность перечисленных факторов обосновывает актуальность разработки специализированного программного инструмента, ориентированного на внутреннюю структуру конкретной компании [1] и обеспечивающего автоматизированный контроль SLA с учетом всех значимых факторов. Потребность в таком решении носит межотраслевой характер: с аналогичными проблемами сталкиваются организации, осуществляющие проектную деятельность, аутсорсинговые компании, провайдеры облачных услуг и внутренние IT-подразделения крупных корпораций [2].

Таким образом, *целью* данной работы является повышение прозрачности и эффективности контроля соблюдения соглашений об уровне сервиса (SLA) в корпоративных процессах за счёт разработки программного инструмента мониторинга выполнения задач, обеспечивающего автоматизированный анализ сроков и условий исполнения с учётом специфики корпоративной среды.

Для достижения обозначенной цели были поставлены следующие *задачи*:

1. Проведение комплексного анализа существующих программных решений в области мониторинга SLA.
2. Анализ проблем и ограничений контроля сроков выполнения задач в рамках рассматриваемой корпоративной системы.
3. Формирование требований и проектирование архитектуры разрабатываемого программного инструмента.
4. Реализация системы мониторинга, удовлетворяющей установленным требованиям.
5. Выполнение экспериментальной оценки эффективности созданного решения на основе практических данных.

Выявленные при сравнительном анализе ограничения обусловили необходимость разработки специализированного корпоративного приложения, архитектура которого представлена на рисунке 1. Основными компонентами системы являются: центральная точка входа запросов к системе (API Gateway), сервис сбора данных (Data Collector), сервис анализа и расчёта SLA (Calculator, Analysis Engine), сервис визуализации и отчетности (Dashboard Visualization), пользовательский интерфейс (GUI). Также для возможности интеграции с внешними системами используются очереди событий (Message Broker).

Заложенная в основу разработки модульная архитектура создает основу для дальнейшего расширения функционала по нескольким перспективным направлениям: интеграция с дополнительными источниками данных позволит обеспечить сквозной контроль SLA по всем категориям процессов [3], а адаптация приложения для последующей интеграции в корпоративные сервисы самих клиентов позволит предоставлять заказчикам прямой доступ к дашбордам с показателями соблюдения SLA [4].

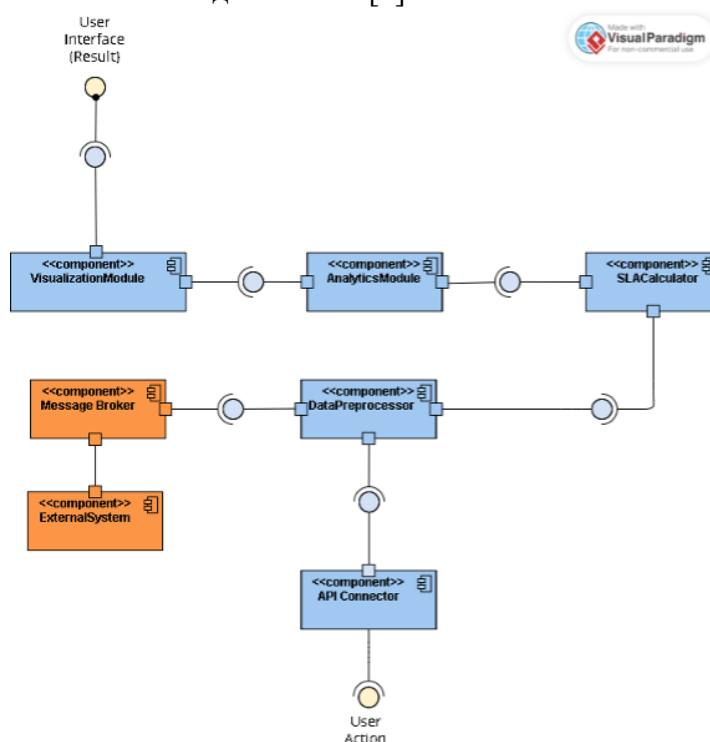


Рисунок 1 – Архитектура системы

Проведенная апробация разработанного решения на базе одного из департаментов компании подтвердила его работоспособность и эффективность. В ходе тестирования на массиве реальных задач было установлено, что автоматизированный расчет SLA с учетом контекстных факторов [5] позволяет существенно снизить трудоемкость формирования отчетности по сравнению с ручным анализом. Применение разработанного инструмента обеспечило выявление значительного числа задач, ранее ошибочно классифицированных как выполненные в срок, за счет корректного учета периодов ожидания и календарных ограничений.

Таким образом, внедрение разработанного приложения способствует повышению прозрачности корпоративных процессов, укреплению репутации компании и минимизации финансовых потерь, связанных с несоблюдением договорных обязательств.

ЛИТЕРАТУРА

1. Главная страница компании Solanteq [Электронный ресурс]. – Режим доступа: <https://solanteq.ru/products-and-solutions/>
2. Labidi T., Mtibaa A., Gargouri F. Cloud SLA Modeling and Monitoring // Proceedings of the 15th International Conference on Software Engineering Research and Practice (SERP'17). Las Vegas, USA, 2017. [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/319867515_Cloud_SLA_Modeling_and_Monitoring
3. YouTrack: документация и поддержка [Электронный ресурс]. – Режим доступа: <https://www.jetbrains.com/help/youtrack/>
4. Yandex Cloud: документация [Электронный ресурс]. – Режим доступа: <https://yandex.cloud/ru/docs>
5. Федотов М. А., Смирнов К. Н. Модели и алгоритмы прогнозирования нарушений соглашений об уровне сервиса в ИТ-сервисах // Информационные технологии и вычислительные системы. 2024. № 4. С. 45–56.

РАЗРАБОТКА МЕТОДОВ АВТОМАТИЗИРОВАННОЙ ГЕНЕРАЦИИ ТЕСТ-КЕЙСОВ ДЛЯ
КАСТОМИЗИРУЕМЫХ API-ЗАПРОСОВ

Современные RESTful API всё чаще поддерживают кастомизацию запросов в реальном времени посредством опциональных параметров, заголовков и динамических тел запросов. Подобная гибкость существенно усложняет процесс тестирования: для конечной точки с n параметрами, каждый из которых принимает v допустимых значений, пространство возможных конфигураций составляет произведение всех v , что делает полный перебор практически невозможным. Ручное проектирование тест-кейсов не масштабируется при комбинаторном росте вариантов запросов, а недостаточное покрытие повышает вероятность необнаруженных дефектов, особенно вызванных взаимодействием параметров [1-3].

Существующие подходы к автоматизации тестирования API имеют характерные ограничения. Контрактное тестирование (Pact, Dredd) проверяет соответствие схеме, но не обеспечивает покрытие комбинаций опциональных параметров. Модельное тестирование (GraphWalker) требует значительных трудозатрат на построение и поддержание моделей. Методы на основе машинного обучения (DeepREST [4], KAT [5] и др. [6-8]) адаптивны, но зависят от объёма обучающих данных и сложны в развёртывании. При этом ни один из перечисленных подходов не учитывает семантику параметров и бизнес-логику предметной области при генерации значений.

В данной работе предлагается двухэтапный подход к генерации тест-кейсов, объединяющий детерминированные комбинаторные методы с коррекцией на основе большой языковой модели (LLM). На первом этапе из OpenAPI-спецификации автоматически извлекаются параметры конечных точек и строятся их домены значений с учётом ограничений схемы (перечисления, диапазоны, форматы). Для опциональных параметров добавляется специальное значение ABSENT, моделирующее их отсутствие. Далее применяется жадный алгоритм IPOG-lite для попарного (pairwise) покрытия, дополненный генерацией граничных значений (min, max, min-1, max+1) и негативных сценариев (отсутствие обязательных параметров, некорректные типы данных, невалидные форматы).

На втором этапе полученный набор тест-кейсов направляется в LLM (Claude API), которая выполняет четыре задачи: замену синтетических значений параметров на семантически корректные (например, «test_string» заменяется на реалистичный адрес электронной почты); добавление бизнес-логических сценариев, не выводимых из формальной спецификации (например, создание заказа со скидкой, превышающей стоимость товара); генерацию человекочитаемых описаний каждого теста; уточнение ожидаемых HTTP-статусов и подсказок по содержанию ответа.

Подход реализован в виде CLI-инструмента на языке Java 17 с использованием Swagger Parser для анализа спецификаций, собственной реализации pairwise-генератора и модуля интеграции с Claude API. Результаты экспортируются в формат Postman Collection v2.1 [9] и в исполняемые JUnit/RestAssured тесты.

Экспериментальная оценка проведена на нескольких OpenAPI-спецификациях с различным числом параметров (от 12 до 28). Для спецификации с 25 параметрами генерация 980 тест-кейсов занимает менее 1 секунды, экспорт — менее 0,25 секунды. Попарное покрытие составляет 0,93–0,95. LLM-коррекция добавляет в среднем 15–20% дополнительных сценариев, не обнаруживаемых комбинаторными методами, и повышает читаемость описаний тестов, что упрощает их сопровождение в рамках CI/CD-процессов [10].

Таким образом, предложенный интегративный подход позволяет сочетать формальную полноту комбинаторного покрытия с семантической осмысленностью, обеспечиваемой

языковой моделью, что делает его практически применимым для автоматизированного тестирования кастомизированных API.

ЛИТЕРАТУРА

1. Cohen D. M., Dalal S. R., Fredman M. L., Patton G. C. The AETG System: An Approach to Testing Based on Combinatorial Design // IEEE Transactions on Software Engineering. – 1997. – Т. 23, № 7. – С. 437–444.
2. Kuhn D. R., Wallace D. R., Gallo A. M. Software Fault Interactions and Implications for Software Testing // IEEE Transactions on Software Engineering. – 2004. – Т. 30, № 6. – С. 418–421.
3. Еммануел, Н. Н. Ч. Тестирование производительности виртуальных машин при использовании гипервизоров VMware Workstation и Oracle Virtualbox при использовании ОС Linux / Н. Н. Ч. Еммануел, А. Сергеев // Theoretical & Applied Science. – 2024. – № 12(140). – С. 25-37. – DOI 10.15863/TAS.2024.12.140.5. – EDN CWIGSA.
4. Corradini F. et al. DeepREST: Deep Reinforcement Learning for REST API Testing // Proceedings of ICST. – 2024.
5. Le T. et al. Knowledge-Aware Test Generation using GPT and Graph Analysis // arXiv preprint. – 2024.
6. Королев, Д. О. Исследование эффективности применения бинарных нейронных сетей при детектировании объекта на изображении / Д. О. Королев, О. Г. Малеев // Информатика и ее применения. – 2023. – Т. 17, № 3. – С. 88-92. – DOI 10.14357/19922264230312. – EDN TOCVAl.
7. Чигасова, М. А. Прогнозирование котировок акций с использованием нейронной сети / М. А. Чигасова, О. Г. Малеев // Современные технологии в теории и практике программирования : Сборник материалов конференции, Санкт-Петербург, 26 апреля 2022 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2022. – С. 180-182. – EDN GRFEDM.
8. Ларионов, В. С. Эффективность распараллеливания метода на основе роя частиц при оптимизации обучения нейронных сетей / В. С. Ларионов, О. Г. Малеев // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2022. – № 7. – С. 71-77. – DOI 10.37882/2223-2966.2022.07.18. – EDN EIZWER.
9. Postman Inc. Postman Collection Format v2.1. [Электронный ресурс] Режим доступа: <https://learning.postman.com/collection-format/> (дата обращения: 08.03.2026).
10. Тимошук, Е. В. Анализ данных профилей сотрудников в социальной сети linkedin / Е. В. Тимошук, И. В. Никифоров, Д. Ф. Дробинцев // Современные технологии в теории и практике программирования : сборник материалов конференции, Санкт-Петербург, 23 апреля 2020 года / Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2020. – С. 133-135. – EDN MXXIHN.

УДК 004.415.53

В.О. Гудков (4 курс бакалавриата),
С.Э. Сараджишвили, к.т.н., доцент

РАЗРАБОТКА УТИЛИТЫ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ ДЛЯ РУЧНЫХ ЗАПРОСОВ К RPC МИКРОСЕРВИСОВ

Тестирование разрабатываемого программного продукта является неотъемлемым этапом в разработке программного обеспечения, который во многом определяет его итоговое качество. В связи с этим улучшение данного процесса представляет собой важнейшую задачу [1]. Несмотря на популярность автоматизированных методов тестирования, трудно переоценить вклад ручного тестирования в качество конечного продукта [2].

При использовании микросервисной архитектуры [3] для проектирования приложений взаимодействие сервисов с применением технологии Remote Procedure Call (RPC) получает всё более широкое распространение за счёт высокой надёжности интеграции между компонентами системы и передовой эффективности передачи данных.

Целью данной работы является упрощение процесса ручного тестирования микросервисного приложения за счёт разработки утилиты для ручного составления запросов к RPC сервисов. Для достижения упомянутой цели необходимо провести обзор и сравнительный анализ методик разработки подобных утилит [4], а затем разработать утилиту в соответствии с выбранной методикой.

Существует три основные методики разработки утилит для ручного тестирования программных продуктов: мобильная, веб- и десктопная разработка. Проведённый сравнительный анализ выделенных методик помог определить ключевые особенности каждой из них. Мобильная разработка строго ограничивает объём доступной утилите памяти, а также значительно усложняет встраивание и вызов нативных библиотек [5]. Веб-разработка увеличивает задержку при отправке запроса к RPC сервиса, вводя дополнительный сетевой уровень в виде HTTP моста, что также создаёт дополнительную точку отказа системы [6]. При этом десктопная разработка позволяет достичь минимальной задержки при обмене данными между сервисами и расширить спектр доступных для выбора средств разработки ввиду отсутствия дополнительных ограничений. По результатам проведённого анализа была выбрана десктопная методика разработки утилиты.

Для разработки приложения был выбран язык программирования Python с использованием подключаемой библиотеки PySide6 для создания графического интерфейса. Выбор данного набора инструментов был обусловлен требованиями заказчика разработки утилиты. Приложение обладает кроссплатформенной архитектурой, что обеспечивает поддержку операционных систем семейств Windows и Linux. Набор элементов графического интерфейса приложения включает в себя окно авторизации в системе, вход в которую осуществляется по логину, паролю и адресу машины, а также основное окно утилиты, в котором происходит указание параметров запроса к RPC сервисов, его отправка и вывод результата в формате json.

Пример основного окна утилиты представлен на рисунке 1. Получившийся графический интерфейс обладает рядом функциональных особенностей, среди которых возможность добавления новых RPC и хранение информации с описанием уже известных, механизм проведения валидации данных на соответствие типов вводимых параметров ожидаемым, поддержка вывода данных со сложной структурой вложенных полей в формате json, разделение полей ввода обязательных и опциональных параметров.

```
{
  "param1": "TEST1",
  "param2": 1111,
  "param3": {
    "sub1": {
      "sub1a": "TEST2",
      "sub1b": "TEST3"
    },
    "sub2": "TEST4"
  },
  "param4": false,
  "param5": "TEST5",
  "opt1": "TEST6",
  "opt2": "TEST7",
  "opt3": "TEST8",
  "opt4": "TEST9",
  "opt5": "TEST10",
  "complex1": {
    "p1": {
      "a": "TEST11",
      "b": "TEST12",
      "c": "TEST13",
      "d": "TEST14"
    },
    "p2": {
      "x": "TEST15",
      "y": "TEST16",
      "z": "TEST17",
      "w": "TEST18"
    },
    "p3": "TEST19"
  }
}
```

Рисунок 1 – Интерфейс окна отправки запросов к RPC сервисов

Проверка скорости и отказоустойчивости отправки запросов с помощью разработанной утилиты проводилась в автоматическом режиме. Для системы под управлением ОС Windows

10 версии 22H2 общее количество вызовов составило 1421, количество неудачных вызовов – 3, а среднее время выполнения запроса – 185664051 нс. Для системы под управлением дистрибутива Linux «РЕД ОС» версии 8 результаты измерений составили 1508, 3 и 210242900 нс соответственно. Аналогично, для системы под управлением дистрибутива Linux «Альт Рабочая станция» версии 10.4 результатами оказались значения 1412, 2 и 174624406 нс соответственно.

В результате была спроектирована и реализована утилита для ручного составления запросов к RPC сервисов. За счёт использования графического пользовательского интерфейса удалось значительно облегчить и ускорить процесс ручного тестирования сервисов приложения. Исходя из полученных в ходе отладки утилиты данных, выбор десктопного подхода разработки оказался оправдан, так как он обеспечил высокую скорость отправки запросов, при этом сохраняя оптимальную отказоустойчивость.

ЛИТЕРАТУРА

1. Haas R. et al. How can manual testing processes be optimized? developer survey, optimization guidelines, and case studies // Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. – 2021. – С. 1281-1291.
2. Bhanushali A. Impact of automation on quality assurance testing: A comparative analysis of manual vs. automated qa processes // International Journal of Advances in Engineering Research. – 2023. – Т. 4. – С. 26.
3. Cerny T. et al. Microservice architecture reconstruction and visualization techniques: A review // 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE). – IEEE, 2022. – С. 39-48.
4. Srivastava N. et al. Software and performance testing tools // Journal of Informatics Electrical and Electronics Engineering. – 2021. – Т. 2. – №. 01. – С. 1-12.
5. Исаченков А. С., Эльсесер Г. П., Гречкина О. В. Разработка мобильного приложения, предоставляющего пользовательский интерфейс для использования нейронной сети, разработанной с использованием фреймворка Keras // Известия Тульского государственного университета. Технические науки. – 2023. – №. 9. – С. 244-250.
6. Баранова Е. М. и др. Разработка безопасного веб-приложения для организации интернет-мероприятий // Известия Тульского государственного университета. Технические науки. – 2024. – №. 9. – С. 334-337.

УДК 004.45

П.О. Гудкова (2 курс магистратуры),
И.В. Никифоров, к. т. н., доцент

АВТОМАТИЗИРОВАННАЯ СИСТЕМА МОНИТОРИНГА И АНАЛИЗА ЗАГРУЖЕННОСТИ СЕРВЕРОВ ЦЕНТРОВ ОБРАБОТКИ ДАННЫХ

Современный этап развития информационных технологий характеризуется экспоненциальным ростом объемов обрабатываемых данных и числа подключаемых устройств, что приводит к стремительному усложнению инфраструктуры центров обработки данных (ЦОД). Традиционные методы планирования [1] ресурсов и ручное управление нагрузкой становятся неэффективными, что влечет за собой финансовые потери от простоев оборудования или его недогрузки, а также избыточное энергопотребление [2]. В связи с этим разработка систем, способных не только констатировать текущее состояние, но и прогнозировать динамику нагрузки, является актуальной научно-практической задачей.

Целью данной работы является повышение качества распределения нагрузки на центры обработки данных за счет разработки интеллектуальной системы мониторинга, обеспечивающей улучшение управления вычислительными ресурсами.

Для достижения поставленной цели необходимо решить следующие задачи: провести анализ предметной области в части методов мониторинга и оценки нагрузки на серверы; выполнить сравнительный анализ существующих подходов к мониторингу, выявив их достоинства и недостатки; разработать архитектуру интеллектуальной системы мониторинга, интегрирующую методы машинного обучения; реализовать предложенный подход в виде программного прототипа; провести экспериментальное исследование разработанной системы для подтверждения повышения качества управления ресурсами.

В ходе исследования был проведен сравнительный анализ методов мониторинга нагрузки. Рассматривались пороговый метод, статистическое прогнозирование, методы глубокого обучения и гибридный подход

Анализ показал, что пороговые методы просты, но не адаптивны, а статистические модели не справляются с нелинейностью современных нагрузок. Методы глубокого обучения, в частности долгая краткосрочная память (LSTM), демонстрируют высокую точность прогнозирования сложных временных рядов [3]. В связи с этим оптимальным компромиссом представляется гибридный подход, который предполагает поэтапное обогащение стандартного стека мониторинга интеллектуальным контуром на основе LSTM [4].

Разработанная архитектура такой системы представлена на рисунке 1. На нижнем уровне (слой источников данных) происходит сбор метрик с серверов с использованием Node Exporter. Слой сбора и передачи реализован на системе Prometheus, которая агрегирует потоки метрик. Ключевым компонентом слоя хранения и обработки является блок ML на Python, использующий исторические данные из базы данных SQLite для обучения моделей LSTM. Обученные модели сохраняются в специальном хранилище и используются для прогнозирования нагрузки. Выявленные прогнозы и аномалии передаются обратно в контур мониторинга. Слой представления базируется на Grafana, которая обеспечивает визуализацию метрик и прогнозов, а также позволяет настраивать триггеры. Модуль оповещений направляет уведомления администратору при обнаружении потенциально опасных ситуаций.

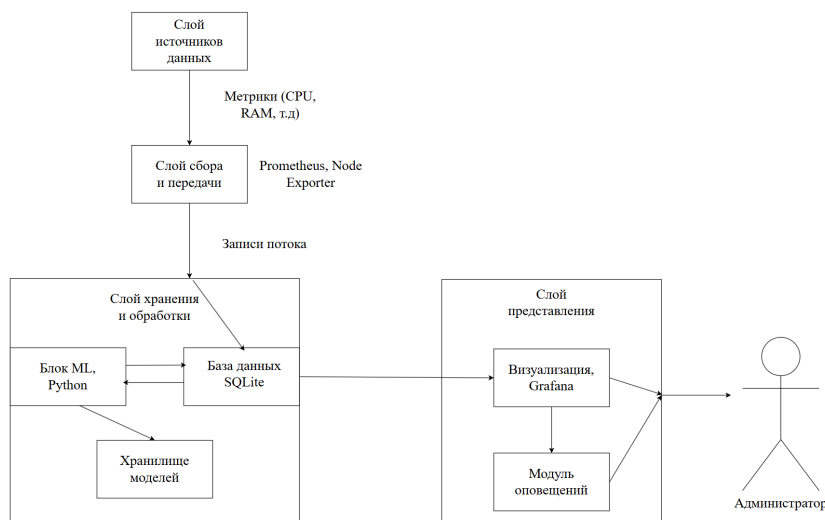


Рисунок 1 – Архитектура системы мониторинга нагрузки

На основе предложенной архитектуры был разработан программный прототип системы на языке Python с использованием библиотек TensorFlow и PyTorch для реализации LSTM-моделей. Интеграция с Prometheus осуществлена через API, визуализация выполнена в Grafana.

Таким образом, в ходе работы был выполнен системный анализ проблемы управления ресурсами ЦОД, проведен сравнительный анализ методов мониторинга, разработана архитектура и реализован программный прототип интеллектуальной системы мониторинга. Экспериментально подтверждено, что предложенный гибридный подход, интегрирующий проверенные решения (Prometheus, Grafana) с методами глубокого обучения (LSTM),

позволяет перейти от реактивного к проактивному управлению ресурсами ЦОД и повысить эффективность их использования [5,6].

ЛИТЕРАТУРА

1. Algorithm for calculating TCO and SCE metrics to assess the efficiency of using a data center / B. Denisenko, M. Tyanutov, I. Nikiforov, S. Ustinov // 2nd International Conference on Computer Applications for Management and Sustainable Development of Production and Industry (CMSD-II-2022), Dushanbe, 21–23 декабря 2022 года. Vol. 12564. – Washington: SPIE-SOC PHOTO-OPTICAL INSTRUMENTATION ENGINEERS, 2023. – P. 1256403. – DOI 10.1117/12.2669285. – EDN TSQXHS.
2. Yadav M. P., Pal N., Yadav D. K. Workload Prediction over Cloud Server using Time Series Data // 2021 11th International Conference on Cloud Computing, Data Science & Engineering. – 2021. – С. 267–272.
3. El Yadari M., El Motaki S., Yahyaouy A., Makany P., El Fazazy K., Gualous H., Le Masson S. Long Short-Term Memory Networks for Server Workload Prediction // 2024 International Conference on Intelligent Systems and Computer Vision (ISCV). – 2021.
4. Saxena D., Singh A. K. Auto-adaptive learning-based workload forecasting in dynamic cloud environment// International Journal of Computers and Applications. – 2020. – С. 1–11.
5. Программные инструменты обработки и визуализации данных. Elasticsearch, Logstash, Kibana, Grafana, Prometheus / И. В. Никифоров, О. А. Юсупова, Н. В. Воинов [и др.]. – Санкт-Петербург : Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. – 140 с. – ISBN 978-5-7422-8075-0. – DOI 10.18720/SPBPU/2/id23-74. – EDN KPGDVS.
6. Методы, алгоритмы и архитектуры распределенной обработки больших данных / И. В. Никифоров, О. А. Юсупова, Н. В. Воинов, А. Д. Ковалев. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2023. – 194 с. – ISBN 978-5-7422-8461-1. – EDN DFTKEX.

УДК 004.4'24

Ю.В. Денисова (ст. преподаватель),
Ю.М. Бородянский, к.т.н., доцент

РАЗРАБОТКА МАТРИЦЫ КЛАССИФИКАЦИИ ЗАДАЧ SDLC ДЛЯ СЕЛЕКТИВНОГО ВНЕДРЕНИЯ ГЕНЕРАТИВНОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОЦЕСС РАЗРАБОТКИ ПО

В нынешнее время активно распространяются LLM (Large Language Models), в частности, в разработке ПО, к примеру, GitHub Copilot. Можно найти исследования [1], где показано, что команда разработчиков, использующая ИИ-инструмент в разработке на 55,8% быстрее выполнила задачу, чем команда, обходившаяся без его использования. Это существенно ускоряет процесс разработки ПО, однако есть и другая сторона: при бесконтрольном использовании подобных инструментов может пострадать безопасность и надежность создаваемых продуктов ввиду избыточного доверия ИИ.

Таким образом, можно сказать, что большинство исследований сводятся к исследованию положительного влияния ИИ-инструментов на разработку, например, в статьях [2], где описан подход с использованием LLM в инженерии ПО и [3] – фундаментальное исследование влияния ИИ на производительность; тогда как в данной статье предпринимается попытка разработки матрицы классификации задач SDLC, которая станет частью модели селективного внедрения ИИ в процесс разработки ПО. При этом стоит также отметить, что использование ИИ-инструментов сопряжено с определенными рисками, например, есть вероятность утечки персональных данных [4], не говоря уже о том, что код, созданный ИИ имеет значительное количество уязвимостей безопасности [5].

Цель работы состоит в повышении надежности и безопасности ПО за счет разработки матрицы классификации задач и алгоритма оценки задач. Для этого необходимо выполнить следующие задачи:

- выделить ключевые метрики риска и сложности задач SDLC;
- разработать систему балльной оценки по осям «Критичность» и «Шаблонность»;
- определить зоны матрицы с соответствующими режимами работы;
- создать алгоритм оценки задачи для определения её положения в матрице;
- верифицировать алгоритм на примерах реальных задач разработки.

Объектом исследования в данном случае выступает процесс разработки ПО в условиях интеграции ИИ-инструментов, а предметом исследования – методы и критерии классификации задач разработки для определения допустимой степени автономности для ИИ-инструментов.

В докладе речь идет о матрице оценки рисков и возможности применения, который является частью модели применения селективного применения ИИ в процессе разработки ПО.

Вид матрицы модуля представлен на рисунке 1.

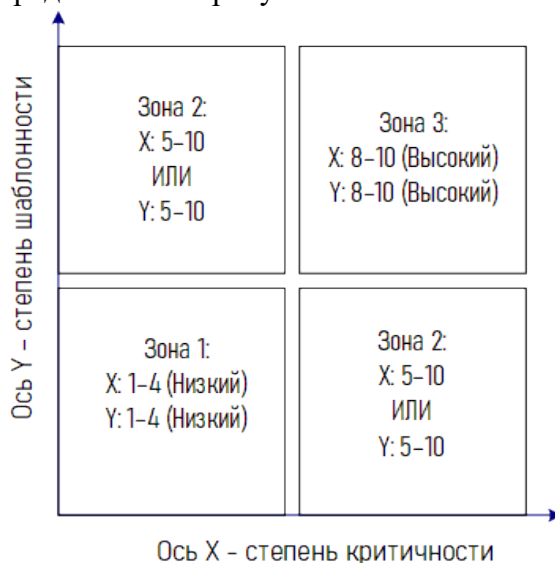


Рисунок 1 – График матрицы оценки рисков и возможности применения

В ходе разработки выделены три зоны, в которые может попасть та или иная задача (от верстки кнопки модального окна до рефакторинга расчета финансовых данных в старых системах). В зависимости от зоны можно сделать представление, насколько можно доверять ИИ решению каждой конкретной задачи, а также оценить риски использования ИИ-инструмента. Помимо этого, для каждой зоны есть свои способы верификации работы, выполненной с помощью ИИ.

После этого разработан подробный алгоритм, позволяющий провести расчеты и определить зону для каждой задачи. Он представляет собой подсчет баллов для каждой из осей: «критичность» и «шаблонность», где 0 – это минимум, а 10 – это максимум.

Для каждой из осей определены свои критерии. Для оси X это безопасность, целостность данных, бизнес-непрерывность и репутационный риск. Для оси Y это наличие паттернов, глубина контекста, уникальность логики задачи, состояние кода.

В конечном счете разработанная матрица позволяет реализовать часть модуля для селективного внедрения ИИ-инструментов в процесс разработки ПО.

В дальнейшем предполагаются исследования для автоматизации сбора метрик для осей матрицы.

ЛИТЕРАТУРА

1. Sida Peng, Eirini Kalliamvakou, Peter Cihon, Mert Demirer. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot // Software Engineering. Изд-во Cornell University, 2023. – 19 С.

2. Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, Haoyu Wang. Large Language Models for Software Engineering: A Systematic Literature Review // Software Engineering. Изд-во Cornell University, 2023. Вып. 220. С. 1 - 79
3. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot // arxiv.org URL: <https://arxiv.org/pdf/2302.06590> (дата обращения: 13.03.2026).
4. Конституция Российской Федерации "Федеральный закон «О персональных данных»" от 27.07.2006 № 152-ФЗ 2006
5. Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, Ramesh Karri. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. // Cryptography and Security. Изд-во Cornell University, 2025. Вып. 68,2. С. 96-105

УДК 004.052.3

А.О. Джичко (4 курс бакалавриата),
И.А. Шемякин, ст. преподаватель

ПРОЕКТИРОВАНИЕ ОТКАЗОУСТОЙЧИВОЙ АРХИТЕКТУРЫ НАГРУЖЕННОГО ВЕБ-СЕРВИСА

В современных условиях разработка высоконагруженных веб-сервисов требует от архитекторов поиска компромиссов, определяемых теоремой CAP [1]. Переход от монолитных приложений к микросервисной архитектуре, решая проблемы масштабируемости, порождает новые вызовы, связанные с обеспечением отказоустойчивости распределённых взаимодействий и целостности данных. Особую остроту эта проблема приобретает для малых команд разработки, которые не могут позволить себе сложность инфраструктуры технологических гигантов.

В данной работе предлагается усовершенствование архитектуры прототипа социальной сети, изначально реализованной как набор микросервисов с синхронным REST-взаимодействием через API Gateway и асинхронной коммуникацией через брокера сообщений RabbitMQ. Несмотря на корректную работу основных сервисов, текущая реализация обладает критическими уязвимостями: отсутствуют механизмы защиты от каскадных сбоев, не гарантируется атомарность при публикации событий и отсутствует комплексная система наблюдения за состоянием распределённой системы.

Целью работы является разработка подхода к модернизации архитектуры, обеспечивающего её отказоустойчивость, масштабируемость и наблюдаемость при сохранении приемлемого уровня сложности для малой команды разработки.

Для достижения поставленной цели решаются следующие задачи:

1. Анализ существующих паттернов обеспечения надёжности для синхронного и асинхронного взаимодействий.
2. Формулирование критериев выбора технологического стека, балансирующего между функциональностью и простотой эксплуатации.
3. Проектирование обновлённой архитектуры и её реализация.

В результате проведённого анализа предложена архитектура (рисунок 1), ключевой особенностью которой является комбинирование нескольких уровней защиты. Для синхронных вызовов между API Gateway и сервисами предлагается внедрение паттерна Circuit Breaker с использованием библиотеки Resilience4j, что позволит локализовать сбои и предотвратить каскадное распространение ошибок [2, 3]. Для обеспечения атомарности операций при асинхронном взаимодействии выбран паттерн Transactional Outbox [1, 2], реализуемый через таблицу-очередь в базе данных каждого сервиса, что гарантирует сохранность события даже при сбое перед отправкой в брокер.

Неотъемлемым компонентом предложенной архитектуры является система наблюдаемости, построенная на стандартах OpenTelemetry. Внедрение распределённой

трассировки позволяет отслеживать полный путь запроса, проходящего как через синхронные вызовы, так и через асинхронные события в RabbitMQ. Сбор метрик и их визуализация дают возможность контролировать ключевые показатели, такие как лаг репликации в CQRS-подобной модели и глубину очередей, что превращает наблюдаемость из инструмента отладки в основу управления надёжностью [4, 5].

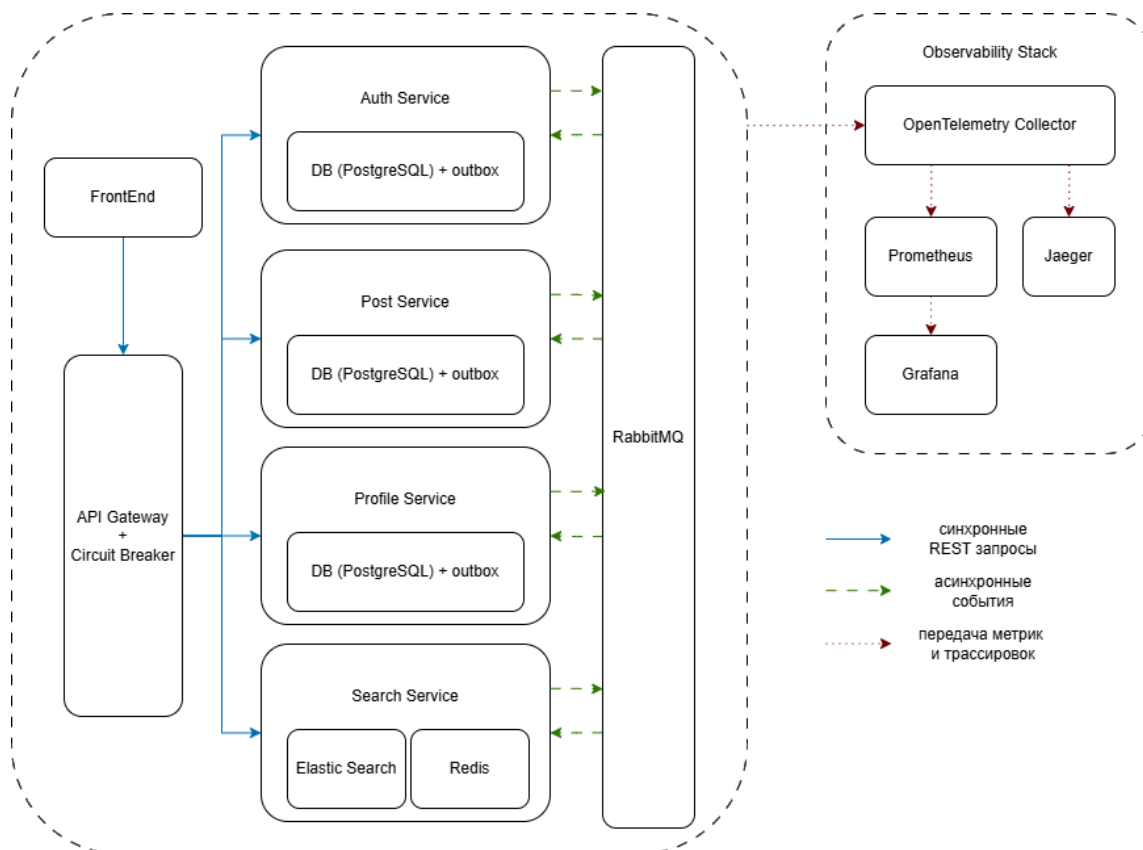


Рисунок 1 – Архитектура системы

Предложенный подход к модернизации архитектуры не только устраняет критические уязвимости исходной реализации, но и формирует фундамент для дальнейшего масштабирования системы. Интеграция паттернов Circuit Breaker, Transactional Outbox и внедрение наблюдаемости на базе OpenTelemetry обеспечивают контролируемую сложность архитектуры. Это создаёт условия для развития платформы, позволяя малой команде наращивать функциональность без потери управляемости и надёжности.

ЛИТЕРАТУРА

1. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. – Sebastopol: O'Reilly Media, 2017. – 614 p.
2. Franco J. et al. A Systematic Literature Review on Microservices Resilience Patterns // IEEE Access. – 2023. – Vol. 11. – P. 45678–45695. – DOI: 10.1109/ACCESS.2023.3274521.
3. Котов В.А., Смирнов А.А. Применение паттернов отказоустойчивости при разработке микросервисных приложений // Программная инженерия. – 2023. – Т. 14, № 3. – С. 115–124. – DOI: 10.17587/prin.14.115-124.
4. Richardson C. Microservices Patterns: With examples in Java. – Shelter Island: Manning Publications, 2019. – 520 p.
5. Сидоров П.П., Иванова Е.А. Методы и инструменты наблюдаемости в микросервисных архитектурах // Современные информационные технологии и ИТ-образование. – 2023. – Т. 19, № 4. – С. 45–55. – DOI: 10.25559/SITITO.019.202304.045-055.

ИНТЕГРАЦИЯ ЖУРНАЛОВ АУДИТА СИСТЕМЫ УПРАВЛЕНИЯ СЕКРЕТАМИ OPENBAO С ЦЕНТРАЛИЗОВАННЫМ СТЕКОМ МОНИТОРИНГА НА ОСНОВЕ LOKI И GRAFANA В KUBERNETES-КЛАСТЕРЕ

Обеспечение безопасности инфраструктуры на базе Kubernetes предполагает не только разграничение доступа к конфиденциальным данным, но и непрерывный мониторинг операций с ними. Особую роль в данном контексте играют системы управления секретами, посредством которых осуществляется хранение и выдача криптографических ключей, паролей, сертификатов и иных чувствительных артефактов [1]. Отсутствие централизованной наблюдаемости за операциями таких систем создаёт существенный пробел в модели безопасности кластера: несанкционированные обращения к секретам, аномальные паттерны чтения или потенциальные утечки остаются незафиксированными в контексте общей инфраструктуры [2].

Целью настоящей работы является разработка и апробация подхода к интеграции журналов аудита системы управления секретами OpenBao с централизованным стеком мониторинга, включающим агрегатор логов Loki, агент сбора Promtail и систему визуализации Grafana, развёрнутыми в производственном кластере k3s на базе Debian 12 [3].

OpenBao – форк HashiCorp Vault, распространяемый под открытой лицензией, – предоставляет встроенный механизм аудита, реализованный в виде подключаемых бэкендов [4]. В рассматриваемой конфигурации задействован файловый бэкенд: каждая операция взаимодействия с хранилищем фиксируется в структурированном JSON-формате с указанием типа операции, пути доступа, идентификатора субъекта аутентификации, временной метки и результата запроса. Данная схема обеспечивает машиночитаемый и семантически однородный поток событий, пригодный для автоматизированной обработки.

Сбор и доставка журналов в Loki осуществляются посредством Promtail, развёрнутого в кластере в виде DaemonSet [5]. Конфигурация агента предусматривает целевое отслеживание файла аудита OpenBao с применением конвейера обработки (pipeline stages): на этапе json-парсинга из каждой записи извлекаются поля operation, path и auth.display_name, которые затем преобразуются в метки Loki. Присвоение метки job со значением openbao-audit обеспечивает изоляцию потока аудита от прочих источников журналирования и позволяет формировать точечные запросы без сканирования нерелевантных данных. Общая архитектура интеграции представлена на рисунке 1.

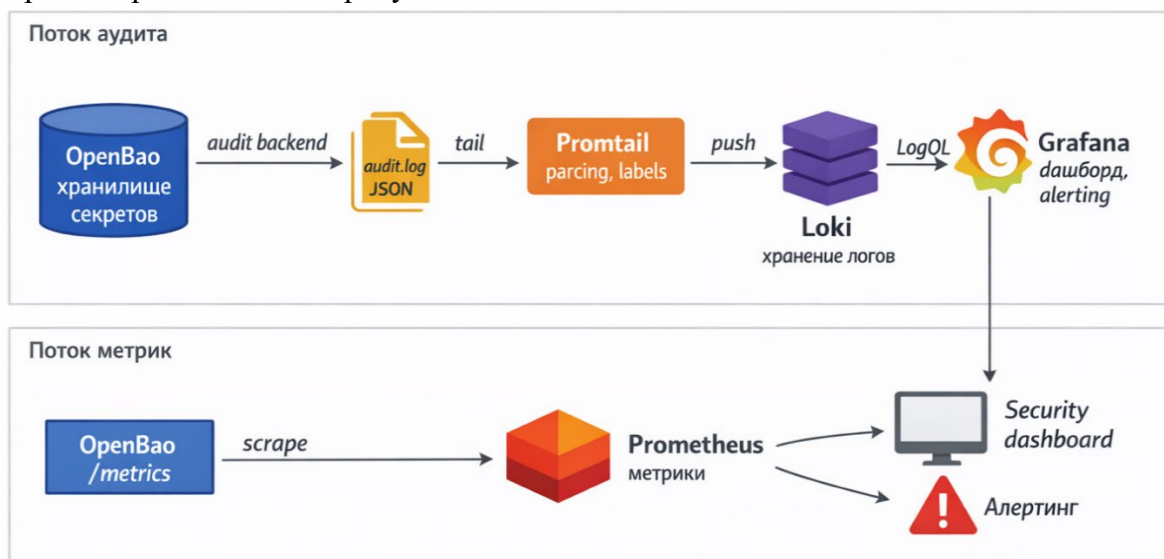


Рисунок 1 – Архитектура интеграции OpenBao с централизованным стеком мониторинга

Хранение и индексация событий выполняются в Loki с использованием встроенного механизма label-based partitioning. Запросы на языке LogQL позволяют реализовывать типовые сценарии анализа безопасности: подсчёт частоты обращений к конкретным путям секретов, фильтрацию отказов аутентификации, выявление аномально высокой интенсивности чтения в заданном временном окне [6]. Метрики производительности самого OpenBao экспортируются в Prometheus через встроенный эндпоинт /metrics и дополняют картину наблюдаемости количественными показателями [7].

На основе описанной архитектуры разработан дашборд Grafana, объединяющий панели нескольких типов: временные ряды запросов к хранилищу с разбивкой по операциям (read, write, delete), таблицы наиболее активных субъектов доступа, а также панель журнала с фильтрацией по критерию ошибок. Совокупность представленных визуализаций обеспечивает оперативный контроль за состоянием системы управления секретами без необходимости обращения к сырым лог-файлам.

Практическая апробация решения проводилась на производственном кластере k3s. Интеграция не потребовала модификации кода прикладных сервисов и реализована исключительно средствами конфигурации инфраструктурных компонентов. Задержка доставки событий от момента операции до отображения в Grafana составила менее десяти секунд, что является приемлемым показателем для задач оперативного мониторинга безопасности.

Предложенный подход демонстрирует возможность построения наблюдаемости за операциями с секретами на основе компонентов, уже присутствующих в типовом мониторинг-стеке Kubernetes-кластера. Дальнейшим направлением развития является корреляция аудит-событий с распределёнными трейсами посредством Grafana Tempo, что позволит связать операции с секретами с конкретными трассами запросов в микросервисной архитектуре и тем самым расширить возможности детектирования аномалий.

ЛИТЕРАТУРА

1. Kubernetes Project. Good Practices for Kubernetes Secrets [Электронный ресурс]. – URL: <https://kubernetes.io/docs/concepts/security/secrets-good-practices/> (дата обращения: 12.03.2025).
2. Berkovitch S. Overcoming Kubernetes Audit Log Challenges in Detection // Wiz Blog. – 2024 [Электронный ресурс]. – URL: <https://www.wiz.io/blog/overcoming-kubernetes-audit-log-challenges> (дата обращения: 12.03.2025).
3. OpenBao Project. Run OpenBao on Kubernetes [Электронный ресурс]. – URL: <https://openbao.org/docs/platform/k8s/helm/run/> (дата обращения: 12.03.2025).
4. OpenBao Project. Audit Devices [Электронный ресурс]. – URL: <https://openbao.org/docs/audit/> (дата обращения: 12.03.2025).
5. Grafana Labs. Promtail Agent | Grafana Loki Documentation [Электронный ресурс]. – URL: <https://grafana.com/docs/loki/latest/send-data/promtail/> (дата обращения: 12.03.2025).
6. Grafana Labs. LogQL Reference | Grafana Loki Documentation [Электронный ресурс]. – URL: https://grafana.com/docs/loki/latest/query/query_reference/ (дата обращения: 12.03.2025).
7. Walsh J. Kubernetes Security: Best Practices for Kubernetes Secrets Management // CNCF Blog. – 2023 [Электронный ресурс]. – URL: <https://www.cncf.io/blog/2023/09/28/kubernetes-security-best-practices-for-kubernetes-secrets-management/> (дата обращения: 12.03.2025).

СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СУБД POSTGRESQL И CLICKHOUSE В КАЧЕСТВЕ ЦЕЛЕВЫХ ХРАНИЛИЩ В ETL-ПАЙПЛАЙНЕ ДЛЯ ОБРАБОТКИ ВРЕМЕННЫХ РЯДОВ

В докладе представлено сравнение производительности СУБД PostgreSQL и ClickHouse при обработке временных рядов в рамках ETL-процесса. Проведен комплексный анализ эффективности обеих систем на различных объемах данных.

Традиционно именно реляционные СУБД, например PostgreSQL, доминировали в сфере управления данными. Но количество данных в последние годы растет с огромной скоростью и не всегда возможностей реляционных систем остается достаточно. Для обработки временных рядов существует ряд требований к системам обработки и хранения данных [6]. Одним из представителей систем, которые справляются с такой задачей, является ClickHouse [1], разработанный компанией Yandex. Он имеет колоночную архитектуру и эффективные алгоритмы сжатия, что значительно повышает его производительность для аналитических запросов, особенно при работе с большим объемом временных рядов.

Несмотря на большое количество уже существующих сравнительных исследований производительности ClickHouse и PostgreSQL [4], лишь немногие систематически сравнивают эти решения в рамках полного ETL-пайплайна, имитирующего реальные условия эксплуатации. Также они не рассматривают полный жизненный цикл данных от генерации и загрузки до выполнения аналитических запросов разной направленности.

Основной целью исследования является комплексное сравнение производительности Postgres и ClickHouse в рамках автоматизированного ETL-пайплайна, также разработать практические рекомендации по выбору оптимальной системы для различных сценариев использования и определить сценарии эффективного применения каждой из СУБД [2].

Задачи исследования:

1. Разработать сценарий комплексного тестирования производительности СУБД в контексте ETL-пайплайна, включая генерацию реалистичных тестовых данных и определение запросов разных типов.
2. Спроектировать и реализовать ETL-пайплайн на базе Apache Airflow, обеспечивающий автоматизированное тестирование обеих систем управления базами данных на различных объемах данных, обеспечить генерацию тестовых данных и сбор результатов.
3. Провести серию экспериментов, включающих загрузку данных, выполнение аналитических запросов на различных объемах информации (от 10 тысяч до 1 миллионов записей), измерить показатели производительности
4. Проанализировать результаты, выявить закономерности в производительности СУБД, определить факторы, влияющие на эффективность каждой системы, представить информацию в доступном виде и сформировать практические рекомендации.

Для проведения сравнительного анализа производительности PostgreSQL и ClickHouse была разработана экспериментальная установка, имитирующая реальный ETL-пайплайн обработки данных временных рядов [2,3,5].

Инфраструктура развернута с использованием Docker и Docker Compose, что обеспечивает воспроизводимость экспериментов и изоляцию тестовой среды. Каждый компонент функционирует в отдельном контейнере. Взаимодействие компонентов организовано через Docker-сеть с использованием механизма DNS-разрешения имен контейнеров.

Для сравнения был разработан генератор реалистичных данных временных рядов, который имитирует логи действий пользователей в веб-приложении.

События генерируются с равным интервалом в 1 секунду в течение 90-дневного периода. Используется степенное распределение, где 20% пользователей генерируют 80% событий, что соответствует принципу распределения в реальных системах, для более точной репрезентации в исследовании.

Сессии пользователей имеют длительность от 1 до 30 событий, с вероятностью завершения сессии в 5% после каждого события, что имитирует реальное поведение пользователя. Также для более точного отображения некоторые типы событий коррелируют с пользователями. То есть некоторые пользователи чаще выполняют определенные типы действий.

Объемы данных выбраны следующие: 10 000 записей - для базового теста, проверки корректности работы пайплайна, 50 000 записей - имитирует суточную нагрузку среднего приложения, 100 000 записей - значительный объем, тестирование масштабируемости, 500 000 и 1 000 000 записей – нагрузочное тестирование

Каждый датасет сохраняется в формате CSV с UTF-8 вместе с заголовком с именами столбцов для обеспечения совместимости с обеими СУБД.

Измерения проводились с использованием монотонного системного времени с точностью до микросекунд. Для каждого запроса выполнялась серия из 5 запусков. Первая итерация отбрасывается, чтобы кэширование не повлияло на измерение, а оставшиеся итерации использовались для расчета метрик.

Для каждого измерения вычислялись среднее арифметическое время выполнения и стандартное отклонение.

Ускорение (Speedup) же вычислялось по формуле:

$$\text{Speedup} = T_{\text{PostgreSQL}} / T_{\text{ClickHouse}} \quad (1),$$

где $T_{\text{PostgreSQL}}$ и $T_{\text{ClickHouse}}$ – среднее время выполнения запроса в СУБД.

По результатам измерения видна нелинейная зависимость: при малых объемах данных (до 100K записей) PostgreSQL демонстрирует лучшую производительность, однако при количестве записей >500K записей ClickHouse имеет значительное преимущество, обеспечивая на массиве в 1M записей общее ускорение в 4,83 раза. Анализ по типам запросов подтвердил архитектурные различия СУБД. ClickHouse быстрее в несколько раз выполняет сложные аналитические запросы (агрегации, временные паттерны), тогда как PostgreSQL на порядки эффективнее при точечных индексных выборках. ClickHouse также показал существенно более высокую скорость загрузки данных (до 18,46 раза) и в 2,1 раза эффективнее сжимает данные на диске благодаря колоночному хранению. На основе этих выводов были сформулированы практические рекомендации: для меньших систем размером менее 100K записей стоит использовать PostgreSQL, тогда как для аналитических нагрузок выше 500K записей оптимален ClickHouse.

ЛИТЕРАТУРА

1. ClickHouse Docs [Электронный ресурс]. — URL: <https://clickhouse.com/docs/ru?ysclid=mkdx7tfqih536621673> (дата обращения: 08.01.2026).
2. Intorf, D., Storey, D., and van Doorn, K. 2024. Apache Airflow Best Practices (дата обращения: 03.01.2026).
3. Продвинутое руководство ClickHouse [Электронный ресурс]. — URL: <https://clickhouse.com/docs/ru/tutorial?ysclid=mkdxb5s6c9321276769> (дата обращения: 08.01.2026).
4. PostgreSQL: Documentation: 18: Chapter 14. Performance Tips [Электронный ресурс]. — URL: <https://www.postgresql.org/docs/current/performance-tips.html> (дата обращения: 08.01.2026).
5. How-to Guide for PostgresOperator [Электронный ресурс]. — URL: https://airflow.apache.org/docs/apache-airflow-providers-postgres/2.2.0/operators/postgres_operator_howto_guide.html (дата обращения: 06.01.2026).
6. Shalina O.V., Andrianova E.E. «The study of combining sql and nosql databases in a heterogeneous system for the development of a project management database» THEORETICAL & APPLIED SCIENCE/ №5 2024, p 14-17

РАЗРАБОТКА СИСТЕМЫ ВЕКТОРИЗАЦИИ И СЕМАНТИЧЕСКОГО ПОИСКА С SQL-ИНТЕРФЕЙСОМ И МАСШТАБИРОВАНИЕМ НА НЕСКОЛЬКО ГРАФИЧЕСКИХ ПРОЦЕССОРОВ

Современные организации накапливают значительные объёмы текстовых данных: техническую документацию, базы знаний, исходный код и переписку. С развитием больших языковых моделей (LLM) возросла потребность в быстром извлечении семантически релевантной информации из корпусов документов для формирования контекста запроса - подход, известный как Retrieval-Augmented Generation [1]. Традиционные системы полнотекстового поиска не способны учитывать смысловую близость запроса и документа. Семантический поиск решает эту проблему путём преобразования текста в числовые векторы фиксированной размерности и нахождения ближайших соседей в векторном пространстве [2]. Вычислительная сложность поиска по миллионам векторов требует применения графических процессоров (GPU), а типичный конвейер включает три отдельных компонента: внешний сервис генерации векторов, хранилище и поисковый движок, что усложняет развёртывание и сопровождение системы.

Анализ существующих решений показывает, что ни одна система не объединяет пять ключевых характеристик: SQL-интерфейс, GPU-ускорение поиска, поддержку параллельных исчислений на нескольких GPU (multi-GPU), встроенную векторизацию и точный поиск (exact kNN). SQL-системы GaussDB-Vector [3] и SingleStore-V [4] не используют GPU и не генерируют вектора. FAISS [5] поддерживает multi-GPU, но не имеет SQL-интерфейса. BANG [6] и FusionANNs [7] применяют приближённые алгоритмы с потерей 5-10% результатов.

Целью работы является разработка интегрированной системы семантического поиска, объединяющей генерацию векторных представлений, хранение и GPU-ускоренный поиск в рамках единого SQL-интерфейса с масштабированием на несколько графических процессоров. Для достижения поставленной цели решены следующие задачи:

1. Анализ существующих решений в области векторного поиска с GPU-ускорением.
2. Разработка архитектуры системы, объединяющей векторизацию, хранение и поиск в рамках SQL-интерфейса.
3. Реализация CUDA-ядер для вычисления косинусного сходства и IVF-индекса.
4. Обеспечение масштабирования вычислений на несколько графических процессоров.
5. Проведение экспериментальной оценки на вычислительном узле с 8 GPU.

Разработанная система позволяет использовать общепринятые SQL-запросы, при исполнении которых ядро системы автоматически преобразует текстовые значения в векторное представление при помощи встроенной нейросетевой модели и осуществляет поиск ближайших соседей, производя вычисления на графических процессорах. Такая архитектура устраняет необходимость во внешних сервисах векторизации и упрощает интеграцию семантического поиска в существующие системы обработки данных. Генерация векторных представлений распараллеливается по нескольким графическим процессорам посредством секционирования входных данных.

При объёме свыше 50 000 векторов система автоматически переключается на IVF-индекс (Inverted File Index), реализованный с нуля на CUDA без использования сторонних библиотек. Индекс группирует векторы в кластеры методом K-means и при поиске просматривает только ближайшие кластеры, сохраняя точность 100% за счёт отсутствия квантизации. В multi-GPU режиме данные и индекс секционируются между графическими процессорами.

Результаты работы получены с использованием вычислительных ресурсов суперкомпьютерного центра Санкт-Петербургского политехнического университета Петра Великого (www.spbstu.ru).

Экспериментальная оценка проводилась на вычислительном узле, оснащённом 8 GPU NVIDIA Tesla V100 (суммарно 256 ГБ видеопамяти), на наборе из 10 млн векторов размерности 128. IVF-индекс обеспечивает латентность поиска 0,59 мс на одном GPU, что в 19,6 раза быстрее полного перебора (11,55 мс). Масштабирование на 8 GPU снижает латентность полного перебора до 2,19 мс (ускорение в 5,3 раза, 456 запросов в секунду). IVF-поиск на 2 GPU достигает латентности 0,44 мс при пропускной способности 2 259 запросов в секунду.

Полученные результаты демонстрируют высокие показатели производительности, а использование общепринятого SQL позволяет значительно снизить затраты на освоение, внедрение и использование системы в промышленной среде.

ЛИТЕРАТУРА

1. Gao Y. [et al.]. Retrieval-Augmented Generation for Large Language Models: A Survey. [Электронный ресурс] Режим доступа: <https://arxiv.org/abs/2312.10997> (дата обращения: 10.03.2026).
2. Mikolov T. [et al.]. Efficient Estimation of Word Representations in Vector Space. [Электронный ресурс] Режим доступа: <https://arxiv.org/abs/1301.3781> (дата обращения: 10.03.2026).
3. Sun Y. [et al.]. GaussDB-Vector: A Large-Scale Persistent Real-Time Vector Database for LLM Applications // Proceedings of the VLDB Endowment. – 2025. – Vol. 18, No. 12. – P. 4951–4963.
4. Chen C. [et al.]. SingleStore-V: An Integrated Vector Database System in SingleStore // Proceedings of the VLDB Endowment. – 2024. – Vol. 17, No. 12. – P. 3772–3785.
5. Douze M. [et al.]. The FAISS Library. [Электронный ресурс] Режим доступа: <https://arxiv.org/abs/2401.08281> (дата обращения: 10.03.2026).
6. Jang J. [et al.]. BANG: Billion-Scale Approximate Nearest Neighbor Search using a Single GPU. [Электронный ресурс] Режим доступа: <https://arxiv.org/abs/2401.11324> (дата обращения: 10.03.2026).
7. Tian B. [et al.]. FusionANNS: An Efficient CPU/GPU Cooperative Processing Architecture for Billion-scale Approximate Nearest Neighbor Search // USENIX FAST. – 2025.

УДК 004.4

П.И. Емшанова (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ NO-CODE ВЕБ-СЕРВИСА ДЛЯ ВИЗУАЛЬНОГО ПРОЕКТИРОВАНИЯ ETL-ПРОЦЕССОВ

Современные организации сталкиваются с экспоненциальным ростом объёмов разнородных данных, требующих предварительной очистки, трансформации и приведения к единому формату перед загрузкой в аналитические системы или хранилища данных. Процесс подготовки этих данных к анализу, известный как ETL (Extract, Transform, Load), является критически важным, но часто трудоёмким этапом. [1,2]

Многие традиционные ETL-инструменты ориентированы на технических специалистов и требуют знания языков программирования (SQL, Python и др.), что затрудняет их использование аналитиками и другими пользователями без навыков разработки [3]. Как следствие, в последние годы наблюдается рост интереса к no-code и low-code платформам [4,5], позволяющим создавать и настраивать сложные процессы с использованием визуальных интерфейсов, что делает их доступными для аналитиков и бизнес-пользователей [6].

Существующие no-code и low-code ETL-инструменты не полностью удовлетворяют потребности организаций на российском рынке. Зарубежные SaaS-решения предполагают обработку данных во внешней облачной инфраструктуре, что противоречит требованиям локализации данных и политике импортозамещения. Российские решения чаще всего представлены ETL-модулями в составе BI-платформ, где подготовка данных рассматривается как вспомогательный этап аналитической визуализации. В результате возможности повторного использования и самостоятельного применения ETL-процессов оказываются ограниченными.

Таким образом, отсутствие решения, сочетающего визуальный подход, локальное развертывание и независимость от BI-платформ, обосновывает актуальность разработки специализированного ETL-инструмента для российского рынка.

Целью данной работы является разработка клиентской части no-code веб-сервиса для визуального проектирования ETL-процессов.

Для достижения этой цели необходимо решение *задач*, перечисленных ниже.

1. Провести анализ существующих подходов к реализации ETL-инструментов и сформулировать требования к пользовательскому интерфейсу no-code решений.
2. Разработать архитектуру клиентского приложения, обеспечивающую визуальное представление ETL-процессов в виде графа операций.
3. Реализовать пользовательский интерфейс загрузки данных из различных источников (CSV, XLSX, базы данных), их обработки и визуального проектирования ETL-процессов.
4. Провести тестирование разработанной клиентской части на предмет соответствия требованиям и удобства использования.

Разрабатываемое приложение реализует клиент-серверную архитектуру. Пользовательский интерфейс ориентирован на визуальное проектирование ETL-процессов без использования программного кода.

Ключевым элементом интерфейса является визуальный редактор, в котором ETL-процесс представляется в виде ориентированного графа. Узлы графа соответствуют отдельным операциям обработки (фильтрация, агрегация, объединение), а направленные связи определяют последовательность их выполнения. Пользователь формирует конвейер обработки данных, перетаскивая элементы из палитры компонентов на рабочую область и настраивая параметры узлов в панели свойств. Пользователь может просмотреть результат выполнения выбранной операции и при необходимости скорректировать параметры узла до перехода к следующим шагам. Такая обратная связь позволяет пользователю анализировать результаты преобразований и корректировать параметры обработки данных.

Клиентская часть реализована с использованием фреймворка React и языка TypeScript. Визуальный редактор ETL-процессов построен на базе библиотеки React Flow. Для управления состоянием используется Zustand, для отображения табличных данных — TanStack Table. Взаимодействие с серверной частью реализовано с использованием Axios, а управление сетевыми запросами и кэширование данных — с помощью библиотеки TanStack Query.

В результате работы создан функциональный прототип клиентской части no-code ETL-инструмента, демонстрирующий возможность визуального проектирования воспроизводимых ETL-процессов.

ЛИТЕРАТУРА

1. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. — 3rd ed. — Wiley, 2013.
2. Simitsis A., Vassiliadis P. Extraction, Transformation, and Loading // Encyclopedia of Database Systems / ed. by L. Liu, M. T. Özsu. — New York: Springer, 2018. — DOI: 10.1007/978-1-4614-8265-9_158.
3. Binzer B., Elshan E., Furstenau D., Winkler T. J. W. Establishing a Low-Code/No-Code-Enabled Citizen Development Strategy // MIS Quarterly Executive. — 2024. — Vol. 23, No. 3. — P. 253–273. — DOI: 10.17705/2msqe.00097.
4. Никулина С. М., Антонова А. А., Иваев М. И. Особенности применения технологии low-code в корпоративных информационных системах // Индустриальная экономика. — 2023. — №3.
5. Sanchis R., García-Perales Ó., Fraile F., Poler R. Low-code as enabler of digital transformation in manufacturing industry // Applied Sciences. — 2020. — Vol. 10, No. 1. — P. 1–19. — DOI: 10.3390/app10010012.
6. Sufi F. Algorithms in Low-Code-No-Code for Research Applications: A Practical Review // Algorithms. — 2023. — Vol. 16, No. 2. — DOI: 10.3390/a16020108.

ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ НА ОСНОВЕ МЕТОДОВ
С ЭКСПОНЕНЦИАЛЬНОЙ РЕЛАКСАЦИЕЙ

Развитие архитектур глубокого обучения и усложнение решаемых ими задач вывели на передний план фундаментальную проблему параметрической оптимизации высокоразмерных невыпуклых функционалов. Чаще всего для этого используют методы первого порядка, например, стохастический градиентный спуск (SGD) или его улучшенные версии вроде Adam и RMSProp. Их способность к быстрой навигации в хаотичных, зашумленных регионах пространства весов с множеством седловых точек обусловила их повсеместное применение. Тем не менее, по мере приближения к глобальному или приемлемому локальному оптимуму топология ландшафта потерь претерпевает существенные изменения. В пространстве параметров появляются так называемые «овраги» — длинные и узкие впадины с резкими склонами [1]. В овражных структурах алгоритмы первого порядка сталкиваются с критическим замедлением сходимости. Градиент в таких областях направлен преимущественно перпендикулярно к дну оврага, что вызывает высокочастотные осцилляции (зигзагообразное движение) между его склонами при полном отсутствии или ничтожно малом продвижении вдоль самого дна в сторону оптимума. Использование адаптивного масштабирования шага в алгоритмах семейства Adam лишь частично смягчает эту проблему, так как подобные методы опираются на диагональную аппроксимацию матрицы кривизны и оказываются бессильными перед оврагами, чьи оси не параллельны осям координат параметрического пространства. Результатом становится эффект «заклинивания» или преждевременной остановки алгоритма [2].

В противовес этому, методы оптимизации второго порядка, использующие информацию о полной матрице Гессе, способны напрямую учитывать геометрию кривизны и осуществлять движение непосредственно к оптимуму, игнорируя иллюзию крутых склонов. Среди них особый интерес представляет метод экспоненциальной релаксации (ЭР), который обладает уникальными свойствами масштабирования градиента через матричную экспоненту. Метод ЭР использует информацию о вторых производных для расчета адаптивного шага вдоль каждого собственного направления пространства весов [3]. В отличие от метода Ньютона, ЭР-метод применяет матричную экспоненту, что гарантирует устойчивость спуска и выравнивание масштабов: алгоритм делает большие шаги вдоль «дна» оврага и микроскопические — по его «склонам». Основная проблема применения классического метода ЭР в машинном обучении — кубическая сложность $O(N^3)$ из-за необходимости спектрального разложения матрицы Гессе. В рамках практической реализации были проанализированы и запрограммированы три подхода:

- *Спектральное разложение* — математически точное вычисление спектра матрицы Гессе. Подходит только для задач малой размерности.
- *Рекурсивный метод* — замена ресурсоемкого вычисления экспоненты итеративным умножением плотных матриц по формуле удвоения шага [1].
- *Метод Ланцоша (Hessian-Free)* — современная оптимизация, реализованная в итоговой программе. Задача проецируется в подпространство Крылова малой размерности ($k \approx 10..20$). Данный подход полностью исключает формирование матрицы Гессе в памяти (используется быстрое вычисление произведения матрицы на вектор через автодифференцирование) [4][5], снижая сложность по памяти с квадратичной до $O(k \cdot N)$, а вычислительную — до линейной.

Тестирование алгоритмов проводилось на задачах с искусственно сгенерированной высокой корреляцией (число обусловленности $>10^4$), а также на реальных прикладных наборах данных: классификация рака груди (медицина), деконволюция сигналов (физика) и

прогнозирование гидродинамического сопротивления яхт. Во всех тестах оптимизированный метод ЭР успешно преодолевал седловые точки и овраги [6]. Для достижения целевого значения функции потерь методу ЭР на базе алгоритма Ланцоша потребовалось порядка 50 итераций, в то время как алгоритм Adam показал сходимость лишь после 3000 итераций, а на задаче деконволюции показал склонность к застреванию.

Программный комплекс реализован на языке Python с использованием библиотеки PyTorch для автоматического дифференцирования и NumPy для проведения матричных вычислений. Архитектура решения включает три модуля оптимизации, соответствующих различным подходам к вычислению шага экспоненциальной релаксации. Спектральный модуль базируется на точном разложении матрицы Гессе, рекурсивный модуль реализует итерационное удвоение шага с использованием только матричных умножений. Наиболее производительный модуль использует метод Ланцоша, который позволяет строить проекцию матрицы Гессе в подпространство Крылова через ее произведение на вектор (Hv-product). Это избавляет от необходимости явного формирования матрицы вторых производных и позволяет масштабировать метод для работы с современными нейросетевыми архитектурами.

Алгоритм экспоненциальной релаксации позволяет использовать математический аппарат методов второго порядка без катастрофических накладных расходов памяти. Данный алгоритм занимает важную нишу в задачах высокой точности, обратных физических задачах и при обучении нейросетей малого и среднего размера со сложной внутренней структурой данных, превосходя классические подходы по надежности сходимости.

ЛИТЕРАТУРА

1. Черноруцкий И. Г. Методы оптимизации в теории управления: Учебное пособие. — СПб.: Питер, 2004. — 256 с.
2. Ludwig O. The Condition Number as a Scale-Invariant Proxy for Information Encoding in Neural Units. — arXiv:2506.16289, 2025.
3. Черноруцкий И. Г. Градиентные методы с экспоненциальной функцией релаксации // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. — 2013. — № 5(181). — С. 11-18.
4. Martens J. Deep learning via Hessian-free optimization // Proceedings of ICML. — 2010. — P. 735-742.
5. Pearlmutter B. A. Fast exact multiplication by the Hessian // Neural Computation. — 1994. — Vol. 6, № 1. — P. 147-160.
6. Черноруцкий И. Г., Дробинцев П. Д. Градиентные методы для задач минимизации большой размерности // МКС и ТУ. — 2024.

УДК: 004.415.23

К.М. Зезина (4 курс бакалавриата),
М.В. Болсуновская, к.т.н., доцент

АУТЕНТИФИКАЦИЯ И АВТОРИЗАЦИЯ В РАСПРЕДЕЛЕННЫХ REALTIME-СИСТЕМАХ

Современные системы, использующие микросервисную архитектуру, могут сталкиваться с проблемами операций аутентификации и авторизации пользователей. Управление идентичностью в микросервисах не может обеспечить необходимый контроль идентификационных данных, по сравнению с монолитной системой, так как при монолитной системе пользователь получает доступ сразу ко всем ресурсам системы без дополнительных проверок, тогда как в случае микросервисной архитектуры, каждый микросервис независим и не знает никакой информации о пользователе [1]. Другая проблема – различность ID-схем между сервисами. Помимо того, что каждый микросервис имеет собственные ключи, каждому из них необходимо иметь сопоставляющую таблицу для обмена токенами с другими

микросервисами. Такой подход увеличивает затраты, как временные и человеческие ресурсы, в случае падения одного из микросервисов, так и денежные из-за возрастающей нагрузки на сервер и падения производительности [2].

Разрабатывая сложный продукт с микросервисной архитектурой необходимо предусмотреть решение всех вышепоставленных проблем. Таким образом, *целью* данной работы является проектирование и реализация интеграционного слоя между централизованным IAM-провайдером и realtime-мессенджером, обеспечивающего единое управление идентичностью пользователей в распределённой системе. В качестве IAM-провайдера был выбран сервис Keycloak [3], поддерживающий протоколы OAuth 2.0, OpenID Connect и SAML 2.0, что обеспечивает безопасную аутентификацию для современных приложений, не требует лицензионных платежей и легко интегрируется в современных корпоративных системах, а в качестве realtime-мессенджера – open-source платформа Tinode [4], которая обеспечивает высокопроизводительную масштабируемую инфраструктуру обмена сообщениями с поддержкой realtime-коммуникаций, push-уведомлений и обмена медиаконтентом.

Для достижения этой цели необходимо решение *задач*, перечисленных ниже.

1. Анализ проблемы несовместимости идентификационных пространств двух систем и обоснование необходимости промежуточного адаптационного слоя.
2. Изучение модели аутентификации Tinode и механизма делегирования аутентификации внешнему сервису через REST Authenticator.
3. Проектирование микросервиса restauth, после определения его интерфейсов, логики маппинга идентификаторов и схемы обработки ошибок.
4. Реализация микросервиса restauth, обеспечивающего трансляцию запросов Tinode в запросы к Keycloak и обратное преобразование ответов и ошибок.
5. Тестирование интеграционного слоя, проверяя корректность аутентификации, авторизации и обработки граничных случаев.

Основными проблемами несовместимости Keycloak и Tinode являются:

1. Совершенно разный формат идентификаторов пользователя. Keycloak идентифицирует пользователей через поле sub в JWT-токене. UUID генерируется Keycloak случайно при создании пользователя и является его внутренним ключом. Tinode, напротив, использует собственный формат UID – int64, закодированный в base64.
2. У Tinode есть механизм делегирования аутентификации – REST Authenticator [5]. Tinode отправляет POST-запросы с JSON-пейлоадом на внешний сервис и ожидает ответ в строго определённом формате. Keycloak же отвечает на запросы в формате JWT – это совершенно другой протокол, другая структура данных, другие эндпоинты.

Схема взаимодействия Keycloak и Tinode представлена на рисунке 1.

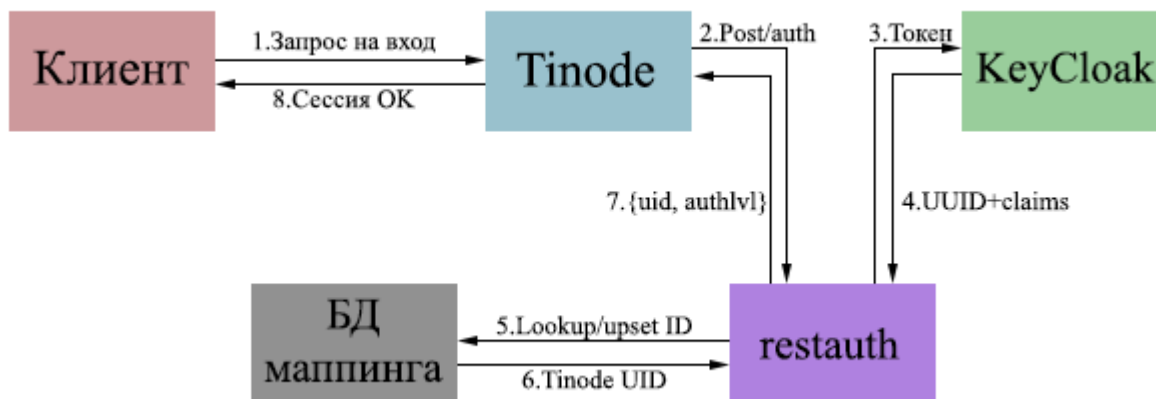


Рисунок 1 – Схема взаимодействия Keycloak и Tinode через сервис restauth

При отправке клиентом запроса на вход в систему обращение направляется к серверу Tinode. Поскольку Tinode не располагает встроенными средствами валидации токенов стороннего IAM-провайдера, он делегирует проверку подлинности внешнему сервису, направляя HTTP POST-запрос на микросервис restauth. Получив запрос, restauth извлекает переданный токен и обращается к Keycloak для его валидации. Keycloak, являясь централизованным источником истины об идентичности пользователей, верифицирует токен и возвращает соответствующие данные. Restauth обращается к собственной базе маппинга с целью установить соответствие между полученным Keycloak UUID и внутренним идентификатором Tinode. Если пользователь входит в систему впервые, restauth инициирует создание новой учётной записи в Tinode и сохраняет связь между двумя идентификаторами в базе. Получив Tinode UID, restauth формирует ответ в формате протокола REST Authenticator. Таким образом, вся несовместимость форматов и протоколов остаётся инкапсулированной внутри restauth и остаётся прозрачной как для Tinode, так и для Keycloak.

ЛИТЕРАТУРА

1. How To Control User Identity Within Microservices. [Электронный ресурс] Режим доступа: <https://nordicapis.com/how-to-control-user-identity-within-microservices/> (дата обращения: 06.03.2026)
2. How We Built a Centralized Identity Layer for Microservices. [Электронный ресурс] Режим доступа: <https://medium.com/freshworks-engineering-blog/centralized-identity-layer-bbda5518160b> (дата обращения: 06.03.2026)
3. Keycloak. Documentation. [Электронный ресурс] Режим доступа: <https://www.keycloak.org/documentation> (дата обращения: 14.02.2026)
4. Tinode. Developers' documentation. [Электронный ресурс] Режим доступа: <https://tinode.co/developers.html> (дата обращения: 25.11.2025)
5. REST or JSON-RPC authenticator. [Электронный ресурс] Режим доступа: <https://github.com/tinode/chat/blob/master/server/auth/rest/README.md#gen-generate-authentication-secret> (дата обращения: 06.03.2026)

УДК 004.65

П. Злобин (3 курс бакалавриата),
Е.Е. Андрианова, ст. преподаватель

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ ГЕНЕРАТИВНОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРОЦЕССАХ ПРОЕКТИРОВАНИЯ СХЕМ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Реляционные СУБД продолжают занимать лидирующие позиции на рынке, заполняя шесть из десяти мест популярных систем [1]. Традиционное проектирование схем отличается высокой трудоёмкостью и сильной зависимостью от уровня эксперта. Ошибки на этапе концептуального моделирования выявляются только в 23 % случаев до внедрения в продакшн [2]. Современные большие языковые модели демонстрируют качественный скачок в задачах генерации схем и SQL-запросов [3].

Цель работы – провести сравнительный анализ эффективности больших языковых моделей (LLM) и специалистов разной квалификации при проектировании схем реляционных баз данных по времени, качеству нормализации и типам ошибок.

В эксперименте приняли участие 14 специалистов различной квалификации: 5 студентов 3–4 курсов, 6 Junior DBA с опытом работы 1–3 года и 3 Senior DBA с опытом более 5 лет. Кроме того, для сравнения тестировались три современные большие языковые модели: GPT-5, Claude 4.5 Sonnet и DeepSeek-R1.

Каждый участник (как человеку, так и модели) должны спроектировать схемы трёх баз данных разной предметной области: интернет-магазин, логистическая система и образовательная платформа внутри СУБД PostgreSQL. Каждая база данных включала 10–11 сущностей. Задача

состояла в последовательном выполнении следующих этапов: выделение сущностей, определение атрибутов с указанием типов данных PostgreSQL, установление связей между таблицами, приведение схемы к третьей нормальной форме и генерация полного DDL-кода.

Качество спроектированных схем оценивалось по шести метрикам: время выполнения задания, степень соответствия третьей нормальной форме, количество избыточных атрибутов, полнота модели данных, правильность выбора типов данных PostgreSQL и корректность созданных индексов. Каждая схема подвергалась независимой экспертной оценке двумя рецензентами с последующим достижением консенсуса.

Время выполнения для специалистов-человеков фиксировалось с момента получения задания до завершения генерации полного DDL-кода с помощью таймера. Для больших языковых моделей время измерялось как полное время работы оператора с моделью: от составления начального промпта до получения и копирования финального DDL-кода. В это время входили все итерации уточнения запроса и исправления ошибок, выполненные оператором. Результаты проектирования по времени приведены в таблице 1.

Таблица 1 – Среднее время проектирования схем баз данных

Группа	Среднее время, мин	Станд. откл.	Интервал времени Min-Max, мин
Студенты	140	42	95-187
Junior DBA	95	34	65-123
Senior DBA	50	19	31-70
GPT-5	18	5	8-26
Claude 4.5	22	6	14-33
DeepSeek	25	6	19-33

GPT-5, как самый быстрый, оказался в 2,8 раза быстрее Senior DBA, в 5,3 раза быстрее Junior DBA и в 7,8 раза быстрее студентов.

GPT-5 показал 100% соответствие третьей нормальной форме, что превосходит результат Senior DBA (91,6 %) и сопоставимое качество по большинству метрик [5]. Полные результаты оценки качества спроектированных схем по всем шести метрикам (степень соответствия третьей нормальной форме, количество избыточных атрибутов, полнота модели данных, правильность выбора типов данных PostgreSQL и корректность созданных индексов) приведены в таблице 2.

Таблица 2 – Среднее время проектирования схем баз данных

Метрика	Студенты	Junior DBA	Senior DBA	GPT-5	Claude 4.5	DeepSeek-R1
Соответствие ЗНФ, %	44,4	75,0	91,6	100	66,6	66,6
Избыточные атрибуты (ср.)	8,3	3,7	0,4	1,2	1,8	3,1
Полнота модели, %	87,4	94,2	98,9	97,1	97,3	91,8
Правильность типов данных, %	68,5	84,2	97,3	93,8	91,5	86,7
Корректность индексов, %	45,0	72,0	94,0	88,0	85,0	74,0

Характерные ошибки LLM носят систематический и предсказуемый характер, что значительно упрощает их выявление и исправление по сравнению с хаотичными ошибками студентов и junior-специалистов [4].

Практические рекомендации основаны на полученных результатах и предполагают трёхуровневый подход к внедрению LLM. Для простых проектов, до 20 таблиц, целесообразно полностью доверять генерацию схемы LLM с последующей проверкой Junior DBA. В проектах средней сложности, от 20 до 50 таблиц, рекомендуется начинать с концептуальной модели, созданной Junior DBA, после чего LLM генерирует DDL-код, а Senior DBA выполняет финальную проверку. Для критически важных систем, более 50 таблиц, сохраняется полный контроль Senior DBA, при этом LLM используется только для автоматизации рутинных операций.

Таким образом, генеративный ИИ позволяет сократить время проектирования схем в 2,8–7,8 раза при сохранении высокого качества. Оптимальная стратегия – использование LLM как инструмента повышения производительности специалистов, а не их замены.

ЛИТЕРАТУРА

1. DB-Engines Ranking - popularity ranking of database management systems. [Электронный ресурс] Режим доступа: <https://db-engines.com/en/ranking> (дата обращения: 04.01.2025)
2. Batini C., Ceri S., Navathe S.B. Conceptual Database Design: An Entity-Relationship Approach. [Электронный ресурс] Режим доступа: <https://searchworks.stanford.edu/view/1964575> (Дата обращения: 04.01.2025)
3. Rajkumar N., Li R., Bahdanau D. Evaluating the Text-to-SQL Capabilities of Large Language Models. [Электронный ресурс] Режим доступа: <https://arxiv.org/pdf/2204.00498> (Дата обращения: 06.01.2025)
4. Ji Z., Lee N., Frieske R. et al. Survey of Hallucination in Natural Language Generation. [Электронный ресурс] Режим доступа: <https://arxiv.org/abs/2202.03629> (Дата обращения: 08.01.2025)
5. Zhang R., Khandelwal A., Ramesh A. et al. Can LLMs Write SQL? A Comparative Study of GPT-4 and Fine-Tuned Models. [Электронный ресурс]. - Режим доступа: <https://arxiv.org/pdf/2404.08727> (Дата обращения: 08.01.2025)

УДК 004.6:004.42

Т.Ю. Иглаков (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ NO-CODE ВЕБ-СЕРВИСА ДЛЯ АВТОМАТИЗАЦИИ ETL-ПРОЦЕССОВ

В последние годы объемы цифровой информации демонстрируют экспоненциальный рост. Способность организации оперативно собирать, обрабатывать и интерпретировать данные стала главным фактором конкурентоспособности. Однако значительная часть информационного потока представляет собой неструктурированные или слабоструктурированные данные (например, CSV-выгрузки или разрозненные Excel-файлы). Без предварительной подготовки такая информация не может быть корректно загружена в аналитические хранилища или BI-системы. Для решения данной задачи требуются специализированные инструменты класса ETL (Extract, Transform, Load) [6].

Анализ предметной области выявил ряд архитектурных и концептуальных недостатков у существующих систем автоматизации. Промышленные Open-Source решения (Apache NiFi, Airbyte) обладают высоким порогом входа и сложным инженерным интерфейсом, практически непригодным для бизнес-пользователей. Зарубежные SaaS-платформы (Fivetran, Skyvia) отличаются удобством, однако несут существенные риски информационной безопасности из-за необходимости передачи корпоративных данных в иностранные юрисдикции (“Vendor Lock-in”). Отечественные решения на текущем рынке (Yandex DataLens, Visiology) преимущественно ориентированы на визуализацию данных и ограничены в функционале выгрузки результатов сложных трансформаций. Таким образом, разработка независимого отечественного онлайн-сервиса класса No-Code для оперативной подготовки данных является высокоактуальной задачей, обеспечивающей технологический суверенитет и удобство масштабирования ИТ-продукта.

Целью работы является проектирование и реализация серверной части веб-сервиса, обеспечивающего баланс между декларативной простотой визуального управления потоками данных (Dataflows) и высокой производительностью безопасной обработки пользовательских датасетов в облачной среде.

Архитектурно-технологический стек разрабатываемой системы базируется на языке Python, который де-факто является стандартом в современной индустрии Data Engineering [3]. В качестве базового фреймворка для реализации серверной логики и предоставления REST API выбран Django (совместно с Django REST Framework). Данный выбор аргументируется наличием зрелой системы ORM, встроенной защитой от типовых веб-уязвимостей и готовой гибкой ролевой моделью для многопользовательского доступа. Основой вычислительного ядра для этапов извлечения и трансформации выступает библиотека pandas. Использование векторизованных операций pandas позволяет отказаться от низкопроизводительных циклических алгоритмов, обеспечивая быструю обработку загружаемых файлов объемом в сотни мегабайт без чрезмерной утилизации процессора.

Для достижения высокой производительности предложен гибридный подход к хранению данных, опирающийся на современные инженерные концепции проектирования высоконагруженных систем и платформ данных [1, 2]. В рамках предложенной архитектуры реляционная СУБД PostgreSQL выполняет исключительно роль управляющего реестра: в ней хранятся метаданные файлов, JSONB-схемы таблиц, конфигурации графов ETL-процессов и пользовательские профили. Подобное разделение ответственности гарантирует, что легкие SQL-запросы транзакционной логики не будут блокироваться операциями чтения тяжелых бинарных данных.

Физическое хранение промежуточных и итоговых пользовательских датасетов реализовано на базе масштабируемого файлового хранилища с использованием аналитического колоночного формата Apache Parquet. Выбор данного формата предоставляет критические технические преимущества по сравнению со стандартными строковыми форматами (CSV, XLSX). Во-первых, строгая типизация данных избавляет систему от затратного процесса распознавания типов на каждом шаге ETL-конвейера. Во-вторых, встроенные алгоритмы сжатия Parquet сокращают занимаемое дисковое пространство в среднем на 60–70%, что снижает требования к серверной инфраструктуре онлайн-сервиса. В-третьих, колоночная структура дает возможность считывать исключительно целевые столбцы, чтократно ускоряет операции ввода-вывода при фильтрации данных. Как подтверждают новейшие исследования, автоматизация процессов обработки на базе подобных оптимизированных архитектур радикально снижает общую трудоемкость интеграции данных и повышает их качество [4, 5].

Таким образом, разработанная архитектура обеспечивает строгую валидацию типов, надежное хранение и высокую скорость трансформации данных на всех этапах жизненного цикла. Практическая значимость работы заключается в формировании детального технического проекта, минимизирующего риски при последующей программной реализации и создающего фундаментальную базу для производительной платформы автоматизации процессов работы с данными как услуги (SaaS).

ЛИТЕРАТУРА

1. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. — СПб.: Питер, 2018. — 640 с.
2. Рейс Дж., Хаусли М. Основы инженерии данных: Планирование и создание надежных систем обработки данных. — СПб.: Питер, 2024. — 464 с.
3. Crickard P. Data Engineering with Python. — Birmingham: Packt Publishing, 2020. — 336 p.
4. Rongala S. Optimizing ETL Processes for High-Volume Data Warehousing in Financial Applications // Journal of Information Systems Engineering and Management. — 2024. — Vol. 10. — P. 700-706.
5. Rongala S., Modalavalasa G. Automating Extract, Transform, Load (ETL) Pipelines using Machine Learning Triggered Workflow Optimization // International Journal of Intelligent Systems and Applications in Engineering. — 2024. — Vol. 12. — No. 3. — P. 4427-4434.
6. Seenivasan D. ETL (Extract, Transform, Load) Best Practices // International Journal of Computer Trends and Technology. — 2023. — Vol. 71. — No 1. — P. 40-44.

РАЗРАБОТКА СИСТЕМЫ АУТЕНТИФИКАЦИИ, АВТОРИЗАЦИИ И АУДИТА
ДЛЯ REST API

Современные требования к безопасности веб-приложений диктуют необходимость создания надежных систем контроля доступа, обеспечивающих защиту данных и отслеживание действий пользователей. Монолитная архитектура систем аутентификации зачастую не позволяет гибко масштабировать решение и адаптировать его под меняющиеся требования безопасности. Таким образом, *целью* данной работы является снижение рисков несанкционированного доступа и повышение наблюдаемости системы за счёт создания распределённой системы аутентификации, авторизации и аудита (AAA) для REST API, интегрированной с API-шлюзом через специализированный протокол.

Для достижения этой цели необходимо решение задач, перечисленных ниже:

1. Проектирование системы декларативного описания правил авторизации и валидации входящих запросов на уровне API-шлюза.
2. Разработка сервиса аутентификации с поддержкой двухфакторной верификации через TOTP и управлением JWT-токенами [1].
3. Создание модуля централизованного логирования для аудита действий пользователей и системных событий с поддержкой уровней детализации.
4. Обеспечение механизма «горячей» смены конфигурации безопасности без остановки работы сервера.
5. Реализация протокола взаимодействия между шлюзом и сервисом аутентификации на базе gRPC [2].

Ключевым компонентом разрабатываемой системы является API [3], который принимает внешние HTTP-запросы и выполняет их первичную валидацию. Конфигурация правил безопасности описывается в YAML-файлах и содержит полную спецификацию доступа к эндпоинтам. Она включает описание обязательности JWT-токена в заголовках, параметры для аутентификации (логин/пароль, коды подтверждения) и структуры данных для аудита запросов. Встроенная система валидации конфигурации (более 37 правил в `config.go` gateway) предотвращает ошибки развёртывания, связанные с некорректным описанием политик безопасности. Например, проверяется уникальность названий параметров, наличие всех используемых типов и корректность вложенности структур.

Для логирования событий аудита был разработан кастомный логгер на языке Golang [4]. Он позволяет выводить сообщения с уровнями `INFO`, `ERROR` и `DEBUG`, причём `DEBUG`-режим дополнительно показывает место в коде (файл и строку), откуда была вызвана функция логирования. В шлюзе реализован middleware `CheckJwtMiddleware`, который логирует каждый входящий запрос: URL, заголовки, параметры пути и тело запроса (функция `generateLogRequestInfo`), что формирует основу для аудита. Взаимодействие между шлюзом и сервисом `service\_jwt` осуществляется через фреймворк `grpc\_unidirection`, реализующий асинхронную модель «запрос-ответ» поверх gRPC, что обеспечивает слабую связанность компонентов.

Сервис аутентификации `service\_jwt` реализует полный цикл управления доступом. При попытке входа пользователя (`LoginUser`) проверяются учетные данные в базе данных, после чего генерируется уникальный ключ запроса и 6-значный код подтверждения. Данные пользователя сохраняются в хранилище Riak [5], а код подтверждения помещается в оперативную память с таймером на ограниченное время жизни (`maxTtlConfirmCode`). При подтверждении кода (`LoginConfirm`) выполняется проверка соответствия, после чего генерируется пара JWT-токенов (Access и Refresh), а информация о пользователе с токенами сохраняется в раздел `ActiveUsers` Riak. Белый список сессий реализует политику «один

пользователь - один активный токен» (функция `CheckToken` в `service\_jwt.go`), обеспечивая контроль одновременных сессий и возможность принудительного выхода.

Разработанный подход был реализован в программных средствах `gateway` и `service\_jwt` на языке Golang. Ключевой особенностью является поддержка динамической смены конфигурации безопасности. По сигналам `SIGUSR1` и `SIGUSR2` сервисы могут перечитать конфигурационные файлы и применить изменения (например, время жизни токенов, уровень логирования аудита или ключ подписи JWT) без остановки работы и потери обрабатываемых запросов. Интеграция со встроенным счетчиком нагрузки позволяет отслеживать количество активных сессий и текущую нагрузку на сервис аутентификации в реальном времени, что критически важно для обнаружения аномальной активности.

ЛИТЕРАТУРА

1. JWT.IO. JSON Web Tokens Introduction [Электронный ресурс] Режим доступа: <https://jwt.io/introduction> (дата обращения: 26.02.2026).
2. Introduction to gRPC [Электронный ресурс] Режим доступа: <https://grpc.io/docs/what-is-grpc/introduction/> (дата обращения: 26.02.2026).
3. Что такое API? Простыми словами для Начинающих [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/964818/> (дата обращения: 26.02.2026).
4. Golang programming language. [Электронный ресурс] Режим доступа: <https://go.dev/> (дата обращения: 26.02.2026).
5. Riak [Электронный ресурс] Режим доступа: <https://riak.com> (дата обращения: 26.02.2026).

УДК 004.65

А.Д. Илюшкин (4 курс бакалавриата),
И.А. Селин, к.т.н., доцент

КОРПОРАТИВНОЕ ХРАНИЛИЩЕ ДАННЫХ И ПОСТАВКА ВЕРИФИЦИРОВАННЫХ ДАННЫХ С ПОМОЩЬЮ СИСТЕМ «1С:АНАЛИТИКА» И «ETL»

В современных реалиях цифровой экономики данные становятся главным ресурсом организации. Одной из главных проблем предприятий остается сильная фрагментация данных. Информация децентрализована по множеству учетных систем, локальных файлов и таблиц. Как следствие, возникает необходимость разделения оперативной и аналитической нагрузок и объединения доменов в рамках единой модели для формирования объективной отчетности [1].

В условиях курса на технологический суверенитет вопросы перехода на отечественное программное обеспечение встают особенно остро. Использование зарубежных платформ несет риски отключения лицензий, потери поддержки и утечки данных [2]. Переход на стек технологий «1С» является логичным решением, позволяющим построить полностью импортонезависимую систему.

Таким образом, целью данной работы является разработка корпоративного хранилища данных (КХД) и обеспечение автоматизированной поставки верифицированных данных в систему бизнес-аналитики с помощью ETL-процессов.

Ключевым этапом разработки стала выработка архитектурной стратегии. В ходе исследования был проведен сравнительный анализ двух фундаментальных подходов: методологии Б. Инмона (нормализованное хранилище) и Р. Кимбола (независимые витрины данных) [3]. Для нивелирования недостатков каждого из методов была спроектирована гибридная двухуровневая архитектура (рисунок 1).

Первый контур архитектуры — слой детальных данных (ODS/DDS), реализующий принципы Б. Инмона, выступает в роли «единой версии правды» и хранит историческую информацию. Второй контур — слой аналитических витрин (Data Marts), построенный по принципам Р. Кимбола, где денормализованная информация оптимизирована для SQL-

запросов BI-системы.

Технологический стек разработанной системы базируется на трехуровневой модели. В качестве ядра физического хранения используется СУБД PostgreSQL 15. Средой разработки и исполнения прикладной логики выбрана платформа «1С:Предприятие 8.3». Для реализации логической модели была создана специализированная конфигурация «с нуля», содержащая 15 целевых объектов метаданных.

За автоматизацию сбора информации отвечает ETL-процесс, включающий извлечение данных из источников, их преобразование (очистку, верификацию) и загрузку в хранилище [4]. Физическая реализация данного конвейера выполнена с использованием решения Modus ETL [5]. Внедрение автоматической верификации на этапе загрузки позволило исключить ошибки ввода и конфликты данных между подразделениями.

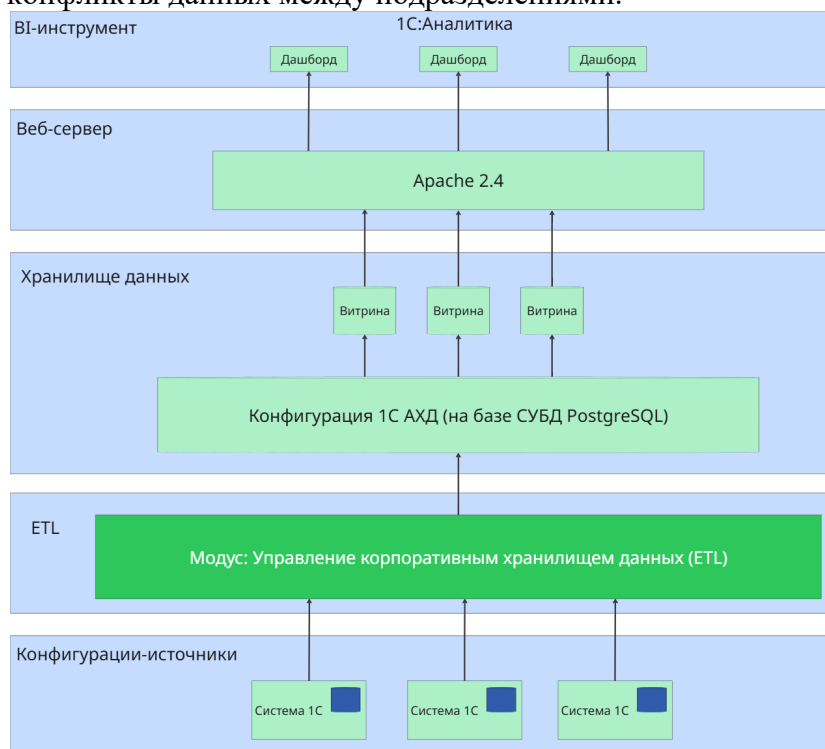


Рисунок 1 – Архитектура системы

В качестве инструмента визуализации используется сервер «1С:Аналитика» [6], взаимодействующий с кластером через веб-сервер Apache 2.4. Использование BI-системы такого класса позволяет обеспечить руководство компании наглядными дашбордами, повысить обоснованность управленческих решений и оперативно выявлять «узкие места» в бизнес-процессах [7].

В результате выполнения работы был успешно развернут отказоустойчивый интеграционный контур и реализован прототип КХД. Настроена бесшовная связка: учетная система – ETL-процесс – КХД – 1С:Аналитика. Разработано 3 интерактивных дашборда и 20 аналитических виджетов. Проведенное тестирование подтвердило стабильность работы: время формирования сложных аналитических отчетов сократилось с десятков секунд до 2-3 секунд. Разработанная система готова к промышленной эксплуатации и масштабированию.

ЛИТЕРАТУРА

1. Бронов С. А., Рогкустов С. В., Рожков С. Е., Максимов Е. В. Проблематика объединения доменов в рамках единой модели корпоративного хранилища данных // Science and technology research. – 2025. – С. 463-469.
2. Волчков Ф. О., Куприянов Д. Ю. Актуальные подходы к проектированию и разработке корпоративного хранилища данных // Ученые записки НИЯУ МИФИ. – 2024. – С. 69-81.
3. Шеховцова И. Р. Практика построения корпоративного хранилища данных в нефтегазовом секторе // ПРОНЕФТЬ. Профессионально о нефти. – 2025. – Т. 10, № 1. – С. 98-107.

4. Воробьев Н. В., Павлова Е. В. ETL процесс на базе платформы 1С:Предприятие // Научно-практический электронный журнал Аллея Науки. – 2019. – № 6(33). – С. 1-10.
5. Modus ETL. База знаний. [Электронный ресурс] Режим доступа: https://kb.modusbi.ru/web/docs_product/modus-etl (дата обращения: 22.02.2026).
6. 1С:Аналитика. [Электронный ресурс] Режим доступа: <https://analitica.ru/> (дата обращения: 22.02.2026).
7. Ломанова Е. В., Ломанова М. В. Опыт использования «1С:Аналитика» в рамках цифровизации управления складским хозяйством предприятия // Новые информационные технологии в образовании: Сборник научных трудов 24-й международной научно-практической конференции. – М.: ООО «1С-Паблишинг», 2024. – Ч. 1. – С. 46-48.

УДК 004.825

В.Е. Казаков (4 курс бакалавриата),
О.А. Юсупова, ст. преподаватель

ИНТЕЛЛЕКТУАЛЬНЫЙ ПОМОЩНИК С ИСПОЛЬЗОВАНИЕМ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ С ПОДХОДОМ RAG

В современных условиях нейронные сети находят широкое практическое применение: согласно статистическим данным, еженедельное количество активных пользователей ChatGPT превышает 800 млн человек [1]. Предшественниками современных решений выступали чат-боты, функционирующие на основе жестко заданных правил (rule-based системы), логика которых строилась на детерминированных цепочках блок-схем [2]. В настоящее время наблюдается переход к системам на базе больших языковых моделей (LLM) [3]. Принцип работы LLM основан на вероятностном прогнозировании последовательности токенов при формировании ответа на запрос. Однако данные модели подвержены феномену «галлюцинаций» – генерации синтаксически корректного и правдоподобного контента, не соответствующего действительности [3]. Для минимизации подобных рисков применяется технология генерации с дополнением данных (Retrieval-Augmented Generation, RAG) [4]. Данная методика предполагает предварительное извлечение информации из баз данных, на основе которой модель формирует итоговый верифицируемый ответ.

Цель работы заключается в разработке интеллектуального помощника для гостиничного сервиса с использованием больших языковых моделей (LLM) с применением метода RAG.

Для достижения этой цели было проведено исследование предметной области, предложена архитектура чат бота на основе RAG (рисунок 1) и реализовано программное средство с предложенной архитектурой на языке программирования Java с использованием фреймворка Spring.

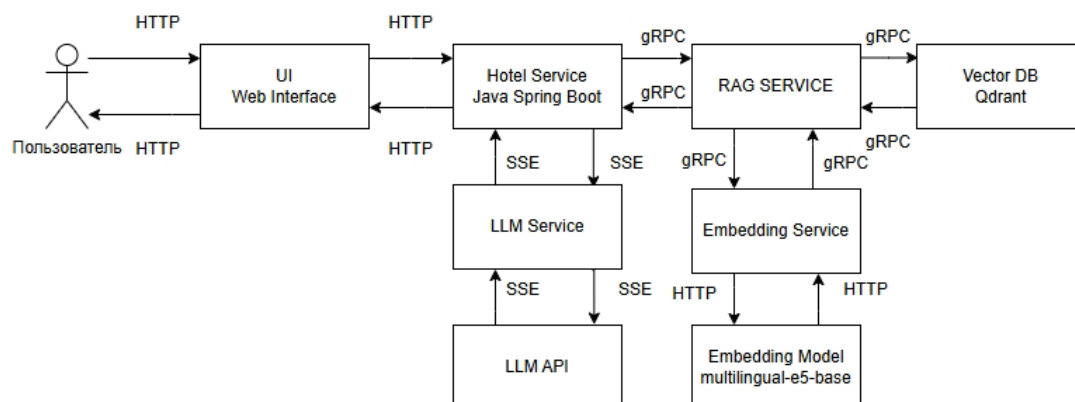


Рисунок 1 – Архитектура чат бота

Процедура формирования ответа включает следующие этапы:

- получение текстового запроса пользователя через графический интерфейс;
- преобразование входных данных в векторное представление (эмбединг) посредством специализированной модели (Embedding Model);
- выполнение семантического поиска наиболее релевантных фрагментов документов в векторном пространстве на основе метрики близости;
- интеграция извлеченного контекста в исходный промпт пользователя (формирование расширенного запроса);
- передача дополненного набора данных большой языковой модели (LLM) для генерации финального высказывания;
- вывод сформированного результата в интерфейс пользователя.

Трансформация исходных данных в векторную форму осуществляется с помощью моделей эмбедингов (Embedding Models) [5]. Статические модели, такие как Word2Vec [5], формируют фиксированные векторы для слов, что ограничивает точность при обработке полисемии и омонимии. Динамические модели (например, BERT [5]) генерируют контекстуально-зависимые представления, определяя значение токена на основе его окружения. Выбор конкретной модели обусловлен целевым языком и требуемой размерностью вектора: увеличение размерности повышает точность семантического представления, но сопряжено с ростом вычислительных затрат и объема используемой памяти.

Для оперирования полученными данными применяются векторные СУБД [5], ключевым отличием которых является поиск по семантическому сходству, а не по строгому совпадению. Оценка релевантности в таких системах базируется на расчете метрик расстояния: евклидова, или косинусного сходства [5].

Эффективное наполнение базы данных требует предварительной сегментации документов на чанки (chunks) [6]. Отсутствие декомпозиции приводит к потере локального контекста и размытию векторного представления. Существуют различные стратегии сегментации:

- Фиксированный размер (Character-based): разбиение по заданному количеству символов; недостатком является риск потери логической связности на границах блоков [6].
- Рекурсивное разбиение по разделителям: последовательное сегментирование на уровне абзацев, предложений и слов до достижения заданного объема [7].
- Семантическая сегментация: определение границ чанков на основе падения косинусного сходства между векторами соседних предложений [7].

Интеграция методики RAG в архитектуру разрабатываемого чат-бота на базе LLM позволяет минимизировать риск генерации галлюцинаций, обеспечивая высокую достоверность ответов на запросы клиентов. Применение данной технологии в рамках автоматизации информационного сервиса отеля способствует повышению качества пользовательского опыта и существенной оптимизации операционной нагрузки на персонал предприятия.

ЛИТЕРАТУРА

1. Sam Altman says ChatGPT has hit 800M weekly active users // TechCrunch URL: <https://techcrunch.com/2025/10/06/sam-altman-says-chatgpt-has-hit-800m-weekly-active-users/> (дата обращения: 08.03.2026).
2. Кутлов, Р. Р. Принципы функционирования чат-ботов и сфера их применения / Р. Р. Кутлов // Информационные технологии : Сборник тезисов XIII Конгресса молодых ученых, Санкт-Петербург, 09–11 апреля 2024 года. – Санкт-Петербург: федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет ИТМО", 2024. – С. 319-320. – EDN ESZCPR.
3. Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Chen Xu, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, Shuming Shi Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models // Computation and Language. – 2025.

4. Поправко, И. В. Механизмы адаптивного обучения интеллектуальных чат-ботов на базе больших языковых моделей в контексте бухгалтерской экспертизы / И. В. Поправко // Вестник евразийской науки. – 2025. – Т. 17, № 2. – EDN EFKUWI.
5. От слов к векторам: как эмбединги помогают моделям понимать нас // Epsilon Metrics URL: <https://blogs.epsilonmetrics.ru/kak-embeddingi-pomogayut-llm-ponyat-nas/> (дата обращения: 08.03.2026).
6. Sinchana Ramakanth Bhat, Max Rudat, Jannis Spiekermann, Nicolas Flores-Herr Rethinking Chunk Size For Long-Document Retrieval: A Multi-Dataset Analysis // Information Retrieval. - 2025
7. Evaluating Chunking Strategies for Retrieval // Chroma Technical Report URL: <https://research.trychroma.com/evaluating-chunking> (дата обращения: 08.03.2026).

УДК 004.415.2

В.И. Карасев (4 курс бакалавриата),
А.В. Самочадин, к.т.н., доцент

СРЕДСТВА ГЕНЕРАЦИЯ UML-ДОКУМЕНТАЦИИ ДЛЯ УНАСЛЕДОВАННОГО КОДА

Унаследованные программные системы (legacy-системы) широко применяются в различных отраслях и нередко эксплуатируются на протяжении десятилетий. Ключевой проблемой сопровождения таких систем является отсутствие или устаревание документации, что существенно затрудняет их понимание, модернизацию и передачу новым разработчикам [1]. Восстановление актуального представления о структуре и поведении системы вручную требует значительных временных и трудовых затрат.

Целью данной работы является разработка методики и прототипа инструмента для автоматизированной генерации UML-документации (включая диаграммы классов, последовательностей и вариантов использования) и формализованных текстовых сценариев для унаследованных систем на основе анализа их исходного кода.

Для достижения поставленной цели решаются следующие задачи: изучение существующих подходов к генерации UML-моделей из исходного кода; анализ специфики унаследованных систем и определение модели извлечения информации; разработка методики анализа кода для генерации текстовых описаний на естественном языке; создание алгоритма агрегации низкоуровневых описаний и потока вызовов в высокоуровневые сценарии использования; реализация программного прототипа.

Анализ существующих решений выявил принципиальный разрыв между инструментами промышленного статического анализа и задач семантически насыщенной документации. Традиционные средства (Doxugen, EMF/MoDisco, коммерческие CASE-инструменты) [2] ориентированы преимущественно на восстановление структурных представлений кода. Они позволяют строить диаграммы классов, однако не обеспечивают автоматического перехода к высокоуровневым поведенческим моделям и текстовым сценариям использования. Современные ИИ-ориентированные платформы (Swimm, Sourcegraph Cody, DocuWriter.ai) в значительной мере решают проблему контекстного понимания кода, однако не предоставляют комплексного автоматизированного решения для генерации полного набора UML-артефактов из legacy-кода [5].

Предлагаемый подход основан на сочетании методов статического анализа и технологий искусственного интеллекта. Платформа реализована на языке Python и включает следующие ключевые компоненты: модуль статического анализа исходного кода для построения абстрактных синтаксических деревьев (AST) и графов вызовов; подсистему семантического обогащения на основе языковых моделей (LLM) [3] для интерпретации идентификаторов и генерации кратких текстовых описаний классов и методов; алгоритм агрегации цепочек вызовов в логически завершённые сценарии использования [4]; модуль генерации UML-диаграмм с использованием текстового DSL PlantUML.

Архитектура системы построена по многоуровневому модульному принципу. Пользовательский интерфейс реализован в виде веб-приложения и обеспечивает загрузку проекта и визуальный просмотр результатов. Управляющий сервис координирует взаимодействие между компонентами. Модули статического анализа, семантической интерпретации и генерации документации функционируют независимо, что обеспечивает расширяемость платформы и возможность подключения новых анализаторов. Подсистема хранения данных поддерживает кэширование результатов и инкрементальный анализ при изменениях кодовой базы.

На основе предложенной методики реализован функциональный прототип в виде веб-приложения. Серверная часть построена на фреймворке FastAPI (Python) и реализует REST API с конечными точками для загрузки файлов и ZIP-архивов, запуска анализа и получения результатов. Парсер определяет язык по расширению файла, маршрутизирует его к соответствующему анализатору и формирует единую языко-независимую модель данных (ClassModel, MethodModel, CallEdge). На её основе генератор диаграмм формирует описания трёх типов: classDiagram, sequenceDiagram и flowchart.

Апробация прототипа на проектах с исходным кодом на языках Python, Go и SQL показала корректное извлечение классов, методов, полей и связей между компонентами. Для демонстрационного проекта, включающего три файла на разных языках, инструмент обнаружил 8 классов, 20 методов и 7 рёбер графа вызовов, а также сформировал полный набор из трёх UML-диаграмм. Корректность определения видимости методов проверена для всех поддерживаемых языков: для Python применяется соглашение об именовании (префикс подчёркивания), для Go — регистр первой буквы идентификатора, для Java/C#/Kotlin/TypeScript — модификаторы доступа [6].

ЛИТЕРАТУРА

1. Cosentino V. A model-based approach for extracting business rules out of legacy information systems // Information Systems. – 2015. – Vol. 53. – P. 117–133.
2. Knodel J., Popescu D. A comparison of static architecture compliance checking approaches // 6th IEEE/IFIP Working Conference on Software Architecture. – 2007. – P. 12.
3. Hu X., Xia X., Lo D., Li G. Deep code comment generation // IEEE/ACM 26th International Conference on Program Comprehension (ICPC). – 2018. – P. 200–210. – DOI 10.1145/3196321.3196334.
4. Черноруцкий И. Г. Методы оптимизации и принятия решений. – СПб.: Лань, 2021. – 384 с.
5. Kaur A., Gill N. S. Automatic documentation generation from source code: A systematic review // Journal of King Saud University – Computer and Information Sciences. – 2023. – Vol. 35. – P. 101–118.
6. Mäder P., Cleland-Huang J. A visual language for modeling traceability in source code // Software and Systems Modeling. – 2022. – Vol. 21. – P. 1305–1330. – DOI 10.1007/s10270-021-00941-6.

УДК 004.05

Д.А. Козяков (4 курс бакалавриата),
М.В. Болсуновская, к.т.н., доцент

ПОВЫШЕНИЕ НАДЁЖНОСТИ ИНФРАСТРУКТУРНОГО КОДА ПОСРЕДСТВОМ СТАТИЧЕСКОГО АНАЛИЗА YAML-ФАЙЛОВ HELM-ЧАРТОВ

Развёртывание современных программных систем в контейнеризированных средах на базе Kubernetes предполагает декларативное описание конфигурации инфраструктуры в виде YAML-файлов, формирующих инфраструктурный код (Infrastructure as Code, IaC). Helm [1] является одним из наиболее распространённых инструментов управления Kubernetes-приложениями, предоставляющим механизмы шаблонизации и параметризации манифестов. Вместе с тем гибкость формата YAML и отсутствие строгой типизации создают предпосылки для возникновения конфигурационных ошибок, проявляющихся на этапе развёртывания или

эксплуатации.

Исследования в области качества инфраструктурного кода показывают, что конфигурационные дефекты являются одной из ключевых причин отказов [2]. В работе [3] предложен подход к автоматизированному выявлению нарушений политик безопасности в Helm-чартах, что подтверждает актуальность систематической валидации конфигураций. Результаты эмпирического исследования дефектов IaC-программ [4] дополнительно свидетельствуют о распространённости типовых ошибок: неверных типах параметров, отсутствии обязательных полей, использовании устаревших API. Комплексный анализ подходов к верификации конфигурационных артефактов Kubernetes указывает на недостаточную проработанность вопросов каскадной статической валидации Helm-чартов с учётом пользовательских ограничений на допустимые значения и условных зависимостей параметров.

Целью данной работы является повышение надёжности инфраструктурного кода за счёт формирования и интеграции каскадного механизма статического анализа YAML-файлов Helm-чартов в процесс непрерывной интеграции.

Для достижения поставленной цели решаются следующие задачи:

1. Анализ существующих инструментов валидации конфигураций Kubernetes и Helm.
2. Систематизация типовых конфигурационных дефектов Helm-чартов.
3. Разработка формализованной схемы статической типизации параметров на основе JSON Schema [5].
4. Проектирование каскадного конвейера проверки и его интеграция в CI/CD-пайплайн.
5. Экспериментальная оценка эффективности предложенного подхода.

Архитектура предложенного решения представлена на рисунке 1. При фиксации изменений в системе контроля версий автоматически инициируется каскад этапов валидации: структурный анализ чарта (Helm lint), проверка параметров values.yaml на соответствие разработанной JSON Schema, рендеринг шаблонов (Helm template) с последующей валидацией сгенерированных манифестов. При обнаружении ошибки на любом этапе слияние изменений в основную ветвь блокируется.

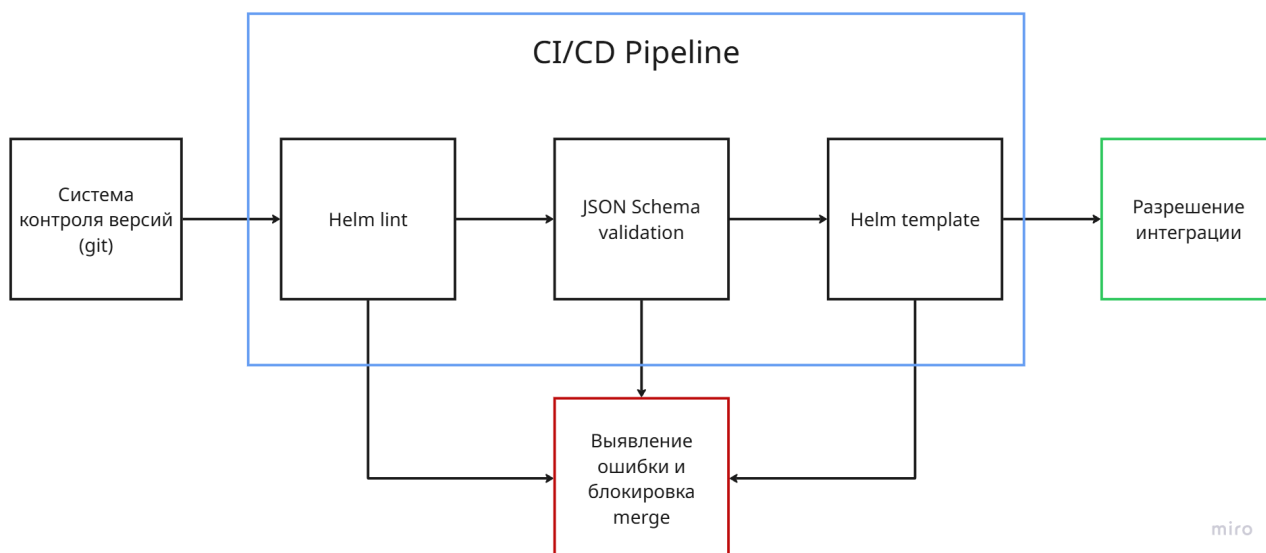


Рисунок 1 – Архитектура каскадного конвейера статического анализа Helm-чартов

В отличие от подходов, ориентированных на анализ отдельных аспектов конфигурации, таких как соблюдение политик безопасности [3], предложенное решение обеспечивает: формализацию пользовательских ограничений через расширенную JSON Schema (числовые диапазоны, перечисления допустимых значений, условные зависимости полей); каскадную трёхуровневую валидацию (параметры, шаблоны, манифесты); автоматическую блокировку некорректных изменений.

Тестирование выполнялось на Helm-чарте микросервисного приложения,

развёртываемого в Kubernetes-кластере. Механизм каскадной статической валидации был интегрирован в CI-пайплайн на базе GitHub Actions. Проведённая интеграция продемонстрировала возможность автоматического выявления конфигурационных ошибок до этапа развёртывания. Полученные результаты подтверждают применимость предложенного подхода в условиях промышленной разработки.

Предложенный подход является масштабируемым и может быть применён в проектах различной сложности, использующих Helm и Kubernetes, что позволяет рассматривать его как элемент контроля качества инфраструктурных решений.

ЛИТЕРАТУРА

1. Helm Documentation [Электронный ресурс]. – Режим доступа: <https://helm.sh/docs/> (дата обращения: 15.01.2026).
2. Zhang Y., Paul U., d'Amorim M., Rahman A. Configuration Defects in Kubernetes // IEEE Transactions on Software Engineering. – 2023. – DOI: (указать при наличии). – Режим доступа: <https://arxiv.org/abs/2512.05062> (дата обращения: 20.01.2026).
3. Minna F., Blaise A., Tuma K., Massacci F. Automated Analysis of Security Policy Violations in Helm Charts [Электронный ресурс]. – 2024. – Режим доступа: <https://www.researchgate.net/publication/397127915> (дата обращения: 20.01.2026).
4. Chiari M., De Pascalis M., Pradella M. Static Analysis of Infrastructure as Code: a Survey [Электронный ресурс]. – 2022. – Режим доступа: <https://arxiv.org/abs/2206.10344> (дата обращения: 25.01.2026).
5. JSON Schema Specification [Электронный ресурс]. – Режим доступа: <https://json-schema.org/specification> (дата обращения: 10.02.2026).

УДК 004.415

Д.С. Кондюков (4 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ ГЕНЕРАЦИИ МОДУЛЬНЫХ И API ТЕСТОВ НА JAVA С ИСПОЛЬЗОВАНИЕМ LLM

В последние годы стремительно развиваются технологии искусственного интеллекта, в частности Large Language Models (LLM), способные к генерации и анализу кода. Модели этого класса, обученные на корпусах исходного кода и технической документации, способны выполнять широкий спектр задач: от дополнения частей программы до генерации документации и модульных тестов [1, 2]. Результаты исследования [3] показали, что генерация модульных тестов с помощью LLM позволяет достигать высокого медианного покрытия по операторам (~ 70 %) и по ветвлениям (~ 52,8 %). Однако не решены проблемы точности и корректности сгенерированных тестов, их компилируемости и соответствия стилю проекта. Применение LLM в реальных проектах требует подбора контекста и адаптации под конкретные проекты.

Частично решается помощью подхода Retrieval-Augmented Generation (RAG). Остаются неизученными вопросы построения эффективной архитектуры RAG-системы для кода, интеграции с инструментами тестирования и оценки качества тестов. В [4] современные модели демонстрируют лишь умеренные результаты при генерации модульных тестов на уровне реальных проектов, а ручное исправление компиляционных ошибок повышает покрытие на 20–40 п.п. Анализ качества таких тестов [5], отразил, что до 68% из них содержат «тест-запахи», снижающие читаемость и сопровождаемость. Модели нуждаются в дополнительной поддержке, ориентированной на контекст и структуру кода.

Целью данной работы является разработка системы автоматической генерации модульных тестов на Java с применением LLM и технологии RAG.

Для достижения этой цели необходимо решение *задач*, перечисленных ниже.

1. Обзор существующих подходов и инструментов генерации тестов с применением LLM.
2. Анализ методов индексирования и поиска контекста в кодовых базах для RAG-систем.
3. Проектирование архитектуры системы генерации тестов с использованием векторного хранилища и LLM.
4. Реализация прототипа данной системы.
5. Тестирование на открытых датасетах и анализ качества сгенерированных тестов по метрикам покрытия, точности и времени выполнения.

Обзор источников показал, что большинство существующих решений для автоматической генерации тестов основываются на эвристических правилах, статическом анализе кода. Несмотря на свою полезность, эти методы ограничены в способности понимать контекст, семантику кода и генерировать осмысленные тестовые сценарии для нестандартных или плохо документированных компонентов системы.

Предлагается решить эти проблемы за счёт интеграции LLM и RAG с векторной базой знаний в процесс генерации.

Архитектура системы генерации модульных тестов на Java с использованием LLM представлена на рисунке 1. Тестируемый проект подключается к системе, которая отслеживает Pull Request (PR)/Merge Request (MR) изменения кода с помощью подключения Webhook. Полученные изменения анализируются системой. На их основе формируется оптимальный промпт с необходимой релевантной частью имеющегося кода из векторной и реляционной баз. Полученные тесты от LLM проходят проверку на корректность и запуск. В случае ошибок компиляции, происходит повторное обращение к LLM с указанием сути ошибки. Успешно сгенерированные тесты добавляются в репозиторий отдельным PR-запросом.

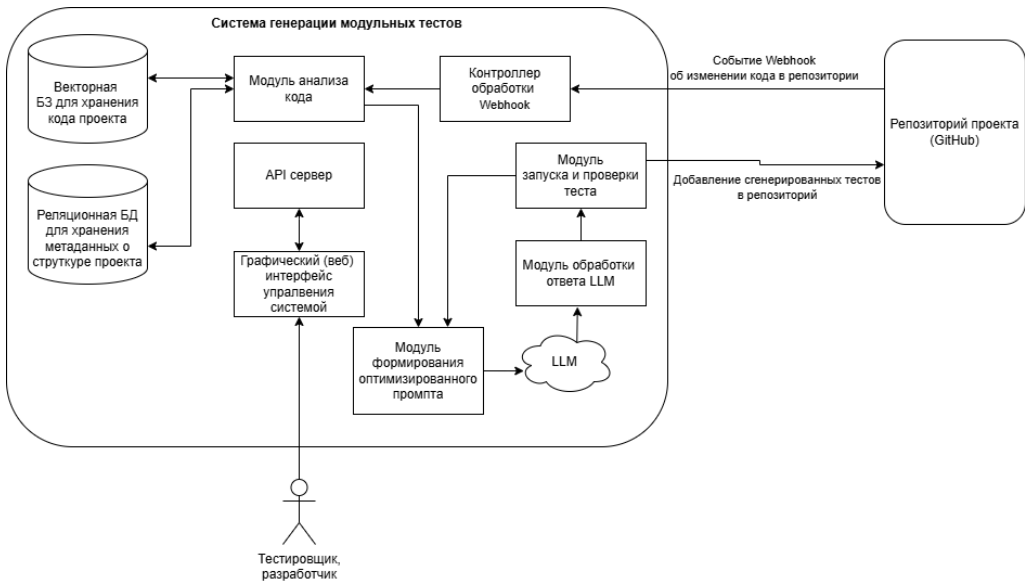


Рисунок 1 – Архитектура разрабатываемой системы

Проведены предварительные эксперименты генерации различными модификациями DeepSeek-r1 (DS) API-тестов для трех эндпоинтов проекта PetClinic. Результаты экспериментов отражены в таблице 1, где НК – некомпилируемые, Р – PASSED (успешно пройденные); F – FAILED (упавшие).

Таблица 1 – Количество сгенерированных методов в тестах проекта PetClinic

LLM / метод	deletePet()	processCreationForm()	processUpdateForm()
DS-coder	0	0	0
DS-7b	5НК	2НК	5НК
DS-8b	2НК	3НК	3НК
DS-14b	3F	5P	2НК
Codestral	2P	3P	4F

В ходе экспериментов менее объемные модели (DS-7b, DS-8b) генерировали некомпilierуемые тесты, в которых содержатся синтаксические ошибки, неверный вызов методов из внешних библиотек и зависимостей. Более крупные модели (DS-14b, Codestral) генерируют компилируемые тесты, но большинство из методов падают (FAILED). Связано это с непониманием LLM более сложной бизнес-логики в проекте, отсутствием примеров работы с Spring Context. В дальнейшем планируется продолжение работы для генерации модульных тестов.

ЛИТЕРАТУРА

1. Yadav D., Mondal S. Evaluating pre-trained Large Language Models on zero shot prompts for parallelization of source code // J. Syst. Softw. – 2025. – Vol. 230. – P. 112543. — DOI 10.1016/J.JSS.2025.112543
2. Roziere [et al.] Code Llama: Open Foundation Models for Code // CoRR. – 2023. — Vol. abs/2308.12950. – DOI 10.48550 / ARXIV.2308.12950.
3. Schafer M. [et al.] An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation // IEEE Trans. Software Eng. – 2024. – Vol. 50, No 1. – P.85-105. – DOI 10.1109/TSE.2023.3334955;
4. Wang Y. [et al.] ProjectTest: A Project-level LLM Unit Test Generation Benchmark and Impact of Error Fixing Mechanisms // CoRR. – 2025. – Vol. abs/2502.06556. – DOI 10.48550/ARXIV.2502.06556.
5. Ouedraogo W. C. [et al.] Test smells in LLM-Generated Unit Tests // CoRR. – 2024. – Vol. abs/2410.10628. – DOI 10.48550 / ARXIV.2410.10628.

УДК 004.658

А.С. Коновалов (2 курс магистратуры),
Т.В. Леонтьева, к.т.н., доцент

ВЕБ-СЕРВИС ОБРАБОТКИ БОЛЬШИХ ДАННЫХ НА ОСНОВЕ ДЕКЛАРАТИВНОГО ОПИСАНИЯ ТРАНСФОРМАЦИЙ

Подготовка данных остаётся одним из наиболее ресурсоёмких этапов аналитического процесса. Согласно отчёту State of Analytics Engineering 2025, качество данных является главным вызовом для команд аналитики, а организация и очистка наборов данных по-прежнему занимает наибольшую долю рабочего времени специалистов [1]. Мировой рынок инструментов подготовки данных оценивается в 3,48 млрд долларов в 2025 году с прогнозируемым среднегодовым ростом 11,3 % до 6,54 млрд долларов к 2031 году [2], что подчеркивает высокую востребованность подобных решений. При этом существующие инструменты обладают существенными ограничениями. Библиотека Pandas требует объёма оперативной памяти, кратно превышающего размер обрабатываемого файла [3]. Фреймворк Apache Spark решает проблему масштаба, однако предъявляет высокие требования к квалификации пользователя.

Целью данной работы является снижение порога входа в обработку больших наборов данных за счёт создания веб-сервиса, реализующего предложенную архитектуру обработки на основе декларативного описания трансформаций. Данная архитектура позволяет аналитикам без навыков программирования интерактивно очищать файлы объёмом до 100 ГБ через визуальный интерфейс.

Для достижения поставленной цели были решены следующие задачи: 1) проведён анализ существующих инструментов подготовки данных и выявлены их ограничения; 2) разработана архитектура Shadow Sampling, обеспечивающая мгновенную обратную связь при работе с файлами произвольного размера; 3) спроектирован и реализован механизм адаптивного выбора вычислительного движка; 4) реализована система декларативного описания трансформаций (контрактов) с поддержкой валидации и предварительного просмотра; 5) проведена экспериментальная оценка характеристик сервиса.

Ключевой концепцией сервиса является архитектура Shadow Sampling, основанная на разделении данных на два уровня. При загрузке файла система считывает первый фрагмент (до 10 МБ) и формирует превью данных, которое немедленно доступно пользователю. Все интерактивные трансформации применяются к превью в реальном времени, обеспечивая мгновенную визуальную обратную связь. Одновременно каждое действие пользователя записывается в структурированный JSON-документ, именуемый контрактом трансформаций. При экспорте контракт применяется к полному набору данных с автоматическим выбором оптимального движка обработки.

Компонент адаптивного выбора движка реализует паттерн Strategy для автоматического определения вычислительного бэкенда на основе размера файла и характера операций. Для файлов объёмом до 200 МБ используется библиотека Pandas, обеспечивающая минимальные накладные расходы. Файлы объёмом от 200 МБ до 2 ГБ обрабатываются в режиме последовательного чтения фрагментов (chunked processing), что позволяет ограничить потребление оперативной памяти константным значением. Для файлов свыше 2 ГБ задействуется Apache Spark, обеспечивающий распределённую обработку. Пользователь не участвует в выборе движка: решение принимается автоматически. Выбор пороговых значений обоснован результатами сравнительного анализа библиотек обработки таблиц, представленного в работах [3, 4].

Серверная часть реализована на языке Python с использованием асинхронного фреймворка FastAPI и ORM SQLAlchemy с драйвером asyncpg. Хранение метаданных осуществляется в СУБД PostgreSQL, управление схемой базы данных выполняется посредством системы миграций Alembic. Файловое хранилище организовано на базе S3-совместимого объектного хранилища MinIO, что обеспечивает совместимость с облачными платформами (AWS S3, Google Cloud Storage). Клиентская часть реализована на React с использованием TypeScript. Развёртывание всех компонентов выполняется посредством Docker Compose.

Система трансформаций поддерживает базовые и расширенные операции, включающие переименование и удаление колонок, заполнение пропущенных значений, фильтрацию, приведение типов данных, удаление дубликатов, замену значений и другие преобразования. Перед выполнением каждая операция проходит валидацию с отслеживанием виртуального состояния колонок, что позволяет обнаруживать ошибки до применения контракта к полному набору данных.

Экспериментальная оценка показала, что сервис обеспечивает формирование превью за время менее 5 секунд при загрузке файла объёмом 10 ГБ. Применение контракта из пяти трансформаций к файлу объёмом 200 МБ завершается в течение 20 секунд при использовании движка Pandas. Экспорт поддерживается в форматах CSV, Parquet, JSON и XLSX.

Таким образом, предложенная архитектура Shadow Sampling позволяет объединить мгновенную интерактивность при работе с превью и надёжную пакетную обработку полного набора данных. Разработанный веб-сервис демонстрирует практическую применимость данного подхода, предоставляя аналитикам инструмент обработки больших файлов без необходимости написания кода.

ЛИТЕРАТУРА

1. dbt Labs. 2025 State of Analytics Engineering Report. [Электронный ресурс] Режим доступа: <https://www.getdbt.com/resources/state-of-analytics-engineering-2025> (дата обращения: 15.03.2026).
2. Mordor Intelligence. Data Wrangling Market – Size, Share, Trends & Forecast, 2025–2030. [Электронный ресурс] Режим доступа: <https://www.mordorintelligence.com/industry-reports/data-wrangling-market> (дата обращения: 15.03.2026).
3. Broihier A., Baziotis S., Kang D., Mendis C. PandasBench: A Benchmark for the Pandas API // arXiv preprint. – 2025. – arXiv:2506.02345.
4. Baziotis S., Kang D., Mendis C. Dias: Dynamic Rewriting of Pandas Code // Proceedings of the ACM on Management of Data (PACMOD). – 2024. – Vol. 2. – No 1. – Article 58. – DOI 10.1145/3639313.

РАЗРАБОТКА И ОБЕСПЕЧЕНИЕ БЕЗОПАСНОГО ФУНКЦИОНИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЯ

На текущий момент времени обеспечение безопасности веб-приложения является одной из важнейших задач при разработке качественного программного продукта, требующей понимания слабых мест, которые могут быть использованы для атаки. Злоумышленник может воспользоваться ошибками, допущенными на этапе разработки, что может привести к утечке конфиденциальных данных клиентов, или к полному отказу работы самого веб-ресурса.

В данной работе рассматриваются наиболее распространенные и критически опасные уязвимости веб-приложений, к которым относятся SQL-инъекция, межсайтовый скриптинг и CSRF-атака [1].

SQL-инъекция является уязвимостью, которая возникает, когда веб-приложение некорректно обрабатывает ввод пользователя и вставляет его напрямую в SQL-запрос. В результате злоумышленник может вставлять в запрос дополнительные фрагменты SQL-кода, что, тем самым, приводит к изменению логики самих запросов или к выполнению произвольных команд в базе данных.

К межсайтовому скриптингу относятся уязвимости, при которых непроверенный или неправильно обработанный пользовательский ввод попадает в контент веб-страницы и интерпретируется браузером, как исполняемый код на языке JavaScript.

И, наконец, CSRF-атака — это уязвимость веб-приложений, благодаря которой злоумышленник может заставить жертву непреднамеренно выполнить действие от имени своей учётной записи на доверенном сайте.

Для применения эффективных способов защиты от данных типов уязвимостей было создано веб-приложение для некоторого интернет-магазина компьютерных комплектующих. При разработке приложения использовалась платформа ASP .NET, модель разработки ASP .NET Core MVC [2], система управления базами данных Microsoft SQL Server и фреймворк для разработки адаптивных веб-приложений Bootstrap. Также, стоит отметить, что при разработке веб-приложения использовались языки программирования C# и JavaScript, язык разметки HTML и язык вёрстки CSS.

Чтобы защитить веб-приложение от SQL-инъекции можно воспользоваться параметризацией запросов. Данную функцию поддерживает технология в .NET под названием LINQ. Пример параметризованного запроса с помощью данной технологии представлен на рисунке 1.

```
partsQuery = partsQuery.Where(p => p.part_name.Contains(searchTerm));
```

Рисунок 1 – Пример параметризованного запроса

Для проверки защиты на странице с каталогом комплектующих введем в строку поиска представленную на рисунке 2 SQL-команду, при помощи которой злоумышленник может добавить новую запись в базу данных.

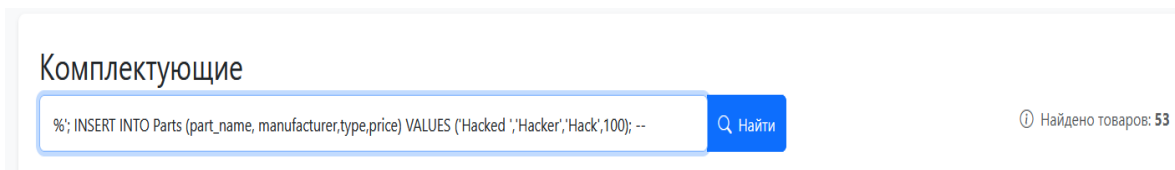


Рисунок 2 – Пример ввода SQL – инъекции

На рисунке 3 видно, что команда по созданию новой записи в базе данных не выполнялась. Вместо сборки SQL-строк вручную используются подготовленные выражения, где SQL-команды и данные передаются отдельно. То есть, данные, введенные пользователем, помещаются в параметры, которые система управления базами данных воспринимает как значения, а не как часть команды и даже если в данных присутствуют символы языка SQL, то они не будут выполняться.

Найдено товаров: 0 по запросу "'; INSERT INTO Parts (part_name, manufacturer,type,price) VALUES ('Hacked ','Hacker','Hack',100); --"

По вашему запросу ничего не найдено

Попробуйте изменить поисковый запрос

← Вернуться к каталогу

Рисунок 3 – Результат защиты от SQL – инъекции

Для обеспечения защиты от межсайтового скриптинга было использовано экранирование входных данных. Проверим данный способ защиты. На странице с каталогом комплектующих введем в строку поиска представленный на рисунке 4 скрипт, который вызывает диалоговое окно с сообщением о уязвимости.

Комплектующие

Найдено товаров: 53

Рисунок 4 – Пример ввода вредоносного скрипта

На рисунке 5 видно, что произошло экранирование входных данных. Все потенциально опасные специальные символы были закодированы в их HTML-сущности. Например, символы “<” и “>” были соответственно заменены на “<” и “>”.

Комплектующие

Найдено товаров: 0 по запросу "<script>alert("уязвимость")</script>"

По вашему запросу ничего не найдено

Попробуйте изменить поисковый запрос

← Вернуться к каталогу

Рисунок 5 – Результат защиты от межсайтового скриптинга

В данной работе защита от CSRF-атаки была обеспечена с помощью добавления специального атрибута, предусмотренного ASP .NET, под названием [ValidateAntiForgeryToken] [3].

Для проверки работоспособности данного способа защиты была создана специальная вредоносная страница, которая отправляет POST-запрос для создания новой записи в базе данных веб-приложения. Код данной страницы представлен на рисунке 6.

```
<!DOCTYPE html>
<html>
<head>
  <title>CSRF Тест</title>
</head>
<body>
  <div class="container">
    <h1>CSRF Тестовая атака</h1>
    <div class="info">
      <p>Данная страница сделана чтобы проверить защиту веб-приложения от CSRF-атак.</p>
    </div>
    <form id="csrfForm" action="https://example.com/Clients/Create" method="POST">
      <input type="hidden" name="last_name" value="Hacked" />
      <input type="hidden" name="first_name" value="Hacker" />
      <input type="hidden" name="phone" value="+79991234567" />
      <input type="hidden" name="email" value="hacker@mail.com" />
      <input type="hidden" name="pass" value="hacked123" />
    </form>
    <p>Нажмите кнопку для отправки вредоносного запроса:</p>
    <button onclick="document.getElementById('csrfForm').submit()">Отправить CSRF-атаку</button>
    <p class="warning">Если после нажатия клиент добавится в систему – приложение УЯЗВИМО!</p>
  </div>
</body>
</html>
```

Рисунок 6 – Код вредоносной страницы для проведения CSRF- атаки

На вредоносной странице при нажатии кнопки передачи данных формы на сервер приложение вернёт ошибку, так как был произведен запрос с неправильным токеном. Результат защиты можно увидеть на рисунке 7.

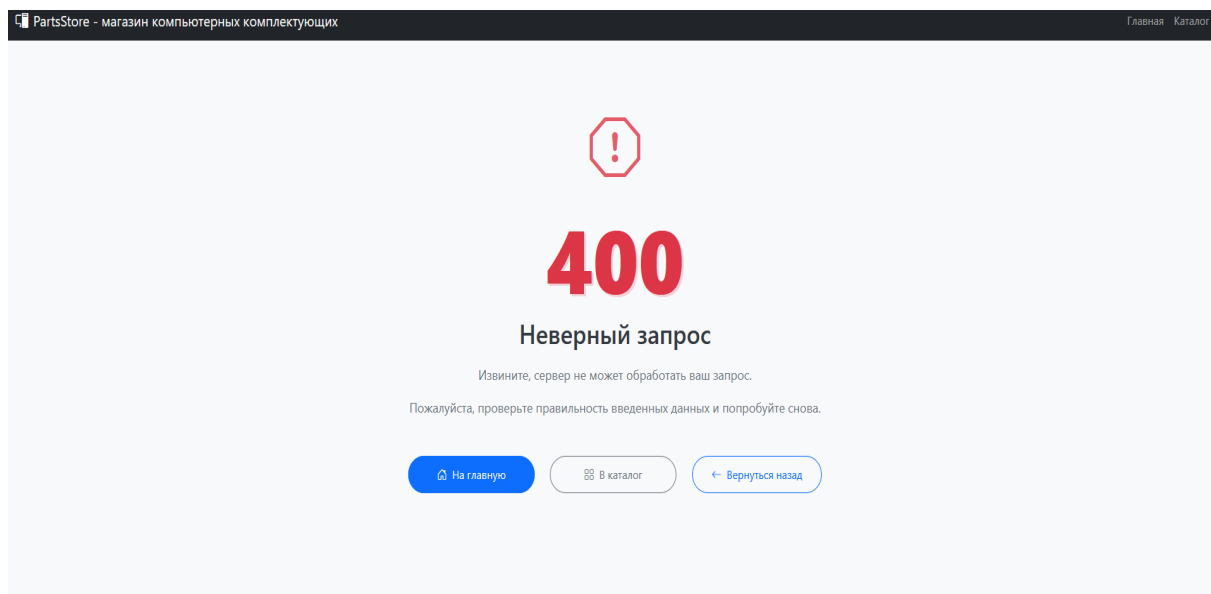


Рисунок 7 – Результат защиты от CSRF - атаки

ЛИТЕРАТУРА

1. OWASP Top 10:2025 [Электронный ресурс] : Список самых распространённых ошибок веб-приложений. – URL: <https://owasp.org/Top10/2025/> (дата обращения 03.03.2026). – Режим доступа: свободный.

2. Эндрю Лок ASP .NET Core в действии / пер. с англ. Д.А. Беликова. 3-е изд. – М.:ДМК Пресс, 2024. – 1046 с.: ил.
3. Документация по ASP.NET | Microsoft Learn [Электронный ресурс] : Документация по ASP.NET. – URL: <https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-9.0> (дата обращения 03.03.2026). – Режим доступа: свободный.

УДК 004.056

Н.В. Коноплёв (2 курс магистратуры),
А.Д. Ковалев, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ ОБНАРУЖЕНИЯ ГОЛОСОВОГО ФИШИНГА В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ НА ОСНОВЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

В последние годы наблюдается устойчивый рост количества атак с использованием социальной инженерии, в том числе голосового фишинга (vishing). Злоумышленники используют телефонные звонки для манипуляции пользователями с целью получения конфиденциальной информации. Развитие технологий синтеза и подмены голоса, а также использование автоматизированных сценариев общения существенно усложняют задачу выявления подобных атак традиционными средствами фильтрации.

Современные подходы к обнаружению голосового фишинга, как правило, основаны либо на анализе акустических характеристик речевого сигнала, либо на семантическом анализе распознанного текста. Так, в работе [1] показана эффективность использования ансамблевых методов машинного обучения для выявления фишинговых голосовых сценариев, а в исследованиях [2, 3] рассматриваются статистические и древовидные модели, применяемые для классификации звонков в режиме, близком к реальному времени. Однако изолированное применение указанных методов не позволяет достичь приемлемой точности в условиях реального времени, особенно при использовании злоумышленниками гибридных атак, сочетающих синтез речи и социальную инженерию.

Целью данной работы является разработка программного комплекса для обнаружения голосового фишинга в режиме реального времени на основе гибридного подхода, объединяющего методы акустического, семантического и поведенческого анализа с использованием алгоритмов машинного обучения.

В рамках предлагаемого решения входящий аудиопоток телефонного звонка (WebRTC/SIP/RTP) подвергается предварительной обработке, включающей шумоподавление, нормализацию уровня сигнала, автоматическую регулировку усиления и выделение речевых сегментов (Voice Activity Detection). Аудиосигнал разбивается на временные фрагменты длительностью от 200 до 1000 мс, что обеспечивает возможность потоковой обработки с минимальной задержкой.

Далее обработанные фрагменты параллельно передаются в три независимых аналитических модуля. Акустический анализатор опирается на подходы, описанные в [4], где рекуррентные нейронные сети используются для анализа временных и спектральных характеристик речи с целью выявления аномалий, характерных для синтезированного или модифицированного голоса. Акустический анализатор оценивает вероятность использования синтезированной или модифицированной речи на основе спектральных и просодических признаков. Семантический анализатор выполняет потоковое распознавание речи с последующей классификацией текста на наличие признаков фишинга, таких как ключевые фразы, шаблонные сценарии давления и запросы конфиденциальных данных. В качестве основы для данного модуля используются методы логистической регрессии и текстовой классификации, описанные в [2], адаптированные для потоковой обработки распознанной речи. Поведенческий анализатор исследует динамику диалога, включая паузы, темп речи,

латентность ответов и эмоциональную окраску, что позволяет выявлять неестественные паттерны общения.

Результаты работы аналитических модулей поступают в блок интеграции и принятия решений, в котором осуществляется взвешенное объединение скорингов с учетом заданных бизнес-правил. Подход к мультимодальной интеграции и агрегации признаков основан на принципах, изложенных в работе [5], где показана эффективность комбинирования акустических, текстовых и поведенческих признаков для повышения точности систем безопасности. При превышении порогового значения риска система инициирует защитные действия, включая уведомление оператора, временное удержание вызова или запуск процедуры дополнительной аутентификации. В противном случае вызов пропускается без вмешательства.

Архитектура программного комплекса представлена на рисунке 1 и ориентирована на модульность, масштабируемость и возможность интеграции в существующие телекоммуникационные инфраструктуры. Особое внимание уделяется вопросам конфиденциальности: в системе сохраняются только агрегированные метрики и минимально необходимые фрагменты данных, используемые для последующего обучения и повышения качества моделей.

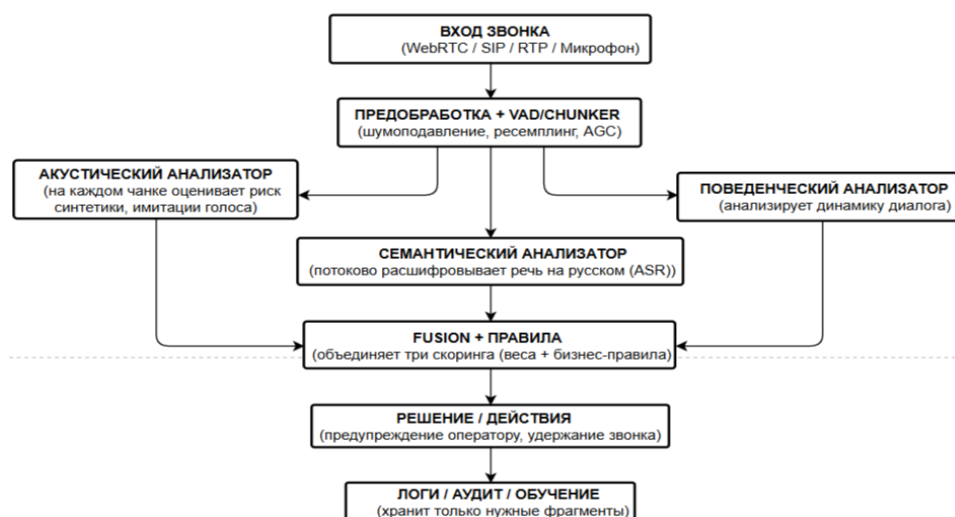


Рисунок 1 – Архитектура гибридной системы обнаружения голосового фишинга

Предложенный подход обеспечивает баланс между точностью обнаружения и требованиями к задержке обработки, что делает возможным его применение в реальных сценариях эксплуатации. Использование гибридной архитектуры позволяет повысить устойчивость системы к новым видам атак и адаптироваться к изменяющимся стратегиям злоумышленников.

ЛИТЕРАТУРА

1. Kumar N. et al. Detecting Vishing Attacks Using Gradient Boosted Decision Trees // Journal of Cybersecurity Research. 2021. Vol. 5. P. 123–139.
2. Li Y., Brown M. Logistic Regression for Voice Phishing Detection // International Conference on Security and Privacy. 2020. P. 45–52.
3. Zhang P. et al. Decision Trees for Real-time Vishing Detection // IEEE Transactions on Information Forensics and Security. 2022. Vol. 17. P. 3412–3425.
4. Ravi K. S. LSTM-based Models for Speech Analysis in Fraud Detection // Neural Computing and Applications. 2023. Vol. 35. P. 15267–15284.
5. Johnson L. et al. Multimodal Fusion for Security Analysis in Voice Communications // IEEE Access. 2024. Vol. 12. P. 7890–7904.

АВТОМАТИЧЕСКИЙ ВЫВОД ГРАФОВ ВЗАИМОДЕЙСТВИЙ В МИКРОСЕРВИСНЫХ АРХИТЕКТУРАХ НА ОСНОВЕ ФРЕЙМВОРКА SPRING

При разработке больших микросервисных систем способы взаимодействия микросервисов между собой часто подвергаются изменениям. После каждого такого изменения необходимо менять схемы, отображаемые в документации, чтобы разработчики и пользователи имели актуальную информацию о структуре системы. Наличие инструмента для автоматической генерации графов взаимодействий микросервисов позволило бы полностью отказаться от их «ручной» поддержки и снизило бы риск устаревания документации. Кроме того, это было бы полезно для быстрого ознакомления с проектами, у которых документации нет вовсе.

Данный проект направлен на разработку функционального прототипа инструмента для анализа структуры микросервисных проектов на основе фреймворка Spring и выделения из нее графов взаимодействий микросервисов.

Реализация решения осуществлялась с использованием следующих технологий:

– Движок CodeQL [1] – применялся для сбора данных о рассматриваемом проекте. Включает в себя набор инструментов для создания и конфигурации баз данных о проекте на основе его исходного кода, а также язык запросов для извлечения сведений из полученных баз. В частности, для исследования Spring-проектов необходимо было использовать библиотеки “java” [2] для анализа основной кодовой базы и “javascript” [3] для анализа фронтенда и конфигурационных файлов.

– Инструмент визуализации Mermaid [4] – применялся для отрисовки диаграмм. Был выбран из-за легкости использования – готовую диаграмму можно отобразить в HTML-документе – и возможности гибкой конфигурации – присвоения узлам и ребрам разных цветов и стилизаций для упрощения восприятия пользователем.

– Язык программирования Python – применялся для обработки результатов запросов и генерации на их основании кода Mermaid-диаграмм. Был выбран из-за простоты использования и наличия удобных библиотек: “csv” для работы с таблицами и “re” для работы с регулярными выражениями. Последняя применялась для создания масок, при помощи которых производилось выделение из полного пути до файла имен и директорий.

На рисунке 1 представлена схема работы итогового прототипа. Получив на вход путь к директории с исходным кодом проекта, инструмент создает необходимые базы данных, производит к ним ряд заранее заготовленных запросов, затем анализирует полученные результаты и предоставляет пользователю HTML-документ, содержащий диаграмму в синтаксисе Mermaid.

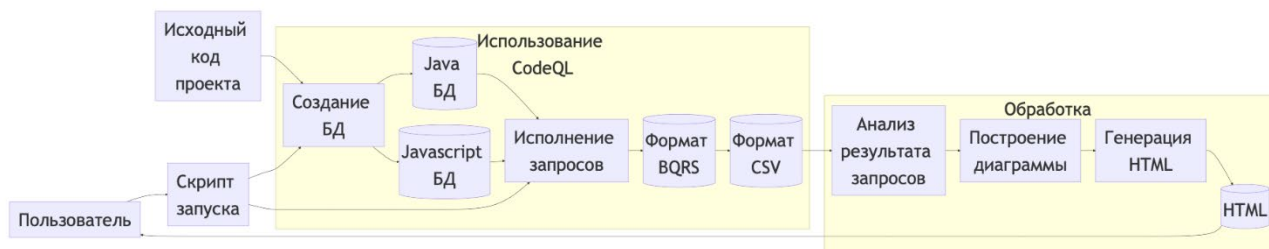


Рисунок 1 – Схема организации инструмента

Ценность использования CodeQL заключается в его возможности межмодульного семантического анализа достижимости. По сравнению с поиском синтаксических шаблонов, это дает более широкие возможности для исследования проектов.

Таким образом, в рамках работы был реализован функциональный прототип, проведена его апробация на тестовых проектах (результат одного из анализов показан на рисунке 2).

Прототип был снабжен документацией, модульными тестами и средствами непрерывной интеграции, результат доступен на Github (<https://github.com/plonerka/microservices-diagrams>).

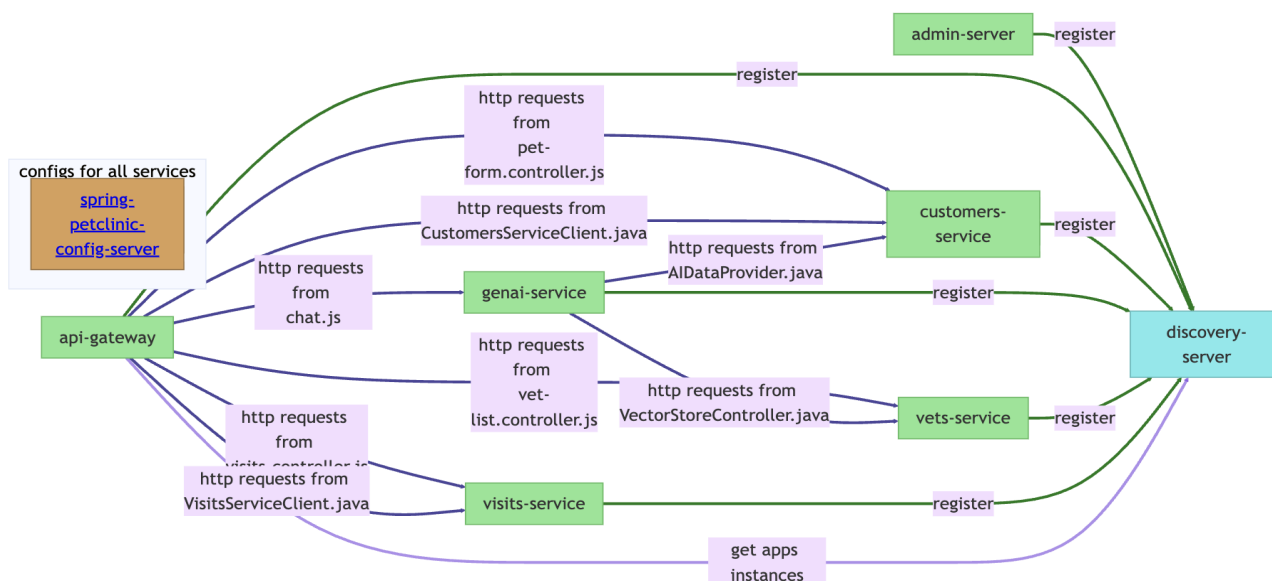


Рисунок 2 – Пример сгенерированной диаграммы тестового проекта [5]

Дальнейшее улучшение проекта может включать в себя увеличение количества распознаваемых свойств микросервисов, для уже распознаваемых свойств – увеличение количества определяемых видов их представления, если таковых несколько. Кроме того, планируется интеграция в инструмент LLM-агента для автоматизации составления запросов к базам данных CodeQL. Это поможет значительно ускорить развитие инструмента за счет того, что запросы не нужно будет составлять вручную на этапе разработки.

ЛИТЕРАТУРА

1. Документация движка CodeQL. [Электронный ресурс] Режим доступа: <https://codeql.github.com/docs/> (дата обращения: 17.03.2026)
2. Документация библиотеки “java” для движка CodeQL. [Электронный ресурс] Режим доступа: <https://codeql.github.com/codeql-standard-libraries/java/> (дата обращения 17.03.2026)
3. Документация библиотеки “javascript” для движка CodeQL. [Электронный ресурс] Режим доступа: <https://codeql.github.com/codeql-standard-libraries/javascript/> (дата обращения: 17.03.2026)
4. Документация Mermaid. [Электронный ресурс] Режим доступа: <https://mermaid.ai/docs/> (дата обращения: 17.03.2026)
5. Проект, использующийся в качестве тестового. [Электронный ресурс] Режим доступа: <https://github.com/spring-petclinic/spring-petclinic-microservices> (дата обращения: 17.03.2026)

УДК 004.415.5

А.К. Кудрявцев (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА ИНСТРУМЕНТА АВТОМАТИЗАЦИИ UI-ТЕСТОВ ПО СЛОВЕСНОМУ ОПИСАНИЮ КЕЙСОВ С ПОМОЩЬЮ AI

В современной разработке программного обеспечения наблюдается устойчивый тренд на ускорение циклов релизов и усложнение пользовательских интерфейсов. В этих условиях обеспечение качества продукта становится критически важным, а доля ручного тестирования остается значительной. Тестировщики тратят большое количество времени на оформление тест-кейсов и написание и поддержку UI-автотестов, которые нужно регулярно поддерживать

в связи с постоянными изменениями в интерфейсе. Одновременно с этим стремительное развитие технологий искусственного интеллекта и обработки естественного языка открывает новые возможности для трансформации процесса тестирования [1, 2].

Анализ современных инструментов автоматизации показывает, что, несмотря на их мощность, они требуют глубоких знаний языков программирования и значительных усилий по поддержке тестовой базы. Существующие AI-решения часто являются дорогостоящими либо решают лишь узкий круг задач [3]. При этом полностью отсутствуют доступные инструменты, позволяющие тестировщику, не обладающему навыками программирования, просто описать кейс на естественном языке и получить готовый исполняемый тест [4].

Таким образом, возникает противоречие между потребностью в быстром создании устойчивых UI-тестов и сложностью их реализации традиционными методами. Это обуславливает актуальность разработки инструмента, который переведет процесс создания тестов в разработку тест-кейсов на естественном языке и сокращении времени на автоматизацию через написание кода.

Целью данной работы является разработка инструмента автоматизации UI-тестов по словесному описанию кейсов.

Для достижения поставленной цели необходимо решение следующих задач:

1. Проанализировать современные подходы и инструменты для автоматизации тестирования мобильных приложений.
2. Исследовать существующие на рынке решения и кейсы применения ИИ в тестировании программного обеспечения.
3. Выбрать инструменты, библиотеки и алгоритмы ИИ для реализации собственного решения тестирования UI мобильных приложений.
4. Разработать архитектуру прототипа системы, способной к распознаванию элементов интерфейса и генерации тестовых сценариев на основе ИИ.
5. Реализовать прототип системы, включая модули для взаимодействия с приложением, анализа интерфейса и выполнения тестовых действий.
6. Провести эксперименты по тестированию выбранного мобильного приложения с использованием разработанного прототипа.
7. Произвести сравнительный анализ эффективности предложенного подхода с традиционными методами автоматизации по ключевым метрикам (время выполнения, покрытие, устойчивость).
8. Сформулировать выводы о целесообразности и перспективах использования искусственного интеллекта в автоматизации тестирования мобильных приложений.

Ключевой особенностью разработанного инструмента является полностью автоматизированная сквозная цепочка взаимодействий, охватывающая все этапы тестирования — от парсинга словесного описания до публикации итоговых отчетов. Пользователю достаточно лишь указать идентификаторы необходимых тестов в базе данных, после чего система самостоятельно извлекает описания, выполняет их разбор с разбиением на шаги, приводит к структурированному JSON-формату, запускает исполнение на целевом устройстве и формирует детализированную отчетность по результатам прогона.

Благодаря полной автоматизации процесса на протяжении всего времени работы инструмента не требуется вмешательство пользователя, что освобождает тестировщика от выполнения рутинных операций по написанию и поддержке автотестов. Высвободившиеся ресурсы могут быть направлены на решение более сложных задач, требующих человеческой экспертизы, а сокращение объема ручного тестирования позволяет существенно повысить общую эффективность процесса обеспечения качества за счет сокращения объема монотонной деятельности [5].

Искусственный интеллект в данном инструменте имитирует поведение человека: он точно так же взаимодействует с приложением, совершая тапы по кнопкам, свайпы по экрану и ввод текста в поля. Только делает это алгоритм, который руководствуется словесной инструкцией и выполняет все действия самостоятельно, без участия пользователя. При этом

каждое действие ИИ фиксируется с помощью автоматического создания скриншотов — система последовательно сохраняет изображения экрана и верифицирует его состояние до и после каждого действия. Благодаря этому можно наглядно проследить весь ход выполнения теста и убедиться, что ИИ действовал в точности по описанию: он нажимал именно на те элементы, на которые нужно, вводил корректные данные и получал ожидаемые результаты на экране [6].

Практическая значимость разрабатываемого инструмента заключается в снижении порога входа в автоматизацию тестирования для специалистов по контролю качества, сокращении времени на написание и поддержку тестовой базы, а также в повышении стабильности тестов за счет достаточности адаптации кейсов к изменениям интерфейса. Инструмент может быть использован как в небольших командах, где нет выделенного специалиста по автоматизации, так и в крупных проектах для ускорения регрессионного тестирования.

В результате работы планируется создать прототип инструмента, соответствующий современным требованиям программной инженерии. Ожидается, что эксперименты покажут сокращение времени создания тестов по сравнению с ручным написанием кода, а также повышение устойчивости тестов к изменениям UI. Разработанный прототип позволит сделать вывод о перспективности применения методов искусственного интеллекта для автоматизации процесса UI-тестирования мобильных приложений.

ЛИТЕРАТУРА

1. Garousi V., Joy N., Jafarov Z., et al. AI-powered software testing tools: A systematic review and empirical assessment of their features and limitations // arXiv preprint. — 2025. — arXiv:2409.00411v3.
2. Yu S., Ling Y., Fang C., et al. LLM-Guided Scenario-based GUI Testing // arXiv preprint. — 2025. — arXiv:2506.05079v3.
3. Ricca F., Marchetto A., Stocco A. A multi-year grey literature review on AI-assisted test automation // Information and Software Technology. — 2025. — Vol. 186. — Art. 107799. — DOI: 10.1016/j.infsof.2025.107799.
4. Fatin S., Barua S., et al. LELANTE: LEveraging LLM for Automated ANdroid TESting // arXiv preprint. — 2025. — arXiv:2504.20896.
5. Kolthoff K., Bartelt C., Maedche A., et al. GUISpector: An MLLM Agent Framework for Automated Verification of Natural Language Requirements in GUI Prototypes // arXiv preprint. — 2025. — arXiv:2510.04791.
6. Garousi V., et al. AI-Powered Software Testing Tools: Full Autonomy Remains a Distant Goal // Zenodo. — 2025. — DOI: 10.5281/zenodo.18171394

УДК 004.422

С.В. Кушнарева (2 курс магистратуры),
В.В. Амосов, к.т.н., доцент

НАБОР СРЕДСТВ ДЛЯ РАЗРАБОТКИ СОЦИАЛЬНОЙ СЕТИ НЕСКОЛЬКИХ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

В условиях стремительного развития цифровой экономики и роста конкуренции на рынке мобильных приложений ключевым фактором успеха становится способность платформ обеспечивать высокий уровень пользовательской вовлеченности. Социальные функции, такие как взаимодействие между пользователями, создание контента и формирование сообществ, играют важную роль в удержании аудитории и повышении лояльности. Однако разработка стабильных, масштабируемых и кастомизируемых социальных механизмов требует от компаний значительных ресурсов: временных, финансовых и экспертных. В связи с этим возникает потребность в универсальных решениях, которые позволили бы интегрировать

социальный функционал в сторонние приложения с минимальными затратами и максимальной эффективностью.

Предлагаемый набор средств для разработки (SDK) [1] направлен на решение проблемы фрагментации разработки: вместо создания отдельных социальных модулей для каждого приложения используется единая кодовая база, что упрощает поддержку, масштабирование и обновление функционала.

Данная работа посвящена разработке [2,3] программного комплекса SDK (Software Development Kit) для интеграции социальных функций в несколько мобильных приложений. Целью исследования является создание универсального компонента, который обеспечивает единый набор социальных возможностей (например, управление профилем, подписки, создание постов) при сохранении возможности кастомизации под индивидуальные требования каждого приложения. Такой подход позволяет сократить время выхода продукта на рынок (Time-to-Market), снизить стоимость разработки и обеспечить согласованность пользовательского опыта (UX/UI) в рамках экосистемы приложений.

Для достижения этой цели необходимо решение задач, перечисленных ниже:

1. Проведение сравнительного анализа существующих подходов.
2. Предложение реализации набора средств для разработки.
3. Реализация данного решения.

В ходе исследования был проведен сравнительный анализ трех ключевых подходов к реализации функционала: индивидуальная разработка для каждого приложения, использование микрофронтендов с WebView и разработка на основе нативного SDK. [4,5,6] Анализ по критериям Time-to-Market, сложности интеграции, гибкости кастомизации и консистентности UX/UI позволил объективно обосновать выбор SDK-подхода как наиболее сбалансированного для решения поставленной задачи. [7]

Ключевым результатом работы является разработанная двухуровневая архитектура SDK, представленная на рисунке 1.[7] На укрупненном уровне реализована многослойная модель на основе Clean Architecture [2,5] с четким разделением на слои Presentation (UI-компоненты), Domain (бизнес-логика) и Data (инфраструктура). На детальном уровне система декомпозирована на вертикальные функционально законченные модули (Profile, SocialGraph, Content, Recommendations), каждый из которых охватывает все три слоя. Такой подход обеспечивает низкое зацепление, высокую связность модулей, тестируемость и независимость бизнес-логики от деталей реализации. [5,6]

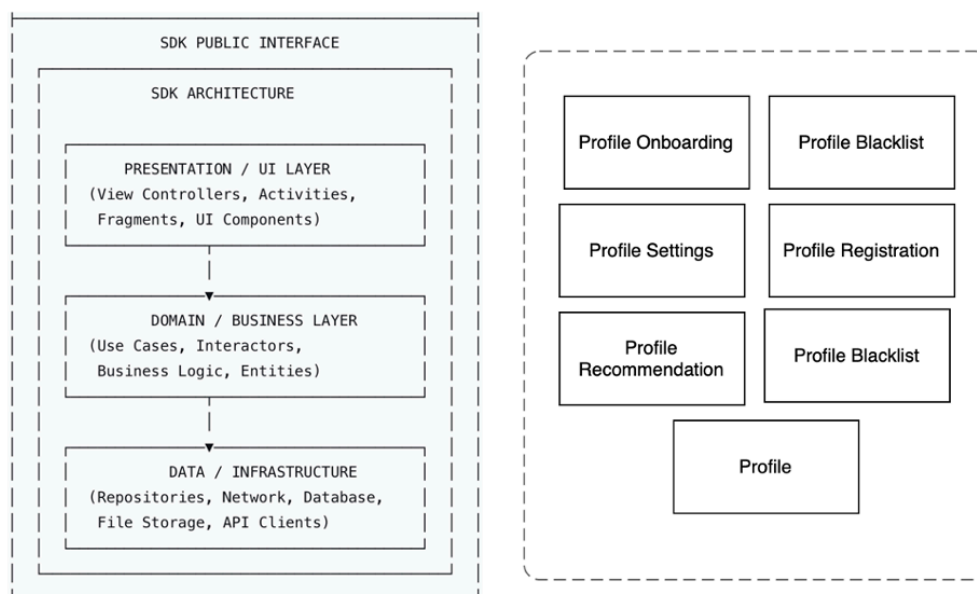


Рисунок 1 – Архитектура: а) многослойная модель; б) функциональные модули

SDK было реализовано для iOS с помощью Swift в среде разработки Xcode. Настройка SDK осуществляется за счет установления параметров инициализации,

установления значения переключателей, обработки событий на стороне интегратора или SDK.[6,7]

Разработанный подход позволяет сократить время вывода продуктов на рынок (Time-to-Market), снизить совокупную стоимость владения (ТСО), обеспечить консистентность ключевой бизнес-логики во всей экосистеме приложений и предоставить интеграторам гибкие возможности для кастомизации пользовательского интерфейса. Предложенная архитектурная и методологическая модель может быть использована для ускорения разработки мобильных приложений с социальным функционалом и оптимизации затрат на их создание и поддержку.

ЛИТЕРАТУРА

1. Гарни Р. Swift. Разработка приложений в среде Xcode для iPhone и iPad с использованием iOS SDK. — М.: Эксмо, 2017. — 416 с.
2. Андронов А.А. Проектирование и разработка мобильных приложений: архитектура и шаблоны проектирования. — М.: ДМК Пресс, 2019. — 342 с.
3. Брукс Ф. Мифический человеко-месяц, или Как создаются программные системы. — СПб.: Символ-Плюс, 2021. — 304 с.
4. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — М.: Питер, 2022. — 366 с.
5. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. — СПб.: Питер, 2021. — 352 с.
6. Apple Developer Documentation. [Электронный ресурс] Режим доступа: <https://developer.apple.com/design/human-interface-guidelines/> (дата обращения: 10.10.2025).
7. Swift.org. Swift API Design Guidelines. [Электронный ресурс] Режим доступа: <https://www.swift.org/documentation/api-design-guidelines/> (дата обращения: 20.10.2025).

УДК 004.89

А.И. Литвак (2 курс магистратуры),
Е.Г. Хитров, д.т.н., доцент

МЕТОДЫ УПРАВЛЕНИЯ КОНТЕКСТОМ В СИСТЕМАХ НА ОСНОВЕ LLM

В современном мире, где важна не только скорость, но и цена и точность выполнения задач, при быстро растущем количестве информации все больше заданий отдается LLM (Large Language Models, большим языковым моделям). При этом, проблемой остается ограниченность размера контекстного окна и отсутствие долговременной памяти. В большом количестве задач, таких как аналитические агенты, чат-боты и мультиагентные системы возникает необходимость структурированного хранения истории взаимодействия, подключения внешних источников знаний, например СУЗ (системы управления знаниями) организации, а также управления релевантностью того контекста, который модель помнит.

Актуальность исследования обусловлена растущей популярностью LLM. Существующие подходы частично решают проблему, однако не обеспечивают стандартизированного механизма для взаимодействия моделей с внешними источниками. собственным контекстом и общим состоянием системы.

Цель работы заключается в аналитическом обзоре различных методов и архитектур управления контекстом и памятью в LLM.

Для достижения этой цели необходимо решение следующих задач:

1. Обзор существующих подходов.
2. Проведение сравнительного анализа.
3. Предложение идеи решения проблемы ограниченности контекстуального окна.

Современные LLM основаны на трансформерной архитектуре нейронной сети [1]. Энкодер отображает входную последовательность представлений символов в последовательность непрерывных представлений. На их основе декодер затем генерирует выходную последовательность по одному элементу за шаг. Принципиальной особенностью является отказ от рекуррентных и сверточных механизмов обработки, модель полностью основана на механизме внимания. Такой подход обеспечивает высокую степень параллелизации вычислений, а значит и существенное ускорение работы. Несмотря на все преимущества архитектуры, одной из главных проблем на данный момент остается ограниченность обрабатываемого контекста заранее заданным максимальным числом токенов. Также существует более широкая задача – отсутствие встроенного механизма долговременного хранения состояния. При каждом новом запросе модель начинает работу с чистого листа, не учитывая предыдущие данные для дальнейшей генерации.

Все методы работы с контекстом и памятью можно разделить на три группы: алгоритмические, использующие внешние источники данных и агентные архитектуры с инструментальным расширением.

Наиболее простыми и популярными методами для сокращения используемого количества токенов являются алгоритмические методы. Они достаточно просты в разработке, не требуют использования новых технологий, но имеют не очень высокую точность. К алгоритмическим можно отнести: методы усечения контекста, суммаризацию и иерархическую память.

К стратегиям усечения контекста относятся, например, FIFO-удаление сообщений из диалога, удаление сообщений по релевантности к текущему запросу, алгоритм Dynamic Context Pruning [2] (выбор релевантного контекста в реальном времени). Основными преимуществами является кроссмодельность и простота реализации, в то время как вероятная потеря релевантной информации является главным ограничением использования этой группы подходов.

Решить проблему потери релевантного смысла в случае использования методов усечения контекста можно с помощью использования суммаризации диалога. Суммаризация – процесс, сжимающий историю запросов, стремясь максимально уменьшить количество используемых токенов при минимальной потере семантической сущности. В простейшем случае дополнительной модели отправляется диалог, а она пытается сохранить наибольший смысл.

В отличие от методов, представленных выше, иерархическая память представляет из себя не плоскую структуру, где каждый токен входного пользовательского сообщения имеет одинаковый вес и значимость для диалога, а многоуровневое представление, где высота уровня в системе прямо пропорциональна обобщенности представления, которую она передает. Примерами реализации являются работы: Hierarchical Long-Term Memory (HiMem) [3] – фреймворк, комбинирующий эпизодическую и заметочную памяти, комбинируя их и Hierarchical Memory Transformer (HMT) [4] – модель последовательно переносит данные из старых сегментов, трансформируя ее в эмбединги, в иерархической памяти.

К методам расширения памяти через внешние источники данных можно отнести RAG. RAG – это процесс оптимизации выходных данных большой языковой модели, в котором перед получением ответа обращение идет к базе знаний за пределами источников обучающих данных. Основной задачей, является защита от галлюцинирования и создание ответа на основе обновляемых данных. Базовая RAG-цепочка состоит из следующих этапов: преобразование запроса на естественном языке в эмбединг, поиск релевантных документов, генерация ответа на основе топ-k найденных фрагментов. Преимуществом данного подхода является не только масштабируемость без увеличения контекстного окна, но и легкое поддержание актуальности. А к проблемам можно отнести невозможность верификации данных, полученных из извлеченных документов, а также сложности, связанные с необходимостью поддержки векторного индекса и эмбедингов при добавлении или изменении документов.

Агенты являются следующей итерацией LLM. Модель самостоятельно выбирает предоставленные инструменты для решения той или иной задачи. В контексте решения проблемы агентная архитектура может быть представлена как способ переноса токеной нагрузки за пределы одного вызова модели. К плюсам подхода можно отнести повышение масштабируемости и комплексное решение проблемы контекстного окна. Тогда как недостатками является огромное множество несовместных реализаций и интеграций инструментов, неявность интерфейсов и большая сложность масштабирования.

Методы с минимальным вмешательством в архитектуру проще в реализации, но уступают по качеству, а архитектурные подходы с внешними источниками знаний обеспечивают более стабильную работу ценой усложнения разработки. При этом ни один из существующих методов не является универсальным. Дополнительной проблемой становится отсутствие стандартизированного протокола взаимодействия между языковой моделью, внешней памятью и инструментами, что затрудняет масштабирование и переиспользование решений. В связи с этим возникает необходимость единого протокола управления контекстом и инструментами, одним из перспективных решений которого является Model Context Protocol (MCP) [5].

ЛИТЕРАТУРА

1. Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. – 2017. – Т. 30.
2. Anagnostidis S. et al. Dynamic context pruning for efficient and interpretable autoregressive transformers //Advances in Neural Information Processing Systems. – 2023. – Т. 36. – С. 65202-65223.
3. Zhang N. et al. HiMem: Hierarchical Long-Term Memory for LLM Long-Horizon Agents //arXiv preprint arXiv:2601.06377. – 2026.
4. He Z. et al. Hmt: Hierarchical memory transformer for efficient long context language processing //Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). – 2025. – С. 8068-8089.
5. MCP. [Электронный ресурс] Режим доступа: <https://modelcontextprotocol.io/docs/>

УДК 004.453

Х.Б. Мирзоев (5 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ И ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ СРЕДСТВ ПРОВЕРКИ ВАЛИДНОСТИ JSON-МОДЕЛЕЙ НА PYTHON

JSON – базовый формат обмена данными в веб-сервисах и распределённых системах. Ошибки в структуре и содержимом JSON-сообщений приводят к отказам интеграций и некорректной бизнес-логике, поэтому требуется надёжная автоматическая проверка соответствия данных заданной схеме [1-3]. В экосистеме Python для этой задачи используются библиотеки jsonschema, Pydantic, Marshmallow, Cerberus, однако их выбор обычно основан на документации и личном опыте разработчиков, а не на систематическом экспериментальном сравнении по корректности, производительности и качеству диагностических сообщений.

Целью работы является экспериментальное сравнение средств проверки валидности JSON-моделей на Python (jsonschema, Pydantic, Marshmallow, Cerberus) в единых условиях по набору функциональных и нефункциональных метрик. Решены следующие задачи: 1) задать модели для тестирования и классы ошибок, покрывающие структурные и семантические нарушения; 2) разработать стенд генерации и валидации наборов данных с измерением времени и потребления памяти; 3) выполнить серии экспериментов при фиксированных и масштабируемых параметрах (глубина дерева, размер графа) для каждого из валидаторов;

4) сопоставить валидаторы по корректности, производительности, использованию ресурсов и качеству диагностических сообщений, сформулировать рекомендации по их выбору.

Эксперименты выполнялись на Acer Nitro AN515-52, Windows 11; процессор Intel64 Family 6 Model 158 (~2304 МГц), ОЗУ 8 ГБ; Python 3.12.4. Используются два набора данных, построенные по единым моделям и классам ошибок. Builtin-набор включает 14 вручную заданных тест-кейсов и применяется как контрольный набор для быстрой проверки корректности реализации правил и интерпретации ошибок (sanity-check); он не используется для генерации синтетического набора. Synthetic-набор формируется автоматически: для каждой модели генерируются валидные экземпляры, после чего к части из них применяются целевые мутации, реализованные кастомными функциями-мутациями (mutate_*_make_invalid) в генераторе стенда, соответствующие заданным классам ошибок (удаление обязательного поля, подмена типа, добавление лишнего поля, нарушение ссылочной целостности). В обоих наборах рассматриваются три модели данных: user, tree и graph; входные данные представлены Python-структурами (dict/list) после десериализации JSON. Структура эксперимента на рисунке 1.

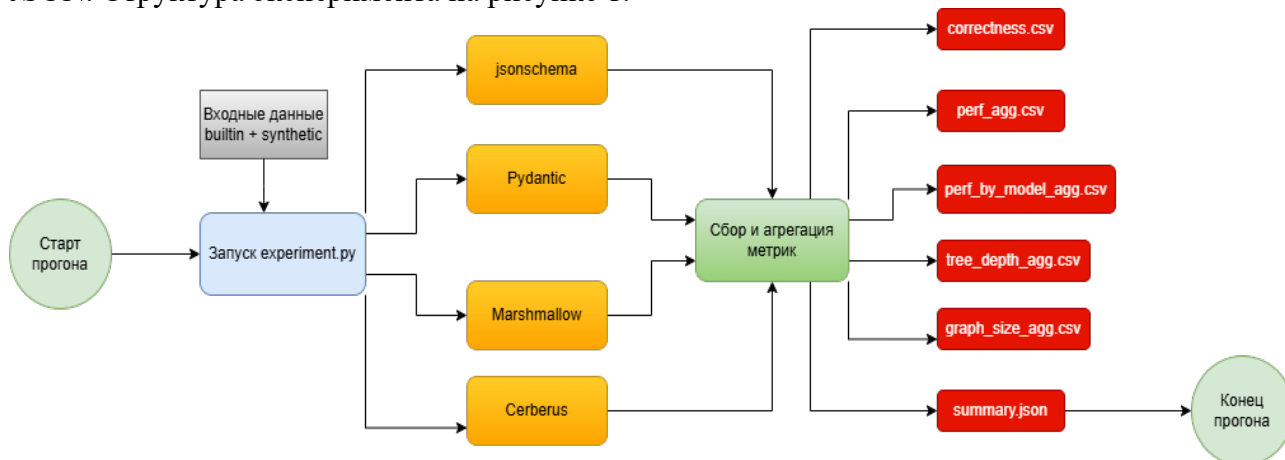


Рисунок 1 – Схема сравнения средств валидации JSON на Python

Результаты эксперимента приведены в таблице 1. Значения получены автоматически из CSV-артефактов эксперимента (perf_agg.csv, perf_by_model_agg.csv, correctness.csv) с помощью скрипта репозитория [1]. Потребление памяти оценивалось по пиковому значению tracemalloc и по приросту RSS ($\Delta\text{RSS} = \text{rss\_peak} - \text{rss\_before}$). Показатель «доля сообщений с путём к ошибке» (has_path) отражает, содержит ли диагностическое сообщение валидатора явный путь до ошибочного поля в терминах структуры JSON (например, settings.notifications.sms), а не номер строки исходного файла. Строка sum соответствует агрегированию по всем кейсам (user+tree+graph), а не среднему трёх строк.

Для проверки равномерности покрытия по классам ошибок и корректности их обнаружения дополнительно сформирован отчёт по типам внедрённых мутаций и их обнаружению валидаторами (injected/found/has_path), формируемый скриптом src/error_types_report.py репозитория [4]. По метрике корректности на обоих наборах данных (builtin и synthetic) все валидаторы совпали с ожидаемым результатом (100 % совпадений по признаку валидности), что подтверждает эквивалентность реализованных правил.

В модели graph дополнительно проверялись семантические ограничения: уникальность идентификаторов узлов и ссылочная целостность ребер. Различия проявляются в нефункциональных характеристиках на synthetic-наборе (таблица 1): Pydantic демонстрирует наивысшую пропускную способность; jsonschema занимает промежуточную позицию; Marshmallow и Cerberus имеют более низкую пропускную способность. По доле сообщений с указанием пути к ошибочному полю pydantic, marshmallow и cerberus дают 100 % для невалидных кейсов, тогда как у jsonschema показатель составляет около 64 %, это означает не пропуск ошибок, а менее детальную локализацию: в части невалидных кейсов jsonschema сообщает о наличии ошибки, но без явного пути до конкретного поля.

Таблица 1 – Сводные результаты на синтетическом наборе (1800 кейсов, 3 прогона)

Валидатор	Структуры	Пропускная способность, валид./с (медиана)	Доля сообщений с путем к ошибке, %	Относительное потребление памяти (Δ RSS / tracemalloc), MiB
jsonschema	user	2816	60	0.047 / 0.216
	tree	204	67	0.023 / 0.220
	graph	75	67	0.020 / 0.181
	sum	162	64	0.074 / 0.220
pydantic	user	20762	100	0.012 / 0.003
	tree	4870	100	0.020 / 0.066
	graph	1089	100	0.020 / 0.084
	sum	2521	100	0.027 / 0.088
marshmallow	user	4080	100	0.020 / 0.126
	tree	365	100	0.020 / 0.122
	graph	128	100	0.020 / 0.175
	sum	268	100	0.043 / 0.179
cerberus	user	295	100	0.035 / 0.168
	tree	621	100	0.020 / 0.136
	graph	26	100	0.227 / 0.524
	sum	66	100	1.242 / 0.625

Таким образом, выбор средства валидации зависит от контекста использования: для высоконагруженных сервисов с жёсткими требованиями к скорости валидации целесообразно применение Pydantic; при необходимости строгого следования стандарту JSON Schema и интеграции с внешними инструментами проверок — jsonschema; в сценариях, где важны удобство описания схем и подробные сообщения об ошибках, оправдано использование Marshmallow, тогда как Cerberus может применяться в относительно простых конфигурациях с небольшими объёмами данных. Исходный код стенда, генераторы наборов данных и агрегированные результаты экспериментов размещены в открытом репозитории [1].

ЛИТЕРАТУРА

1. Ong T.C., Pradhananga R., Holve E.G., Kahn M.G. Framework for Classification of Electronic Health Data Extraction–Transformation–Loading Challenges in Data Network Participation. 2017.— DOI 10.5334/egems.222.
2. Abdelaal M., Hammacher C., Schoening H. REIN: A Comprehensive Benchmark Framework for Data Cleaning Methods in ML Pipelines // Proc. of EDBT. 2023 — P. 499—511.—URL: <https://openproceedings.org/2023/conf/edbt/3-paper-49.pdf>
3. Redyuk S., Kaoudi Z., Markl V., Schelter S. Automating Data Quality Validation for Dynamic Data Ingestion // Proc. of EDBT. 2021.— P. 61—72.—DOI 10.5441/002/edbt.2021.07
4. Mirzoev K.B., Parkhomenko V.A. json-validation-bench // GitHub – URL:

УДК 004.4'244

Д.Н. Мартюшова (4 курс бакалавриата),
А.В. Самочадин, к.т.н., доцент

РАЗРАБОТКА ПЛАТФОРМЫ QUXIMO ДЛЯ УПРАВЛЕНИЯ ЖИЗНЕННЫМ ЦИКЛОМ МИКРОСЕРВИСНЫХ ПРИЛОЖЕНИЙ

Современные программные системы всё чаще создаются в виде распределённых приложений, состоящих из множества независимых сервисов. Использование микросервисной архитектуры позволяет повысить масштабируемость, отказоустойчивость и гибкость развития программных продуктов [1,2]. Однако такой подход усложняет управление инфраструктурой, процессы развертывания и сопровождения программных систем.

Одним из решений данной проблемы является использование облачных платформ класса Platform as a Service (PaaS), обеспечивающих автоматизацию процессов разработки, интеграции и эксплуатации программных приложений [3]. Такие платформы позволяют разработчикам сосредоточиться на реализации бизнес-логики, передав задачи управления инфраструктурой специализированным инструментам.

Целью проекта является разработка платформы Quximo, обеспечивающей разработчикам централизованный доступ к инструментам управления исходным кодом, процессам непрерывной интеграции и доставки программного обеспечения, механизмам аутентификации и безопасного хранения конфиденциальных данных.

В рамках данной работы рассматривается платформа Quximo — PaaS-решение, предназначенное для создания единой среды разработки и эксплуатации микросервисных приложений. Основной задачей платформы является упрощение процессов развертывания сервисов и управления инфраструктурой распределённых систем.

Для достижения поставленной цели были сформулированы следующие *задачи*:

1. Провести анализ современных архитектурных решений для построения платформенных сред разработки программного обеспечения.
2. Разработать архитектуру платформы Quximo, обеспечивающую интеграцию инструментов разработки и управления инфраструктурой.
3. Реализовать прототип платформы и провести экспериментальное тестирование её функциональности.
4. Оценить эффективность использования платформенного подхода при разработке микросервисных приложений.

Архитектура платформы основана на принципах микросервисной архитектуры и контейнеризации приложений, предполагающих разделение функциональности системы на независимые сервисы, взаимодействующие через сетевые интерфейсы [1]. Развертывание приложений осуществляется в контейнерной инфраструктуре, управляемой системой оркестрации Kubernetes, обеспечивающей масштабирование, балансировку нагрузки и устойчивость системы к отказам [5].

Ключевым элементом платформы является DevOps-конвейер, автоматизирующий процессы сборки, тестирования и развертывания приложений. Управление исходным кодом и реализация процессов CI/CD осуществляется с использованием системы GitLab, позволяющей автоматически выполнять сборку программных компонентов, запуск тестов и публикацию новых версий сервисов.

Основная логика платформы реализована на языке программирования Go, обеспечивающем высокую производительность и эффективную работу с параллельными процессами. Для взаимодействия между сервисами используется протокол gRPC, позволяющий организовать быстрый обмен данными между компонентами распределённой системы.

Для управления инфраструктурой применяются инструменты Infrastructure as Code — Terraform и Helm. Terraform используется для управления инфраструктурными ресурсами, а Helm — для конфигурирования и развертывания приложений в Kubernetes-кластере.

Безопасность системы обеспечивается специализированными сервисами управления доступом и хранения секретов. Управление идентификацией и авторизацией пользователей реализовано с использованием Keycloak, а безопасное хранение конфиденциальных данных и секретов осуществляется с помощью HashiCorp Vault.

Для обеспечения наблюдаемости системы используются инструменты Prometheus и Grafana. Prometheus выполняет сбор и хранение метрик работы сервисов и инфраструктуры, а Grafana обеспечивает визуализацию и анализ показателей системы посредством интерактивных дашбордов.

Таким образом, технологический стек платформы включает язык Go, Kubernetes, Terraform, Helm, GitLab для управления исходным кодом и CI/CD, протокол gRPC для

межсервисного взаимодействия, а также Keycloak и HashiCorp Vault для обеспечения безопасности системы.

Архитектура платформы представлена на рисунке 1.

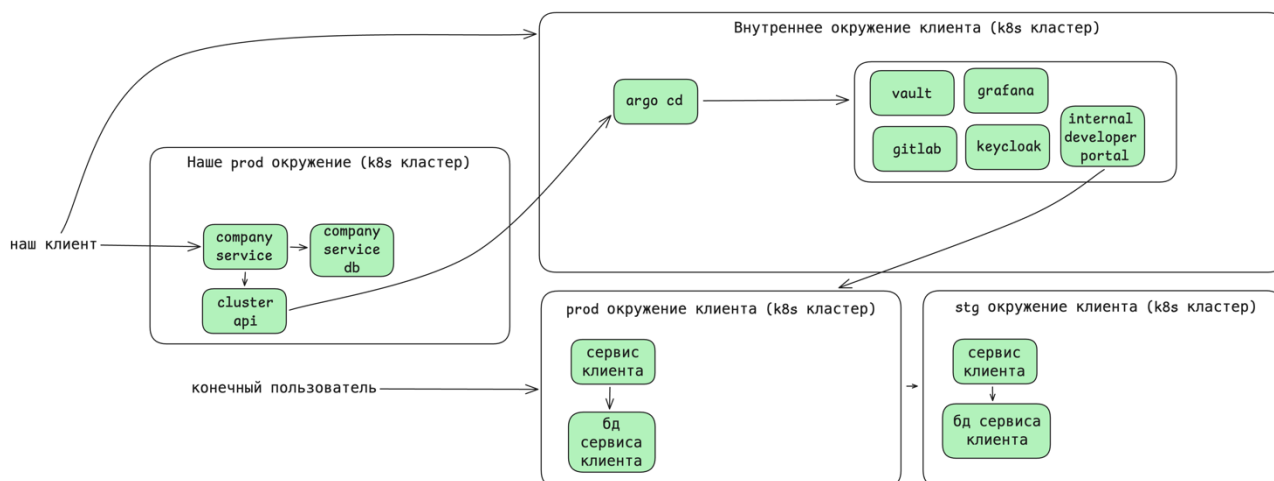


Рисунок 1 – Архитектура платформы Quximo

В ходе выполнения работы был разработан прототип платформы Quximo, демонстрирующий возможность создания интегрированной PaaS-среды для разработки и эксплуатации микросервисных приложений. Результаты показывают, что использование платформенного подхода позволяет сократить время развертывания сервисов и повысить уровень автоматизации процессов разработки.

Разработанное решение может применяться при создании распределённых программных систем, а также в образовательных и исследовательских проектах, связанных с изучением современных технологий программной инженерии.

ЛИТЕРАТУРА

1. Pahl C. Containerization and the PaaS Cloud // *IEEE Cloud Computing*. – 2015. – Vol. 2. – No. 3. – P. 24–31.
2. Dragoni N., et al. Microservices: Yesterday, Today, and Tomorrow // *Present and Ulterior Software Engineering*. – Springer, 2017. – P. 195–216.
3. Zhang Q., Chen M., Li L. Cloud computing: state-of-the-art and research challenges // *Journal of Internet Services and Applications*. – 2010. – Vol. 1. – P. 7–18.
4. Leite L., Rocha C., Kon F., Milojicic D., Meirelles P. A Survey of DevOps Concepts and Challenges // *ACM Computing Surveys*. – 2020. – Vol. 52. – No. 6.
5. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Borg, Omega, and Kubernetes // *Communications of the ACM*. – 2016. – Vol. 59. – No. 5. – P. 50–57.
6. Ahmad H., Treude C., Wagner M., Szabo C. Smart HPA: Resource-Efficient Auto-Scaling for Microservice Architectures // *arXiv / Software Engineering Research*. – 2024.

УДК 004.942

Д.А. Меркадо Оудалова (2 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент

ФОРМАЛИЗОВАННЫЙ ПОДХОД К ВЫБОРУ МЕТОДА MODEL CHECKING ДЛЯ BPMN-МОДЕЛЕЙ

Рост автоматизации усилил зависимость от сложных цифровых систем, что вызвало широкое использование BPMN для моделирования процессов и формальной верификации, осложняемой гетерогенностью моделей: детерминированные или вероятностные, временные или нет, с параллелизмом, циклами и стохастикой [1]. Кроме того, существующие методы верификации обладают различной масштабируемостью, точностью и вычислительной

сложностью [2,3], что делает задачу выбора оптимального подхода к верификации BPMN-моделей нетривиальной.

Таким образом *целью* работы является разработать формализованный критерий и алгоритм выбора метода model checking в зависимости от свойств BPMN-модели.

Для достижения этой цели были решены следующие задачи:

1. Исследовать существующие методы проверки BPMN и формализовать модель с ключевыми характеристиками (детерминированность, стохастичность и др.).
2. Провести сравнительный анализ методов MC и разработать критерии классификации требований к верификации.
3. Предлагать алгоритм выбора подхода к верификации и наиболее подходящие методы.

Для разработки алгоритма принятия решения по выбору подхода MC были рассмотрены современные методы с учетом характеристик моделей BPMN. С этой целью модель рассматривается не только как графическая нотация, но и как формальная структура.

Используется определение модели сотрудничества BPMN [1]: $BC = \{BP_i, M, SL\}$ где каждый внутренний бизнес-процесс определяется как $BP_i = \{V_i, E_i, L_i\}$ при этом множество узлов имеет вид $V_i = V_{A_i} \cup V_{E_i} \cup V_{G_i}$ а $E_i \subseteq V_i \times V_i$ задает последовательные потоки внутри пула, тогда как $M \subseteq \bigcup_{i \neq j} V_i \times V_j$, описывает межпуловые взаимодействия посредством сообщений.

Такая формализация позволяет интерпретировать BPMN-модель как систему переходов с анализом в терминах пространства состояний. Ее свойства определяются структурами V_i, E_i и M однозначность переходов задает детерминированность, вероятностные ветвления стохастичность, временные события расширяют пространство временными параметрами, циклы влияют на его ограниченность, а параллелизм через V_{G_i} и M определяет размер и сложность верификации. Исходя из этих характеристик формальной модели, выбор техники проверки осуществляется следующим образом. Classical MC применяется к небольшим детерминированным моделям с последовательными задачами и небольшим количеством шлюзов, обеспечивая полноту [2], с другой стороны, Symbolic MC подходит для крупных процессов с множеством задач, подпроцессов и событий, где переходы состояний могут быть представлены математическими формулами [4], в то время как Bounded MC используется при наличии повторяющихся циклов или ограниченных петель, позволяя получать контрпримеры без необходимости полного обхода пространства состояний [4]. Statistical MC позволяет оценивать вероятности в случае случайных решений или условных событий [5], аналогично Probabilistic/Quantitative MC оценивает производительность, затраты и риски при критических решениях, например, при альтернативных маршрутах в сложных шлюзах [6] и в последнюю очередь, Timed/Hybrid MC сочетает задачи с определенной длительностью, временные события и непрерывную динамику, моделируя поведение рабочих процессов реалистично [7]. Резюмируя вышеизложенное, результатом исследования являются *матрица сравнительного анализа* в таблице 1 и *предложенный алгоритм выбора* на рисунке 1, систематизирующие соответствие структурных свойств BPMN-модели и методов MC. Данный инструментарий формализует логику принятия решений, позволяя определить подход к верификации с учетом временных, вероятностных и структурных ограничений модели

Таблица 1 – Сравнительный анализ методов Model Checking для BPMN

Тип модели	Рекомендуемый метод	Полнота	Масштабируемость
Малые детерминированные	Classical	Полная	Низкая
Крупные детерминированные	Symbolic	Полная	Средняя/Высокая
С глубокими циклами	Bounded	Частичная	Высокая
Стохастические	Statistical	Вероятностная	Очень высокая
Формально вероятностные	Probabilistic	Полная	Средняя
Временные	Timed	Полная	Низкая

Algorithm 1 Model Checking Technique Selection

```

1: Input:  $BC = \{BP_i, M, SL\}$ 
2: Output: Model Checking technique  $T$ 

3: Analyze each  $BP_i = (V_i, E_i, L_i)$ 
4: if there are temporal constraints in  $V_{E_i}$  or durations associated with  $V_{A_i}$  then
5:   return Timed/Hybrid MC
6: end if
7: if there are probabilistic transitions in  $E_i$  or in  $M$  then
8:   if exact quantitative analysis is required then
9:     return Probabilistic MC
10:  else
11:    return Statistical MC
12:  end if
13: end if
14: if the structure of  $E_i$  contains cycles with depth  $> k$  then
15:   return Bounded MC
16: end if
17: Estimate the state-space size  $|S|$  as a function of:
     $|V_i|$ , the degree of parallelism in  $V_{G_i}$ ,
    and synchronization induced by  $M$ 
18: if  $|S|$  is large and can state transitions can be expressed as math formula then
19:   return Symbolic MC
20: else
21:   return Classical MC
22: end if

```

Рисунок 1 – Предложенный алгоритм выбора метода МС на основе характеристик BPMN-модели

Таким образом, можно сделать вывод, что выбор метода верификации зависит от структурных свойств модели BPMN и целей проверки, что обеспечивает эффективность, масштабируемость и надежность анализа.

ЛИТЕРАТУРА

1. Zhang Z. и др. Collaborative Modeling of BPMN and HCPN: Formal Mapping and Iterative Evolution of Process Models for Scenario Changes // Information. publisher, 2025. Т. 16, № 4.
2. Шестакова А. Ю., Сабуткевич А. М., Никифоров И. В. Подход к верификации моделей BPMN 2.0 с использованием алгоритма Model Checking // Современные технологии в теории и практике программирования. Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. С. 141–143.
3. Petrov O. M. Finding More Bugs with Software Model Checking using Delta Debugging // Proc. Inst. Syst. Program. Ras. 2023. Т. 35, № 3. С. 151–162.
4. Костюков Ю. О. Автоматический вывод индуктивных инвариантов программ с алгебраическими типами данных =: Automatic inference of inductive invariants of programs with algebraic data types: в 2-х т.: диссертация ... кандидата физико-математических наук: 2.3.5. Санкт-Петербург, 2023. 213 с.
5. Legay A. и др. Statistical Model Checking // Computing and Software Science: State of the Art and Perspectives / под ред. Steffen B., Woeginger G. Cham: Springer International Publishing, 2019. С. 478–504.
6. Cabral R., Bolin D., Rue H. Robustness, model checking, and hierarchical models // J. R. Stat. Soc. Ser. B Stat. Methodol. 2025. Т. 87, № 3. С. 632–652.
7. Jamroga W. и др. Towards Probabilistic Strategic Timed CTL: arXiv:2602.10824. arXiv, 2026.

УДК 004.42

Д.В. Митрошин (4 курс бакалавриата),
А.В. Петров, ст. преподаватель

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ РАЗЛИЧНЫХ СТРАТЕГИЙ ОРГАНИЗАЦИИ WORKER POOL ПРИ ОБРАБОТКЕ КОНКУРЕНТНЫХ ЗАДАЧ НА ЯЗЫКЕ GO

Современные программные системы часто функционируют в условиях высокой нагрузки, требующей эффективной организации параллельной обработки задач [1]. Особенно актуальна данная проблема для backend-сервисов, систем обработки данных и микросервисных архитектур, где производительность напрямую зависит от эффективности использования вычислительных ресурсов. Неэффективная организация конкурентного

выполнения может приводить к избыточному потреблению памяти, увеличению времени обработки задач и снижению устойчивости системы.

Одним из распространённых подходов к решению данной проблемы является использование паттерна Worker Pool, представляющего собой метод организации обработки задач с помощью ограниченного числа рабочих потоков выполнения [2]. Данный подход позволяет контролировать степень параллелизма и снижать нагрузку на планировщик выполнения, однако эффективность его применения зависит от выбранной стратегии управления количеством воркеров.

Целью данной работы является исследование эффективности различных стратегий организации Worker Pool при обработке конкурентных задач, а также определение оптимальных условий их применения.

Для достижения поставленной цели были сформулированы следующие задачи:

- анализ существующих подходов организации конкурентной обработки задач;
- реализация различных моделей обработки задач на языке Go;
- проведение экспериментального сравнения исследуемых подходов;
- оценка влияния типа нагрузки на эффективность использования Worker Pool.

В рамках исследования были реализованы модели обработки задач, включающие последовательное выполнение, обработку задач с созданием отдельной goroutine для каждой задачи, использование фиксированного пула воркеров, а также динамического пула воркеров с адаптацией к текущей нагрузке [3].

Реализация экспериментального прототипа выполнена на языке Go с использованием стандартных средств конкурентного программирования, включая goroutines, каналы взаимодействия и механизмы синхронизации пакета sync [4]. Для проведения экспериментов использовались два типа задач: CPU-bound задачи, моделирующие вычислительную нагрузку, и IO-bound задачи, имитирующие операции ожидания ввода-вывода [5].

Оценка эффективности рассматриваемых подходов производилась на основе следующих характеристик:

- общее время выполнения набора задач;
- объём используемой оперативной памяти;
- количество одновременно выполняемых goroutines.

Результаты проведённых экспериментов показали, что последовательная обработка задач демонстрирует наихудшие показатели производительности. Подход с созданием отдельной goroutine для каждой задачи обеспечивает высокую скорость выполнения, однако сопровождается значительным увеличением потребления памяти и числа активных потоков выполнения [6].

Использование фиксированного Worker Pool позволило снизить потребление ресурсов при сохранении высокой производительности обработки задач. Динамический пул воркеров показал наилучшие результаты при переменной нагрузке за счёт адаптации количества воркеров к текущему числу задач.

В ходе исследования также установлено, что оптимальное количество воркеров зависит от характера нагрузки. Для вычислительных задач наилучшие результаты достигаются при количестве воркеров, близком к числу доступных процессорных ядер. Для задач, связанных с ожиданием операций ввода-вывода, увеличение числа воркеров позволяет повысить общую производительность системы [7].

Практическая значимость работы заключается в возможности применения полученных результатов при проектировании высоконагруженных программных систем и сервисов, реализованных на языке Go. Полученные результаты могут быть использованы для выбора оптимальной стратегии организации конкурентной обработки задач.

ЛИТЕРАТУРА

1. Donovan A. A., Kernighan B. W. The Go Programming Language. – Boston: Addison-Wesley Professional, 2015. – 380 p.

2. Cox-Buday K. Concurrency in Go: Tools and Techniques for Developers. – Sebastopol: O'Reilly Media, 2017. – 302 p.
3. Pike R. Go Concurrency Patterns // Google I/O Conference. – 2012. – [Электронный ресурс]. Режим доступа: <https://go.dev/talks/2012/concurrency.slide/> (дата обращения: 16.03.2026).
4. The Go Authors. Go Documentation. – [Электронный ресурс]. Режим доступа: <https://go.dev/doc/> (дата обращения: 16.03.2026).
5. The Go Authors. Effective Go. – [Электронный ресурс]. Режим доступа: https://go.dev/doc/effective_go (дата обращения: 16.03.2026).
6. Brown A. High Performance Go: Optimizing Go Programs for Speed and Efficiency. – Birmingham: Packt Publishing, 2023. – 356 p.
7. Go Optimization Guide. – [Электронный ресурс]. Режим доступа: <https://gopherf.dev> (дата обращения: 16.03.2026).

УДК 004.415.2

Н.С. Курташ (4 курс бакалавриата),
С.А. Молодяков, д.т.н., профессор

ПРОГРАММНЫЕ СЕРВИСЫ ДЛЯ АВТОМАТИЗИРОВАННОЙ ВЕРИФИКАЦИИ СИСТЕМ КОМПЬЮТЕРНОГО ЗРЕНИЯ С ИСПОЛЬЗОВАНИЕМ СИМУЛЯЦИОННЫХ ДАННЫХ

Развитие технологий беспилотного движения железнодорожного транспорта требует создания систем технического зрения, способных в реальном времени обнаруживать препятствия, распознавать элементы ЖД инфраструктуры и обеспечивать надежную навигацию. Блок обнаружения препятствий (БОП), разрабатываемый в АО «НИИАС», использующий камеры, лидары и вычислительные модули с нейронными сетями, должен соответствовать требованиям безопасности уровня SIL 3-4, что подразумевает вероятность опасного отказа менее 10^{-7} на час работы [1]. Натуральные испытания системы требуют много времени и имеют высокую стоимость. Также они не позволяют покрыть все граничные случаи.

Целью работы является разработка распределенной системы автоматизированной валидации - платформы Test Stage, обеспечивающей полный цикл тестирования систем компьютерного зрения на синтетических данных от железнодорожного симулятора.

Анализ готовых решений показал, что похожие системы из разных индустрий (CARLA [2], AirSim, LGSVL) не учитывают специфику железнодорожной инфраструктуры (геометрию пути, стрелочные переводы, элементы железнодорожной сигнализации), а готовый ЖД симулятор AutomatedTrain (Siemens, Deutsche Bahn) - проприетарный и не раскрывает технической реализации [3]. Данная работа посвящена разработке программных компонентов, обеспечивающих интеграцию блока обнаружения препятствий с собственным железнодорожным симулятором и организацию процесса автоматизированной валидации сервисов блока обнаружения препятствий.

В рамках работы разработана микросервисная архитектура с event-driven взаимодействием через Apache Kafka (рисунок 1). Система разделена на слабо связанные компоненты: симулятор генерирует эталонные данные (ground truth) и передает их по TCP компоненту Data Converter; Data Converter конвертирует данные из Protobuf в формат ZCM и передает объекты сцены в тестируемую систему БОП по UDP Multicast используя ZeroCM, одновременно с этим публикует полученные эталонные данные в ground-truth топик Kafka; БОП выполняет детекцию и возвращает результаты (predictions) по ZCM-каналу; Data Converter сериализует распознанные данные объектов в Protobuf и публикует их в prediction-topic. Сервис Metric Collector подписывается на оба топика, синхронизирует события по временным меткам и в реальном времени вычисляет метрики качества детекции: Precision, Recall, IoU, mAP, ошибки измерения расстояния и другие.

Выбор технологий обусловлен требованиями каждого участка, а также необходимостью интеграции разнородных сервисов, реализованных на различных языках программирования

(C#, C++) и использующих различные форматы данных. Apache Kafka [4] обеспечивает персистентность данных, что позволяет получать оперативные метрики тестируемой системы в режиме реального времени, а после теста перечитывать лог и генерировать подробный отчет по тестовому сценарию с увеличенным числом рассчитываемых показателей. Protocol Buffers выбран основным форматом сериализации, потому что он работает и с компонентами на C++ и с Go и с симулятором на C#. Кроме того, бинарное кодирование в 3-10 раз компактнее JSON при скорости десериализации 500-1000 МБ/с в C++. ZCM (Zero Communications and Marshalling) используется для передачи данных к БОП по UDP Multicast с задержкой менее 1 мс в локальной сети. Критичные к задержкам компоненты (Data Converter) реализованы на C++20, а сервисы потоковой обработки - на Go с использованием горутин для параллельной обработки разделов Kafka.

Система хранения данных оптимизирована под разные типы нагрузки: Redis применяется для оперативного доступа к промежуточным данным, VictoriaMetrics - для хранения и обработки временных рядов метрик, а ClickHouse - для выполнения аналитических запросов с агрегацией больших объемов данных. По результатам нагрузочного тестирования измеренная инфраструктурная задержка составляет 15-40 мс. Система устойчиво обрабатывает поток данных 10 Гбит/с от конфигурации симулятора с двумя камерами (3840×2160, 15 кадр/с) и двумя лидарами (300 000 точек/скан, 10 Гц).

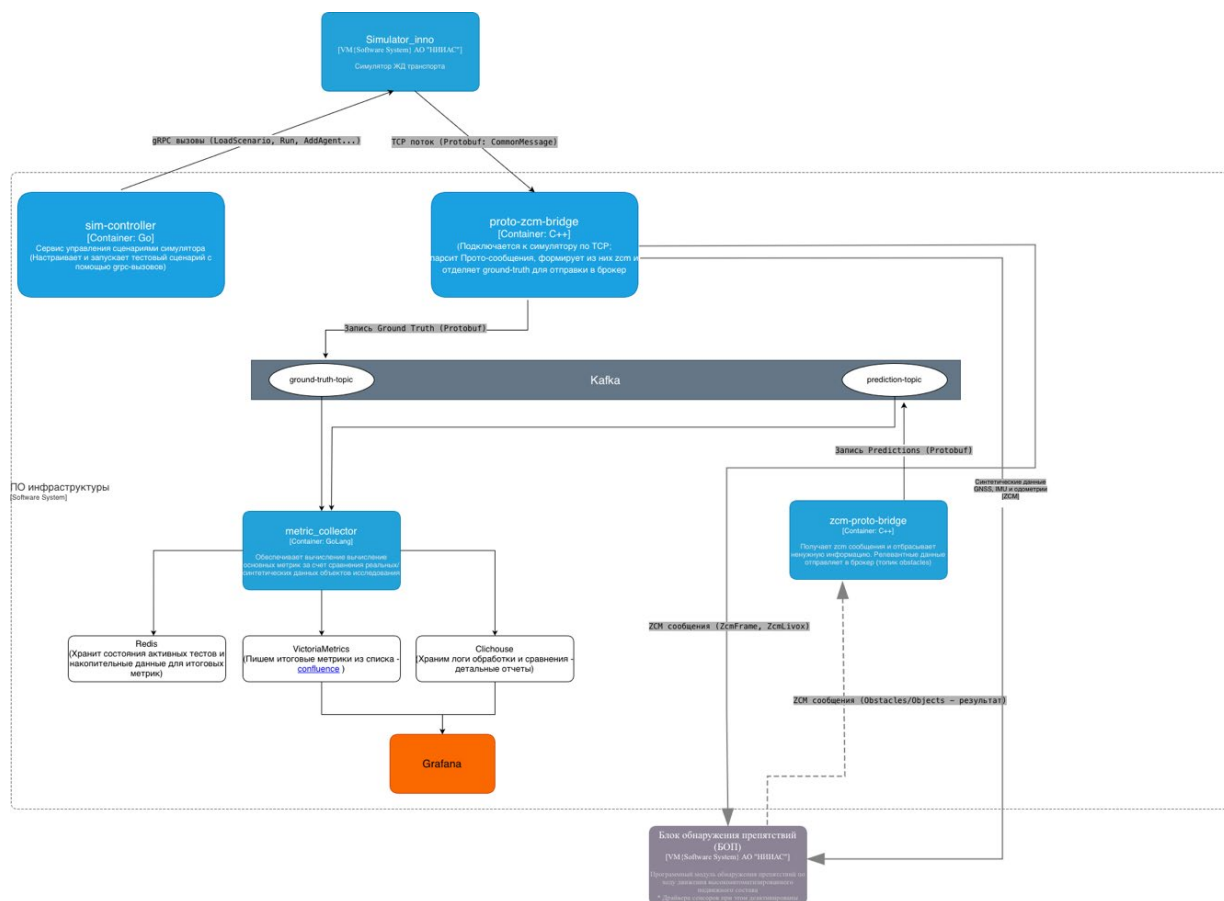


Рисунок 1 – Архитектура системы Test Stage

Горизонтальная масштабируемость обеспечивается stateless архитектурой сервисов: Data Converter и Metric Collector не хранят состояния, что позволяет добавлять экземпляры без координации, а Kafka автоматически перебалансирует разделы при изменении числа потребителей.

Разработанная система внедрена в промышленную эксплуатацию в АО «НИИАС» в рамках проекта «Цифровой Полигон» и обеспечивает полный цикл TestOps: генерацию синтетических сценариев, автоматизированное тестирование БОП, отчеты о результатах и интеграцию с CI/CD-пайплайнами. Дальнейшее развитие системы построено на оптимизации

алгоритмов синхронизации по временным меткам, расширении набора вычисляемых метрик и масштабировании на конфигурации с большим числом сенсоров.

ЛИТЕРАТУРА

1. EN 50129:2018. Railway applications - Communication, signalling and processing systems - Safety-related electronic systems for signalling. - Brussels : CENELEC, 2018. - 194 p.
2. Dosovitskiy A., Ros G., Codevilla F. et al. CARLA: An open urban driving simulator // Proceedings of the 1st Conference on Robot Learning (CoRL). - 2017. - P. 1-16.
3. Groh K., Schüller T., Frey F. et al. Towards a Scenario-Based Assessment Framework for Automated Train Operations // IEEE Intelligent Vehicles Symposium (IV). - 2023. - P. 1-6. - DOI 10.1109/IV55152.2023.10186701.
4. Kreps J. I heart logs: Event data, stream processing, and data integration. - Sebastopol : O'Reilly Media, 2014. - 60 p.
5. Narkhede N., Shapira G., Palino T. Kafka: The Definitive Guide. - 2nd ed. - Sebastopol : O'Reilly Media, 2024. - 478 p.
6. ClickHouse Documentation [Электронный ресурс]. - URL: <https://clickhouse.com/docs> (дата обращения: 01.03.2026).

УДК 004.415

О.А. Мурзич (5 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ И ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ CONSTRAINT-BASED СРЕДСТВ ТЕСТИРОВАНИЯ НА PYTHON

В работе рассматривается метод тестирования программного обеспечения на основе ограничений (constraint-based testing, СВТ), при котором корректность поведения системы формализуется в виде набора ограничений на входные данные, выходные значения и промежуточные состояния [1]. *Целью* исследования является исследование возможностей и ограничений библиотек для СВТ в Python и их экспериментальное сравнение. Решены следующие *задачи*: 1) анализ теоретических основ СВТ и его места среди смежных подходов к обеспечению качества программного обеспечения; 2) сравнение библиотек Typeguard, Beartype и Icontract; 3) разработка программного обеспечения для экспериментального сравнения указанных средств; 4) проведение вычислительного эксперимента; 5) анализ полученных результатов и оценка практической применимости библиотек.

СВТ рассматривается как формально обоснованный подход, в котором тестовые данные и механизмы определения корректного поведения тестируемого объекта (тест-оракулы) формируются на основе явно заданных ограничений на входы, выходы и промежуточные состояния [2]. В данной работе тест-оракул определяется как композиция проверок, реализуемых рассматриваемыми библиотеками, каждая из которых обеспечивает проверку отдельных аспектов контрактов (типов, предикатов, предусловий и постусловий). Typeguard реализует динамическую проверку аннотаций типов и может рассматриваться как средство проверки типовых контрактов. Beartype ориентирована на проверку аннотированных типов и пользовательских предикатов, интерпретируя типы как композицию ограничений. Icontract предоставляет декларативный механизм задания предусловий и постусловий.

Для экспериментов разработан стенд для сравнительного анализа перечисленных библиотек. В качестве системы под тестированием использовался набор коротких функций обработки текстовых данных, которые формально удовлетворяют типовым и контрактным ограничениям, но в экспериментальных наборах данных им намеренно добавлялись ошибки конкретных классов для проверки возможности их обнаружения. *Архитектура* пайплайна обработки данных представлена на рисунке 1. Код запуска экспериментов, входные и выходные данные выложены в репозиторий [3].

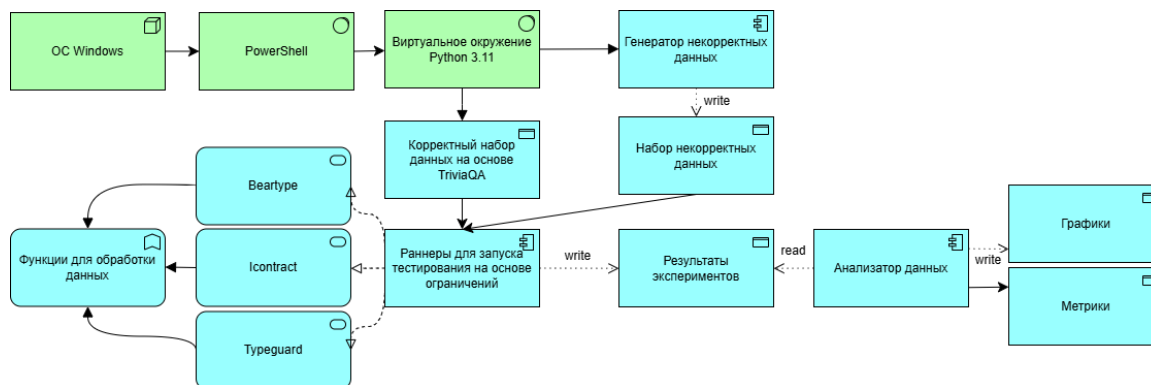


Рисунок 1 – Схема обработки данных

В качестве входных данных использовались JSON-объекты с полями вопроса, ответа и списка альтернативных ответов. Корректный набор данных основан на подвыборке набора данных TriviaQA [4], а ошибочный набор формировался через прямую генерацию конкретных дефектов посредством прямого внедрения классов этих дефектов, а также ошибок в данных при сохранении типовой корректности их структуры. Для каждой библиотеки автоматически подсчитывалось количество внесенных в данные и реально обнаруженных дефектов. Основной количественной метрикой сравнения являлась доля корректно обнаруженных дефектов для каждой библиотеки. Дополнительно измерялось распределение классов ошибок и время выполнения.

Эксперименты проводились на локальной машине (Intel Core i5-10300H 2.50 ГГц, 16 Гб RAM, NVIDIA GTX 1650, Windows 10 x64). Результаты экспериментов в таблице 1 показывают, что для каждой библиотеки наблюдается 100% обнаружение внесенных дефектов (30 000 на одну библиотеку, всего 90 000 дефектов), при этом распределение по классам ошибок существенно различается между библиотеками, что подтверждает их функциональную специализацию. Совпадение количества TypeCheckError и BeartypeCallHintParamViolation у Beartype и Typeguard связано с тем, что каждая ошибка приводит к срабатыванию обеих библиотек одновременно, если она касается типового несоответствия или параметров функции.

Таблица 1 – Сводная таблица результатов экспериментов

Библиотека	BeartypeCallHintParamViolation	ViolationError	TypeCheckError	Exception
Beartype	21279	0	0	8721
Icontract	0	0	17105	12895
Typeguard	0	21279	0	8721

Среднее время выполнения функций представлено на рисунке 2. Библиотека Icontract показывает самое высокое время выполнения, так как как проверяет предусловия и постусловия с использованием декораторов и дополнительных проверок, что добавляет накладные вычисления на каждом вызове функции.

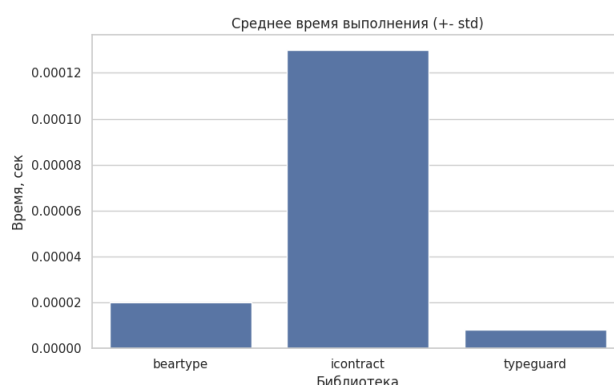


Рисунок 2 – Среднее время выполнения тестов по библиотекам

Таким образом, Typeguard целесообразно использовать для базового контроля типов, Beartype – для проверки формы данных и простых предикатов, а Icontract – для задания предусловий и постусловий при наличии формализованных требований. Эксперимент показал, что библиотеки частично взаимозаменяемы, однако покрывают разные классы ошибок, а их комбинированное использование снижает риск пропуска дефектов. Разработанный стенд и артефакты эксперименты размещены в открытом репозитории [3].

ЛИТЕРАТУРА

1. Gil S., Capel M., Olea G. Automatic test cases generation from formal contracts // Information and Software Technology. – 2024. – Vol. 172. – P. 107467. – DOI 10.1016/j.infsof.2024.107467.
2. Fleischmann M., Fraser G., Seidl C., Zeller A. Constraint-Based Test Oracles for Program Analyzers // Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24). – New York: Association for Computing Machinery, 2024. – P. 344–355. – DOI 10.1145/3691620.3695035.
3. Murzich O., Parkhomenko V. Experimental Comparison of CBT testing tools // GitHub – URL: https://github.com/MurzichOA-z5130903-10301/CBT_conference – 2025
4. Joshi M., Choi E., Weld D., Zettlemoyer L. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP). – Copenhagen, 2017. – P. 1601–1611.

УДК 004.453

Д.А. Набережнов (4 курс бакалавриата),
О.А. Юсупова, ст. преподаватель

РАЗРАБОТКА ПЛАГИНА DATABASE USER FEDERATION ДЛЯ ИНТЕГРАЦИИ KEYCLOAK С СУЩЕСТВУЮЩИМИ БАЗАМИ ДАННЫХ ПОЛЬЗОВАТЕЛЕЙ

Современные корпоративные информационные системы включают большое количество разнородных сервисов и приложений, каждое из которых может использовать собственный механизм аутентификации и хранения учетных данных пользователей. Подобная фрагментация усложняет управление доступом и повышает риски нарушения безопасности. Для решения данной проблемы применяются системы управления идентификацией и доступом (Identity and Access Management, IAM), обеспечивающие централизованную аутентификацию и механизм единой точки входа (Single Sign-On, SSO).

Одним из распространенных решений в области IAM является платформа Keycloak[1], предоставляющая развитые средства управления пользователями, аутентификацией и авторизацией. Однако при внедрении Keycloak в существующую инфраструктуру организаций возникает проблема интеграции с системами, в которых учетные записи пользователей уже хранятся во внешних реляционных базах данных и тесно связаны с бизнес-логикой приложений. Миграция таких данных в хранилище пользователей Keycloak либо внедрение дополнительных каталогов пользователей зачастую требует значительных трудозатрат и может повлечь серьезные изменения в существующих программных системах.

Целью данной работы является разработка плагина Database User Federation[2] для Keycloak, позволяющего интегрировать платформу IAM с существующими базами данных пользователей без необходимости изменения логики работы сторонних приложений и дополнительной разработки с их стороны. Реализация такого механизма позволяет существенно упростить внедрение SSO в существующие информационные системы.

Для достижения поставленной цели были решены следующие задачи: проведено исследование архитектуры платформы Keycloak и механизмов расширения её функциональности; выполнен анализ существующих способов интеграции внешних источников пользователей, включая LDAP User Federation и использование Admin REST API; предложен подход к интеграции Keycloak с существующими реляционными базами данных

пользователей на основе реализации собственного поставщика пользователей; реализован программный плагин Database User Federation; продемонстрирована работоспособность разработанного решения на примере подключения внешней базы данных пользователей.

Анализ существующих решений показал, что использование LDAP User Federation требует развертывания отдельного каталога пользователей, миграцию пользователей в данный каталог и последующее сопровождение. Интеграция через Admin REST API, в свою очередь, приводит к разработке сложной логики управления пользователями через REST запросы от сторонней системы.

Предлагаемый подход основан на использовании механизма Service Provider Interface (SPI), предоставляемого платформой Keycloak для расширения функциональности системы. Разработанный плагин реализует собственного поставщика пользователей, обеспечивающий прямое взаимодействие Keycloak с внешней реляционной базой данных. При аутентификации пользователя Keycloak выполняет обращение к базе данных через реализованного поставщика, получая необходимые учетные данные и атрибуты пользователя.

Реализация решения выполнена на языке программирования Java с использованием системы сборки Apache Maven. Для доступа к внешним базам данных применяется интерфейс JDBC[5], что обеспечивает возможность работы с различными системами управления базами данных. Для повышения производительности используется пул соединений HikariCP[3]. Разработка и тестирование проводились в контейнеризированной среде с использованием Docker[4].

В результате выполнения работы был разработан плагин Database User Federation для платформы Keycloak. Проведенные испытания демонстрационной версии подтвердили возможность аутентификации пользователей, данные которых хранятся во внешней базе данных. Использование разработанного решения позволяет снизить трудозатраты на интеграцию существующих информационных систем с платформой SSO.

ЛИТЕРАТУРА

1. Keycloak Documentation. URL: <https://www.keycloak.org/documentation>
2. Keycloak Server Developer Guide. URL: https://www.keycloak.org/docs/latest/server_development/index.html#_user-storage-spi
3. Wooldridge B. HikariCP. URL: <https://github.com/brettwooldridge/HikariCP>
4. James Turnbull. “The Docker Book” Version v17.03.0 – 2017. – 400pp.
5. Oracle. Java Database Connectivity (JDBC) A. URL: <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

УДК 004.042

Д.Д. Недорезов (4 курс бакалавриата),
А.В. Самочадин, к. т. н., доцент,
Т.Н. Самочадина, ст. преподаватель

РАЗРАБОТКА СРЕДСТВ ГЕНЕРАЦИИ АВТОМАТИЗИРОВАННЫХ ТЕСТОВ ДЛЯ РАЗЛИЧНЫХ ПЛАТФОРМ

Рост сложности программных систем и разнородность платформ увеличивают стоимость регрессионного тестирования. Существующие инструменты либо автоматизируют только запуск тестов (JUnit, Selenium), либо генерируют модульные тесты, слабо связанные с бизнес-сценариями. Современные ИИ-решения часто работают как «черные ящики», не учитывая структуру кодовой базы и корпоративные стандарты.

Целью данной работы является снижение трудозатрат на разработку и сопровождение автотестов путем создания системы на основе больших языковых моделей (LLM), генерирующей исполняемый код тестов по текстовому описанию сценариев с учетом

контекста конкретного проекта. Для достижения цели решаются задачи анализа предметной области, разработки методологии генерации, реализации программного прототипа.

Предлагаемый подход базируется на архитектуре RAG (Retrieval-Augmented Generation). Система индексирует кодовую базу проекта, включая существующие тесты и Page Object-ы [2], разбивая их на фрагменты (chunking) и сохраняя векторные представления в индексе (FAISS). При поступлении текстового описания сценария (например, пользовательской истории) выполняется поиск семантически близких фрагментов кода. Найденный контекст вместе с запросом подставляется в специализированный шаблон промпта (например, для XCTest [1]), который направляется в LLM. Модуль генерации получает ответ модели, проводит пост-обработку и возвращает пользователю готовый код автотеста, стилистически соответствующий стандартам проекта.

Разработанная система имеет модульную клиент-серверную архитектуру, обеспечивающую гибкость и масштабируемость. Пользовательский интерфейс (GUI) реализован на TypeScript и включает два основных функциональных блока: Генератор и Поиск. Взаимодействие фронтенда с бэкендом осуществляется через слой API [4].

Бэкенд, написанный на Python [5], состоит из двух ключевых подсистем: Core и RAG Indexer. Core реализует основной пайплайн генерации:

- Pipeline координирует последовательность операций: получение запроса от GUI, обращение к RAG-подсистеме за контекстом, формирование промпта, вызов LLM и пост-обработку результата.
- Builder отвечает за конструирование промптов на основе шаблонов (из директории Prompts) и найденного контекста.
- API предоставляет REST-интерфейс для фронтенда и внешних сервисов (например, для интеграции с GitLab CI/CD).

RAG Indexer [3] обеспечивает семантический поиск по кодовой базе проекта и существующим тестам.

- Indexer выполняет разбиение исходного кода на фрагменты (chunking), вычисляет их векторные представления (эмбединги) и сохраняет в индекс (FAISS).
- Searcher по текстовому запросу пользователя находит наиболее релевантные фрагменты в индексе.
- Scheduler автоматизирует периодическое обновление индекса при изменениях в репозитории (например, по триггерам из GitLab).

Система поддерживает интеграцию с несколькими большими языковыми моделями через унифицированный адаптер.

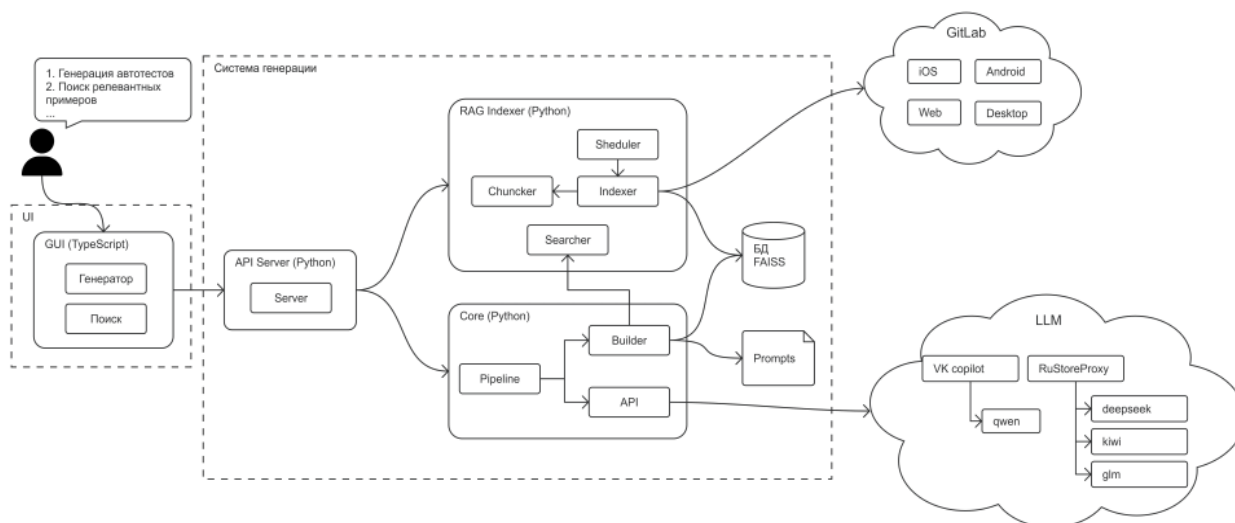


Рисунок 1 – Архитектура системы генерации

Таким образом, предложенная архитектура обеспечивает тесную связь между пользовательскими сценариями, контекстом проекта и возможностями современных LLM, что позволяет получать готовые к исполнению тесты с минимальными затратами на ручную доработку.

ЛИТЕРАТУРА

1. G. W. Bieleza, «Comparison of tools for creating and conducting automated tests», [Электронный ресурс]. Режим доступа: <https://ph.pollub.pl/index.php/jcsi/article/view/3804>. (Дата обращения: 17.01.26).
2. Паттерны проектирования в автоматизации тестирования. [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/jugru/articles/338836/> (дата обращения: 18.02.2026).
3. An Introduction to RAG [Электронный ресурс] Режим доступа: <https://io-technology-io-technology.vercel.app/articles/introduction-to-rag> (Дата обращения: 17.02.2026)
4. FastAPI Documentation [Электронный ресурс] Режим доступа: <https://fastapi.tiangolo.com/> (Дата обращения: 11.03.2026)
5. Python Programming Language. [Электронный ресурс] Режим доступа: <https://www.python.org/> (Дата обращения: 21.01.2026)

УДК 004.4

К.В. Новиковский (аспирант),
А.П. Преображенский, д.т.н., профессор

МЕТОД ИНТЕЛЛЕКТУАЛЬНОГО РАСПРЕДЕЛЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ В ПРОГРАММНЫХ СИСТЕМАХ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

Современные программные системы, ориентированные на обработку больших объёмов данных и выполнение ресурсоёмких вычислений, в настоящее время всё чаще функционируют в условиях распределённой вычислительной среды [1]. В этом контексте эффективность функционирования подобных систем в значительной степени определяется качеством алгоритмов распределения вычислительных задач между доступными вычислительными ресурсами [2]. При этом вычислительные ресурсы распределённых систем, как правило, характеризуются выраженной неоднородностью по своим аппаратным и программным параметрам, а интенсивность поступления задач и их вычислительная сложность могут существенно изменяться во времени. Совокупность указанных факторов, как следствие, приводит к снижению эффективности традиционных статических и эвристических методов планирования, не учитывающих динамику состояния вычислительной среды и изменения характеристик нагрузки [1].

В условиях динамически изменяющейся нагрузки и неоднородности вычислительных ресурсов особую актуальность приобретает задача разработки адаптивных методов распределения вычислительных задач, способных учитывать текущее состояние системы и корректировать принимаемые решения в процессе функционирования [2]. При этом дополнительную сложность представляет необходимость оперативного принятия решений в условиях неполной и зашумлённой информации о состоянии вычислительных узлов. В связи с этим одним из перспективных направлений повышения эффективности алгоритмов распределения является применение методов машинного обучения, позволяющих аппроксимировать сложные нелинейные зависимости между характеристиками нагрузки и эффективностью выполнения вычислительных задач [3].

Исходя из изложенного, целью данной работы является разработка метода интеллектуального распределения вычислительных задач в программных системах на основе применения методов машинного обучения для оценки состояния вычислительной среды и поддержки принятия решений в алгоритме планирования. Для достижения поставленной цели в рамках исследования решаются задачи формализации процесса распределения вычислительных задач, определения информативных характеристик состояния

вычислительных ресурсов, а также построения нейросетевой модели, обеспечивающей прогнозирование эффективности назначения задач на вычислительные узлы [3].

В работе рассматривается распределённая вычислительная система, состоящая из множества вычислительных узлов, каждый из которых характеризуется вектором состояния, включающим параметры текущей загрузки и доступных вычислительных ресурсов. Входной поток вычислительных задач описывается совокупностью характеристик, определяющих их вычислительную сложность и требования к используемым ресурсам [1]. В этом случае задача интеллектуального распределения вычислительных задач формулируется как задача выбора такого соответствия между задачами и вычислительными узлами, при котором достигается минимизация средней задержки выполнения задач при одновременном обеспечении равномерной загрузки вычислительных ресурсов системы [2].

Для решения указанной задачи в рамках данной работы предлагается метод интеллектуального распределения вычислительных задач, основанный на использовании многослойной персептронной нейронной сети (MLP) в качестве компонента алгоритма планирования [3]. Нейросетевая модель применяется для аппроксимации зависимости между текущим состоянием вычислительной среды и ожидаемой эффективностью выполнения задач на отдельных вычислительных узлах. Входными параметрами модели выступают характеристики нагрузки и состояния ресурсов, такие как степень загрузки процессора, объём доступной оперативной памяти и интенсивность входного потока задач, тогда как выходным параметром является числовая оценка целесообразности назначения задачи соответствующему вычислительному узлу [4].

Для оценки эффективности предложенного метода было проведено моделирование процесса распределения вычислительных задач в распределённой программной системе с динамически изменяющейся нагрузкой. В рамках моделирования осуществлялось сравнение работы предлагаемого алгоритма интеллектуального планирования с использованием нейросетевой модели MLP и базового эвристического алгоритма распределения задач [2]. В качестве критериев оценки рассматривались показатели средней задержки выполнения задач, степени равномерности загрузки вычислительных ресурсов, а также устойчивости алгоритма к изменению интенсивности входного потока задач. Полученные результаты моделирования показали, что применение нейросетевого компонента позволяет повысить адаптивность алгоритма распределения и обеспечить более эффективное использование вычислительных ресурсов по сравнению с базовым подходом.

Таким образом, полученные результаты подтверждают целесообразность применения методов машинного обучения в задачах интеллектуального распределения вычислительных задач в программных системах [3, 5]. Предложенный метод может быть использован при проектировании и модернизации распределённых вычислительных платформ, ориентированных на обработку потоковых и ресурсоёмких задач. В дальнейшем представляется целесообразным развитие предложенного подхода за счёт расширения набора входных параметров нейросетевой модели, а также проведение экспериментальных исследований в условиях реальных вычислительных сред.

ЛИТЕРАТУРА

1. Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. – Boston: Pearson Education, 2017. – 686 p.
2. Кузнецов С. Д., Семёнов А. В. Методы распределения вычислительных ресурсов в распределённых вычислительных системах // Программирование. – 2018. – № 6. – С. 45–58.
3. Goodfellow I., Bengio Y., Courville A. Deep Learning. – Cambridge: MIT Press, 2016. – 775 p.
4. Афанасьев А. Н., Пильщиков А. И. Интеллектуальные методы управления вычислительными процессами в распределённых системах // Вестник компьютерных и информационных технологий. – 2020. – № 4. – С. 12–21.
5. Mao H., Alizadeh M., Menache I. Resource management with deep reinforcement learning // Proceedings of the 15th ACM Workshop on Hot Topics in Networks. – Atlanta, USA, 2016. – P. 50–56.

СОПОСТАВЛЕНИЕ ФРЕЙМВОРКОВ КОНТРОЛЯ КАЧЕСТВА ДАННЫХ И МОНИТОРИНГА ПРИ ПРОГНОЗИРОВАНИИ ВРЕМЕННЫХ РЯДОВ

При эксплуатации систем прогнозирования деградация качества прогнозов часто обусловлена дефектами входных данных и изменением статистических свойств потоков данных (concept drift). В [1, 2] эти факторы рассматриваются как одни из наиболее распространённых источников сбоев и ухудшения метрик в промышленной эксплуатации. *Целью работы* является разработка и апробация технологии тестирования пайплайна прогнозирования временных рядов с использованием модели Prophet и сопоставление инструментов контроля качества данных (обнаружение пропусков, дубликатов и нарушений бизнес-ограничений по диапазонам значений) в Great Expectations (GX) и мониторинга дрейфа на основе Evidently, Alibi Detect и NannyML.

Экспериментальная схема включает базовую модель прогнозирования Prophet и набор сценариев контролируемого внесения дефектов в тестовый интервал. Для каждого сценария формируется отчёт каждого фреймворка и рассчитываются показатели обнаружения относительно известной разметки внесённых дефектов (ground truth). Код экспериментов и артефакты отчётов доступны в репозитории [3]. Схема эксперимента приведена на рисунке 1.

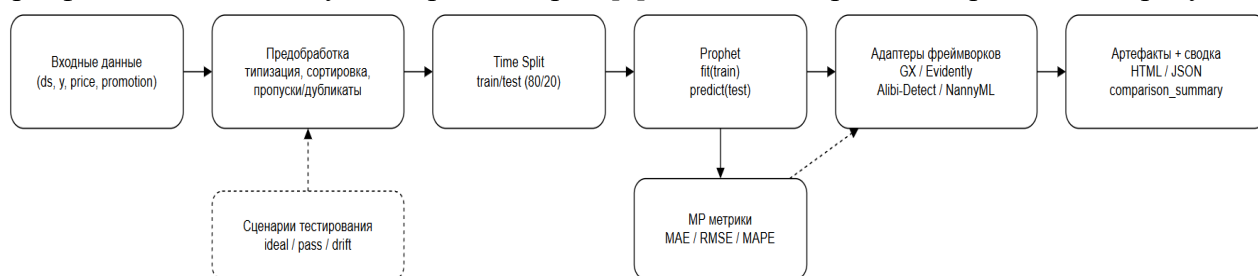


Рисунок 1 – Схема этапов экспериментального конвейера

Исходные данные – синтетический временной ряд спроса (y) с двумя внешними факторами: $price$ (числовой) и $promotion$ (бинарный). Он содержит сезонную компоненту и стохастический шум, при этом систематический тренд и плановые сдвиги распределения в базовом сценарии отсутствуют. Данные упорядочиваются по ds и делятся по времени на обучающую и тестовую части в пропорции 80/20. Эксперименты выполнены на ПК (Windows 10, Python 3.10.11, AMD Ryzen 5 2500U 2.00 GHz, RAM 16 ГБ). Есть 3 сценария: *ideal* (без внесённых дефектов), *pass* (data quality defects: пропуски и дубликаты), *dr* (drift/outliers: дрейф, выбросы и выход за допустимый диапазон). Для *dr* также задаётся правило, что $price$ из $[80; 120]$, а выбросы внедряются в y и $price$.

Для каждого типа дефекта на тестовом интервале задаётся эталонное число внедрённых нарушений E (внедрено). Инструмент контроля формирует набор срабатываний; их общее количество S (сработало). Для типов, где возможно сопоставление с эталоном (пропуски, дубликаты, выбросы, выход за диапазон; для дрейфа — по перечню контролируемых признаков), выделяется число корректных срабатываний KKK (корректно). Полнота обнаружения R рассчитывается как отношение K и E при положительном E . Если E равно 0, то S интерпретируется как число дополнительных (ложноположительных) срабатываний.

В таблицах 1 и 2 приведены результаты экспериментов. Нагрузка (время выполнения и пиковое потребление памяти) на наборах малый (*small*) и большой (*big*) представлены в таблице 3. *Small* содержит 1000 наблюдений, а *big* – 40 000; они имеют одинаковую схему столбцов: ds , y , $price$, $promotion$ и используются только в нагрузочном эксперименте и формируются без внесения дефектов.

Таблица 1 – Количество внесённых дефектов на тестовом интервале (E_test)

Сценарий	Объём тест. интервала, n_test	Пропуски у, E	Пропуски price, E	Дубликаты ds, E	Дрейф признаков (число признаков), E	Выбросы по у, E	Выбросы по price, E	Нарушение диапазона price [80;120] E
ideal	4000	0	0	0	0	0	0	0
pass	4120	85	207	204	0	0	0	0
dr	4000	0	0	0	3	258	144	4000

Таблица 2 – Результаты обнаружения дефектов на тестовой части

Сценарий	Тип проверки	Число внедрённых дефектов, E_test	GX	Evidently	Alibi-Detect	NannyML
pass	Пропуски в у	85	85/85	н/д	н/д	н/д
	Пропуски в price	207	207/207	н/д	н/д	н/д
	Дубликаты ds	204	204/204	н/д	н/д	н/д
	Дрейф (признаки)	0	2/3	2/3	2/3	3/3
	Выбросы в у	0	н/д	6/6	6/6	3/3
	Выбросы в price	0	н/д	0	0	0/3
	Выход price за диапазон	0	0	н/д	н/д	н/д
dr	Дрейф (признаки)	3	3/3	3/3	3/3	3/3
	Выбросы в у	258	н/д	258/258	258/258	258/258
	Выбросы в price	144	н/д	144/144	144/144	144/144
	Выход price за диапазон	4000	4000/4000	н/д	н/д	н/д

В столбце E_test указано число дефектов соответствующего типа, внедрённых в тестовую часть данных. В ячейках по инструментам значения приведены в формате «обнаружено / внедрено». При E_test > 0 это количество корректно выявленных дефектов. При E_test = 0 приводится число дополнительных срабатываний инструмента. Обозначение «н/д» означает отсутствие соответствующей проверки в используемой конфигурации инструмента. Контрольный сценарий ideal не включён в таблицу из-за отсутствия дефектов и срабатываний.

Таблица 3 – Нагрузочные характеристики фреймворков на наборах small и big, n=3

Фреймворк	Время, с (small)	Память, МБ (small)	Время, с (big)	Память, МБ (big)
GX	0.214 ± 0.011	353 ± 0	0.306 ± 0.056	361.5 ± 0.5
Evidently	2.074 ± 0.065	374.5 ± 2.5	2.465 ± 0.339	389.5 ± 1.5
Alibi-Detect	0.029 ± 0.000	365.5 ± 0.5	0.077 ± 0.015	376.5 ± 0.5
NannyML	1.188 ± 0.043	377 ± 2	1.558 ± 0.099	389 ± 2

Таким образом, ideal не даёт дополнительных срабатываний, что подтверждает корректность конфигурации. Потребление памяти сопоставимо, а по времени выполнения лидирует Alibi-Detect: на small он быстрее Evidently примерно в 72 раз (2.074/0.029 с), на big — в 32 раза (2.465/0.077 с); относительно NannyML — примерно в 41 раз на small и в 20 раз на big.

ЛИТЕРАТУРА

1. Hinder F., Vaquet V., Hammer B. One or two things we know about concept drift — a survey on monitoring in evolving environments. Part A: detecting concept drift // *Frontiers in Artificial Intelligence*. 2024. Vol. 7. DOI: 10.3389/frai.2024.1330257.

2. Priestley M., O'Donnell F., Simperl E. A Survey of Data Quality Requirements That Matter in ML Development Pipelines // Journal of Data and Information Quality. 2023. Vol. 15, No. 2. P. 1–39.
3. Orzhekhovskiy A., Parkhomenko. V. ML testing frameworks comparison (репозиторий проекта) URL: <https://github.com/alexandor09/ml-testing-frameworks-comparison>.

УДК 004.453

Ю.Д. Петрова (4 курс бакалавриата),
А.О. Жаранова, ст. преподаватель

АВТОМАТИЗИРОВАННОЕ СРЕДСТВО ПЛАНИРОВАНИЯ АСТРОНОМИЧЕСКИХ НАБЛЮДЕНИЙ С ПРОГНОЗИРОВАНИЕМ ЛОКАЛЬНОЙ ВИДИМОСТИ ПОЛЯРНЫХ СИЯНИЙ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

Планирование астрономических наблюдений сопряжено со значительными временными и финансовыми затратами: подготовка оборудования и выезд на место наблюдений создают высокие риски потерь при срыве сеанса из-за неверной оценки условий [1]. В связи с этим возникает потребность в автоматизированном построении планов астрономических наблюдений. Анализ существующих инструментов – Sky Tonight, Star Walk 2, Astrospheric – показал, что они решают задачу частично: сервисы не интегрируют разнородные данные в единый план и не предоставляют локального прогноза видимости полярных сияний [2]. Таким образом, целью работы является разработка автоматизированного средства планирования астрономических наблюдений, включающего ML-модель локального прогноза видимости полярных сияний и серверную архитектуру, обеспечивающую масштабируемую агрегацию разнородных данных.

Ключевой технической задачей при построении плана является конкурентная агрегация данных из независимых источников: библиотек `astropy`, `astroplan` [3] и `astroquery` для запросов к JPL Horizons API [4] для расчёта эфемерид, Open-Meteo API [5] для метеопрогноза и NOAA SWPC [6] для геофизических индексов.

Для устранения последовательного накопления задержек применяется `asyncio.gather` на двух уровнях при конкурентной агрегации данных из независимых источников и параллельном формировании временных срезов плана. В качестве фреймворка выбран FastAPI с асинхронной моделью выполнения, что позволяет обрабатывать одновременные запросы без блокировки сервера.

Для снижения нагрузки на внешние сервисы применяется кэширование на базе Redis. Система обслуживает пользователей с различными координатами и датами наблюдений, что исключает прямое переиспользование готовых планов. Тем не менее, составляющие плана – эфемериды, погода, геофизические данные – воспроизводимы в пределах периодов их актуальности, что позволяет задать дифференцированные TTL: 24 часа для астрономических эфемерид (детерминированы в пределах суток), 60 минут для метеопрогноза (периодичность обновления Open-Meteo), 30 минут для геофизических индексов (периодичность публикации NOAA). Ключи формируются с округлением координат до $0,1^\circ$ (~ 11 км), что позволяет пользователям одного региона разделять кэш без значимой потери точности. Геофизические данные обновляются по паттерну Background Refresh, что исключает задержки при обращении API к кэшу.

Для прогнозирования локальной видимости полярных сияний разработана ML-модель классификации на основе градиентного бустинга (XGBoost). Целевая переменная – факт визуальной наблюдаемости сияния – получена из записей камеры всего неба обсерватории Hankasalmi (Финляндия) [7]. Датасет агрегирует три группы признаков: глобальные геофизические параметры из NOAA SWPC, локальные метеорологические условия из Open-Meteo и астрономический контекст.

Выбор градиентного бустинга обусловлен его доказанной эффективностью на структурированных табличных геофизических данных [8] и высокой интерпретируемостью

через важность признаков, что позволяет верифицировать физическую обоснованность модели. В отличие от существующих глобальных прогнозов, модель учитывает локальные факторы, непосредственно определяющие возможность визуального наблюдения, что позволяет повысить точность прогноза для конкретной точки наблюдения по сравнению с глобальными индексами геомагнитной активности.

Архитектура серверной части приведена на рисунке 1.

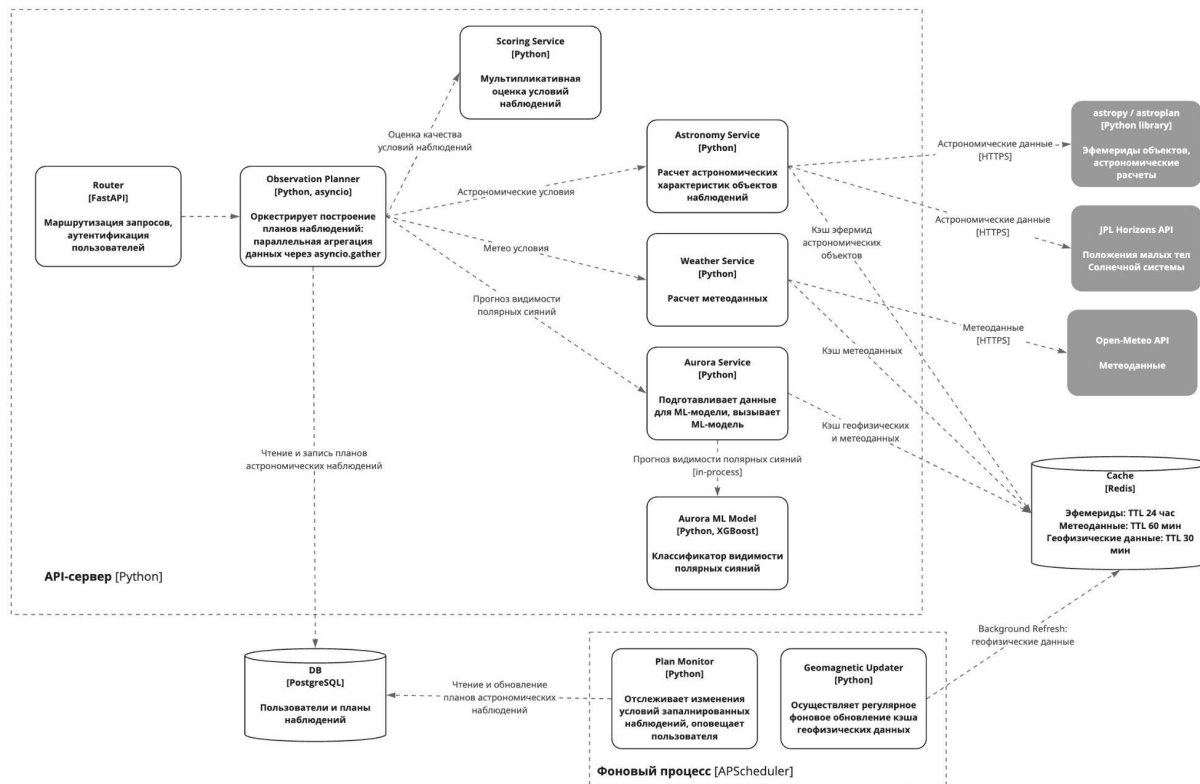


Рисунок 1 – Архитектура серверной части приложения

Система разделена на два независимых сервиса: асинхронный REST API на базе FastAPI и фоновый процесс на базе APScheduler – это обеспечивает независимое масштабирование API и позволяет разделить нерегулярные пользовательские запросы и строго периодическое обновление геофизических данных. Фоновый процесс выполняет три задачи: обновление геофизических данных в Redis каждые 30 минут, мониторинг активных планов с переоценкой условий и отправку email-уведомлений при их ухудшении.

Таким образом, реализованное средство автоматизирует построение плана наблюдений с учётом астрономических, метеорологических и геофизических условий.

ЛИТЕРАТУРА

1. Toyib M., Pramesti G. A., Herachwati N. Enhancing Public Engagement and Science Communication Through Participatory Astronomy: Insights from the Surabaya Astronomy Club's Star Party Model // JCOM: Journal of Science Communication. – 2024. – Vol. 23, Iss. 07. – P. 55-70.
2. Mooney M. K. и др. Evaluating Auroral Forecasts Against Satellite Observations Under Different Levels of Geomagnetic Activity // Journal Of Geophysical Research: Space Physics. – 2024. – no. 8 (129).
3. Morris B. M. и др. astroplan: An Open Source Observation Planning Package in Python // The Astronomical Journal. – 2018. – Vol. 155. – No. 3. – P. 128.
4. JPL Horizons Queries. [Электронный ресурс] Режим доступа: <https://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html> (дата обращения: 15.03.2026).
5. Открытый метеорологический API Open-Meteo. [Электронный ресурс] Режим доступа: <https://open-meteo.com/> (дата обращения: 13.03.2026).

6. Национальное управление океанических и атмосферных исследований (NOAA). [Электронный ресурс] Режим доступа: <https://www.swpc.noaa.gov/> (дата обращения: 13.03.2026).
7. Сайт обсерватории Hankasalmi Observatory. [Электронный ресурс] Режим доступа: <https://murtoinen.jklsirius.fi/> (дата обращения: 13.03.2026).
8. Feng H. и др. A Kp-Driven Machine Learning Model Predicting the Ultraviolet Emission Auroral Oval // Journal of Geophysical Research: Machine Learning and Computation. – 2025. – Vol. 2. – No. 2.

УДК 004.453

Н.В. Плиски (4 курс бакалавриата),
К.А. Туральчук, к.т.н., доцент

РАЗРАБОТКА ИНСТРУМЕНТА АВТОМАТИЧЕСКОГО АНАЛИЗА АРХИТЕКТУРНЫХ ОСОБЕННОСТЕЙ ПРОГРАММНЫХ СИСТЕМ НА C#

Современные программные системы характеризуются высокой сложностью архитектуры и значительным объёмом исходного кода. В условиях активного развития программных продуктов особую актуальность приобретает задача автоматизированного анализа архитектурных характеристик программных систем, позволяющего выявлять потенциальные проблемы проектирования, архитектурные аномалии и нарушения принципов программной инженерии [1, 2].

Одним из перспективных подходов к решению данной задачи является применение метрик программной инженерии, характеризующих структурные, когнитивные и архитектурные свойства исходного кода [1, 6]. Такие метрики позволяют количественно оценить сложность классов, степень связности компонентов системы и уровень их внутренней согласованности [6]. Однако анализ большого количества метрик вручную является трудоёмким и не позволяет эффективно выявлять нетипичные архитектурные структуры.

Целью данной работы является разработка инструментального средства для автоматизированного анализа архитектуры программных систем на основе совокупности метрик программной инженерии и методов машинного обучения.

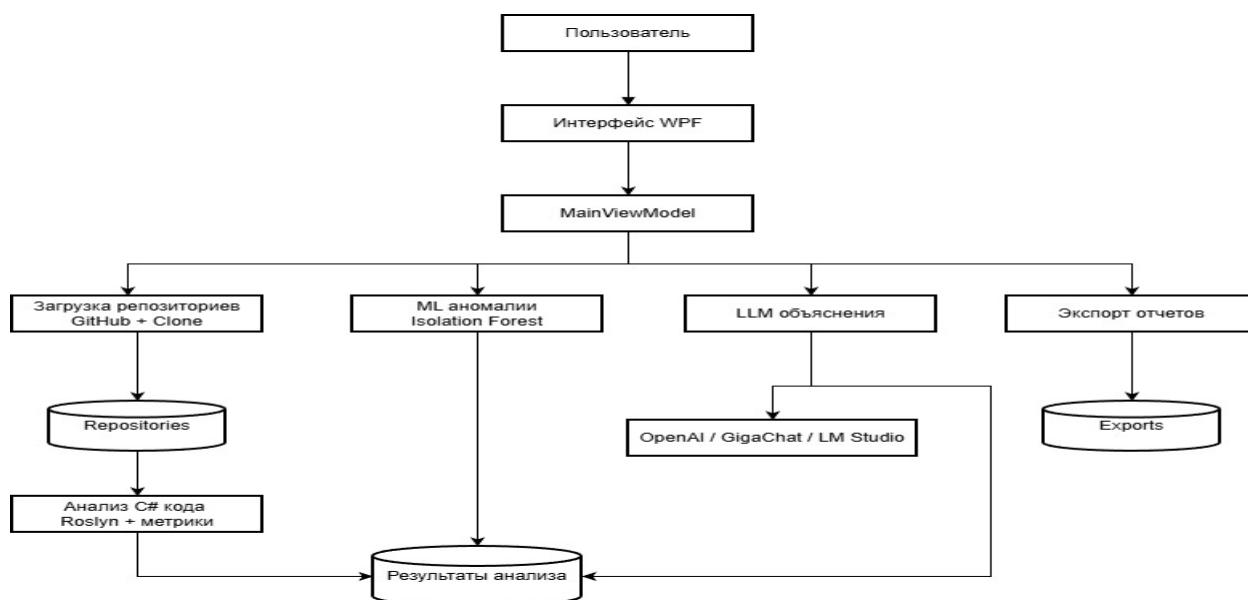


Рисунок 1 – Архитектура разрабатываемой системы анализа программного кода

Архитектура разрабатываемого программного средства представлена на рисунке 1. Система реализована по модульному принципу и включает несколько функциональных подсистем, обеспечивающих полный цикл анализа архитектуры программных проектов.

Взаимодействие пользователя с системой осуществляется через графический интерфейс, реализованный на платформе WPF. Управление логикой приложения выполняется в слое представления с использованием шаблона MVVM.

Подсистема загрузки данных обеспечивает получение исходного кода проектов из удалённых репозиторий. Для этого используется интеграция с сервисом GitHub, позволяющая автоматически клонировать выбранные репозитории и формировать локальный набор проектов для последующего анализа.

Анализ исходного кода основан на построении синтаксического дерева и семантической модели для каждого файла C#. На уровне классов рассчитываются базовые характеристики размера и сложности: Метрика LOC, число методов, полей, свойств, публичных методов, конструкторов, суммарная, максимальная и средняя цикломатическая сложность, максимальная глубина вложенности, число длинных методов, а также характеристики наиболее длинного и наиболее сложного метода [1].

Для оценки зацепления и связности используются метрики CBO, RFC, FRF, FanIn, FanOut, LCOM4, TCC, LCC, CAMC, SCOM, RCI, CBMC, IC BMC и др. [6]. Дополнительно формируются диаграммы доступа к полям и взаимодействия методов в нотации PlantUML.

Интегральная оценка качества класса вычисляется как взвешенная функция рисков, связанных с размером, сложностью, зацеплением, когезией и возможным нарушением принципа SRP [2]. На основе пороговых значений автоматически выявляются признаки God Class, Large Class, Long Methods, High Complexity, Deep Nesting, High Coupling, Feature Envy, Low Cohesion, Possible SRP Violation и Data Class [3]. Это позволяет ранжировать классы по проблемности и быстро выделять участки кода, требующие рефакторинга [7].

Для выявления нетипичных архитектурных структур используется модуль анализа аномалий, основанный на алгоритме машинного обучения Isolation Forest [5], относящемся к методам обнаружения выбросов в многомерных наборах данных [4]. Данный алгоритм позволяет выявлять объекты, существенно отличающиеся от основной совокупности наблюдений, без необходимости предварительной разметки обучающей выборки. В рамках разработанной системы каждый класс программного проекта представляется в виде вектора архитектурных метрик, характеризующих размер, сложность, связность и когезию. На основе анализа распределения этих признаков модель определяет степень отклонения класса от типичных архитектурных характеристик, что позволяет выявлять потенциальные архитектурные аномалии.

Для повышения интерпретируемости результатов анализа реализуется модуль генерации объяснений на основе больших языковых моделей (OpenAI, GigaChat или LM Studio). На основе рассчитанных метрик и результатов обнаружения аномалий формируются текстовые пояснения, описывающие возможные архитектурные проблемы и причины их возникновения.

Результаты анализа могут быть экспортированы пользователем в виде отчётов для дальнейшего анализа.

ЛИТЕРАТУРА

1. McConnell S. Code Complete. – Microsoft Press, 2004. – 960 p.
2. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Prentice Hall, 2017. – 432 p.
3. Fowler M. Refactoring: Improving the Design of Existing Code. – Addison-Wesley Professional, 2018. – 448 p.
4. Breunig M., Kriegel H., Ng R., Sander J. LOF: Identifying Density-Based Local Outliers // Proceedings of the ACM SIGMOD International Conference on Management of Data. – Dallas, USA, 2000. – P. 93–104.
5. Liu F., Ting K., Zhou Z. Isolation Forest // Proceedings of the IEEE International Conference on Data Mining (ICDM). – Pisa, Italy, 2008. – P. 413–422.
6. Spinellis D. Code Quality: The Open Source Perspective. – Addison-Wesley Professional, 2006. – 480 p.
7. Mens T., Tourwé T. A Survey of Software Refactoring // IEEE Transactions on Software Engineering. – 2004. – Vol. 30. – No 2. – P. 126–139.

РАЗРАБОТКА ИНСТРУМЕНТА УНИВЕРСАЛЬНОГО МОНИТОРИНГА ПРОИЗВОДИТЕЛЬНОСТИ ПРИЛОЖЕНИЙ

Современные программные системы часто функционируют в условиях высокой нагрузки и распределённой инфраструктуры, включающей множество сервисов и вычислительных узлов. В таких условиях обеспечение стабильной и эффективной работы программных продуктов требует постоянного контроля их производительности и состояния используемых ресурсов. Мониторинг позволяет выявлять перегрузку вычислительных ресурсов и проблемы производительности приложений, анализировать динамику использования ресурсов и своевременно реагировать на возникающие проблемы [4, 5].

Существующие системы мониторинга, такие как Prometheus, Zabbix и другие решения, предоставляют широкий набор инструментов для сбора и анализа метрик [2]. Однако многие из них ориентированы на определённые типы инфраструктуры или требуют сложной настройки и интеграции. В связи с этим возникает необходимость разработки универсального инструмента мониторинга, который позволит упростить процесс сбора и анализа метрик производительности приложений и обеспечит гибкость использования в различных программных средах.

Целью данной работы является разработка инструмента универсального мониторинга производительности приложений, обеспечивающего сбор, обработку и визуализацию ключевых системных и прикладных метрик.

Для достижения поставленной цели предполагается решение следующих задач:

1. Провести анализ существующих инструментов и методов мониторинга производительности программных систем.
2. Разработать архитектуру системы мониторинга, включающую модули сбора, обработки и хранения данных.
3. Реализовать агента мониторинга для сбора системных и прикладных метрик.
4. Разработать серверную часть системы для обработки поступающих данных и их хранения.
5. Создать веб-интерфейс для визуализации и анализа показателей производительности приложений.
6. Провести тестирование разработанного инструмента и оценить его эффективность.

Разрабатываемая система мониторинга будет построена по модульному принципу и включать несколько основных компонентов. Ключевым элементом системы является агент мониторинга, который устанавливается на контролируемом узле и выполняет сбор системных и прикладных метрик. К таким метрикам могут относиться загрузка центрального процессора, использование оперативной памяти, сетевые показатели, дисковая активность, а также показатели работы приложений, например время отклика и количество обрабатываемых запросов [4].

Агент мониторинга планируется реализовать на языке программирования Go [1], который обеспечивает высокую производительность, эффективную работу с сетевыми соединениями и удобные средства разработки распределённых систем. Собранные метрики передаются на серверную часть системы посредством сетевых запросов, где выполняется их обработка и сохранение в базе данных.

Серверная часть инструмента предназначена для централизованной обработки и хранения данных мониторинга. Она обеспечивает агрегацию метрик, их структурирование и подготовку к дальнейшему анализу. Для хранения данных может использоваться специализированная база данных временных рядов [2], позволяющая эффективно работать с большим объёмом метрик и анализировать их изменения во времени.

Для взаимодействия с пользователем будет разработан веб-интерфейс, предоставляющий средства визуализации показателей производительности. Интерфейс позволит отображать динамику метрик в виде графиков и диаграмм, а также анализировать состояние приложений в режиме реального времени. Использование средств визуализации данных позволит упростить выявление аномалий и определить факторы, влияющие на производительность системы [3].

Разрабатываемый инструмент предполагается использовать как в учебных проектах, так и при разработке программных систем различной сложности. Разрабатываемый инструмент может применяться для мониторинга различных типов приложений и инфраструктур.

Ожидаемым результатом работы является программный инструмент, позволяющий осуществлять универсальный мониторинг производительности приложений, обеспечивающий сбор и анализ метрик в реальном времени, а также удобную визуализацию данных для повышения эффективности анализа состояния программных систем.

ЛИТЕРАТУРА

1. Golang programming language. [Электронный ресурс] Режим доступа: <https://go.dev/> (дата обращения: 22.02.2026)
2. Prometheus Documentation. [Электронный ресурс] Режим доступа: <https://prometheus.io> (дата обращения: 15.01.2026)
3. OpenTelemetry платформа наблюдаемости распределённых систем. [Электронный ресурс]. Режим доступа: <https://opentelemetry.io> (дата обращения: 11.03.2026)
4. Кулямин В. В. Методы мониторинга и анализа поведения программных систем // Программирование. – 2017. – № 4. – С. 3–25.
5. Радченко Г. И. Распределенные системы мониторинга и управления производительностью в облачных средах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2021. – Т. 10, № 2. – С. 45-59.

УДК 004.415

А.А. Поливанова (2 курс магистратуры),
П.Д. Дробинцев, к.т.н., доцент

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПРОЦЕССА ВНЕСЕНИЯ ИЗМЕНЕНИЙ В ИТ-ИНФРАСТРУКТУРУ НА ОСНОВЕ АНАЛИТИЧЕСКИХ ЗАПИСОК

В современных промышленных и индустриальных компаниях, таких как группа компаний «Газпром нефть», функционируют масштабные и сложные ИТ-инфраструктуры, обеспечивающие работу ключевых бизнес-процессов. Эффективное управление изменениями в ИТ-среде требует точной, своевременной и формализованной документации [3, 6]. Одним из ключевых элементов данного процесса являются аналитические записки — документы, содержащие информацию об иницилируемых изменениях и их влиянии на инфраструктуру.

На текущий момент заполнение и корректировка аналитических записок выполняются преимущественно вручную, при заполнении блоков в шаблонах форматов word и excel. Однако шаблоны довольно обширны и содержат все возможные типы аналитических записок. Это приводит к таким проблемам, как высокой трудоемкости процесса, вероятности появления ошибок при вводе данных, необходимости повторных согласований, отсутствию единого подхода к оформлению документации [2, 5].

Автоматизация процесса подготовки аналитических записок позволит повысить точность вводимых данных, сократить время подготовки документации и снизить нагрузку на сотрудников [1]. В связи с этим актуальность исследования заключается в разработке подхода к оптимизации процесса изменений в ИТ-инфраструктуре путём создания решения, обеспечивающего автоматизированное заполнение аналитических записок и проверку их корректности [4].

Цель исследования — сокращение трудоёмкости процесса подготовки и корректировки аналитических записок за счёт создания веб-ресурса, который позволил бы сократить количество ошибок и неточностей к минимуму на стадии заполнения документа, а также обеспечил проверку корректности данных.

Для достижения поставленной цели необходимо решить задачи, перечисленные ниже.

1. Проанализировать существующие методы организации документооборота в промышленных компаниях.
2. Изучить современные технологии разработки веб-приложений для автоматизации внутренних процессов.
3. Исследовать подходы к проектированию интерфейсов (UI/UX), направленные на повышение удобства и точности заполнения документов.
4. Разработать архитектуру веб-ресурса для формирования и проверки аналитических записок.
5. Провести сравнительный анализ предложенного решения с традиционными способами оформления аналитических записок по показателям трудоёмкости и точности.

В соответствии с поставленными задачами было разработано веб-приложение AZ-Portal, реализующее автоматизированное заполнение и проверку аналитических записок. Архитектура системы построена на основе современных технологий веб-разработки [7]. Ключевым результатом стала реализация специализированных шаблонов. Система обеспечивает сквозной процесс создания заявки — от выбора шаблона, проверки нейросетью до генерации IaC, с возможностью сохранения черновиков и последующего восстановления незавершённых документов.

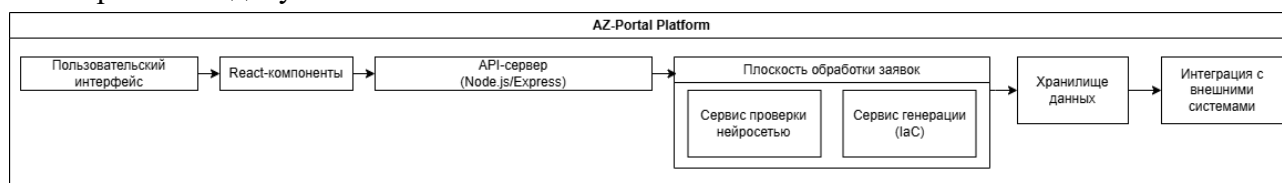


Рисунок 1 – Архитектура системы

Управление шаблонами заявок производится путём обращения к React-компонентам при помощи навигационных запросов на создание, отображение и обновление специализированных форм. Подход был реализован в программном средстве на языке TypeScript для строгой типизации данных. Платформа осуществляет обработку пользовательских запросов на создание аналитических записок, в которых описываются настройки и параметры всех компонентов инфраструктуры. К поддерживаемым типам заявок относятся: запрос доступов (МСЭ), создание объектов в AD, создание виртуальных машин (ВМ), а также создание комплексных запросов.

Поддерживаемые платформы развёртывания – любая современная веб-инфраструктура с поддержкой Node.js (включая облачные платформы и корпоративные серверы).

ЛИТЕРАТУРА

1. Sabutkevich A.M., Nikiforov I.V., Samochadin A.V. A method for modeling of individual agent behavior in the process-network paradigm of discrete-event simulation // Computing, Telecommunications and Control. 2025. Т. 18, № 1. С. 23–35. DOI: 10.18721/JCSTCS.18102
2. Verlaine B., Jureta I., Faulkner S. Aligning a Service Provisioning Model with the ITIL v3 Life Cycle // Journal of Service Science Research. – 2014. – Vol. 6. – No 2. – P. 165–190.
3. Axelos. ITIL Foundation: ITIL 4 Edition. – London: TSO (The Stationery Office), 2019. – 260 p.
4. Кузнецов С. Д. Основы современных информационных технологий. – Москва: Интернет-Университет Информационных Технологий, 2016. – 640 с.
5. Atlassian IT Service Management Guide. [Электронный ресурс] Режим доступа: <https://www.atlassian.com/itsm> (дата обращения: 09.03.2026)

6. COBIT 2019 Framework: Governance and Management Objectives. – Rolling Meadows: ISACA, 2019. – 320 p.
7. TypeScript is JavaScript with syntax for types. [Электронный ресурс] Режим доступа: <https://www.typescriptlang.org/> (дата обращения: 09.03.2021)

УДК 629.526:004.8

А.В. Примак (2 курс магистратуры),
К.В. Пшеничная, к.т.н., доцент,
В.А. Капонен, ст. преподаватель

МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧЕ ОПРЕДЕЛЕНИЯ ОСТАТОЧНОГО СОПРОТИВЛЕНИЯ МАЛОМЕРНОГО СУДНА

Расчёт сопротивления движению — ключевой этап проектирования судна. Результаты расчета влияют на выбор конструктивных решений, экономическую эффективность и безопасность эксплуатации судна. Общепринятой практикой является использование эмпирических формул для расчета остаточного сопротивления, например [1]. При этом в разных фирмах формулы отличаются и нет единой формулы для различных диапазонов изменения числа Фруда. Применение методов машинного обучения (МО) дополняет и расширяет возможности традиционных технических расчётов и является актуальным [2, 3].

Рассматриваемая задача относится к задачам регрессии [4]. Выходной параметр (отношение остаточного сопротивления к весовому водоизмещению, y) зависит от шести входных параметров: коэффициента продольной полноты (x_1), абсциссы центра величины (x_2), отношения ширины к осадке (x_3), отношения длины к ширине (x_4), отношения длины к объемному водоизмещению в степени $1/3$ (x_5), числа Фруда (x_6). Зависимость выходного параметра от входных является сложной и нелинейной.

Целью работы является сравнительный анализ трёх моделей машинного обучения (искусственная нейронная сеть, решающее дерево и случайный лес). Для обучения моделей МО использовались экспериментальные данные, полученные в результате проведения 308 полномасштабных экспериментов в лаборатории гидромеханики Делфтского технического университета [5].

Программная реализация моделей машинного обучения осуществлялась на языке Python с применением библиотеки Scikit-learn [6]. В качестве моделей были рассмотрены:

1. Полносвязная нейронная сеть прямого распространения (MLP). Проведен анализ 30 архитектур, лучшей архитектурой является архитектура $6 \times 20 \times 5 \times 5 \times 20 \times 1$ с функцией активации ReLU в скрытых слоях, обученная методом обратного распространения ошибки с оптимизатором Adam. Качество моделей оценивалось метриками MSE (среднеквадратическая ошибка) и R^2 (коэффициент детерминации). Для лучшего варианта нейронной сети $MSE = 0,0018$ и $R^2 = 0,9745$.
2. Решающее дерево (DT). Решающее дерево используется для получения результата на основе правил. Дерево представляет собой иерархическую систему бинарных вопросов о значениях входных переменных (признаков). Метод позволяет интерпретировать результаты. Полученные метрики оценки: $MSE = 0,0003$ и $R^2 = 0,9959$.
3. Случайный лес (Random forest). Случайный лес — ансамблевый метод, который объединяет множество решающих деревьев, что позволяет улучшить точность предсказаний. Лучшим вариантом был ансамбль из 150 деревьев с параметрами: $\text{min\_samples\_leaf} = 2$, $\text{min\_samples\_split} = 6$, $\text{max\_depth} = 15$, $\text{bootstrap} = \text{True}$, $\text{max\_samples} = 0.75$, $\text{max\_features} = \text{None}$. Качество полученной модели: $MSE = 0,0003$, $R^2 = 0,9962$.

На рисунке 1 приведены диаграммы рассеяния для исследуемых моделей машинного обучения.

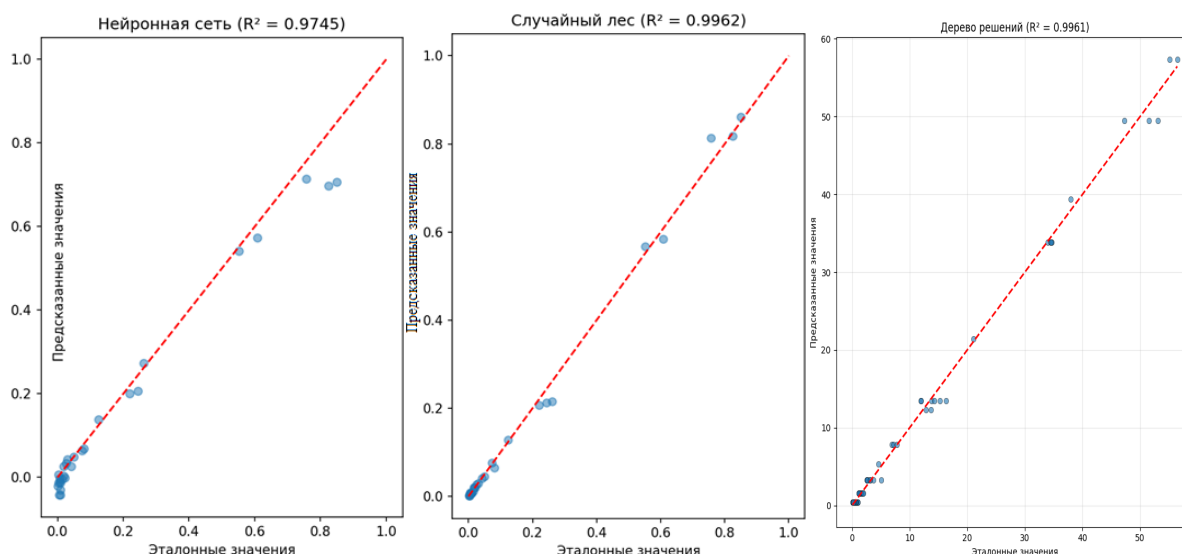


Рисунок 1 – Диаграммы рассеяния сравниваемых моделей

Сравнительный анализ показывает, что все три модели способны с приемлемой точностью аппроксимировать остаточное сопротивление маломерного судна. Однако наилучшие результаты получены для модели случайного леса: коэффициент детерминации $R^2 = 0.9962$ свидетельствует о практически идеальном совпадении предсказанных значений с эталонными. Нейронная сеть также продемонстрировала высокое качество ($R^2 = 0.9745$), но уступила ансамблевому методу. Дерево решений, несмотря на меньшую точность, также может быть полезным, так как позволяет в отличие от других моделей интерпретацию полученного результата.

Использование разработанной программы, реализующей три метода МО, позволяет создавать конкурентоспособные модели для прогнозирования гидродинамических характеристик маломерных судов. При этом традиционные методы расчета сопротивления маломерного судна сохраняют свою ценность, так как они более наглядны и проще в интерпретации. Перспективным является гибридный подход, сочетающий использование традиционных методов и методов МО.

ЛИТЕРАТУРА

1. J. Gerritsma, J.A. Keuning and A. Versluis. Sailing Yacht Performance in Calm Water and Waves. ReportNo. 938-P. 1993. Delft University of Technology, Ship Hydromechanics Laboratory.
2. Нечаев Ю.И. Современные проблемы информатики и вычислительной техники. – Санкт-Петербург: Арт- Экспресс. 2018. – 315 с.
3. Sandi Baressi Šegota, Nikola Anđelić, Jan Kudláček, Robert Čep. Artificial neural network for predicting values of residuary resistance per unit weight of displacement, Pomorski zbornik 57 (2019), 9-22.
4. Комличенко В.Н., Федосенко В.А., Купрейчик А.С. Сравнительный анализ различных архитектур нейронных сетей для задач регрессии. // Экономика и качество систем связи. – 2025. №1 – 110-121.
5. Гидродинамика яхты. [Электронный ресурс]. Режим доступа: <http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics#> (дата обращения 05.03.2026).
6. Scikit-learn. Машинное обучение на Python. [Электронный ресурс]. Режим доступа: https://scikit-learn.org/stable/supervised_learning.html (дата обращения 05.03.2026).

РАЗРАБОТКА ПРОГРАММНОГО МЕНЕДЖЕРА ЛОКАЛЬНО РАЗВЕРТЫВАЕМЫХ
БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

В последние годы методы обработки естественного языка (Natural Language Processing, NLP) становятся всё востребованнее и распространёнее благодаря появлению и развитию больших языковых моделей (Large Language Models, LLM) [1]. Такие модели способны решать широкий спектр задач: генерацию текста, семантический анализ, поиск сходства между текстовыми объектами, распознавание речи и другие [2]. Однако использование LLM в исследовательских и прикладных системах сопряжено с рядом технических сложностей. Модели могут иметь большой размер, требовать специальных механизмов загрузки, хранения и конфигурации, а также регулярно обновляться. Кроме того, для корректной работы некоторых систем необходимо управлять несколькими типами моделей одновременно и обеспечивать удобный механизм их выбора и замены.

Целью данного проекта является разработка программного обеспечения по управлению локальными большими языковыми моделями, обеспечивающее их установку, хранение, конфигурацию и использование в составе внешних приложений.

Для достижения поставленной цели необходимо решить следующие задачи:

1. провести анализ существующих подходов к управлению и использованию больших языковых моделей в прикладных системах;
2. сформировать функциональные требования к разрабатываемому программному обеспечению;
3. реализовать серверную часть системы, обеспечивающую хранение информации о моделях, их установку, конфигурацию и взаимодействие с внешними источниками моделей;
4. реализовать пользовательский интерфейс, позволяющий управлять установленными моделями, просматривать информацию о них и выбирать активные модели для выполнения различных задач по генерации;
5. протестировать разработанное ПО, в том числе с применением разных языковых моделей.

Для реализации программы был выбран следующий технологический стек. Серверная часть системы реализуется на языке Python [3] (с применением LangChain – библиотеки для разработки приложений на основе LLM и организации взаимодействия с ними, в том числе для структуризации ответов LLM [4]) с использованием веб-фреймворка FastAPI, который обеспечивает удобную разработку REST-интерфейсов, предназначенных для взаимодействия компонентов системы посредством HTTP-запросов. FastAPI позволяет автоматически генерировать документацию API, поддерживает асинхронную обработку запросов [5] и хорошо подходит для разработки микросервисных приложений. Для хранения метаданных установленных моделей используется система управления базами данных PostgreSQL, которая обеспечивает надежное хранение данных, поддержку сложных запросов и масштабируемость [6]. Пользовательский интерфейс системы реализован на языке C# с использованием фреймворка Avalonia UI, который позволяет создавать кроссплатформенные графические приложения [7]. В архитектуре клиентской части применяется паттерн MVVM (Model-View-ViewModel), обеспечивающий разделение логики представления и бизнес-логики приложения.

Одним из ключевых компонентов является библиотека llama.cpp, предназначенная для эффективного выполнения больших языковых моделей на локальных вычислительных устройствах. Данный инструмент реализует оптимизированный механизм запуска моделей в формате GGUF и позволяет использовать современные LLM без необходимости в специализированной серверной инфраструктуре. Важным преимуществом llama.cpp является

поддержка квантованных моделей. Квантование представляет собой метод оптимизации нейронных сетей, при котором параметры модели переводятся из высокоточной числовой формы (например, 16- или 32-битных значений) в более компактное представление. Такой подход позволяет существенно снизить требования к объёму оперативной памяти и видеопамати, сохраняя при этом приемлемое качество работы модели.

В качестве источников моделей используются открытые хранилища репозитория машинного обучения, такие как HuggingFace и ModelScope, предоставляющие доступ к большому количеству современных языковых моделей [8].

Полная схема архитектуры разрабатываемого ПО менеджера LLM и взаимодействия между его основными компонентами отражена на рисунке 1.

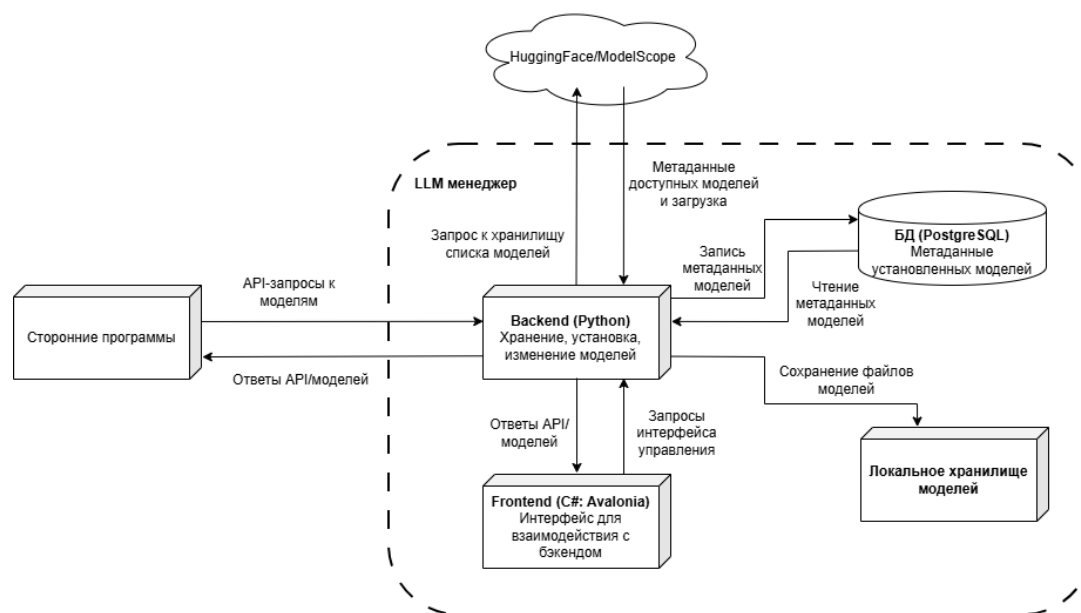


Рисунок 1 – Архитектура разрабатываемого менеджера LLM

ЛИТЕРАТУРА

1. Дудихин В. В., Кондрашов П. Е. Методология использования больших языковых моделей для решения задач государственного и муниципального управления по интеллектуальному реферированию и автоматическому формированию текстового контента // Государственное управление. Электронный вестник, 2024, № 105, с. 169–179. DOI: 10.55959/MSU2070-1381-105-2024-169-179
2. Кипяткова И. С., Долгушин М. Д., Кагиров И. А. Аналитический обзор применения больших языковых моделей для автоматического распознавания речи. Информационно-управляющие системы, 2026, № 1, с. 19–35. doi:10.31799/1684-8853-2026-1-19-35, EDN: DSRKFE.
3. Рашка Себастьян, Мирджалили Вахид Python и машинное обучение. Машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2. – Санкт-Петербург: Диалектика, 2020. – 848 с.
4. LangChain Docs. [Электронный ресурс] Режим доступа: <https://docs.langchain.com/oss/python/langchain/middleware/overview> (дата обращения: 10.01.2026)
5. Fast API. [Электронный ресурс] Режим доступа: <https://fastapi.tiangolo.com/> (дата обращения: 23.02.2026)
6. PostgreSQL: Documentation [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/> (дата обращения: 15.02.2026)
7. Avalonia Docs [Электронный ресурс] Режим доступа: <https://docs.avaloniaui.net/> (дата обращения: 19.02.2026)
8. Hugging Face – Documentation. [Электронный ресурс] Режим доступа: <https://huggingface.co/docs> (дата обращения: 18.02.2026)

СРАВНЕНИЕ МЕТОДОВ ПРИОРИТИЗАЦИИ РЕГРЕССИОННЫХ ТЕСТОВ

Методы приоритизации формируют оптимальный порядок выполнения модульных тестов. Total и Additional [1] дают базовый ориентир по эффективности, но не всегда учитывают неоднородную важность покрываемых методов и часто теряют качество на проектах с плотным перекрытием тестов. Современные подходы, ориентированные на анализ влияния изменений и различимость покрытия, показывают более устойчивое поведение. Их эффективность, в свою очередь, зависит от структуры проекта и распределения дефектов.

Целью работы является сравнительная оценка методов Law of Minimum (далее - LoM), Dis-LoM и Coverage Clustering [2] (далее – CC), Total и Additional на едином экспериментальном стенде для задач приоритизации регрессионных тестов.

LoM использует вероятности влияния методов классов, полученные из анализа изменений и зависимостей вызовов. Для каждого теста учитывается, какие значимые методы он покрывает, после чего тест получает итоговый балл по наиболее важным компонентам. В результате выше в порядке оказываются тесты, затрагивающие наиболее критичные участки кода. Dis-LoM развивает LoM и снижает приоритет тестов с избыточно похожим покрытием. После базовой оценки вводится штраф за сходство с уже выбранными тестами (по коэффициенту Жаккара): чем сильнее дублирование покрытия, тем ниже текущий приоритет. CC решает задачу через геометрию покрытия. Каждый тест представляется бинарным вектором строки матрицы C, расстояние между тестами вычисляется в евклидовой метрике, после чего выполняется агломеративная кластеризация. Внутри кластеров применяется стратегия Additional, а финальный порядок формируется round-robin объединением тестов из разных кластеров.

Архитектура стенда приведена на рисунке 1.

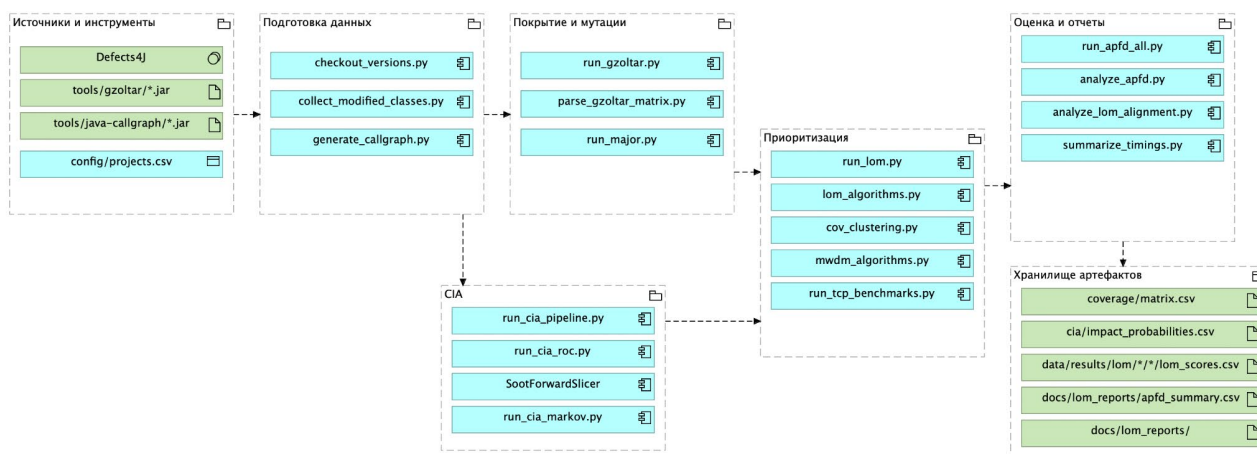


Рисунок 1 – Архитектура разработанного стенда приоритизации

Экспериментальная часть выполнена на проектах Defects4J [3] (Jsoup-93, Gson-18, Csv-16, JacksonDatabind-112, JacksonXml-6, Compress-47, Time-27, Codec-18, JacksonCore-26, Lang-65) с использованием GZoltar для сбора покрытия на уровне методов и Major для генерации мутантов. Методы запускались на одинаковых входных данных, что устраняет влияние различий в источниках покрытия и позволяет интерпретировать различия APFD как эффект именно алгоритма приоритизации. $APFD = 1 - \sum_{i=1}^m TF_i / mn + \frac{1}{2}$, где n - число тестов, m - число мутантов, TF_i - позиция первого теста, обнаружившего i-й мутант. Чем выше APFD, тем раньше по порядку обнаруживаются дефекты. Средние значения APFD по семи проектам следующие: DisLom - 0.97, Lom - 0.93, CC - 0.86, Additional - 0.82, Total - 0.8.

Полученные результаты показывают устойчивое преимущество Dis-LoM по APFD. На пяти проектах Dis-LoM улучшает LoM, на двух - совпадает с ним. Наиболее заметный прирост наблюдается на проекте Codec (около +0.20 APFD относительно LoM), что подтверждает практическую ценность штрафа за сходство покрытия. В проектах Csv и Lang различия между LoM и Dis-LoM минимальны, что указывает на более слабый эффект диверсификации для соответствующей структуры тестов.

Таким образом, наиболее надежным подходом для раннего обнаружения дефектов является Dis-LoM, поскольку он сочетает оценку влияния изменений с контролем избыточного дублирования покрытия. LoM остаётся сильной базой приоритизации. СС может быть полезен в сценариях, где приоритетом является максимальная структурная диверсификация ранних запусков, который ранее не сравнивался с Dis-LoM и LoM [1]. Артефакты экспериментов и стенд размещены в открытом репозитории [4].

ЛИТЕРАТУРА

1. Ufuktepe E., Tuglular T. Application of the Law of Minimum and Dissimilarity Analysis to Regression Test Case Prioritization // IEEE Access. 2023. Vol. 11.
2. Mahdiah M., Mirian-Hosseiniabadi S.-H., Mahdiah M. Test Case Prioritization Using Test Case Diversification and Fault-Proneness Estimations // Automated Software Engineering. 2022. Vol. 29, no. 2.
3. Just R., Jalali D., Ernst M. D. Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs // Proceedings of ISSTA. 2014.
4. Rudenko M., Parkhomenko V. TCP methods // URL: <https://github.com/oneren38/tcp-methods>

УДК 004.415.2

А.В. Потехин (4 курс бакалавриата),
Е.Е. Андрианова, ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ МОНИТОРИНГА И ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Производительность приложений, использующих реляционные базы данных, во многих случаях определяется эффективностью выполнения SQL-запросов. Оптимизация запросов в эксплуатации осложняется изменчивостью рабочей нагрузки, большим количеством различающихся запросов и необходимостью учитывать статистику их фактического выполнения.

Методы оптимизации запросов могут рассматриваться в двух уровнях: логической и физической оптимизации [1, 2]. Логическая оптимизация заключается в преобразовании исходного SQL-запроса в семантически эквивалентную форму, потенциально более эффективную. Физическая оптимизация связана с выбором конкретных способов выполнения (алгоритмов соединений, использования индексов и т.п.) оптимизатором СУБД.

Использование больших языковых моделей также рассматривается как перспективный подход к переписыванию SQL-запросов в рамках логической оптимизации. В ряде работ [3, 4, 5] показана возможность генерации семантически эквивалентных вариантов запросов и получения улучшений производительности при последующей верификации корректности и оценке эффекта по фактическим метрикам выполнения (например, GenWrite, LLM-R2, R-Bot).

Целью данной работы является разработка системы, позволяющей автоматически получать более эффективные варианты запросов через мониторинг и выявление критичных запросов, требующих оптимизации.

Для достижения поставленной цели необходимо решить следующие задачи, перечисленные ниже.

1. Обзор существующих решений по оптимизации и мониторингу SQL запросов.

3. Liu J., Mozafari B. Query Rewriting via Large Language Models. — 2024. — URL: <https://arxiv.org/abs/2403.09060> (дата обращения: 14.03.2025).
4. Li Z., Yuan H., Wang H., Cong G., Bing L. LLM-R2: A Large Language Model Enhanced Rule-based Rewrite System for Boosting Query Efficiency // Proc. VLDB Endow. — 2024. — Vol. 18, № 1. — P. 53–65 — URL: <https://arxiv.org/abs/2404.12872> (дата обращения: 14.03.2026).
5. Sun Z., Zhou X., Li G., Yu X., Feng J., Zhang Y. R-Bot: An LLM-based Query Rewrite System // Proceedings of the VLDB Endowment. — 2025. — Vol. 14, No. 1. — P. 1-13 — URL: <https://arxiv.org/abs/2412.01661> (дата обращения: 14.03.2026).
6. Transaction Processing Council (TPC). [Электронный ресурс] Режим доступа: <https://www.tpc.org/> (дата обращения: 14.03.2026).
7. Rajvanshi S., Yadav H., Singh A. Overview of Java Programming // International Research Journal of Engineering and Technology (IRJET). — 2020. — Vol. 7. — No. 5. — P. 7538–7540. — URL: <https://www.irjet.net/archives/V7/i5/IRJET-V7I51454.pdf> (дата обращения: 14.03.2026).
8. Spring Framework. [Электронный ресурс] Режим доступа: <https://spring.io/projects/spring-framework> (дата обращения: 14.03.2026).

УДК 004.453

М.Д. Скарлыгина (4 курс бакалавриата),
А.Н. Яковлев, ст. преподаватель

СИСТЕМА ОРКЕСТРАЦИИ ФАЗЗИНГ-ТЕСТИРОВАНИЯ НА ОСНОВЕ СУПЕРКОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Фаззинг-тестирование занимает особое место в современном процессе обеспечения качества программного обеспечения, позволяя выявлять уязвимости и дефекты, недоступные для статического анализа и формальной верификации. Фаззинг-тестирование (fuzzing) представляет собой автоматизированную технику подачи на вход программы большого количества некорректных, неожиданных или случайных данных с целью обнаружения сбоев, ошибок выполнения и потенциальных уязвимостей [1]. Эффективность фаззинга напрямую зависит от интенсивности генерации и подачи входных данных: чем больше тестовых воздействий удастся сгенерировать и применить к целевой программе в единицу времени, тем выше вероятность обнаружения скрытых ошибок. Современные фаззеры, такие как AFL++ [6], способны генерировать миллионы тестовых воздействий в секунду, однако достижение по-настоящему высоких показателей требует колоссальных вычислительных мощностей, которые способны предоставить суперкомпьютерные комплексы.

Существующие решения для автоматизации фаззинга, такие как BugBane или Sydr-fuzz, ориентированы на локальные или статические окружения и не учитывают специфику работы с системами пакетной обработки заданий (SLURM), характерными для высокопроизводительных вычислительных комплексов [2]. Это обуславливает актуальность создания системы оркестрации, адаптированной для суперкомпьютерной среды.

Целью работы является реализация системы оркестрации фаззинг-тестирования для суперкомпьютерного комплекса, обеспечивающей полный цикл: от описания сценария до реляционного анализа результатов. Следует выделить следующие задачи исследования:

1. Анализ существующих подходов к распараллеливанию фаззинга и организации взаимодействия экземпляров фаззера [3] (на примере архитектуры AFL++).
2. Разработка архитектуры системы, включающей реляционное хранилище, модуль генерации заданий для SLURM и парсер результатов.
3. Интеграция механизмов оптимизации: алгоритм автоматического определения числа узлов и методы снижения накладных расходов на ввод-вывод.
4. Реализация конвейера трансляции результатов фаззинга в реляционную модель.
5. Аprobация системы на реальном программном проекте.

Система построена на принципе декларативного управления, где база данных выступает центральным компонентом, определяющим логику работы оркестратора. Архитектура включает три уровня:

- *Уровень описания сценариев.* Пользователь взаимодействует с системой через CLI-команды. В базе данных создаются сущности: таблица `targets` содержит информацию об объекте тестирования (путь к исполняемому файлу, скомпилированному с инструментацией AFL++, расположение корпуса начальных семплов); таблица `scenarios` связывает целевой объект с конфигурацией фаззера и задает временные ограничения и ожидаемую интенсивность генерации данных.
- *Уровень оптимизированного планирования.* При активации сценария модуль оркестрации на основе эмпирической модели определяет оптимальное число узлов для оптимизации распараллеливания тестирования. Система анализирует заявленную интенсивность тестирования и применяет пороговые правила использования узлов суперкомпьютера. Модуль генерации формирует SLURM-скрипт с оптимизациями: запрос узлов с InfiniBand (`--constraint=ib`) для снижения задержек при синхронизации процессов, монопольное использование узлов (`--exclusive`), перенос корпуса тестовых данных на локальные SSD-диски для исключения обращений к сетевой файловой системе [4].
- *Уровень сбора и трансформации результатов.* После завершения фаззинга модуль парсинга выполняет ETL-процесс: извлечение данных из директории результатов (файлы статистики, директории `crashes/` и `queue/`), интерпретацию внутреннего формата AFL++ и загрузку данных в таблицы `afl_runs` (общая статистика) и `afl_crashes` (информация о каждом краше с привязкой к файлу-триггеру) [5].

Результаты работы были получены с использованием вычислительных ресурсов суперкомпьютерного центра Санкт-Петербургского политехнического университета Петра Великого. В ходе апробации была выполнена проверка работоспособности полного цикла: создание сценария, автоматическое определение ресурсов, сбор и структурирование результатов. В ходе экспериментов парсер зафиксировал в базе данных следующие метрики: общее количество выполненных итераций, число найденных уникальных путей, количество крашей. Каждый краш соотнесен с конкретным входным файлом.

Применение реляционной модели позволяет выполнять аналитические запросы, недоступные при работе с сырыми логами. Например, SQL-запрос к данным запусков с разным числом узлов дает возможность оценить эффективность масштабирования для конкретной программы. Группировка крашей по функциям позволяет идентифицировать наиболее проблемные участки кода.

В результате работы спроектирована и реализована система оркестрации фаззинг-тестирования для суперкомпьютерной среды. Основным результатом является создание замкнутого цикла: описание сценария в реляционной модели, оптимизированное планирование ресурсов, параллельный запуск на кластере и автоматическая трансформация результатов в структурированное представление. Практическая значимость работы заключается в автоматизации ресурсоемкого процесса тестирования и предоставлении инструмента для SQL-анализа накопленных данных. Дальнейшее развитие предполагает интеграцию средств автоматической классификации ошибок и поддержку других фаззеров.

ЛИТЕРАТУРА

1. Барабанов А.В., Падарян В.А., Козлов А.Ю. Sydr-fuzz: конвейер для поиска уязвимостей на основе гибридного фаззинга // Труды Института системного программирования РАН. – 2022. – Т. 34. – № 5. – С. 47-62.
2. Иванцов Р.Е., Платонов В.В. Применение методов фаззинг-тестирования для поиска уязвимостей в программном обеспечении // Вопросы кибербезопасности. – 2021. – № 4(44). – С. 55-64.
3. Lemieux C., Sen K. FairFuzz: A Targeted Mutation Strategy for Increasing Greybox Fuzz Testing Coverage // Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. – 2018. – P. 475-485.

4. SLURM Workload Manager Documentation. [Электронный ресурс]. – Режим доступа: <https://slurm.schedmd.com/> (дата обращения: 10.03.2026).
5. Zalewski M. American Fuzzy Lop. [Электронный ресурс]. – Режим доступа: <https://lcamtuf.coredump.cx/afl/> (дата обращения: 05.03.2026).

УДК 004.056

И.С. Смирнова, О.А. Коган, А.И. Слапик (4 курс бакалавриата),
Н.В. Совтус, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО ИНСТРУМЕНТА АНАЛИЗА ЗЕРКАЛИРОВАННОГО СЕТЕВОГО ТРАФИКА ДЛЯ ТЕСТИРОВАНИЯ ПРОМЫШЛЕННЫХ ETHERNET-КОММУТАТОРОВ

Современные сети Ethernet широко применяются в промышленных системах автоматизации, телекоммуникационной инфраструктуре и распределённых вычислительных системах. Для диагностики сетевой инфраструктуры и анализа сетевых потоков управляемые коммутаторы поддерживают режим зеркалирования портов (port mirroring), при котором копии пакетов с выбранных портов передаются на отдельный порт мониторинга. Данный механизм позволяет осуществлять анализ сетевого трафика без вмешательства в рабочие сетевые соединения и используется при тестировании сетевого оборудования и обнаружении аномалий сетевой активности.

Анализ сетевого трафика является важным инструментом обеспечения надёжности и безопасности компьютерных сетей, позволяя выявлять нарушения в передаче данных, аномальные сетевые события и проблемы функционирования сетевых устройств [1]. Для захвата и анализа сетевых пакетов широко применяются универсальные программные анализаторы сетевых протоколов, например Wireshark, позволяющий выполнять детальный разбор сетевых пакетов и анализ протоколов различных уровней сетевой модели [2].

Несмотря на широкие функциональные возможности подобных систем, они ориентированы преимущественно на интерактивный анализ сетевых протоколов и диагностику сетевых взаимодействий. В задачах тестирования сетевого оборудования особый интерес представляет автоматизированная оценка корректности функционирования режима зеркалирования портов, включая анализ полноты копирования пакетов и количественную оценку характеристик сетевого потока при высоких скоростях передачи данных.

В условиях высокой нагрузки на сетевое оборудование возможно возникновение ситуаций, при которых зеркалируемый трафик передаётся не полностью, что приводит к потере части пакетов на зеркалированном порту. В связи с этим актуальной является разработка специализированных программных средств диагностики сетей Ethernet, позволяющих выполнять автоматизированный анализ сетевого трафика и выявлять возможные потери пакетов [3].

Целью данной работы является разработка программного средства анализа зеркалированного сетевого трафика, предназначенного для экспериментального исследования характеристик режима зеркалирования портов в промышленных Ethernet-коммутаторах.

Для достижения поставленной цели решаются следующие задачи:

- анализ существующих методов захвата и анализа сетевого трафика;
- разработка архитектуры программного средства анализа зеркалированного трафика;
- разработка алгоритма обнаружения потерь пакетов на основе анализа последовательных номеров;
- реализация методов измерения интенсивности сетевого потока;
- проведение экспериментального исследования характеристик режима зеркалирования портов.

Разработанный программный инструмент осуществляет захват сетевых пакетов с зеркалированного порта коммутатора и выполняется их анализ. Архитектура системы включает модуль захвата сетевых пакетов, модуль анализа последовательных номеров пакетов, модуль статистической обработки данных и модуль формирования результатов.

Модуль захвата пакетов выполняет приём сетевого трафика с сетевого интерфейса и передачу полученных данных в подсистему анализа. Модуль анализа выполняет обработку последовательных номеров пакетов (sequence number), что позволяет выявлять пропуски пакетов и автоматически определять уровень потерь трафика. Использование последовательных номеров является распространённым подходом при анализе сетевых потоков и позволяет выявлять нарушения целостности передачи данных [4].

Для количественной оценки характеристик сетевого потока реализованы алгоритмы расчёта следующих показателей:

- интенсивность потока пакетов (Packets Per Second, PPS);
- пропускная способность канала (Mbps);
- доля потерянных пакетов на основе анализа последовательных номеров.

Архитектура разработанного программного инструмента представлена на рисунке 1.

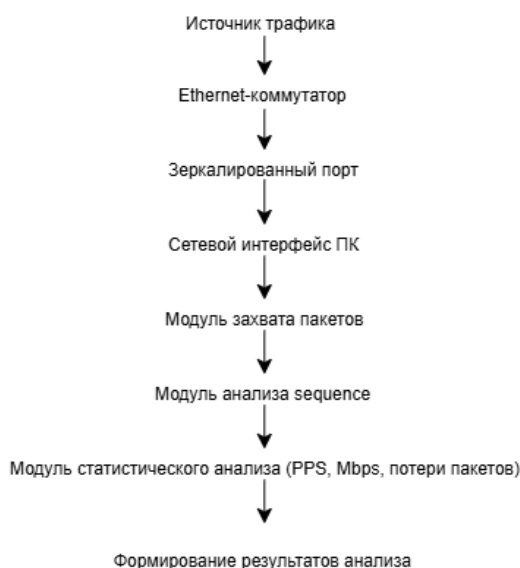


Рисунок 1 – Архитектура программного инструмента анализа зеркалированного трафика

Разработанное программное средство позволяет проводить экспериментальные исследования характеристик режима зеркалирования портов. В ходе экспериментов выполняется захват зеркалированного трафика и автоматический расчёт статистических характеристик сетевого потока. Полученные результаты позволяют выявлять случаи неполного копирования пакетов, а также оценивать пропускную способность системы мониторинга.

Предложенный инструмент может применяться при испытаниях сетевого оборудования, а также при диагностике и мониторинге промышленных сетей Ethernet. Разработка специализированных средств анализа зеркалированного трафика расширяет возможности автоматизации процессов тестирования сетевых устройств и повышает точность оценки их характеристик.

ЛИТЕРАТУРА

1. Обеспечение информационной безопасности компьютерных сетей на основе анализа сетевого трафика. – Москва: Изд-во МГТУ, 2016.
2. Orebaugh A., Ramirez G., Beale J. Wireshark & Ethereal Network Protocol Analyzer Toolkit. – Syngress, 2007.
3. Разработка и реализация средства диагностики сетей Ethernet // Сборник научных трудов по информационным технологиям. – 2022.

4. Stevens W. R., Wright G. R. TCP/IP Illustrated. Volume 1: The Protocols. – Addison-Wesley, 2011.
5. Tanenbaum A., Wetherall D. Computer Networks. – Pearson, 2021.

УДК 004.056

С.А. Семенов (3 курс бакалавриата),
Н.В. Воинов, к.т.н., доцент

РЕАЛИЗАЦИЯ RAID-МАССИВОВ С ТРЕМЯ КОНТРОЛЬНЫМИ СУММАМИ ДЛЯ ВОССТАНОВЛЕНИЯ ДАННЫХ

Переход к цифровой экономике сопровождается резким ростом объемов создаваемой и обрабатываемой информации, к которой требуется мгновенный доступ, но при этом она должна быть надежно защищена от потери. В качестве решения этой задачи широко применяются RAID (Redundant Array of Independent Disks - избыточный массив независимых дисков) системы хранения данных. На сегодняшний день самым распространенным вариантом является RAID 5 и 6 с одной и двумя контрольными суммами (паритетами), покрывая больше 70% всех установок корпоративных RAID систем [1]. Если целостность данных более критична, например, для сбора астрономических данных или обработке финансовых транзакций, можно использовать 3 контрольные суммы. Тем самым можно будет восстановить до 3 отказавших дисков.

В процессе решения были систематизированы существующие подходы к многопаритетному кодированию и разработана система, поддерживающая три контрольные суммы. Такая система позволит восстановить данные после потери до трех дисков и будет применяться в центрах обработки данных (ЦОД) клиентов [2].

В ходе работы были решены следующие задачи:

1. Изучены существующие решения для генерации контрольных сумм и восстановления данных RAID 6 и RAID с 3 паритетами.
2. Реализована основная функциональность.
3. Проведено тестирование созданного решения.

RAID-система может быть реализована как аппаратно, так и программно. В данном проекте выбран программный подход, он сильно удешевляет производство и развертывание и при умеренных нагрузках на процессор сопоставим по скорости с аппаратной системой. Ключевыми факторами для корпоративных клиентов остается надежность и поддержка. Учитывая прекращение продаж и поддержки зарубежных решений в РФ, выбор пал на российскую платформу Alt Linux – популярную, с открытой лицензией и высокой производительностью [3]. Для максимальной эффективности использования доступных ресурсов платформы нужно использовать не уровень пользователя операционной системы, а уровень ядра. Так как целевой платформой разработки является русская версия операционной системы GNU/Linux, разумным является разработка модуля для ядра Linux. Linux имеет монолитную архитектуру ядра с возможностью загрузки внешних модулей, что используется для загрузки драйверов устройств [4]. Модуль был разработан на языке ядра – C. Для сборки проекта использовалась система сборки Kbuild [5].

Первый паритет создаётся простым XOR-сокращением содержимого всех дисков. То есть каждый бит каждого диска объединяется логическим «исключающим И» (XOR), и результат записывается в первый контрольный блок.

Второй паритет генерируется с помощью арифметики конечного поля Галуа $GF(2^8)$. Для каждого диска i данные умножаются на примитивный (действительный) элемент α в степени i (α^i). Полученные произведения затем сокращаются операцией XOR, что формирует второй контрольный блок. Конечное поле имеет свойство, позволяющее заменить деление (необходимо для решения полученных линейных уравнений) на умножение на обратный элемент поля [6].

Третий паритет строится аналогичным способом, но с другим набором степеней. Данные диска i умножаются на $\alpha^{(2*i)}$, после умножения результаты дисков сокращаются операцией XOR, и образуется третий контрольный блок. Такая конструкция позволяет сформировать систему уравнений и восстановить до трёх одновременно потерянных дисков.

Тестирование проводилось двумя методами: unit-тестирование (проверка отдельных функций) и функциональное нагрузочные тестирование при помощи утилиты fio – проверка полноценной работы функциональности.

В дальнейшем планируется внедрение фоновое сканирование скрытых повреждений (SDC). В RAID-5 такое сканирование лишь обнаруживает, но не восстанавливает скрытые повреждения. RAID-6 способен исправлять их, но только при условии, что все диски остаются в работе. В RAID-системах с тремя контрольными суммами возможность восстановления сохраняется даже при отказе одного диска, что сильно повышает надежность системы.

ЛИТЕРАТУРА

1. Redundant Array of Independent Disks (RAID) Market Overview [Электронный ресурс] Режим доступа: <https://www.industryresearch.biz/market-reports/redundant-array-of-independent-disks-raid-market-107795> (дата обращения: 02.03.2026).
2. Чигасова, М. А. Прогнозирование котировок акций с использованием нейронной сети / М. А. Чигасова, О. Г. Малеев // Современные технологии в теории и практике программирования : Сборник материалов конференции, Санкт-Петербург, 26 апреля 2022 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2022. – С. 180-182. – EDN GRFEDM.
3. Н. С. Нестерова, Д. В. Волков РОССИЙСКИЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ ALT LINUX: ОБЗОР И ОСОБЕННОСТИ // Международный журнал гуманитарных и естественных наук. 2025. №7-2 (106).
4. Kernel modules [Электронный ресурс] Режим доступа: https://linux-kernel-labs.github.io/refs/heads/master/labs/kernel_modules.html (дата обращения: 02.03.2026).
5. Building External Modules [Электронный ресурс] Режим доступа: <https://docs.kernel.org/kbuild/modules.html> (дата обращения: 02.03.2026).
6. Intelligent RAID 6 Theory Overview and Implementation [Электронный ресурс] Режим доступа: <https://www.lamsade.dauphine.fr/~litwin/cours98/Doc-cours-clouds/raid6-intel.pdf> (дата обращения: 02.03.2026).

УДК 004.4

Я.А. Спирчина (4 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

РАЗРАБОТКА МОДУЛЯ АНАЛИЗА SDK ДЛЯ ОБРАБОТЧИКА APK-ФАЙЛОВ RUSTORE

Мобильные приложения всё чаще используются в нашей повседневной жизни. Согласно статистике [6] ежегодно во всем мире загружается 255 млрд мобильных приложений. С ростом их количества растёт и их сложность. Разработчики активно стали использовать сторонние библиотеки и SDK, чтобы ускорить создание приложений и добавить в них готовые функции. Однако без точных данных о том, какие именно SDK [1] используются разработчиками, сложно понимать текущую ситуацию на рынке.

В экосистеме RuStore, где пользователи загружают тысячи приложений, автоматический анализ SDK становится ключевым инструментом для развития бизнеса. Когда станет известно, что разработчик использует сторонний SDK, ему можно будет предложить перейти на решение от Rustore. Это создаст дополнительный источник дохода через комиссии, а также укрепит позиции Rustore как технологической платформы.

Цель моей работы - создать анализатор SDK, который автоматически определяет, какие библиотеки используются в приложениях для платформы Android и передавать эти данные для дальнейшего анализа. Сейчас такая информация часто собирается вручную или существующие инструменты не встроены в процесс загрузки приложений, из-за чего данные получаются не полные и порой ошибочные. Мой анализатор решает эту проблему и покажет, какие SDK используют разработчики.

Реализация функционала будет на языке Java. Для самого анализа будет внедрена сторонняя библиотека `net.dongliu.apk.parser` [5], которая предназначена для чтения и парсинга APK-файлов. Она позволяет извлекать информацию из манифеста, ресурсов и подписей приложений без необходимости распаковывать весь APK [2]. С её помощью мы будем анализировать DEX-классы и файл `AndroidManifest.xml` [4]. По названиям пакетов DEX-классов [3] будут собирать данные об SDK.

Предполагаемая архитектура:

1. Загрузка приложений: Пользователь загружает APK-файлы, которые попадают в сервис `application-publicator` для валидации и проверки данных.

2. Передача через Kafka: Обработанные данные передаются в сервис `app-processor`. Это сделано для того, чтобы сервис, который обрабатывает данные, мог продолжить свою работу, не дожидаясь завершения анализа в `app-processor`. Это значительно увеличивает производительность и масштабируемость системы.

3. Анализ приложений: В `app-processor` происходит анализ apk-файлов на SDK с использованием библиотеки `net.dongliu.apk.parser`, результаты анализа сохраняются в базу данных. Также будет реализовано автоматическое очищение базы данных раз в две недели с использованием DTF-механизма.

4. Данные проверяются на наличие активной и опубликованной в RuStore версии и передаются в DWH [7] через Kafka для дальнейшего анализа и отчетности.

Внедрение данного функционала в микросервисную архитектуру создаст основу для дальнейшего развития системы, позволяя легко добавлять новые функции и улучшать существующие.

Также дополнительно будет реализована внешняя ручка, позволяющая принудительно просканировать любое приложение, загруженное ранее в RuStore, по его `package_name`.

Таким образом, главным результатом будет – разработанный анализатор SDK, который решает проблему недостатка информации о текущих тенденциях и предпочтениях разработчиков. А также откроет новые возможности для предложения собственных SDK от Rustore, что может привести к увеличению доходов через комиссии и укреплению позиций платформы на рынке.

ЛИТЕРАТУРА

1. SDK [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/SDK> -(дата обращения: 12.03.2026)
2. APK [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/APK> -(дата обращения: 12.03.2026)
3. Dex-classes [Электронный ресурс] – URL: https://translated.turbopages.org/proxy_u/en-ru.ru.d56eba5f-69b4268a-0166cce5-74722d776562/https/stackoverflow.com/questions/14230573/role-of-classes-dex-file-in-an-apk-file - (дата обращения: 13.03.2026)
4. Manifest [Электронный ресурс] – URL: https://translated.turbopages.org/proxy_u/en-ru.ru.d56eba5f-69b4268a-0166cce5-74722d776562/https/stackoverflow.com/questions/14230573/role-of-classes-dex-file-in-an-apk-file -(дата обращения: 13.03.2026)
5. `net.dongliu.apk.parser` [Электронный ресурс] – URL: <https://github.com/hsiafan/apk-parser?ysclid=mk0ytkynn295628364> -(дата обращения: 12.03.2026)

6. Скачивание приложений [Электронный ресурс] – URL : <https://apptractor.ru/info/analytics/v-2025-godu-kolichestvo-zagruzok-prilozheniy-snova-sokratilos-no-rashody-potrebiteley-vyrosli-pochti-do-156-mlrd.html?ysclid=mmqpebc7uo743068909> -(дата обращения: 12.03.2026)
7. DWH [Электронный ресурс] – URL : <https://habr.com/ru/articles/852910/> -(дата обращения: 12.03.2026)

УДК 004.415.53

М.Б. Спиваков, Т.Р. Костюченко (4 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ГЕНЕРАТОРА АВТОМАТИЗИРОВАННЫХ ТЕСТОВ НА ОСНОВЕ ИИ

При разработке современного программного обеспечения команды создают сотни тестовых сценариев. Хранить их в обычных таблицах и документах неудобно: сложно отслеживать версии, статусы и собирать аналитику [1]. Для решения этой проблемы в данной работе спроектирована и разработана серверная часть централизованной системы управления тестированием. Она позволяет вести ИТ-проекты, управлять проектными командами и хранить тестовые сценарии, а также включает модуль автоматической генерации тестов с помощью локальных нейросетей.

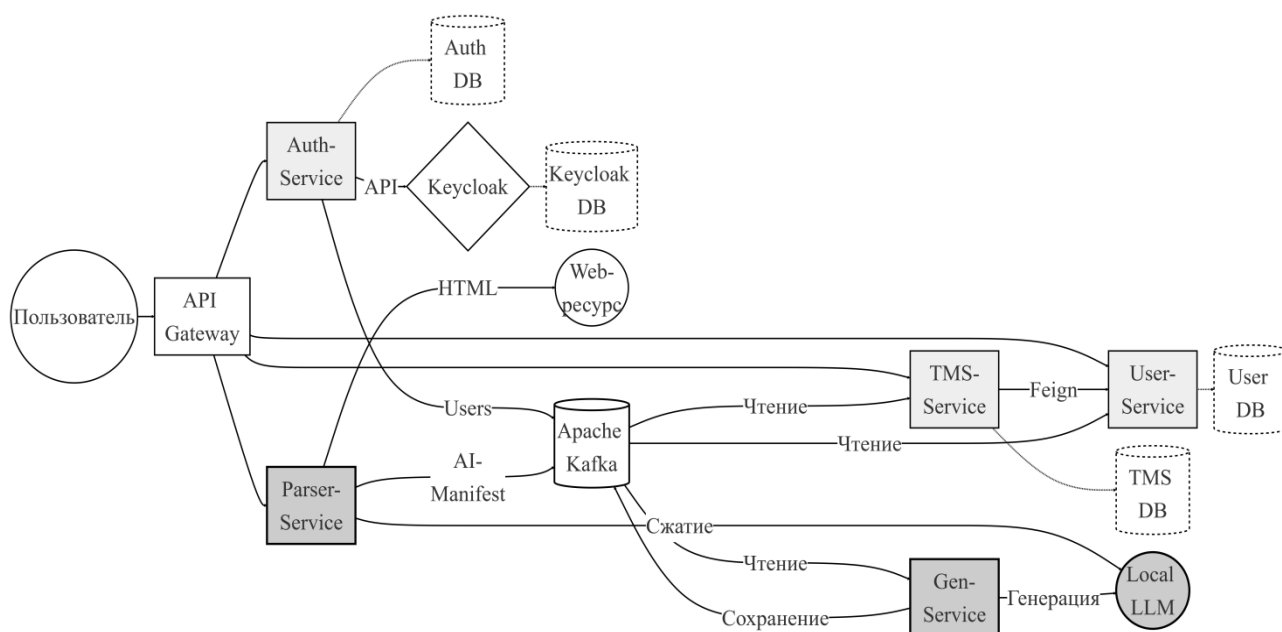


Рисунок 1 – Архитектура системы

На рисунке 1 показано, что архитектура разрабатываемой системы построена на микросервисах [2]. Точкой входа для пользователей служит Spring Cloud Gateway. Для поиска сервисов используется реестр Spring Cloud Eureka, а за безопасность и выдачу токенов отвечает Keycloak [4]. Чтобы падение одного сервиса не ломало всю систему, применяется паттерн «база данных на сервис». Данные хранятся в изолированных реляционных БД PostgreSQL, что гарантирует их надежность и целостность [3].

Микросервисы общаются друг с другом двумя способами. Синхронные запросы идут через REST API с помощью клиента OpenFeign. Фоновые задачи и обмен событиями работают асинхронно через брокер сообщений Apache Kafka с помощью Spring Cloud Stream [5]. В системе используется событийно-ориентированная архитектура. Например, когда сервис авторизации регистрирует пользователя, он отправляет событие в Kafka. Модуль профилей читает это сообщение и обновляет свою локальную базу. Такая денормализация избавляет сервисы от необходимости постоянно запрашивать информацию друг у друга по сети [3]. Это повышает скорость работы и отказоустойчивость всей платформы.

Главная инновационная особенность платформы – конвейер для создания тестов. Чтобы исходный код и коммерческая тайна не попали в открытый доступ, все вычисления происходят на локальном AI-сервере. Процесс разделен на два микросервиса. Сначала парсер загружает веб-страницу и с помощью нейросети сжимает её код, оставляя только важные смысловые элементы. Затем эта краткая выжимка через Kafka передается генератору. На её основе вторая модель пишет готовый код Unit-тестов с учетом возможных ошибок [6]. Такое разделение экономит ресурсы сервера, обходит ограничения на размер текста для LLM и снижает риск генерации логически некорректного кода.

В результате разработана серверная часть современной системы управления тестированием. Использование микросервисов, Kafka и изолированных баз данных делает систему масштабируемой и защищенной от сбоев. Внедрение локального искусственного интеллекта берет на себя рутинную работу по написанию тестов, экономя время QA-инженеров и повышая качество выпускаемого ПО.

ЛИТЕРАТУРА

1. Златоустов И. В., Копша О. Ю. Автоматизация тестирования: ускорение инноваций с помощью интеллектуальных тестовых сценариев // Вестник науки. – 2025. – Т. 2, № 5 (86). – С. 816-823.
2. Ньюмен С. Создание микросервисов. 2-е изд. – СПб.: Питер, 2023. – 624 с.
3. Kleppmann M. Designing Data-Intensive Applications. – O'Reilly Media, 2017. – 616 p.
4. Walls C. Spring in Action, Sixth Edition. – Manning Publications, 2022. – 520 p.
5. Stopford B. Designing Event-Driven Systems. – O'Reilly Media, 2018. – 171 p.
6. Титеев А. М. Оптимизация процесса создания автотестов веб-приложений с использованием LLM и структурного анализа HTML // Моделирование и анализ данных. – 2025. – Т. 15, № 4. – С. 87-103.

УДК 004.05

С.Ф. Солодовников (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент,
С.М. Устинов, д.т.н, профессор

МЕТОДИКА АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ ОБЛАЧНЫХ СЕРВИСОВ В РАСПРЕДЕЛЕННЫХ ИНФРАСТРУКТУРАХ

Современные облачные платформы становятся ключевым элементом цифровой инфраструктуры организаций, обеспечивая гибкость, масштабируемость и высокую скорость доставки сервисов [1]. Модель Platform as a Service (PaaS) предоставляет разработчикам возможность создавать и развертывать приложения без необходимости управления аппаратными ресурсами, что существенно ускоряет процессы разработки и снижает эксплуатационные затраты. Однако усложнение архитектуры PaaS-платформ, включающих десятки микросервисов, интеграций и внешних зависимостей, приводит к росту требований к качеству и надежности [2].

Дополнительные сложности создают разнообразие сценариев использования, различия между типами инсталляций (публичное облако, частное облако, standalone-режим), а также высокая стоимость вычислительных ресурсов, необходимая для длительных тестовых прогонов [3]. В этих условиях возрастает потребность в системной методике автоматизированного тестирования, позволяющей обеспечить воспроизводимость, масштабируемость и снижение трудозатрат [4].

Целью работы является снижение трудоемкости обеспечения качества облачных PaaS-сервисов в распределённых инфраструктурах за счет внедрения методики автоматизации тестирования пользовательских сценариев (ПСИ), реализованной с использованием стека технологий на языке Python.

Задачи, необходимые для достижения цели представлены ниже.

1. Изучить существующие программные методики испытаний.
2. Представить сравнительный анализ подходов запуска тестов в виртуализированной инфраструктуре с целью выявления их достоинств и недостатков.
3. Реализовать фундамент тестового фреймворка, учитывая особенности ПМИ и способов инсталляции PaaS.
4. Покрыть тестами продукты Kafka, PostgreSQL, Trino для осуществления автоматизации прохождения приемо-сдаточных испытаний, обеспечения качества за счет регресс тестирования.
5. Настроить автоматические запуски тестов в CI/CD, чтобы находить ошибки на ранних этапах, ускорять выпуск продукта и отображать динамику исправлений дефектов.
6. Настроить механизм динамического формирования дочерних цепочек сборки (выбор окружения, продукта, тест кейса).
7. Осуществить генерацию Allure отчетов и интеграцию с TMS.

Предлагаемая в работе методика направлена на создание системного, воспроизводимого и масштабируемого подхода к автоматизации ПСИ, ранее выполнявшихся вручную. До внедрения автоматизации прохождение сценариев занимало значительное количество человеко-часов, сопровождалось высокой вероятностью пропуска ошибок и не позволяло выполнять полный регрессионный цикл при каждом изменении кода [5].

Методика применима к комплексным распределенным системам с гетерогенной архитектурой, проектам с длительным и дорогостоящим прогоном тестового набора, а также системам с отложенной согласованностью данных и асинхронными процессами [6].

На текущий момент удалось выполнить все перечисленные задачи, а перечень покрытых тестовых сценариев представлен в Таблице 1.

До внедрения методики длительность прохождения ПСИ по одному продукту составляла 16 человеко-часов. Для прохождения ПСИ по 5 продуктам на 3 окружениях требовалось 240 человеко-часов. После внедрения длительность одного тестового прогона сократилась до 2 часов 30 минут, а полный цикл по 5 продуктам на 3 окружениях занимает всего 2,5 человеко-часа. Автоматизация обеспечивает значительную экономию времени и ресурсов по сравнению с ручным тестированием.

Таблица 1 – Матрица функционального покрытия

Категория тестовых сценариев	Проверяемая функция	Kafka	Trino	Postgres
Управление кластером	Создание, удаление кластера	+	+	+
	Доступность кластера	+	+	+
Управление пользователями и ролями	Работа с пользователями	+	+	+
	Работа с ролями	+	+	+
Сетевые настройки	Изменение IP / Port	+	+	+
Масштабирование	Вертикальное масштабирование	+	+	+
	Горизонтальное масштабирование	+	+	+
Интеграции	Kafka Bridge	+	-	-
	Kafka UI	+	-	-
	Подключение к HIVE, PostgreSQL, MySQL, ClickHouse, Iceberg	-	+	-
Конфигурация	Изменение настроек продукта	+	+	+
Бэкапы	Полный бэкап / восстановление	-	-	+
	Инкрементальный бэкап	-	-	+

Разработанная методика автоматизации тестирования пользовательских сценариев PaaS-платформы на базе Python-стека доказала свою эффективность: трудоемкость сократилась в десятки раз, обеспечена воспроизводимость и масштабируемость, а интеграция в CI/CD позволила выполнять полный регрессионный цикл при каждом изменении. Подход применим к комплексным распределенным системам и решает ключевые проблемы качества облачных инфраструктур [7]. Внедрение подобных методик является перспективным направлением для повышения качества и скорости доставки облачных сервисов в современных ИТ-организациях.

ЛИТЕРАТУРА

1. Ткаченко, А. А. Метод объединения данных для автоматизированного анализа и управления ИТ-инфраструктурой компании / А. А. Ткаченко, И. В. Никифоров, В. А. Головин // Неделя науки ИКНХ : Материалы докладов Всероссийской научно-практической конференции, Санкт-Петербург, 14 апреля – 16 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 27-29. – EDN LYYWWI.
2. Deng S., et al. Cloud-native computing: A survey from the perspective of services. *Proceedings of the IEEE*. 2024. Vol. 112, No. 1. pp. 12–46.
3. Несмеянов, П. П. Основные проблемы в облачном тестировании / П. П. Несмеянов // Международный журнал гуманитарных и естественных наук. – 2023. – № 6-3(81). – С. 136-138. – DOI 10.24412/2500-1000-2023-6-3-136-138. – EDN PKUNDM.
4. Бевзенко, С. А. Особенности нагрузочного тестирования в облачных сервисах / С. А. Бевзенко // Наукосфера. – 2023. – № 8-1. – С. 73-77. – EDN FVZUTM.
5. Преимущества и стратегии облачного тестирования / Н. П. Барабак, М. М. Лукьяев, А. Р. Белова, А. Р. Хабаров // Информационные ресурсы и системы в экономике, науке и образовании: сборник статей XIV Международной научно-практической конференции, Пенза, 29–30 апреля 2024 года. – Пенза: Автономная некоммерческая научно-методическая организация «Приволжский Дом знаний», 2024. – С. 15-18. – EDN NVTLQY.
6. Blokland K., Mengerink J., Pol M. *Testing Cloud Services: How to Test SaaS, PaaS & IaaS*. Rocky Nook, Inc., 2013.
7. Сафронов, Д. Автоматизация подготовки окружения в среде Kubernetes с помощью расширения инструмента Kind для тестирования функциональности горизонтального масштабирования приложений/ Д. Сафронов, И. В. Никифоров // Современные технологии в теории и практике программирования: Сборник материалов конференции, Санкт-Петербург, 26 апреля 2022 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2022. – С. 202-204. – EDN WYLSKN.

УДК 004.453

Г.В. Сырбу, С.К. Федорова (1 курс магистратуры),
О.Г. Малеев, к.т.н., доцент

ПЛАТФОРМА КОНТЕКСТНОГО ЧАТ-БОТА НА ОСНОВЕ GRAPHRAG ДЛЯ РАБОТЫ С БОЛЬШИМИ ОБЪЕМАМИ ДОКУМЕНТОВ

Современные системы обработки естественного языка на основе больших языковых моделей (Large Language Models, LLM) позволяют эффективно решать задачи анализа и генерации текста. Однако ограниченный размер контекстного окна существенно затрудняет их применение для работы с большими объемами документов [1].

Одним из эффективных подходов к решению данной проблемы является архитектура Retrieval-Augmented Generation (RAG), объединяющая генеративные модели и механизмы поиска релевантной информации во внешних источниках [2]. В рамках данного подхода документы индексируются, а в модель передаются только наиболее релевантные фрагменты.

Развитием RAG является GraphRAG, использующий граф знаний для моделирования связей между сущностями и фрагментами текста, что позволяет учитывать семантические зависимости и улучшать качество ответов [3].

Целью работы является разработка масштабируемой платформы контекстного чат-бота, способного эффективно работать с большими объемами документов за счёт использования GraphRAG и распределённой архитектуры.

Для достижения цели решаются следующие задачи:

1. Анализ подходов построения RAG-систем.
2. Разработка архитектуры платформы с поддержкой LightRAG как основной реализации и GraphRAG как расширения для работы с графами знаний.
3. Реализация индексации документов с использованием векторного поиска и графа знаний.
4. Разработка масштабируемой системы обработки запросов.
5. Обеспечение кластерного развёртывания.

Разработанная система представляет собой распределённую платформу, включающую backend-сервис, асинхронные модули индексации и обработки запросов (workers), взаимодействующие через брокера сообщений, а также инфраструктурные компоненты. Backend реализован на Java (Spring Boot), предоставляет REST API и управляет запуском Kubernetes Job для индексации и обработки данных, обеспечивая управление контекстами и чат-сессиями. Взаимодействие компонентов осуществляется через брокера сообщений Kafka, а история диалогов и служебные метрики (например, счётчики активных запросов) хранятся в Redis. Архитектура разработанной системы приведена на рисунке 1.

Модуль индексации выполняет разбиение документов на фрагменты и вычисление эмбедингов, которые сохраняются в PostgreSQL с расширением pgvector, выступающий основным векторным хранилищем для реализации семантического поиска [4]. Формируется граф знаний, отражающий связи между сущностями и фрагментами документов, при этом используется одна из реализаций — LightRAG или GraphRAG в зависимости от выбранной конфигурации системы.

Обработка запросов выполняется Query Workers, которые получают запросы через Kafka, извлекают релевантные данные из векторного индекса и графа и передают их в языковую модель. Поддерживаются различные режимы RAG, что позволяет адаптировать стратегию поиска.

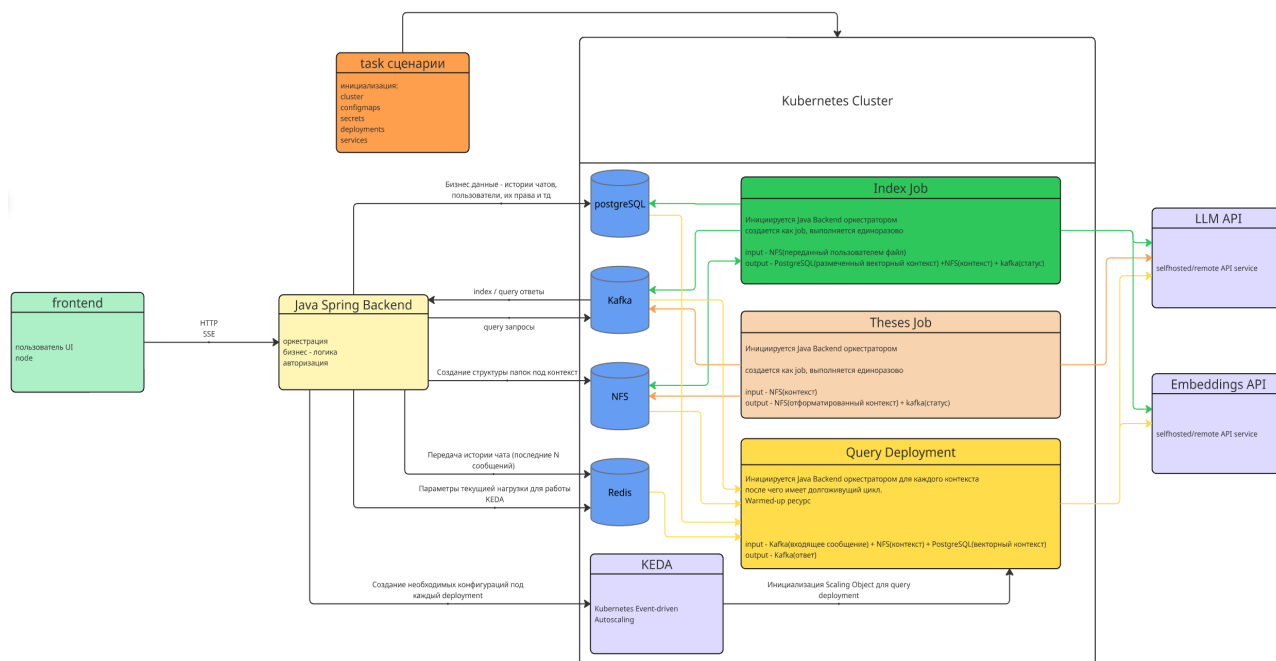


Рисунок 1 – Архитектура системы

Система ориентирована на кластерное развёртывание с использованием Kubernetes, что обеспечивает масштабируемость и отказоустойчивость. Для балансировки нагрузки используется Kafka, а автоматическое масштабирование реализовано с помощью KEDA на основе метрик очередей Kafka (Kafka lag) [5].

Предложенная архитектура обеспечивает эффективную обработку запросов к большим объемами документов и повышает качество ответов за счёт использования графовых связей.

ЛИТЕРАТУРА

1. Vaswani A., Shazeer N., Parmar N., et al. Attention Is All You Need // Advances in Neural Information Processing Systems. – 2017. – Vol. 30. – URL: <https://arxiv.org/abs/1706.03762> (дата обращения: 17.03.2026).
2. Lewis P., Perez E., Piktus A., et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – URL: <https://arxiv.org/abs/2005.11401> (дата обращения: 17.03.2026).
3. Edge D., Trinh H., Cheng N., et al. From Local to Global: A GraphRAG Approach to Query-Focused Summarization // arXiv preprint. – 2024. – URL: <https://arxiv.org/abs/2404.16130> (дата обращения: 17.03.2026).
4. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs // IEEE Transactions on Big Data. – 2019. – Vol. 7. – No. 3. – P. 535–547. – DOI: 10.1109/TBDATA.2019.2921572.
5. Burns B., Grant B., Oppenheimer D., Brewer E., Wilkes J. Kubernetes: Up and Running. – Sebastopol: O'Reilly Media, 2019. – 329 p.

УДК 004.047

Е.И. Старунская (1 курс магистратуры),
И.В. Никифоров, к.т.н., доцент,
М.С. Соломонов, ассистент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА АКТИВНОСТИ РЕПОЗИТОРИЕВ GITHUB

GitHub является крупнейшей платформой для хостинга проектов с системой контроля версий Git и сервисом совместной разработки программного обеспечения. Анализ активности репозиторий на GitHub представляет научный интерес для нескольких областей [1]. Во-первых, понимание жизненного цикла проектов позволяет выявить факторы, влияющие на их долговечность и успешность. Исследование [2] показало, что более половины проектов прекращают активность в течение первых четырёх лет существования, что подчёркивает важность изучения механизмов выживаемости open source проектов. Во-вторых, динамика популярности языков программирования позволяет прогнозировать тренды в индустрии разработки программного обеспечения [3].

Данное исследование опирается на обширный датасет (2398913 репозиторий, 587 Мб), собранный с помощью GitHub API, и охватывает период с 2008 по 2025 годы, что позволяет проследить полную историю развития платформы и выявить ключевые закономерности в активности проектов.

Цель работы – анализ репозиторий GitHub посредством разработки веб-приложения на базе Django для автоматизированного сбора, хранения и анализа данных репозиторий GitHub. Для достижения данной цели были поставлены следующие задачи, перечисленные ниже.

1. Рассмотреть распределение активных репозиторий по годам их создания.
2. Выявить преобладающие языки программирования в активных сейчас репозиториях, которые созданы в 2012 и 2020 годах для оценки языковых трендов в долгоживущих проектах.
3. Проанализировать распределение репозиторий по году последнего коммита.
4. Исследовать динамику создания репозиторий за весь период существования платформы.
5. Разработать веб-приложение на базе Django для сбора и анализа репозиторий.

6. Сделать обобщенный вывод для понимания трендов, тенденций и актуальных направлений развития проектов с открытым исходным кодом.

Для сбора данных о репозиториях GitHub был использован API GitHub. Для каждого репозитория собирались следующие поля: уникальный идентификатор (id), название (full_name), язык программирования (language), дата создания (created_at) и дата последнего коммита (pushed_at). Данные собирались с сортировкой по звездам.

Схема приложения представлена на рисунке 1. Приложение разделено на три Django-приложения (apps). За сбор данных отвечает приложение scraper, за аналитику — приложение analysis, веб-интерфейс и модель данных находятся в приложении main [4]. Пользователь через браузер отправляет POST-запрос на эндпоинт /collect. Класс RepoViewSet (views.py, приложение main) передаёт управление классу ScraperService. ScraperService инициализирует класс App, который создаёт асинхронные задачи по месяцам за заданный период. Для получения токенов доступа используется класс TokenProvider. Класс Fetcher выполняет постраничные асинхронные HTTP-запросы к GitHub API через библиотеку httpx, валидирует ответ через Pydantic-модель SearchResult и при получении ошибки 403 Forbidden выбрасывает исключение ApiRateException для смены токена. Полученные данные сохраняются в базу данных PostgreSQL через Django ORM.

При обращении к одному из четырёх аналитических эндпоинтов (GET /research/*) RepoViewSet вызывает AnalyseService. Сервис вызывает методы класса Analyzer, который с помощью функций Django ORM извлекает статистику из базы данных: распределение активных репозиториях по годам создания, распределение последних коммитов и топ языков программирования. Полученные данные передаются в PictureGenerator, который формирует гистограммы и круговые диаграммы с помощью Matplotlib и сохраняет их в PNG-файлы. AnalyseService возвращает в RepoViewSet путь к изображению и данные, после чего view рендерит HTML-шаблон с графиком и возвращает его браузеру.

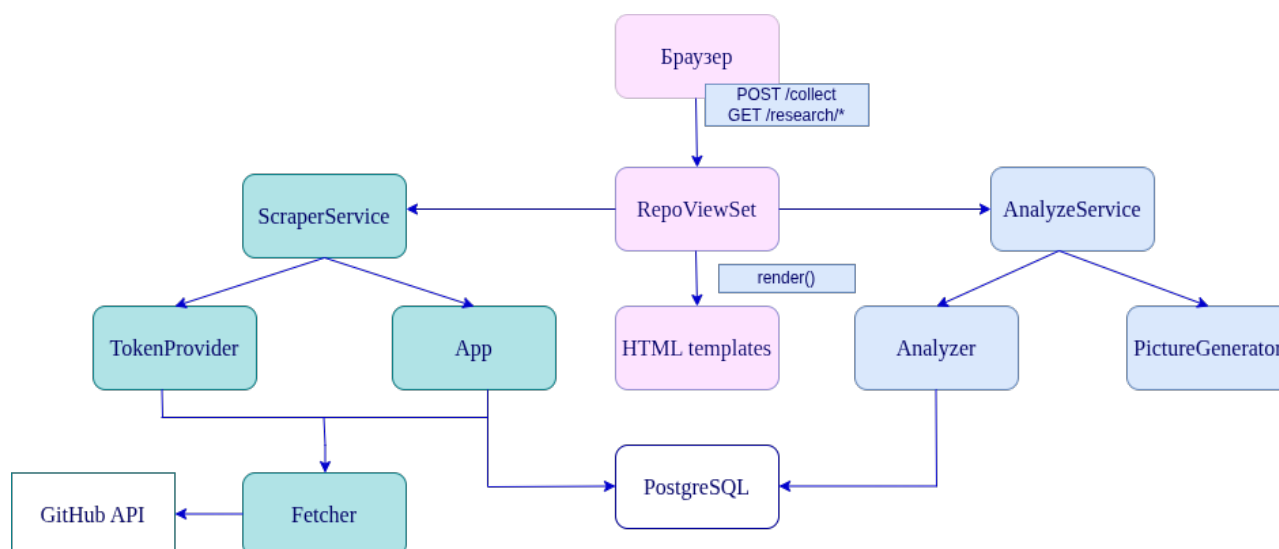


Рисунок 1 – Схема работы приложения

Разработанное веб-приложение автоматизирует сбор данных через GitHub API и предоставляет интерактивный интерфейс для визуализации четырёх видов аналитики.

Выявлен значительный рост числа создаваемых репозиторий в период 2012-2013 годов (рисунок 2 а). Начиная с 2013 года, количество ежегодно создаваемых репозиторий стабилизировалось [5]. Анализ показал, что существуют проекты, активные с момента основания платформы в 2008 году (17-летний возраст) (рисунок 2 б). Количество долгоживущих активных репозиторий увеличивается с каждым годом, что демонстрирует устойчивость значительной части open source проектов. Критерием активности служило наличие коммита после 1 января 2024 года.

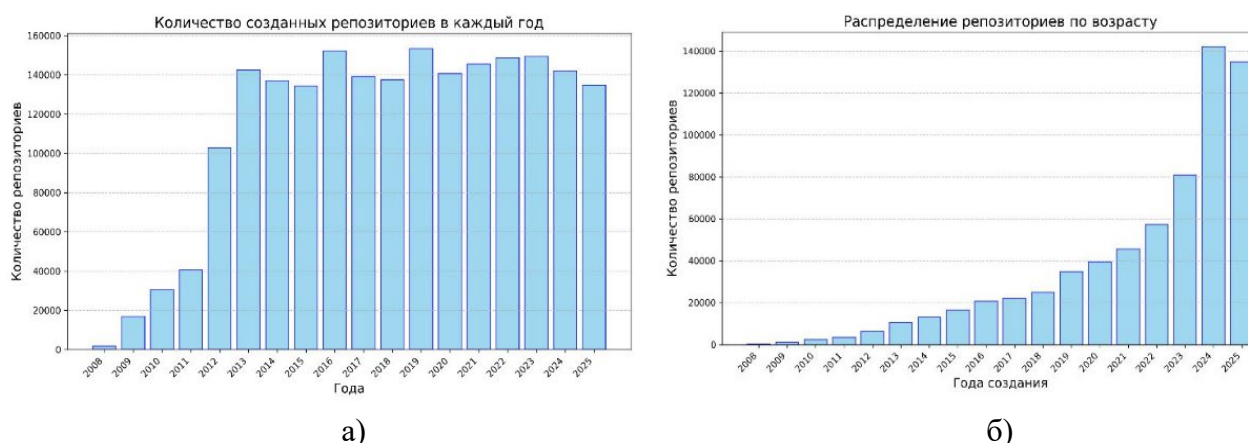


Рисунок 2 – Распределение репозиторий: а) по возрасту активных репозиторий; б) количество созданных репозиторий по годам

Для репозиторий, активных с 2012 года, характерно доминирование Python (15,5%), PHP (14,6%) и JavaScript (14,0%), что отражает популярность веб-разработки. В репозиториях, созданных с 2020 года, доля Python возросла до 26%, что подтверждает выводы исследования [5, 6] о его лидерстве в области Data Science и машинного обучения. Значительный рост продемонстрировал TypeScript (13%), отражающий переход от JavaScript к типизированному программированию. Доля Java снизилась с 10,2% до 7,4%, что связано с появлением альтернативных JVM-языков (Kotlin, Scala).

Анализ даты последнего коммита выявил минимальную активность завершения проектов в начальный период существования платформы (2008-2010 годы) с последующим ростом. Значительное количество репозиторий с последним коммитом в 2025 году (более 400 тысяч) свидетельствует о высокой текущей активности платформы.

Полученные результаты характеризуют экосистему популярных open source проектов на GitHub и отражают тренды среди активно развиваемых репозиторий [7]. Выводы о доминировании Python и роста TypeScript верны для верхней части распределения по популярности; полная популярность языков программирования может отличаться при анализе всей совокупности репозиторий.

Выявленные закономерности в жизненном цикле репозиторий и динамике популярности языков программирования могут быть использованы для прогнозирования технологических трендов и принятия решений о выборе технологического стека.

ЛИТЕРАТУРА

1. Методы, алгоритмы и архитектуры распределенной обработки больших данных / И. В. Никифоров, О. А. Юсупова, Н. В. Воинов, А. Д. Ковалев. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2023. – 194 с. – ISBN 978-5-7422-8461-1. – EDN DFTKEX.
2. Ait A., Izquierdo J. L. C., Cabot J. An empirical study on the survival rate of GitHub projects Proceedings of the 19th International Conference on Mining Software Repositories (MSR). – ACM, 2022. – P. 365–375.
3. Big data processing system for analysis of GitHub events / N. Voinov, K. Rodriguez Garzon, I. Nikiforov, P. Drobintsev // Proceedings of 2019 22nd International Conference on Soft Computing and Measurements, SCM 2019 : 22, St. Petersburg, 23–25 мая 2019 года. – St. Petersburg, 2019. – P. 187–190. – DOI 10.1109/SCM.2019.8903782. – EDN RDBGKV.
4. Программные инструменты обработки и визуализации данных. Elasticsearch, Logstash, Kibana, Grafana, Prometheus / И. В. Никифоров, О. А. Юсупова, Н. В. Воинов [и др.]. – Санкт-Петербург : Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. – 140 с. – ISBN 978-5-7422-8075-0. – DOI 10.18720/SPBPU/2/id23-74. – EDN KPGDVS.
5. Gousios G., Spinellis D. GHTorrent: GitHub's data from a firehose // Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR). – 2012. – P. 12–21.

6. Tambad S., Nandwani R., McIntosh S. K. Analyzing programming languages by community characteristics on Github and StackOverflow //arXiv preprint arXiv:2006.01351. – 2020.
7. Hudson Borges, Andre Hora, Marco Tulio Valente, "Understanding the Factors that Impact the Popularity of GitHub Repositories", 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2016.

УДК 004.65

И.В. Устинов (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент

МЕТОДИКА МИГРАЦИИ ТЕЛЕМЕТРИЧЕСКИХ ДАННЫХ ИЗ СТРОКОВОЙ РЕЛЯЦИОННОЙ СУБД MYSQL В КОЛОНОЧНОЕ ХРАНИЛИЩЕ CLICKHOUSE

В телеметрических системах основную нагрузку на хранилище со временем начинают формировать не транзакционные операции, а длительное хранение координат и событий, выборки истории по устройствам и аналитические запросы по большим временным диапазонам. Как отмечается в работах по архитектурам обработки больших данных, при росте объема информации возникает необходимость разделения транзакционного и аналитического контуров, а также использования специализированных подходов к хранению и обработке данных [1]. На начальном этапе такие системы часто строятся на реляционной базе данных MySQL, поскольку эта СУБД проста в эксплуатации, хорошо поддерживается прикладными библиотеками и подходит для базового OLTP-сценария [2]. Однако при росте объема исторических данных до миллиардов строк и сотен гигабайт появляются типичные проблемы: увеличивается время выполнения выборок, растет объем индексов, усложняется аналитическая обработка и повышается стоимость хранения.

В рамках работы рассматривается промышленная микросервисная система, в которой хранятся события и координаты устройств. Основной проблемой стала работа с историческими данными в MySQL: для ряда запросов по событиям устройства и временным диапазонам время ответа достигло десятков секунд и более, что делало использование такого хранилища медленным для аналитических и сервисных задач. В качестве целевого хранилища выбран ClickHouse – колоночная СУБД, ориентированная на высокую скорость обработки больших объемов данных и аналитические сценарии [3]. В связи с этим была поставлена задача переноса исторического и аналитического контура данных из MySQL в ClickHouse без остановки системы и без потери данных.

Цель работы заключается в разработке и апробации методики миграции телеметрических данных из строковой реляционной СУБД MySQL в колоночное хранилище ClickHouse в распределенной микросервисной системе. Практический смысл этой задачи состоит в организации безопасного перехода, при котором система продолжает работать, новые данные не теряются, а чтение может быть переключено на новое хранилище только после подтверждения корректности миграции данных. Подобный поэтапный переход с возможностью контролируемого возврата к исходной конфигурации применяется и в других задачах миграции вычислительной инфраструктуры, где критичны минимизация риска и сохранение работоспособности продуктивного контура [4].

Предлагаемая методика состоит из нескольких последовательных этапов, представленных на рисунке 1. На первом этапе исторические данные заранее выгружаются из MySQL в виде дампов и загружаются в ClickHouse пакетно. Это позволяет перенести основной объем телеметрии до включения нового хранилища в рабочий контур. На втором этапе в продуктивную систему выкатывается новая версия приложения, после чего начинается параллельная запись новых данных одновременно в MySQL и ClickHouse. При этом чтение в этот момент еще остается на стороне MySQL, что позволяет не затрагивать пользовательские и служебные сценарии до завершения проверки. На третьем этапе выполняется дозагрузка так называемого «хвоста» – данных, которые появились в MySQL после снятия дампа, но до

начала параллельной записи. Это важный момент, поскольку именно на этом интервале обычно возникает риск появления разрыва между старым и новым хранилищем. После дозагрузки выполняется сверка данных и только затем чтение переключается на ClickHouse. Некоторое время запись в MySQL и ClickHouse продолжается параллельно как страховочный механизм, после чего MySQL выводится из основного контура. Такой подход согласуется с практикой поэтапной миграции и балансировки данных, при которой основной объем переносится заранее, а критический переходный интервал обрабатывается отдельно и под контролем [5].

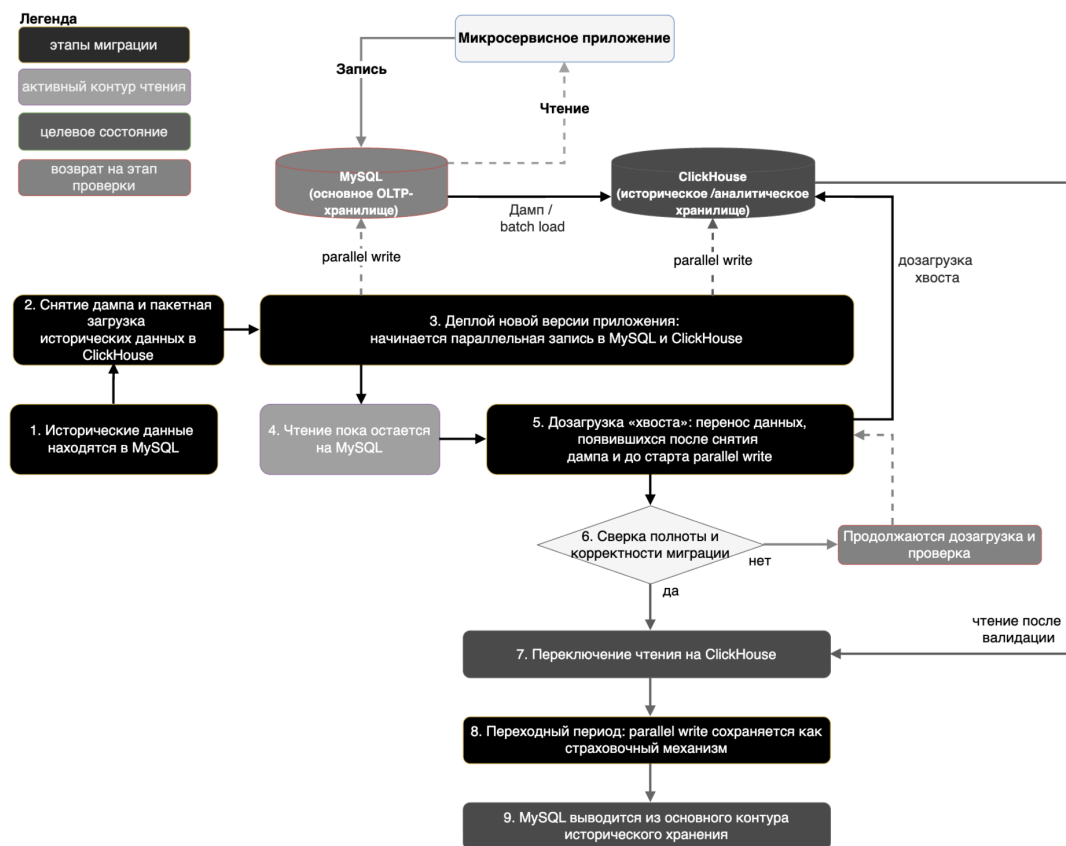


Рисунок 1 – Методика миграции телеметрических данных из MySQL в ClickHouse

В работе учитываются и особенности трансформации данных. Для таблицы событий одно из полей в MySQL хранилось в бинарном виде и на стороне приложения интерпретировалось как шестнадцатеричная строка. При миграции в ClickHouse это поле было решено хранить сразу в hex-представлении, чтобы данные были читаемыми и не требовали дополнительного преобразования при проверке и анализе. Для таблицы координат применяется доменная оптимизация: последовательности точек могут быть сокращены алгоритмом Дугласа-Пекера, предназначенным для уменьшения числа точек линии при сохранении ее формы в пределах заданного отклонения [6]. Это позволяет уменьшить число хранимых координат при сохранении формы маршрута в заданных пределах точности. Такой подход особенно важен для систем, в которых накопление координат идет непрерывно и основной рост объема данных связан именно с треками устройств.

Оценка характеристик демонстрационной версии проводилась на реальных структурированных телеметрических данных – событиях и координатах устройств. Объем таблицы событий составил 2,86 млрд строк. Суммарный объем хранения в MySQL составил 244,9 GB, тогда как в ClickHouse тот же объем событий занял 41,5 GB, то есть почти в 6 раз меньше. Такое сокращение объема согласуется с особенностями колоночного формата хранения и эффективного сжатия данных в ClickHouse [3, 7]. Для загрузки исторических дампов общим объемом 44,32 GB потребовалось около 10 часов 7 минут, а средняя скорость загрузки составила около 78 тыс. строк в секунду. В рассматриваемой реализации перенос

исторических данных выполнялся через дампы, которые стали резервной точкой восстановления. Согласованность данных в переходный период обеспечивалась параллельной записью в обе СУБД и последующей дозагрузкой инкрементального «хвоста». Дополнительно были измерены задержки выполнения типовых запросов. Для аналитических сценариев по событиям устройства ClickHouse показал существенный выигрыш по сравнению с MySQL: выборка последних 200 событий устройства с фильтрацией в одном из сценариев выполнялась за 0,08 с в ClickHouse против 54,17 с в MySQL. Для отдельных запросов ускорение достигло сотен и тысяч раз. При этом для коротких точечных выборок по небольшому объему строк MySQL мог показывать сопоставимые или меньшие задержки, что соответствует ее назначению как OLTP-СУБД [2]. Таким образом, сравнение показало, что переход на ClickHouse оправдан прежде всего для исторического и аналитического контура телеметрии (см. рисунок 2).

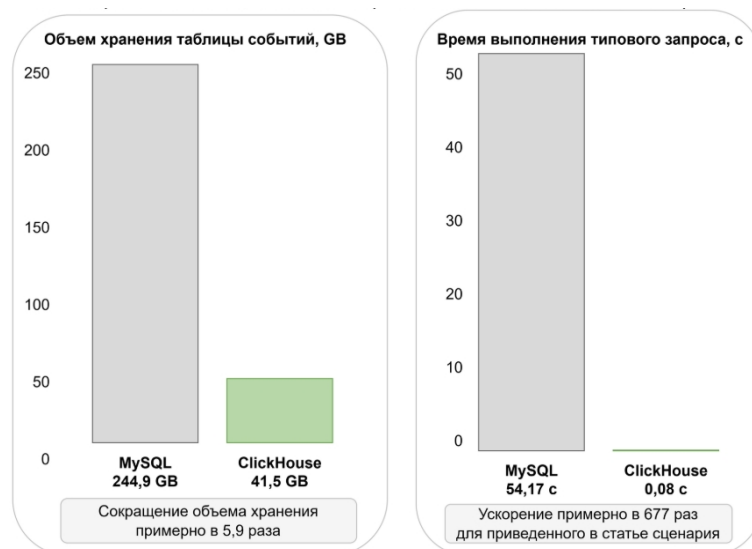


Рисунок 2 – Преимущества перехода с MySQL на ClickHouse

Предложенная методика позволяет перенести телеметрические данные в ClickHouse без длительного простоя, с контролем полноты и корректности, а также с безопасным поэтапным переключением чтения. Практические результаты показывают, что для систем с большими объемами координат и событий использование ClickHouse позволяет существенно снизить объем хранения и ускорить выполнение аналитических запросов. Это делает такой подход применимым для промышленных распределенных микросервисных систем, где исторические данные должны не только сохраняться, но и оставаться доступными для быстрой обработки.

ЛИТЕРАТУРА

1. Никифоров И. В., Юсупова О. А., Воинов Н. В., Ковалев А. Д. Методы, алгоритмы и архитектуры распределенной обработки больших данных : учебное пособие. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2023. – 194 с. – ISBN 978-5-7422-8461-1. – DOI 10.18720/SPBPU/2/id23-715.
2. MySQL 8.4 Reference Manual. Chapter 17. The InnoDB Storage Engine. [Электронный ресурс]. – Режим доступа: официальная документация MySQL. – Дата обращения: 15.03.2026.
3. ClickHouse Docs. Architecture Overview. [Электронный ресурс]. – Режим доступа: официальная документация ClickHouse. – Дата обращения: 15.03.2026.
4. Петров А. А., Никифоров И. В., Устинов С. М. Алгоритм миграции ESXi-кластеров между разными vCenter серверами с возможностью возврата к исходной конфигурации // Информационно-управляющие системы. – 2022. – № 2(117). – С. 20-31. – DOI 10.31799/1684-8853-2022-2-20-31. – EDN FIGLAF.
5. Петров А. А., Никифоров И. В., Зайцев И. В. Балансировка данных для программы объединения систем хранения данных // Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции, Санкт-Петербург, 22 апреля 2021 года.

– Санкт-Петербург : Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2021. – С. 216-218. – EDN TUCTXK.

6. Douglas D. H., Peucker T. K. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature // The Canadian Cartographer. – 1973. – Vol. 10. – P. 112-122. – DOI 10.3138/FM57-6770-U75U-7727.

УДК 004.89

А.М. Чернецов (4 курс бакалавриата),
А.В. Гончаров, ст. преподаватель

СОВРЕМЕННЫЕ СПОСОБЫ ВЗАИМОДЕЙСТВИЯ С API БЕЗ ЯВНОГО ДЕКЛАРАТИВНОГО УПРАВЛЕНИЯ НАМЕРЕНИЯМИ

Вместе со стремительным развитием искусственного интеллекта возникает потребность в поиске областей для его максимально эффективного и уместного применения [1]. Одной из таких сфер является разработка программного обеспечения, где использование генеративных моделей способно значительно оптимизировать рутинные действия участников проектных команд. Транскрибация и резюмирование встреч, интеллектуальные помощники по поиску информации во внутренней документации, системы генерации с расширенным поиском (RAG) [2] – эти и многие другие инструменты базируются на глубокой интеграции с технологиями искусственного интеллекта.

Среди возможных способов взаимодействия с ИИ особую популярность приобрели диалоговые интерфейсы(чаты), где модель выступает в качестве интеллектуального участника диалога с пользователем [3]. Такой формат значительно упрощает взаимодействие с системами, обладающими сложным и многоуровневым интерфейсом. К примерам таких продуктов, в первую очередь, можно отнести системы управления проектами и баг-трекинга.

Однако техническая реализация подобного взаимодействия долгое время оставалась нетривиальной задачей. До недавнего времени для интеграции ИИ требовалось определение доступного набора пользовательских сценариев и реализация жестко заданных намерений (интентов) в рамках целеориентированных диалоговых систем [4]. Подобная архитектура была крайне сложна в поддержке и неэффективна при расширении доступных функций. С развитием языковых моделей на смену этому подходу пришел новый способ взаимодействия посредством вызова инструментов (Tool Calling или Function Calling) [5]. Это позволило удобнее интегрироваться с внешними сервисами и обеспечить более широкую функциональность. Но в новом подходе также были свои сложности. Для взаимодействия с ИИ использовались специфические программные библиотеки, разрабатываемые либо open-source сообществом, либо крупными компаниями под собственные генеративные модели. Потребность в изучении различных реализаций, адаптация спецификаций при смене моделей и необходимость в собственной реализации части функционала – все это ограничивало скорость разработки программных средств с использованием ИИ.

Таким образом, целью данной работы является создание масштабируемой архитектуры системы управления проектами с AI-ассистентом, обеспечивающей низкую связность между бизнес-логикой и модулем искусственного интеллекта.

Для достижения этой цели необходимо решение задач, перечисленных ниже.

1. Анализ существующих подходов к интеграции больших языковых моделей и вызовов внешних API.
2. Проектирование распределенной архитектуры с применением протокола MCP для разделения зон ответственности.
3. Реализация серверной части (ядра провайдера данных) на платформе Java и изолированного AI-агента на языке Python.
4. Оценка эффективности предложенного архитектурного решения.

В качестве основного подхода к решению поставленных задач был выбран протокол MCP [6], направленный на стандартизацию и унификацию процесса эффективного использования искусственного интеллекта в разрабатываемых программных комплексах. Благодаря введению понятных ролей, стандартизированному интерфейсу взаимодействия, вариативному и гибкому транспорту для общения, новый протокол оптимизировал работу с языковыми моделями. MCP обеспечивает широкие возможности для безопасной реализации и масштабирования предоставляемых инструментов.

Базовыми терминами рассматриваемого протокола являются три понятия [7]:

- MCP host: приложение с искусственным интеллектом, которое взаимодействует с одним или несколькими MCP client для получения внешних данных или выполнения различных функций;
- MCP client: компонент системы, который предоставляет MCP host подключение к MCP server. Он также обеспечивает приложение с искусственным интеллектом необходимым контекстом (список доступных инструментов, ответы от сервера и другие опциональные функции);
- MCP server: программный комплекс, предоставляющий контекст для MCP host через MCP client, который выступает в роли связующего звена.

Примером архитектуры с использованием протокола MCP (рисунок 1) может служить бакалаврская выпускная квалификационная работа на тему «Разработка серверной части системы управления командными проектами с AI-ассистентом».

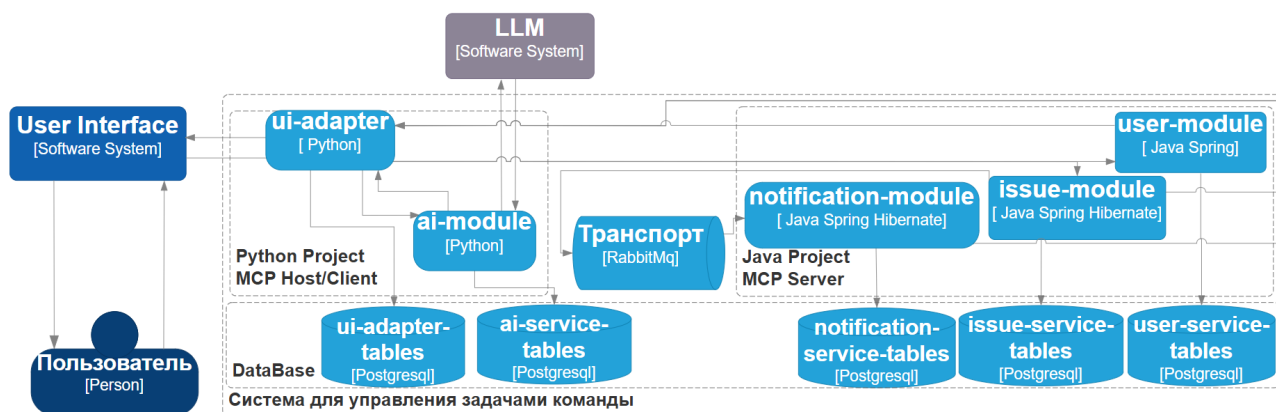


Рисунок 1 – Архитектура с использованием MCP

Роль сервера (MCP Server) выполняет базовое ядро системы на платформе Java. Данный модуль инкапсулирует бизнес-логику работы с задачами (issue-module), пользователями (user-module) и уведомлениями (notification-module), транзакционное взаимодействие с реляционной базой данных и механизмы авторизации. Важной архитектурной особенностью является то, что ядро не содержит логики обработки естественного языка. Вместо этого оно выступает надежным провайдером данных, динамически предоставляя схему доступных методов (например, создание задач, поиск участников команды, изменение статусов) в стандартизированном формате.

Роль клиента и хоста (MCP host, client) берет на себя изолированный сервис, разработанный на языке Python. Выбор данного языка обоснован его статусом индустриального стандарта в сфере машинного обучения и работы с LLM. Главной задачей данного сервиса является управление процессами взаимодействия с искусственным интеллектом. Он принимает неструктурированный текстовый запрос пользователя, запрашивает у Java-ядра актуальный манифест доступных инструментов по протоколу MCP и транслирует их вместе с контекстом диалога в большую языковую модель.

Подобное архитектурное решение позволяет добавлять новые функции в систему управления проектами без необходимости модификации AI-агента – новые методы ядра автоматически подхватываются через обновляемый манифест. Это значительно ускоряет цикл

разработки, снижает связность между микросервисами и обеспечивает высокую масштабируемость программного комплекса.

ЛИТЕРАТУРА

1. Hou X. et al. Large language models for software engineering: A systematic literature review //ACM Transactions on Software Engineering and Methodology. – 2024. – Т. 33. – №. 8. – С. 1-79.
2. Lewis P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks //Advances in neural information processing systems. – 2020. – Т. 33. – С. 9459-9474.
3. Ni J. et al. Recent advances in deep learning based dialogue systems: A systematic survey //Artificial intelligence review. – 2023. – Т. 56. – №. 4. – С. 3055-3155.
4. Zhang Z. et al. Recent advances and challenges in task-oriented dialog systems //Science China Technological Sciences. – 2020. – Т. 63. – №. 10. – С. 2011-2027.
5. Qin Y. et al. Toolllm: Facilitating large language models to master 16000+ real-world apis //arXiv preprint arXiv:2307.16789. – 2023.
6. Hou X. et al. Model context protocol (mcp): Landscape, security threats, and future research directions //ACM Transactions on Software Engineering and Methodology. – 2025.
7. Model Context Protocol. [Электронный ресурс] Режим доступа: <https://modelcontextprotocol.io/docs/learn/architecture> (дата обращения 12.03.2026)

УДК 004.457

О.Р. Чумакова (2 курс магистратуры),
А.Д. Ковалев, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ТЕХНИЧЕСКОГО И UX АУДИТА ВЕБ-САЙТОВ С ПОМОЩЬЮ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Современные веб-интерфейсы представляют собой сложные динамические системы, качество взаимодействия с которыми напрямую определяет успех цифровых продуктов — конверсию, удержание пользователей и лояльность аудитории [1, 2]. Однако существующие методы оценки пользовательского опыта либо опираются на трудоёмкий экспертный анализ, либо ограничиваются поверхностной автоматизированной проверкой отдельных страниц [3]. Инструменты, обеспечивающие глубокий анализ, требуют установки дополнительного ПО на проверяемый сайт или имеют высокую стоимость корпоративных лицензий, что делает их недоступными для малого бизнеса, стартапов и исследовательских задач [4].

Один из способов преодоления указанных ограничений — создание открытого программного обеспечения, сочетающего методы гибридного сбора данных и машинного обучения для целостного аудита веб-сайта как единой системы [5]. За счёт работы исключительно с публичными данными и отсутствия необходимости в доступе к закрытому коду проверяемого ресурса такой подход позволяет анализировать любые веб-сайты, автоматически выявлять проблемы навигации, визуальной согласованности и доступности, а также формировать конкретные, технически реализуемые рекомендации.

Таким образом, целью данной работы является снижение трудоёмкости и повышение эффективности процесса технического и UX-аудита веб-сайтов за счёт разработки подхода, интегрирующего статический парсинг, динамическую симуляцию поведения пользователей и методы машинного обучения.

Для достижения поставленной цели необходимо решение следующих задач:

1. Проведение системного сравнительного анализа существующих решений и формулировка требований к предлагаемому подходу.
2. Разработка теоретической модели многоуровневого аудита.
3. Проектирование и реализация модульной программной архитектуры системы.
4. Обоснование выбора технологического стека и средств реализации.
5. Экспериментальная проверка эффективности разработанного прототипа.

Ключевым компонентом разрабатываемого подхода является гибридный сбор данных, объединяющий статический парсинг структуры сайта для извлечения DOM-дерева и визуальных атрибутов, а также динамическую симуляцию поведения для генерации поведенческих метрик. Архитектура системы, представленная на рисунке 1, реализует сквозной конвейер обработки данных.

На первом этапе выполняется сбор данных об интернет-ресурсе. Для заданного URL система извлекает HTML код и CSS стили, создаёт скриншоты страниц, проверяет работоспособность ссылок и анализирует структуру ресурса. Параллельно, с помощью динамической симуляции могут собираться пользовательские события: клики, прокрутки и время взаимодействия с элементами интерфейса.

На втором этапе собранные данные проходят аналитическую обработку. Для каждого интерфейсного блока выделяются характеристики, описывающие его свойства: геометрические (размеры, положение, отступы), визуальные (цветовая гамма, контрастность, размер шрифта) и структурные (тип элемента, его роль в навигации). Полученная информация структурируется и подготавливается для дальнейшего анализа.

Основой третьего этапа является ML-ядро системы. С помощью метода кластеризации пользовательские сессии сегментируются по паттернам поведения. Алгоритмы предсказания оценивают вероятные сценарии взаимодействия. Корреляционный анализ выявляет зависимости между действиями пользователей. Выполняется анализ пути пользователя для обнаружения проблемных зон навигации.

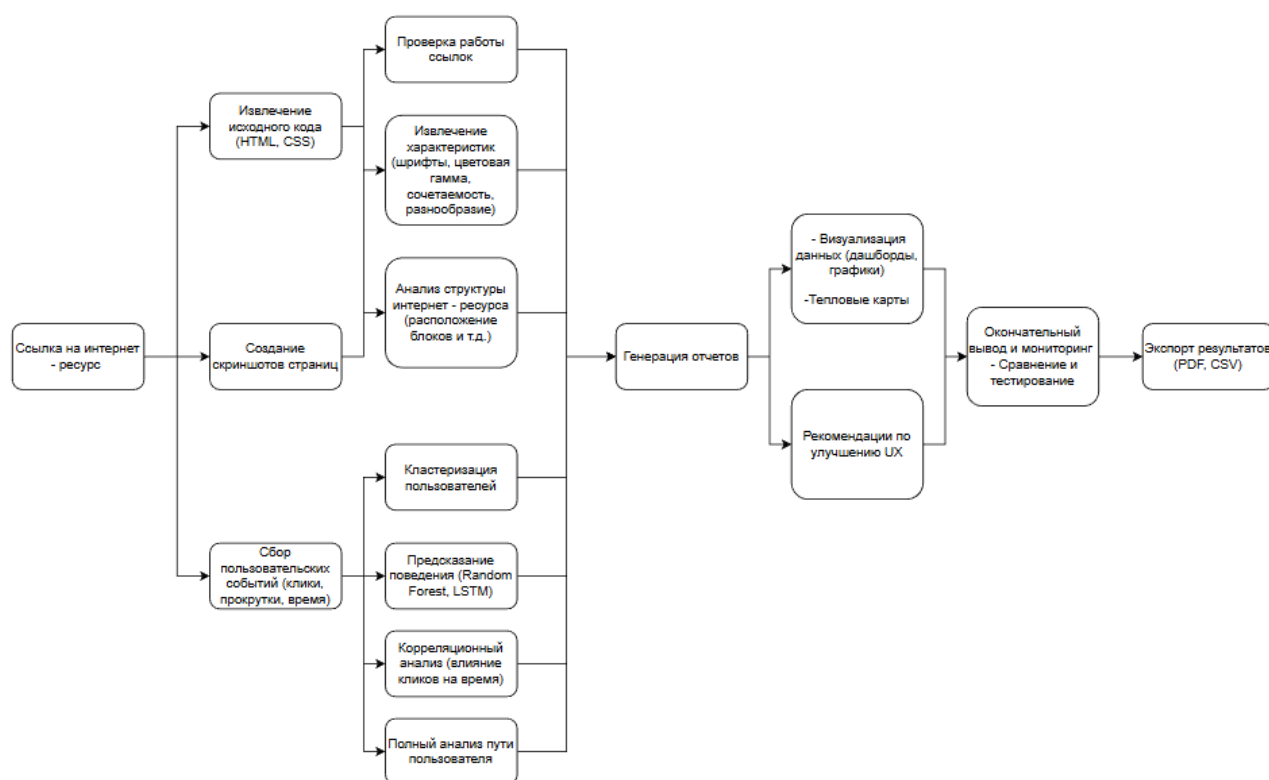


Рисунок 1 – Архитектура предлагаемого решения

Непосредственная генерация отчёта производится на основе агрегированных данных о нарушениях, результатах ML анализа и встроенного механизма приоритизации, учитывающего серьёзность, распространённость и влияние проблемы на ключевые пользовательские сегменты.

ЛИТЕРАТУРА

1. Круг С. Веб-дизайн: книга Стива Круга, или Не заставляйте меня думать! / С. Круг. – 2-е изд. – СПб.: Питер, 2017. – 208 с.
2. Нильсен Я., Будиу Р. Юзабилити мобильных устройств / Я. Нильсен, Р. Будиу. – М.: Вильямс, 2013. – 368 с.

3. Современные тенденции в автоматизированной оценке юзабилити и поведенческие факторы в алгоритмах поисковых систем // Киберленинка. – 2021. – URL: <https://cyberleninka.ru/article/n/sovremennye-tendentsii-v-avtomatizirovannoy-otsenke-yuzabiliti-i-povedencheskie-factory-v-algoritmah-poiskovyh-sistem/viewer> (дата обращения: 10.02.2026).
4. Принятие решений в области совершенствования цифровых сервисов на основе UI/UX исследований // Киберленинка. – 2023. – URL: <https://cyberleninka.ru/article/n/prinyatie-resheniy-v-oblasti-sovershenstvovaniya-tsifrovyyh-servisov-na-osnove-ui-ux-issledovaniy-vliyanie-ispolzovaniya-tehnologii/viewer> (дата обращения: 11.02.2026).
5. 15 критериев для оценки дизайна сайта / Crystal Digital. – 2023. – URL: <https://crystal-digital.ru/blog/15-kriteriev-htoby-oczenit-dizajn-sajta/> (дата обращения: 15.12.2025).

УДК 004.415

А.А. Финогенова (2 курс магистратуры),
П.Д. Дробинцев, к.т.н., доцент

ИИ АГЕНТ ДЛЯ АВТОМАТИЧЕСКОГО ЗАПОЛНЕНИЯ И РАСПОЗНАВАНИЯ КОРРЕКТНОСТИ ТЕКСТОВОГО ДОКУМЕНТА НА ОСНОВЕ ПОЛЬЗОВАТЕЛЬСКОГО ЗАПРОСА

Современные облачные платформы (IaaS) позволяют оперативно выделять вычислительные мощности, однако процесс их конфигурирования требует строгого соблюдения регламентов информационной безопасности и сетевой связности. Ручное формирование архитектурных спецификаций и листов согласования в крупных корпоративных сетях сопряжено с высокой вероятностью ошибок в адресации, масках подсетей и распределении зон ответственности, что приводит к задержкам в развёртывании сервисов и повышению рисков ИБ [1, 2].

Один из способов абстрагироваться от рутинных операций администрирования – делегировать процессы извлечения технических параметров и заполнения спецификаций интеллектуальным агентам на базе больших языковых моделей (LLM) [3]. Снизив влияние человеческого фактора на этапе подготовки заявок, мы упростим взаимодействие между бизнес-заказчиком и ИТ-департаментом, обеспечим автоматическую проверку на соответствие корпоративным стандартам и значительно сократим время цикла предоставления ресурсов.

Одной из наиболее перспективных технологий в этой области является использование многоагентных систем (Multi-Agent Systems), где специализированные программные компоненты выполняют изолированные задачи – от извлечения сущностей до финальной верификации данных [4].

Таким образом, целью данной работы является снижение трудоёмкости управления жизненным циклом инфраструктурных запросов за счёт создания интеллектуального инструментального средства на базе агентного рабочего процесса (Agentic Workflow). Для достижения этой цели необходимо решение задач, перечисленных ниже.

- Обзор существующих подходов к автоматизации технического документооборота с применением LLM.
- Проведение сравнительного анализа эффективности различных моделей Open Source в задачах извлечения инфраструктурных сущностей.
- Разработка многоагентного подхода к генерации и верификации архитектурных заданий.
- Реализация данного подхода в виде программного прототипа.
- Экспериментальная оценка точности и снижения временных затрат.

Одним из основных компонентов разрабатываемого подхода является блок верификации («агент-судья»), осуществляющий кросс-проверку извлечённых данных на соответствие сетевым стандартам предприятия. Архитектура системы представлена на рисунке 1.

Взаимодействие пользователя с системой происходит через интерфейс естественного языка, после чего агент проводит поиск необходимых параметров в базе знаний (RAG) [5, 6], формирует цифровой двойник документа и инициирует процедуру автоматической проверки корректности.

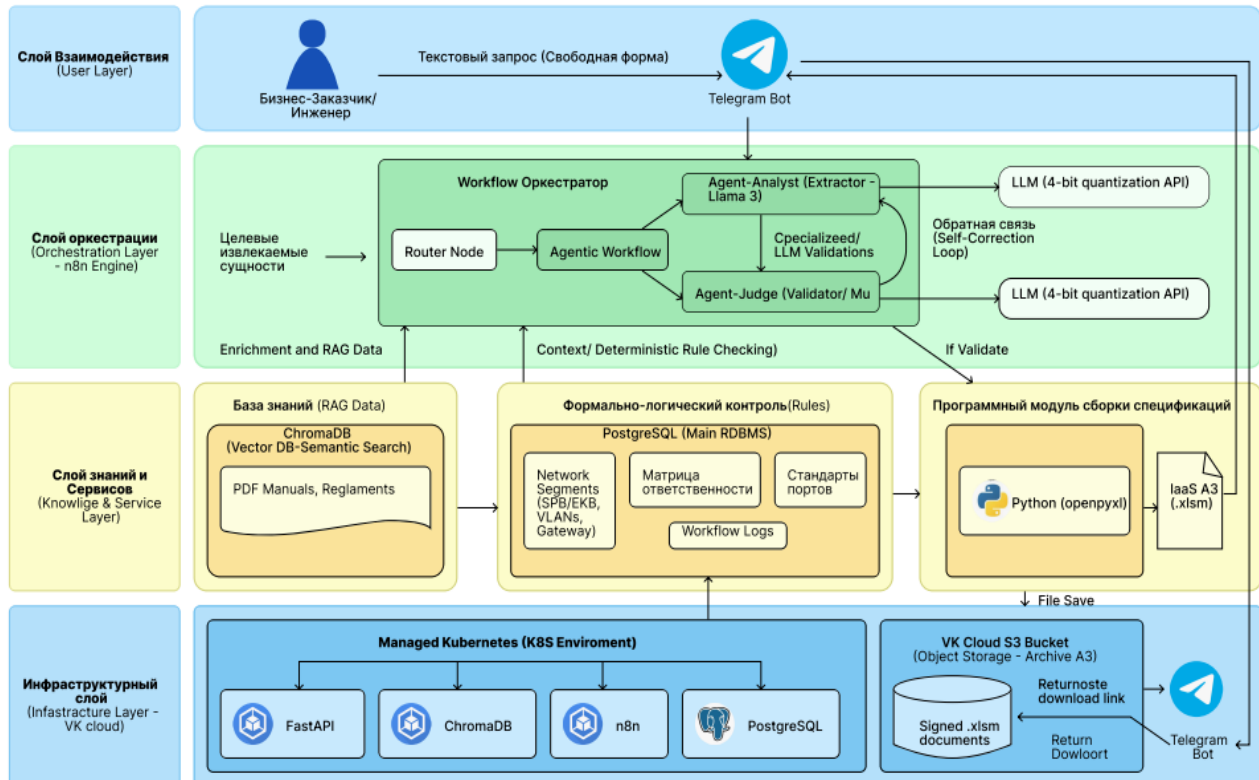


Рисунок 1 – Архитектура системы

Непосредственное формирование итоговой спецификации производится путём заполнения формализованных шаблонов в формате .xslm с использованием специализированных библиотек обработки данных.

Подход был реализован в программном средстве на языке Python с использованием платформы оркестрации n8n. Инструментарий включает в себя интеграцию со справочниками сетевых сегментов (СПб/ЕКБ) и матрицей ответственности сотрудников. Сравнительный анализ демонстрационной версии показал высокую точность извлечения параметров (vCPU, RAM, VLAN) даже из неполных запросов, а внедрение блока автоматизированной верификации позволило практически полностью исключить ошибки, связанные с некорректным указанием IP-адресов и портов доступа.

ЛИТЕРАТУРА

1. Беспалова Н. В. Безопасность облачных технологий // Computational nanotechnology. 2024. №5.
2. Карпов И. А., Семенов А. С. Автоматизация управления ИТ-инфраструктурой и оркестрация контейнеров в условиях импортозамещения // Программная инженерия. – 2025. – Т. 16, № 2. – С. 34-41.
3. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need // Advances in Neural Information Processing Systems. – 2017. – Vol. 30. – P. 5998-6008.
4. Wu Q., Bansal G., Zhang J. et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation // arXiv preprint arXiv:2308.08155. – 2023. – 38 p.
5. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 9459-9474.
6. Федоров В. О., Поляков Р. А. Большие языковые модели с поисковой расширенной генерацией: обзор и перспективы // Оригинальные исследования. – 2023. – Т. 13, № 12. – С. 15-22.

СИСТЕМА АНАЛИЗА ПОСЕЩАЕМОСТИ ТОРГОВЫХ ЗАЛОВ НА ОСНОВЕ ВИДЕО С
КАМЕР НАБЛЮДЕНИЯ

В последние годы в сфере розничной торговли всё большее значение приобретает анализ данных о поведении посетителей. Одним из ключевых показателей эффективности работы магазина является посещаемость торгового зала и характер перемещения клиентов внутри него. Такая информация позволяет оценивать эффективность планировки магазина, оптимизировать размещение товаров и принимать управленческие решения на основе объективных данных.

Традиционные методы анализа посещаемости имеют ряд ограничений. Например, ручной подсчёт посетителей требует значительных трудозатрат и часто приводит к неточным результатам. Анализ кассовых данных позволяет учитывать только покупателей, совершивших покупку, но не отражает общее количество посетителей. Аппаратные счётчики на входе фиксируют лишь факт пересечения линии и не дают информации о перемещениях людей внутри торгового пространства. В связи с этим в последние годы активно развивается направление видеоаналитики, основанное на применении методов компьютерного зрения и машинного обучения [1].

Использование видеопотоков с камер наблюдения позволяет получать более подробную информацию о поведении посетителей. С помощью современных алгоритмов можно обнаруживать людей на видеозаписи, отслеживать их перемещения и анализировать потоки посетителей в различных зонах торгового зала [2]. Подобные системы позволяют строить тепловые карты активности, оценивать плотность потока клиентов и определять наиболее посещаемые участки помещения.

Целью данной работы является анализ современных подходов к разработке систем видеоаналитики для оценки посещаемости торговых залов, а также формирование концепции системы, которая может использоваться для решения данной задачи.

В рамках работы были рассмотрены методы обнаружения объектов на изображениях. В настоящее время наиболее эффективными являются нейросетевые алгоритмы обнаружения, основанные на сверточных нейронных сетях. Одним из широко используемых подходов является алгоритм Faster R-CNN, который обеспечивает высокую точность обнаружения на изображениях [3]. Однако для задач обработки видеопотока в реальном времени чаще применяются одноэтапные методы обнаружения объектов, такие как модели семейства YOLO (You Only Look Once), которые позволяют значительно повысить скорость обработки данных.

После обнаружения объектов на кадрах видеопоследовательности необходимо решать задачу многообъектного отслеживания. Она заключается в сопоставлении обнаруженных объектов на последовательных кадрах и поддержании уникального идентификатора для каждого человека. Для решения этой задачи применяются различные алгоритмы отслеживания, среди которых одним из современных подходов является метод ByteTrack, позволяющий эффективно отслеживать объекты даже при частичных перекрытиях [4].

На основе результатов обнаружения и отслеживания можно выполнять аналитическую обработку данных. В частности, система может фиксировать пересечение виртуальных линий на входе в торговый зал и тем самым подсчитывать количество посетителей. Кроме того, возможно построение тепловых карт перемещений, позволяющих визуализировать распределение потоков людей внутри помещения.

Предлагаемая концепция системы анализа посещаемости включает несколько основных модулей: модуль получения видеопотока с камер наблюдения, модуль обнаружения людей, модуль отслеживания объектов и аналитический модуль, формирующий статистические

показатели посещаемости. Также одним из элементов системы является визуальный редактор помещения, позволяющий создавать схему торгового зала и размещать на ней камеры видеонаблюдения. Данный элемент системы учитывает геометрию помещения, направление обзора и зоны видимости каждой камеры. На основе полученных данных реализуется алгоритм сопоставления видеопотоков с разных камер с их расположением на карте помещения, что позволяет объединять данные наблюдения в единую систему координат и анализировать перемещения посетителей по торговому залу.

Такая модульная архитектура позволит адаптировать систему под различные конфигурации помещений и используемое оборудование.

Результаты проведенного анализа показывают, что современные методы компьютерного зрения позволяют эффективно использовать существующую инфраструктуру видеонаблюдения для получения аналитической информации о посещаемости торговых залов. Разработка подобной системы может быть полезна для оптимизации работы торговых залов и повышения эффективности управления.

ЛИТЕРАТУРА

1. Retail store customer behavior analysis system: Design and Implementation // Proceedings of the International Conference on Computer Vision and Pattern Recognition. – 2023. – P. 145–156.
2. Ицзя, Л. Применение нейронных сетей для оценки трафика магазина / Л. Ицзя, О. Г. Малеев // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 26–27 апреля 2023 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. – С. 54-55. – EDN WJRDNR.
3. Ren S. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // Advances in Neural Information Processing Systems. — 2015. — T. 28. — С. 91–99.
1. 4. ByteTrack: Multi-Object Tracking by Associating Every Detection Box / Y. Zhang [и др.] // European Conference on Computer Vision. – 2022. – P. 1–21.
5. Person detection and re-identification in open-world settings of retail stores and public spaces (2025) // Computer Vision and Image Understanding. – 2025. – Vol. 234. – P. 103–115.

УДК 004.42

А.А. Шожат (2 курс магистратуры),
Т.В. Леонтьева, к.т.н., доцент

ПРИМЕНЕНИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ПЕРСОНАЛИЗАЦИИ ТРЕНИРОВОЧНОГО ПРОЦЕССА

В последние годы наблюдается стремительный рост интереса к вопросам здоровья, физической активности и формирования индивидуального подхода к тренировкам. Это обусловлено несколькими важными тенденциями.

Согласно исследованиям [1] международных аналитических агентств (Statista, Grand View Research), рынок мобильных фитнес-приложений демонстрирует устойчивый рост и ожидается, что его объем превысит 30 млрд долларов к 2028 году. Всё больше пользователей предпочитают проводить тренировки дома или на улице, используя удобные цифровые инструменты для контроля прогресса, получения рекомендаций и взаимодействия с сообществом.

Универсальные тренировочные схемы теряют актуальность [2], так как не учитывают индивидуальные особенности организма. Люди с разным уровнем подготовки, целями (снижение веса, набор массы, реабилитация и т.д.) и физическими ограничениями требуют различного подхода [3]. На первый план выходит персонализация — возможность получить рекомендации, оптимально подходящие конкретному пользователю.

Развитие технологий машинного обучения и упрощение их интеграции в прикладные решения делает возможным использование ИИ даже в мобильных приложениях [4]. Сегодня достаточно доступных библиотек и вычислительных мощностей, чтобы реализовать рекомендательные алгоритмы и адаптивные модели в рамках пользовательских приложений. Это открывает возможность сделать персонализацию действительно интеллектуальной, а не статической.

Разрабатываемая система рекомендаций упражнений предназначена для автоматического подбора индивидуальных программ тренировок на основе анализа базы упражнений и предпочтений пользователя. В основе решения лежит метод контентной фильтрации, который предполагает создание профиля каждого упражнения на основе его характеристик и последующее сравнение этих профилей с вектором предпочтений пользователя.

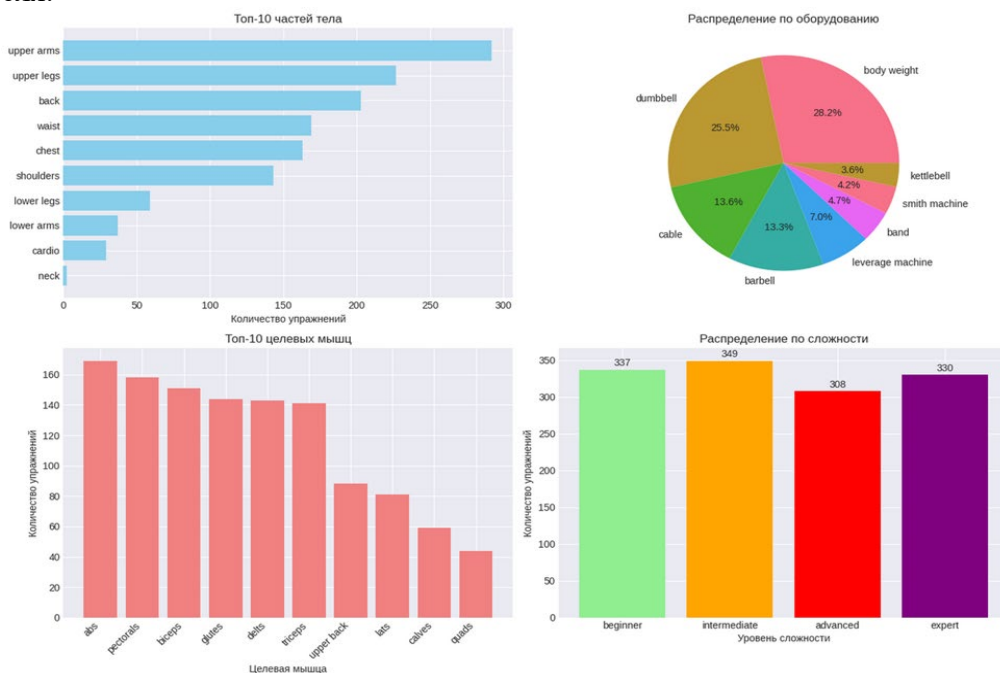


Рисунок 1 – Предобработка данных.

В ходе работы было предусмотрено несколько этапов, первый из которых посвящен анализу исходных данных и выявлению их структуры. Второй этап включал предобработку и очистку данных, третий - проектирование признакового пространства, четвертый - разработку алгоритма рекомендаций, и заключительный этап был направлен на тестирование и визуализацию полученных результатов.

На этапе предобработки данных было выполнено объединение разрозненных столбцов с информацией о вторичных мышцах в единый признак, что представлено на рисунке 1. В ходе анализа исходного датасета было обнаружено, что данные о вторичных мышцах распределены по семи различным группам. Такая структура данных характерна для многих реальных датасетов, где информация может дублироваться или быть разбита на множество полей без четкой системы. Для решения этой проблемы потребовалась разработка специального алгоритма агрегации, который собирает все значения из семи столбцов в единый список, удаляет дубликаты и формирует целостное представление о вторичных мышцах для каждого упражнения.

Для преобразования категориальных признаков в числовой формат был применен метод One-Hot Encoding [5]. Кодированию подверглись части тела, типы оборудования, целевые мышцы, вторичные мышцы и уровень сложности. Поскольку общее количество уникальных значений по некоторым категориям превышало разумные пределы, было принято решение кодировать только наиболее часто встречающиеся значения. В итоге была сформирована матрица признаков размерностью 1324 упражнения на 56 бинарных признаков. Для сравнения профиля пользователя с профилями упражнений используется метрика косинусного сходства.

В разработанной системе искусственный интеллект реализован через алгоритмы машинного обучения, которые лежат в основе механизма персонализации. Использование ИИ проявляется в нескольких аспектах работы. Во-первых, система автоматически извлекает скрытые закономерности из данных: она не просто сопоставляет категории, а выявляет неочевидные связи между упражнениями, например, какие группы мышц чаще всего работают совместно или какие упражнения могут быть взаимозаменяемы при отсутствии определенного оборудования. Во-вторых, реализован интеллектуальный механизм взвешивания признаков, где модель самостоятельно определяет вклад каждого параметра в итоговую рекомендацию, обучаясь на структуре данных.

Таким образом, созданная система предоставляет пользователю возможность получать индивидуальные программы тренировок, учитывающие его личные предпочтения и ограничения. Благодаря визуализации данных и прозрачности алгоритмов пользователь может понимать, на каких основаниях были предложены те или иные упражнения.

ЛИТЕРАТУРА

1. A personalized mobile app for physical activity. [Электронный ресурс]. Режим доступа: <https://pubmed.ncbi.nlm.nih.gov/35898287/> (дата обращения: 10.03.2026)
2. Personalization Paradox in Behavior Change Apps. [Электронный ресурс]. Режим доступа: <https://dl.acm.org/doi/10.1145/3449190> (дата обращения: 10.03.2026)
3. Scoping review PubMed. [Электронный ресурс]. Режим доступа: <https://pubmed.ncbi.nlm.nih.gov/35249995/> (дата обращения: 10.03.2026)
4. P3FitRec: Privacy-Preserving Personalized Fitness Recommender System. [Электронный ресурс]. Режим доступа: https://www.researchgate.net/publication/367131152_Privacy-Preserving_Personalized_Fitness_Recommender_System_P_3_FitRec_A_Multi-level_Deep_Learning_Approach (дата обращения: 10.03.2026)
5. One Hot Encoding: что это такое и как применять в машинном обучении. [Электронный ресурс]. Режим доступа: <https://sky.pro/wiki/analytics/one-hot-encoding-cto-eto-takoe-i-kak-primenyat-v-mashinnom-obuchenii/> (дата обращения: 14.03.2026)

УДК 004.75

Р.Р. Янбеков (1 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент

АЛГЕБРАИЧЕСКИЕ МОДЕЛИ СОГЛАСОВАННОСТИ РАСПРЕДЕЛЁННЫХ СИСТЕМ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ

Современные высоконагруженные информационные системы — сети сотовых операторов, стриминговые сервисы, поисковые системы — используют распределённые системы хранения и обработки данных с гетерогенной инфраструктурой. Такие системы характеризуются критическими параметрами: пиковые сетевые задержки достигают 500–2000 мс при межконтинентальных запросах [1, 2], отказы узлов затрагивают до 16% аппаратных комплексов в год [3], а требования к доступности составляют 99,99% для критичных систем. При этом использование строгой координации снижает пропускную способность на 40–60% [4], что обуславливает актуальность поиска альтернативных подходов к обеспечению согласованности данных.

Целью данной работы является повышение доступности и снижение задержек обработки данных в распределённых системах за счёт обеспечения согласованности без глобальной координации, используя алгебраические свойства операций. Для достижения цели были поставлены следующие задачи: проведение сравнительного анализа существующих моделей согласованности и подходов к репликации данных; исследование алгебраических структур (полурешётки, полугруппы, моноиды) применительно к описанию операций над распределёнными состояниями; формулировка методов определения корректных

асинхронных операций, преобразования операций в бесконфликтные формы и адаптивного выбора стратегии слияния состояний.

В ходе исследования проведён анализ традиционных протоколов координации — 2PC [5], Paxos [6], Raft [4] — и установлены их фундаментальные ограничения: высокая стоимость координации (до $O(n^2)$ сообщений на раунд), невозможность обеспечения сильной согласованности при сетевых разделениях (CAP-теорема [7]), снижение доступности при потере лидера и рост задержек записи при масштабировании кластера. Экспериментальные данные подтверждают зависимость задержек записи от числа узлов [1]. Результаты сравнения представлены на рисунке 1.

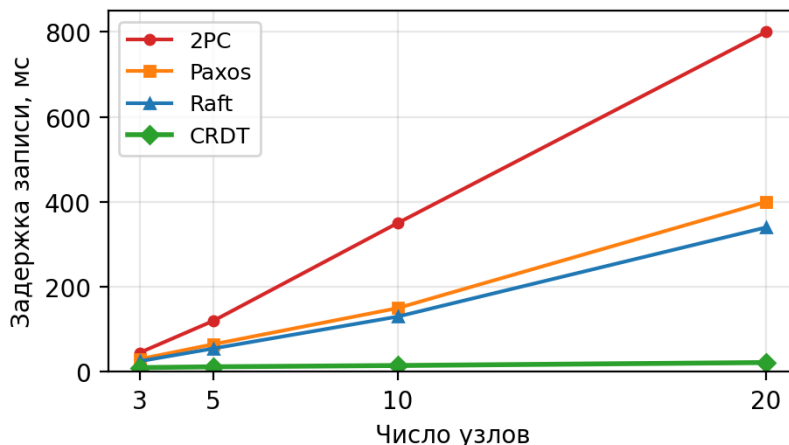


Рисунок 1 – Зависимость задержек записи от числа узлов

Как видно из рисунка 1, бесконфликтные реплицируемые типы данных (Conflict-free Replicated Data Types, CRDT) демонстрируют практически постоянные задержки (10–22 мс) независимо от числа узлов, тогда как координационные протоколы показывают рост задержек на порядок (45→800 мс для 2PC), что подтверждает перспективность данного подхода [1, 8].

Сравнительный анализ способов репликации (синхронная, полусинхронная, асинхронная) и моделей (Master–Slave, Master–Master, Leaderless) показал, что для обеспечения высокой доступности и минимальных задержек оптимальной является асинхронная репликация в Leaderless-архитектуре. Анализ уровней гарантий согласованности обосновал выбор модели строгой согласованности в конечном счёте (Strong Eventual Consistency, SEC), обеспечивающей детерминированное слияние параллельных обновлений без координации [8].

Ключевым инструментом достижения SEC являются CRDT, алгебраические свойства которых — коммутативность, ассоциативность и идемпотентность — гарантируют автоматическое слияние параллельных обновлений [8]. Исследованы три вида CRDT: CmRDT (Operation-based), CvRDT (State-based) и Δ -CRDT (Delta). CmRDT обеспечивают минимальный трафик, но требуют надёжной доставки; CvRDT устойчивы к потерям, но создают значительную нагрузку на сеть; Δ -CRDT представляют оптимальный баланс [9].

Анализ существующих реализаций CRDT (Riak, AntidoteDB, Automerge, Yjs) выявил общие ограничения: фиксированный набор типов данных, отсутствие автоматической верификации корректности операций и фиксированная стратегия слияния без учёта условий сети [8, 9, 10]. Выявлены три ключевые проблемы существующих подходов к обеспечению согласованности в распределённых системах: отсутствие формальных критериев определения корректных асинхронных операций; невозможность автоматического преобразования произвольных операций в бесконфликтные формы; отсутствие адаптивного метода выбора стратегии слияния состояний.

Научная новизна определяется разработкой формального аппарата для автоматической верификации операций над распределёнными состояниями на основе алгебраических свойств (коммутативность, идемпотентность, монотонность), который позволяет решить перечисленные выше проблемы. Предложены три метода: метод определения корректных

асинхронных операций на основе аксиом полурешётки; метод преобразования произвольных операций в бесконфликтные формы путём декомпозиции на элементарные CRDT-примитивы; метод адаптивного выбора стратегии слияния состояний в зависимости от типа данных и топологии сети. Практическая значимость заключается в применении разрабатываемых методов для повышения эффективности промышленных распределённых систем в телекоммуникациях, электронной коммерции и IoT.

ЛИТЕРАТУРА

1. Bailis P., Venkataraman S., Franklin M.J., Hellerstein J.M., Stoica I. Quantifying eventual consistency with PBS // *Proceedings of the VLDB Endowment*. – 2014. – Vol. 7, No. 4. – P. 279–290.
2. Кольцов А.А., Никифоров И.В. Автоматизация определения регрессии производительности СХД // *Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции, Санкт-Петербург, 22 апреля 2021 года*. – СПб.: СПбПУ, 2021. – С. 211–214.
3. Ford D., Labelle F., Popovici F.I. et al. Availability in globally distributed storage systems // *Proceedings of the 9th USENIX Symposium on OSDI*. – 2010. – P. 61–74.
4. Ongaro D., Ousterhout J. In search of an understandable consensus algorithm // *Proceedings of the 2014 USENIX ATC*. – 2014. – P. 305–319.
5. Gray J. Notes on data base operating systems // *Operating Systems: An Advanced Course*. – Springer, 1978. – P. 393–481.
6. Lamport L. The part-time parliament // *ACM Transactions on Computer Systems*. – 1998. – Vol. 16, No. 2. – P. 133–169.
7. Никифоров И.В., Юсупова О.А., Воинов Н.В., Ковалев А.Д. Методы, алгоритмы и архитектуры распределенной обработки больших данных. – СПб.: Санкт-Петербургский политехнический университет Петра Великого, 2023. – 194 с.
8. Shapiro M., Preguiça N., Baquero C., Zawirski M. Conflict-free replicated data types // *Proceedings of the 13th SSS*. – 2011. – P. 386–400.
9. Shapiro M., Preguiça N., Baquero C., Zawirski M. A comprehensive study of Convergent and Commutative Replicated Data Types // *INRIA Research Report*. – 2011. – No. 7506.
10. Kleppmann M., Beresford A.R. A conflict-free replicated JSON datatype // *IEEE TPDS*. – 2017. – Vol. 28, No. 10. – P. 2733–2746.

УДК 004.45

В.Д. Яровой (2 курс магистратуры),
И.В. Шошмина, к.т.н., доцент

РАЗРАБОТКА NFS-СЕРВЕРА НА ЯЗЫКЕ RUST В ПОЛЬЗОВАТЕЛЬСКОМ ПРОСТРАНСТВЕ LINUX

Сетевые файловые системы являются фундаментальным компонентом распределенных вычислительных инфраструктур, обеспечивающим прозрачный удалённый доступ к данным. В работе реализуется протокол NFS (Network File System) версии 3 [1], который остаётся одним из наиболее распространенных решений для организации совместного доступа к файлам в UNIX-подобных системах и используется в корпоративных хранилищах данных, высокопроизводительных вычислительных кластерах и облачных средах.

Традиционная реализация NFS-сервера функционирует на уровне ядра операционной системы, что обеспечивает высокую производительность, однако существенно ограничивает переносимость, усложняет отладку и повышает риски нарушения стабильности системы при возникновении ошибок. Реализация NFS-сервера в пространстве пользователя позволяет обеспечить изоляцию от ядра ОС, упрощенную разработку и возможность развертывания на различных платформах. Вместе с тем серверы в пространстве пользователя требуют

тщательного проектирования для достижения приемлемой производительности и безопасного управления памятью.

Целью работы является разработка NFS-сервера для Linux систем в пространстве пользователя с поддержкой протокола NFS версии 3 и протокола монтирования MOUNT версии 3.

Для достижения поставленной цели необходимо решение следующих задач:

1. Реализация стека протоколов RPC (Remote Procedure Call) и XDR (External Data Representation) для сетевого взаимодействия в соответствии с RFC 5531 [2] и RFC 4506 [3].
2. Реализация интерфейса виртуальной файловой системы (VFS) с полным набором операций протокола NFS версии 3, которые описаны в RFC 1813 [1].
3. Реализация протокола монтирования MOUNT версии 3 согласно RFC 1813 [1].
4. Разработка асинхронной конвейерной архитектуры обработки запросов.

NFS сервер реализован на языке программирования Rust [4] — системном языке, ключевым преимуществом которого является модель безопасности памяти, обеспечивающая обнаружение и предотвращение ошибок на этапе компиляции, таких как обращение к недопустимой памяти, гонки данных и использование освобожденных ресурсов. Данные свойства особенно важны для серверного программного обеспечения, требующего надежности и производительности.

Архитектура сервера построена на асинхронной модели с использованием среды выполнения Tokio[5], предоставляемой отдельной библиотекой. Данная архитектура позволяет принимать TCP-соединения, извлекать запросы и выполнять их, формируя ответ согласно спецификации протокола.

Также реализован парсинг сообщений с поддержкой типов данных всех 22 процедур протокола NFS версии 3, перечисленных в RFC 1813 [1].

Для эффективной работы с сетевыми буферами разработан специализированный аллокатор памяти, реализующий стратегию предварительного выделения и повторного использования буферов фиксированного размера через неблокирующие каналы, что минимизирует аллокации в куче при обработке запросов записи и чтения.

Интерфейс виртуальной файловой системы (VFS) спроектирован как абстрактный слой (trait), изолирующий логику протокола от конкретной реализации файлового хранилища. Это позволяет подключать различные файловые бэкенды без изменения сетевого и протокольного кода сервера.

В результате проведенного исследования и реализации был создан полностью функционирующий NFS-сервер версии 3, работающий в пользовательском пространстве и написанный на языке Rust. Данный сервер может быть использован как самостоятельное приложение, так и быть встроен в процесс, выполняющийся в Tokio [5] для уменьшения копирования памяти при реализации файловой системы в пространстве пользователя Linux, позволяющей избежать прохождения данных через модуль ядра FUSE.

ЛИТЕРАТУРА

1. RFC 1813 - NFS Version 3 Protocol Specification // URL: <https://datatracker.ietf.org/doc/html/rfc1813>
2. RFC 5531 - RPC: Remote Procedure Call Protocol Specification Version 2 // URL: <https://datatracker.ietf.org/doc/html/rfc5531>
3. RFC 4506 - XDR: External Data Representation Standard // URL: <https://datatracker.ietf.org/doc/html/rfc4506>
4. Rust Programming Language // URL: <https://www.rust-lang.org/>
5. Tokio - An asynchronous runtime for the Rust programming language // URL: <https://tokio.rs/>

УДК 004.032.24, 004.85

П.Г. Авдышева, А.Р. Зиатдинова (4 курс бакалавриата),
А.Н. Яковлев, ст. преподаватель

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ НЕЙРОСЕТЕВОГО АНСАМБЛЕВОГО
АЛГОРИТМА RANDOM FOREST НА АРХИТЕКТУРЕ GPU (TESLA K40) С
ИСПОЛЬЗОВАНИЕМ СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

Случайный лес (Random Forest) [1] является одним из наиболее эффективных методов ансамблевого обучения в машинном обучении. Алгоритм был предложен Лео Брейманом в 2001 году и с тех пор получил широкое распространение благодаря своей высокой точности и устойчивости к переобучению. В основе алгоритма лежит теорема о мудрости толпы: множество слабых моделей в сочетании дают сильного предсказателя. Однако его ключевая особенность — построение множества независимых решающих деревьев — приводит к линейному росту вычислительных затрат с увеличением объема данных и числа деревьев в ансамбле [2][3]. На стандартном персональном компьютере эффективность параллелизации быстро падает из-за ограниченности вычислительных ресурсов, что делает невозможным или крайне неэффективным обучение моделей на датасетах большого объема. Целью данной работы является исследование возможности преодоления этих ограничений путем реализации нейросетевого аналога Random Forest на суперкомпьютерной платформе с использованием GPU-ускорения.

Исследования проводились с использованием вычислительных ресурсов суперкомпьютерного центра Санкт-Петербургского политехнического университета Петра Великого на суперкомпьютере «Политехник – РСК Торнадо», узлы которого оснащены GPU NVIDIA Tesla K40 [4]. В связи с невозможностью прямого переноса классического алгоритма RF на архитектуру GPU, разработан специализированный нейросетевой ансамбль на базе фреймворка PyTorch [5]. В предложенной модели каждое «дерево» ансамбля представлено в виде полносвязной нейронной сети (SimpleNet) с регуляризацией. Ансамблевая природа классического Random Forest имитируется посредством генерации бутстрэп-выборок для каждой сети, их параллельного обучения на CPU через библиотеку для конвейерной обработки joblib и последующей агрегации результатов голосованием. Основные вычисления (прямой расчет предсказаний, обратное распространение ошибки) выполняются на GPU с использованием CUDA. Исследование включало анализ производительности и качества модели на синтетических данных, а также решение практической задачи бинарной классификации биоразлагаемости химических соединений (датасет QSAR biodegradation) [6].

Исследования включали изменение гиперпараметров модели и конфигураций вычислительных ресурсов в системе управления задачами Slurm.

В ходе исследований на синтетических данных было установлено, что критическое влияние на качество классификации нейросетевого ансамбля оказывает не только количество базовых моделей, но и архитектура сети и выбор гиперпараметров обучения. Применение пакетной нормализации (Batch Normalization) и оптимизатора Adam позволило повысить точность модели с 65% до 96.6%. При решении прикладной задачи прогнозирования биоразлагаемости разработанный алгоритм продемонстрировал стабильную точность 81.52% на тестовой выборке. Анализ производительности выявил нелинейную зависимость времени обучения от выделяемых ресурсов. Конфигурация с 4 ядрами CPU и 16 ГБ оперативной памяти обеспечила время обучения, сопоставимое с конфигурацией, использующей 24 ядра CPU и 48 ГБ памяти. Использование двух вычислительных узлов с двумя GPU оказалось неэффективным для данного объема данных (1055 образцов) из-за накладных расходов на межпроцессное взаимодействие над временем полезных вычислений.

Результаты работы демонстрируют эффективность реализации ансамблевых алгоритмов с использованием нейросетевого подхода на GPU-архитектурах. Применение суперкомпьютерных систем позволяет достигать высокой производительности и гибко управлять вычислительными ресурсами в пределах их эффективного масштабирования. Для задач с относительно небольшим объемом данных (порядка 1000 образцов) оптимальной является конфигурация с использованием одного GPU-узла. Определение пределов эффективного масштабирования для решения более ресурсоемких задач рассматриваемого класса является направлением дальнейших исследований.

ЛИТЕРАТУРА

1. Breiman L. Random Forests // Machine Learning. – 2001. – Vol. 45, No. 1. – P. 5–32.
2. Babylonian Journal of Machine Learning. Random Forest Algorithm Overview / Ed. by S. Al-Jawad. – Babylon Publishers, 2023. – 18 p.
3. Уткин Л. В. Анализ данных и машинное обучение [Электронный ресурс]: курс лекций. – URL: <https://levutkin.github.io/teaching> (дата обращения: 10.03.2026).
4. Суперкомпьютерный центр «Политехнический» [Электронный ресурс]. – СПбПУ. – URL: <https://scc.spbstu.ru/resources/> (дата обращения: 10.03.2026).
5. Официальная документация PyTorch [Электронный ресурс]. – URL: <https://pytorch.org/docs/stable/index.html> (дата обращения: 10.03.2026).
6. QSAR Biodegradation Data Set [Электронный ресурс] // UCI Machine Learning Repository. – 2002. – URL: <https://archive.ics.uci.edu/dataset/254/qsar+biodegradation> (дата обращения: 10.03.2026).

УДК 004.8

З.М. Абубакова (2 курс магистратуры),
Н.М. Лыченко, д.т.н., профессор

ПРОГНОЗИРОВАНИЕ КЛАССОВ ИНДЕКСА КАЧЕСТВА ВОЗДУХА НА ОСНОВЕ ГРАФОВЫХ НЕЙРОННЫХ СЕТЕЙ

Загрязнение атмосферного воздуха является одной из острейших экологических проблем г. Бишкек. В зимний отопительный период концентрации мелкодисперсных частиц PM_{2.5} многократно превышают допустимые нормы вследствие массового сжигания угля, температурных инверсий и котловинного рельефа города [1]. Традиционные методы прогнозирования качества воздуха, такие как статистические модели ARIMA (AutoRegressive Integrated Moving Average) или рекуррентные нейронные сети Long Short-Term Memory (LSTM Прогнозирование классов индекса качества воздуха на основе графовых нейронных сетей) и Gated Recurrent Unit (GRU), рассматривают каждую станцию мониторинга

независимо, что не позволяет учитывать пространственный перенос загрязнителей и общие метеорологические факторы [2].

Целью работы является повышение точности прогнозирования классов индекса качества воздуха (AQI) за счёт совместного моделирования пространственных и временных зависимостей между станциями мониторинга. Для достижения цели выполнен анализ существующих методов, построен взвешенный граф станций с учётом географических расстояний, примененна архитектура ST-GAT+TCN (Spatio-Temporal Graph Attention Network with Temporal Convolutional Network), проведено обучение модели и сравнение с базовыми подходами.

На рисунке 1 представлена структура сети, положенной в основу модели прогноза. Ее особенность заключается в комбинировании динамического внимания Graph Attention Network v2

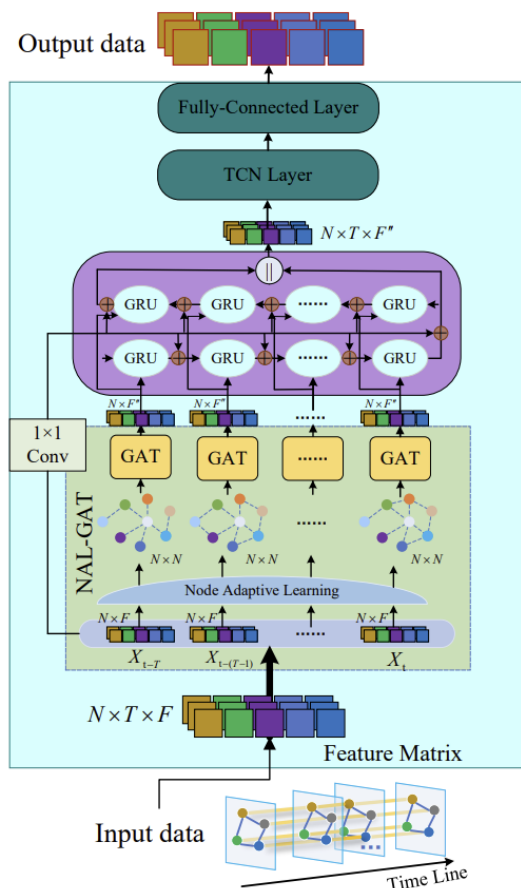


Рисунок 1 – Структура Spatio-temporal Causal Graph Attention Network [3]

(GATv2Conv) и улучшенного блока Temporal Convolutional Network (Improved TCN) для задачи классификации AQI в условиях разреженной сети станций. Граф станций построен на основе реальных географических расстояний по формуле гаверсина с экспоненциальным затуханием весов ребер $w_{(ij)} = \exp(-d_{(ij)}/\sigma)$, где связи формируются в пределах порога 50 км. GATv2Conv обеспечивает контекстно-зависимую агрегацию информации от соседних станций, а улучшенный TCN-блок, который позволяет эффективно извлекать долгосрочные временные зависимости без проблемы затухающих градиентов.

Исходные данные охватывают 2021-2023 гг. с временным шагом 3 часа и включают: метеорологические параметры (скорость ветра, температура, влажность, давление, осадки, точка росы), концентрации PM2.5 по каждой станции и бинарный признак отопительного сезона. Целевая переменная представлена дискретными классами AQI: 0 (хороший), 1 (умеренный), 2 (нездоровый для чувствительных групп) и 3 (нездоровый). Предобработка данных включала заполнение пропусков и нормализацию числовых признаков.

Модель решает задачу многозадачной классификации с использованием взвешенной функции потерь с весами классов, рассчитанными с учётом их частоты (усиление редких

классов 2 и 3). Модель ST-GAT+TCN функционирует следующим образом: на каждом временном шаге данные проходят через слой GATv2Conv [3, 4], который агрегирует пространственный контекст с помощью динамических коэффициентов внимания, вычисляемых после конкатенации признаков целевого и соседнего узлов с применением функции активации. Полученные представления поступают в блок Improved TCN [5], где каузальные дилатированные свёртки с gated-механизмом обеспечивают параллельную обработку и захват долгосрочных паттернов. Финальный линейный классификатор формирует вероятности классов AQI для каждой станции. Введение бинарного признака позволяет лучше учитывать сезонную динамику загрязнения.

В результате экспериментов модель продемонстрировала более высокие значения метрик ассурасу и macro-F1 по сравнению с базовыми моделями LSTM и GRU. Ошибки преимущественно возникают между соседними классами AQI, что свидетельствует о способности модели отслеживать постепенное изменение уровня загрязнения. Использование взвешенного графа и динамического внимания особенно положительно сказалось на качестве предсказания критических классов 2 и 3. Модель прогноза реализована на языке Python с использованием библиотек PyTorch и PyTorch Geometric.

Разработанный метод позволяет повысить точность прогнозирования качества воздуха в условиях ограниченной сети станций мониторинга и может быть использован при создании систем экологического мониторинга и раннего предупреждения населения об опасных уровнях загрязнения воздуха.

ЛИТЕРАТУРА

1. World Bank. Air Quality Analysis for Bishkek: PM2.5 Source Apportionment and Emission Reduction Measures. Washington, DC, 2023.
2. Bai S., Kolter J. Z., Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling // arXiv preprint arXiv:1803.01271. 2018.
3. Zhao W., Zhang S., Zhou B., Wang B. STCGAT: A Spatio-temporal Causal Graph Attention Network for traffic flow prediction in Intelligent Transportation Systems (arXiv:2203.10749) [Preprint]. arXiv, 2022.
4. Brody S., Alon U., Yahav E. How attentive are graph attention networks? // ICLR. 2022.
5. Sundaramoorthy R.A. et al. Implementing heuristic-based multiscale depth-wise separable adaptive temporal convolutional network for ambient air quality prediction // Scientific Reports. 2024. Vol. 14. Article number: 68793.

УДК 004.021

И.И. Азаров (2 курс магистратуры),
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ ЧУВСТВИТЕЛЬНОСТИ ПАРАМЕТРОВ АЛГОРИТМА АЕМ-СОМ ПРИ РАЗРЕШЕНИИ КОНФЛИКТОВ В ГРУППОВОМ ПРИНЯТИИ РЕШЕНИЙ

В задачах группового принятия решений, основанного на методе анализа иерархий, появляется проблема несовместимости индивидуальных суждений экспертов с коллективным мнением. При корректном агрегировании матриц парных сравнений итоговая коллективная матрица может значительно расходиться с индивидуальными матрицами эксперта по согласованности, что повышает разногласия экспертного мнения с групповым. Данную проблему решает алгоритм редуцирования противоречий экспертов АЕМ-СОМ [1].

Целью работы является исследование, прототипирование и анализ чувствительности алгоритма АЕМ-СОМ. Для достижения цели выполнены следующие задачи: анализ механизма выбора корректируемых элементов, проектирование итерационной процедуры с параметром допустимости ρ , оценка влияния масштаба изменений параметра на итоговую метрику несовместимости и проверка поведения алгоритма на синтетических данных.

АЕМ-COM вводит понятие метрики несовместимости GCOMPI [1], характеризующую величину противоречия между индивидуальной матрицей эксперта к коллективного решения, где 0 – полная согласованность, а также параметр сдвига ρ , определяющий величину корректировки одного элемента матрицы эксперта за итерацию, варьируясь от 0 до 1, где 1 означает полную замену мнения в целом. Рекомендуемое максимальное значение параметра обычно составляет не более, чем 0.5. Алгоритм изменяет только коллективную матрицу, выявляя элемент с максимальным вкладом в GCOMPI и корректируя его в относительных величинах в пределах допустимого диапазона ρ . Итерации продолжаются до тех пор, пока изменение метрики не станет незначительным или не будет достигнут локальный минимум. Такой подход позволяет рассматривать алгоритм как локальный модуль согласования в групповом принятии решений, применимый на этапе уточнения коллективного решения перед итоговым агрегированием.

Нами разработан прототип стенда для исследования АЕМ-COM с открытым исходным кодом [2]. Его архитектура отображена на рисунке 1, иллюстрирующий последовательность вычислений и взаимодействие компонентов. Система реализована на языке Python.

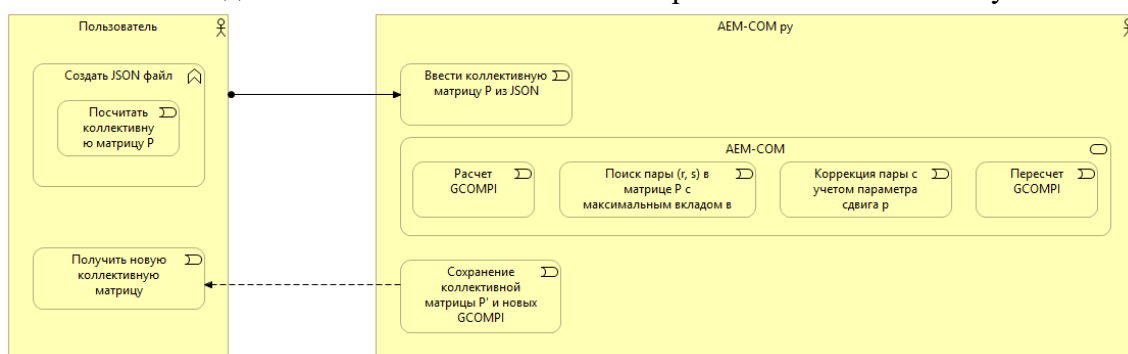


Рисунок 1 – Архитектура стенда для исследования АЕМ-COM

В рамках экспериментального тестирования проведено несколько типов экспериментов. Сценарий эксперимента влияет на согласованность исходных оценок экспертов в их матрице, моделируя различные конфликтные ситуации в итоговой коллективной матрице. Каждый сценарий эксперимента отражен в таблице 1, где под «идеальной согласованностью» понимается максимально возможное отсутствие противоречий в мнениях экспертов, под «реалистичными оценками» – возможно наличие небольших конфликтов, которые обычно требуют минимальных корректировок для компромисса, под «максимальным конфликтом» понимаются максимально несогласованные матрицы экспертов, а под «два противоположных лагеря» – разделение экспертов на 2 группы с противоречивым мнением между группами. Полученные наборы данных генерировались на основе случайно формируемых векторов приоритетов для каждого из n экспертов. В рамках тестирования создавалась группа из 3 экспертов с матрицами размера 5 на 5. На основе этого вектора строилась индивидуальная согласованная матрица парных сравнений для каждого эксперта. В созданную матрицу итеративно вводятся случайные отклонения, достигая задаваемого индекса согласованности. Регулируя итоговый индекс согласованности матрицы, моделируются различные конфликтные ситуации. В случае отсутствия отклонений получается полностью согласованная матрица, соответствующая сценарию «Идеальная согласованность».

Сравнение проводилось с агрегированными с помощью геометрического среднего матрицами без сдвига полученных значений независимо от изначально полученного значения метрики несовместимости GCOMPI.

Экспериментальное тестирование включало 56 синтетических наборов данных, из которых 15 наиболее показательных отражены в таблице 1 с $\rho \in \{0.05, 0.25, 0.50\}$. На большинстве наборов зафиксировано снижение метрики несовместимости на 3–6%, на предварительно согласованных матрицах — на 1–3%, а на случайных данных с выраженным конфликтом — до 6%. При крайне противоречивых структурах предпочтений возможно ухудшение показателя на 2–4%, что подтверждает локальный характер метода,

чувствительность к параметру ρ и зависимость от распределения оценок экспертов в конфликте.

Таблица 1 – Результаты ключевых экспериментов на синтетических данных

Сценарий эксперимента	GCOMPI до	GCOMPI после	Параметр сдвига ρ
Идеальная согласованность	1,84	1,84	0,05
			0,25
			0,50
Реалистичные оценки	6,02	6,00	0,05
		5,93	0,25
		5,86	0,50
Максимальный конфликт	13,29	13,28	0,05
		13,27	0,25
		13,26	0,50
Два противоположных лагеря	8,07	8,05	0,05
		7,95	0,25
		7,87	0,50
Бессмысленные значения	11,34	11,41	0,05
		11,67	0,25
		11,92	0,50

Полученные результаты подтверждают, что АЕМ-COM целесообразно применять как локальный инструмент для уменьшения метрики несовместимости, когда требуется уменьшить расхождение между коллективным и индивидуальным мнением эксперта без изменения исходных суждений. Алгоритм демонстрирует наибольшую эффективность на умеренно конфликтные данные и при корректно настроенном параметре ρ . Также, результаты эксперимента согласуются с работой [1], в которой показано, что применение данного подхода позволяет снизить значение метрики несовместимости примерно на 9% в демонстрационном примере, а также достигать 40-60% потенциально возможного уменьшения несовместимости в симуляционных экспериментах.

ЛИТЕРАТУРА

1. Aguarón J., Escobar M., Moreno-Jiménez J.M. AEM-COM algorithm for reducing incompatibility in group AHP. - Springer, 2023.
2. Azarov I., Parkhomenko V. AEM-COM Model prototype with synthetic test examples // GitHub URL: <https://github.com/DM-groups/aem-py> (дата обращения 07.03.2026)

УДК 004.421

А.В. Багрова (3 курс бакалавриата),
И.Г. Черноуцкий, д.т.н., профессор

РЕАЛИЗАЦИЯ АЛГОРИТМА РОЕВОЙ ОПТИМИЗАЦИИ ДЛЯ СИСТЕМ ЛОГИСТИКИ МНОЖЕСТВА НЕЗАВИСИМЫХ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ

В последние годы роевой интеллект, моделирующий коллективное поведение децентрализованных самоорганизующихся систем, стал мощной парадигмой для решения широкого спектра вычислительно трудоёмких задач [1]. Его ключевая идея состоит в том, что взаимодействие множества относительно простых агентов, каждый из которых действует на основе локальных правил, приводит к возникновению разумного глобального поведения, которое позволяет решать сложные задачи.

Децентрализованный характер и адаптивность роевого интеллекта позволяют спроецировать его на робототехнические системы, в которых группа взаимодействующих устройств должна координировать свои действия для достижения общей цели [2]. Подобные системы находят применение в логистике, мониторинге, инспекции инфраструктуры,

поисково-спасательных операциях и других областях, где требуется согласованное выполнение множества задач в изменяющихся внешних условиях [3, 4].

Однако прямой перенос биологических паттернов на гетерогенные робототехнические рои сталкивается с рядом технологических барьеров. В отличие от биологических прототипов, технические агенты (роботы) обладают жёстко заданными и зачастую неодинаковыми характеристиками: различной грузоподъёмностью, скоростью и ёмкостью аккумуляторов. Это превращает классическую задачу поиска пути в сложную многокритериальную проблему оптимизации с ограничениями, где алгоритм должен балансировать между скоростью принятия решений, энергетической автономностью каждого робота и учётом динамически изменяющегося риска движения по заданным путям.

Среди метаэвристических методов, основанных на роевом интеллекте, особое место занимает алгоритм муравьиной колонии (Ant Colony Optimization, ACO), основанный на коллективном поведении социальных насекомых [5]. Его широкое применение обусловлено тем, что ACO естественно работает на графовых структурах и позволяет строить решения как последовательности переходов между узлами с использованием феромонной памяти и эвристической информации [6]. Однако классические варианты ACO демонстрируют избыточную стохастичность на финальных стадиях поиска или, напротив, преждевременную стагнацию в локальном оптимуме [7]. В задачах управления робототехническим роем этот недостаток становится особенно критичным, поскольку пространство решений включает не только выбор путей, но и распределение задач между роботами при наличии индивидуальных ограничений и изменяющихся параметров среды.

В связи с этим, целью данной работы являлась разработка программной реализации на языке C++ модифицированного варианта алгоритма, ключевой особенностью которого является введение экспоненциально затухающего фактора случайности в механизм принятия решений. Такая система обеспечивает на ранних этапах более интенсивное исследование пространства решений, а по мере накопления информации плавно переходит к эксплуатации полученных наиболее перспективных конфигураций путей, достигая более высокоточной оптимизации. Это позволяет достигать быстрой сходимости без ущерба для разнообразия поиска, сохраняя при этом низкую вычислительную сложность, что критично для бортовых вычислительных устройств роботов.

В рамках данной работы предложенный алгоритм рассматривается в контексте задачи маршрутизации и координации группы гетерогенных программно-аппаратных комплексов в динамической среде, в которой необходимо определить эффективные маршруты множества роботов для доставки продукции многим потребителям при минимизации суммарной длины путей, времени транспортировки и использования ресурсов.

Для анализа алгоритмического поведения были проведены измерительные исследования на стандартных эталонных задачах комбинаторной оптимизации, которые используются как контролируемые прокси для задач маршрутизации в логистических системах.

ЛИТЕРАТУРА

1. Abdulghani B. A., Abdulghani M. A. A Comprehensive Review of Ant Colony Optimization in Swarm Intelligence for Complex Problem Solving // *Acadlore Transactions on AI and Machine Learning* – 2024. – Vol. 3, No. 4 – P. 214-224.
2. Brambilla M., Ferrante E., Birattari M., Dorigo M. Swarm robotics: A review from the swarm engineering perspective // *Swarm Intelligence* – 2013. – Vol. 7, No. 1. – P. 1-41.
3. Dorigo M., Theraulaz G., Trianni V. Swarm Robotics: Past, Present, and Future // *Proceedings of the IEEE* – 2021. – Vol. 109, No. 7. – P. 1152-1165.
4. Zedadra O., Guerrieri A., Jouandeau N., Spezzano G., Seridi H., Fortino G. Swarm Intelligence and IoT-Based Smart Cities: A Review // *The Internet of Things for Smart Urban Ecosystems* – Cham:Springer, 2019. – P. 177-200.
5. Dorigo M., Gambardella L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem // *IEEE Transactions on Evolutionary Computation* – 1997. – Vol. 1, No. 1. – P. 53-66.

6. Skinderowicz R. Improving Ant Colony Optimization efficiency for solving large TSP instances // Applied Soft Computing – 2022. – Vol. 120. – Art. 108653.
7. Misra B., Chakraborty S. Ant Colony Optimization – Recent Variants, Application and Perspectives // Applications of Ant Colony Optimization and Its Variants – Singapore:Springer, 2024. – P. 1-17.

УДК 004.4'422

А.М. Балышев (2 курс бакалавриата),
Д.С. Косарев (инженер-исследователь)

РАЗРАБОТКА КОМПИЛЯТОРА ДЛЯ ЗАЩИТЫ ОТ АТАК НА ЦЕПОЧКИ ПОСТАВОК

Атакой на цепочку поставок программного обеспечения (англ. software supply chain attack) называют ситуацию, при которой злоумышленник получает доступ к программной системе путём внедрения вредоносного кода в компонент, используемый организацией при разработке или развертывании этой системы. Ещё в 1984 году Кен Томпсон в лекции "Reflections on Trusting Trust" описал атаку, при которой вредоносный код внедряется компилятором в генерируемые им программы, что делает аудит исходного кода компилируемых программ недостаточным для выявления уязвимостей [1].

Одним из наиболее распространенных подходов к борьбе с атаками на цепочки поставок является Diverse Double-Compiling [2]. Метод состоит в том, чтобы проверять воспроизводимость артефактов при сборке несколькими независимыми компиляторами. Если удаётся получить побитово идентичные артефакты, то вероятность того, что сборка была скомпрометирована, крайне мала, поскольку злоумышленнику пришлось бы скомпрометировать несколько независимых сборок идентичным образом. Подобный подход широко используется в проектах с открытым исходным кодом: большое количество пользователей могут независимо собирать проекты из публично доступных исходных файлов и сверять результаты. Одной из основных проблем этого подхода является высокая сложность достижения детерминированности сборок. Окружение сборки требуется явно описывать и полностью повторять: архитектура процессора, версия операционной системы и версии зависимостей должны совпадать, иначе результаты не будут воспроизводимы, даже если сборка не была скомпрометирована. Известно, что даже наиболее популярные языковые экосистемы имеют проблемы с обеспечением воспроизводимости сборок без дополнительной конфигурации и/или внесения исправлений в их реализацию [3].

Rukaml [4] — проект по разработке компилятора функционального языка, развиваемый на математико-механическом факультете СПбГУ, преимущественно в рамках семестровых учебных практик. Rukaml написан на OCaml и позволяет порождать RISC-V 64 ASM, AMD64 ASM и LLVM IR. Система сборки dune, используемая в Rukaml, уделяет большое внимание воспроизводимости сборок, что достигается с помощью механизмов фиксации зависимостей [5], изоляции сборки в песочнице [6] и вероятностной проверки воспроизводимости [7]. В рамках работы предлагается обеспечить возможность собирать с помощью Rukaml языковые инструменты и обеспечить защиту собираемых программ от атак на цепочки поставок. Ожидается, что преимущества выбранной экосистемы позволят добиться воспроизводимости сборок на уровне текстового представления программы — на уровне языков ассемблера. Текстовое представление человекочитаемо, что позволит распознавать скрытую логику путём ручной проверки результата компиляции. Это полезно в случае, если проверка воспроизводимости артефактов провалилась, чтобы проанализировать расхождения результатов и зафиксировать попытку атаки, а также в качестве дополнительной гарантии безопасности. Кроме того, в практике компиляторной инженерии известны эффективные оптимизации объема генерируемого компилятором кода [8], применение которых существенно упрощает процесс аудита.

Важно понимать, что такой подход требует наличия доверенного транслятора языка ассемблера, что является отдельной инженерной задачей, остающейся за рамками работы над Rukaml.

Целью работы в рамках проекта Rukaml является реализация поддержки языковых конструкций, необходимых для разработки компиляторов и других языковых инструментов, а также обеспечение защиты собираемых программ от атак на цепочки поставок.

Для достижения этой цели необходимо решение *задач*, перечисленных ниже.

1. Реализация поддержки алгебраических типов данных и сопоставления выражений с образцом.
2. Расширение стандартной библиотеки примитивами форматированного вывода, работы с файловой системой и обработки аргументов командной строки.
3. Оптимизация размера порождаемого компилятором кода.
4. Обеспечение воспроизводимости сборок.

Реализация решения даст пользователю возможность описывать и обрабатывать промежуточные представления программ такие, как деревья абстрактного синтаксиса и компилировать программы, работающие в режиме утилиты командной строки. Достижение детерминированности сборок позволит осуществлять проверку воспроизводимости артефактов. Выбранный подход — компактное текстовое представление — позволит проводить ручное выявление скрытой логики. Это даст возможность собирать языковые инструменты в среде, защищенной от атак на цепочки поставок. На момент написания заявки решение *находится в разработке*.

ЛИТЕРАТУРА

1. Ken Thompson. "Reflections on Trusting Trust", 1984
2. David A. Wheeler et al. "Fully Countering Trusting Trust through Diverse Double-Compiling", 2009
3. Giacomo Benedetti et al. "An Empirical Study on Reproducible Packaging in Open-Source Ecosystems", 2025
4. Rukaml. [Электронный ресурс] Режим доступа:
<https://github.com/Kakadu/rukaml/blob/master/README.md> (дата обращения: 17.03.2026)
5. Dune Locking. [Электронный ресурс] Режим доступа:
<https://dune.readthedocs.io/en/latest/tutorials/dune-package-management/locking.html> (дата обращения: 15.03.2026)
6. Dune Sandboxing. [Электронный ресурс] Режим доступа:
<https://dune.readthedocs.io/en/stable/concepts/sandboxing.html> (дата обращения: 15.03.2026)
7. Dune Reproducibility Check. [Электронный ресурс] Режим доступа:
<https://dune.readthedocs.io/en/stable/caching.html#reproducibility> (дата обращения: 15.03.2026)
8. Intel, Methods to Optimize Code Size. [Электронный ресурс] Режим доступа:
<https://www.intel.com/content/www/us/en/docs/dpcpp-cpp-compiler/developer-guide-reference/2025-1/methods-to-optimize-code-size.html> (дата обращения: 15.03.2026)

УДК 004.94

И.В. Гуляев (4 курс бакалавриата),
И.В. Никифоров, к.т.н., доцент

АРХИТЕКТУРА ОБЛАЧНОЙ ПЛАТФОРМЫ ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ GO

В современном мире имитационное моделирование является главным способом изучения поведения сложных динамических систем [1]. Имитационное моделирование представляет собой описание моделируемой системы с нужным уровнем детализации в виде компьютерной программы, которая позволяет имитировать поведение системы с различными параметрами во времени. Существует три основных подхода имитационного моделирования:

системная динамика, дискретно-событийное моделирование и агентное моделирование [2]. Дискретно-событийное моделирование [3] – это подход, в котором система моделируется как последовательность дискретных во времени событий. Событием называется момент изменения состояния системы. А между наступлениями событий время не моделируется. Основным недостатком дискретно-событийного моделирования является время исполнения модели, которое для большого количества экспериментов может быть очень большим. Исправить данный недостаток можно с помощью распределения экспериментов на многоядерном оборудовании. В этом случае может возникнуть необходимость в приобретении данного оборудования, его настройки, а также решении проблемы поиска области применения оборудования после окончания всех экспериментов. Эту проблему можно решить с помощью облачных вычислений [4], где пользователю надо будет оплатить только время использования вычислительных ресурсов (pay-as-you-go). Однако для настройки облачной инфраструктуры нужно обладать навыками, которых у потенциального пользователя может не быть.

Именно по этой причине актуальной является задача разработки высокопроизводительной облачной платформы дискретно-событийного моделирования, которая предоставляет пользователю простой и удобный интерфейс для создания модели [5], запуска экспериментов и получения качественных результатов симуляции за разумное время.

Целью данной работы является повышение производительности исполнения имитационной модели путем разработки системы имитационного моделирования с применением дискретно-событийного подхода, позволяющую проводить симуляции в облачной инфраструктуре.

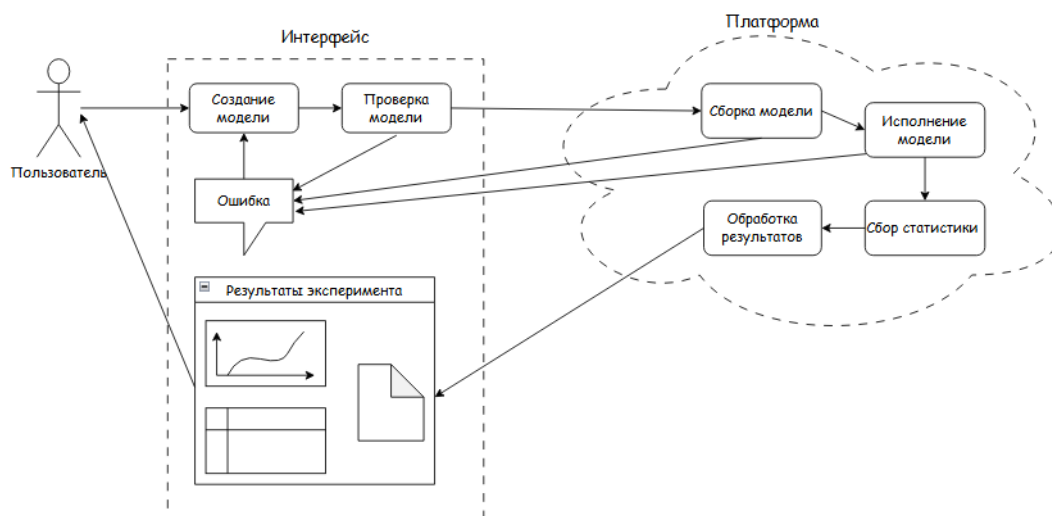


Рисунок 1 – Концептуальная схема облачной платформы

Концептуальная схема, представленная на рисунке 1, предлагает модель облачного сервиса – программное обеспечение для моделирования как услуга (SimSaaS) [6]. В рамках данной модели пользователь имеет доступ только к интерфейсу программного обеспечения, в котором он может создавать модели, запускать эксперименты и просматривать результаты. А на процесс сборки модели, исполнения модели и сбор статистики пользователь не может никак повлиять.

Для реализации функциональности концептуальной схемы была разработана техническая архитектура облачной платформы дискретно событийного моделирования, как показано на рисунке 2. В качестве инструмента реализации данной архитектуры был выбран язык программирования Go, поскольку он является основным языком программирования в разработке программного обеспечения для облачных вычислений [7], а за счет встроенного механизма конкуренции, такого как легковесные потоки и каналы, служит отличным выбором для создания высокопроизводительного движка дискретно-событийного моделирования.

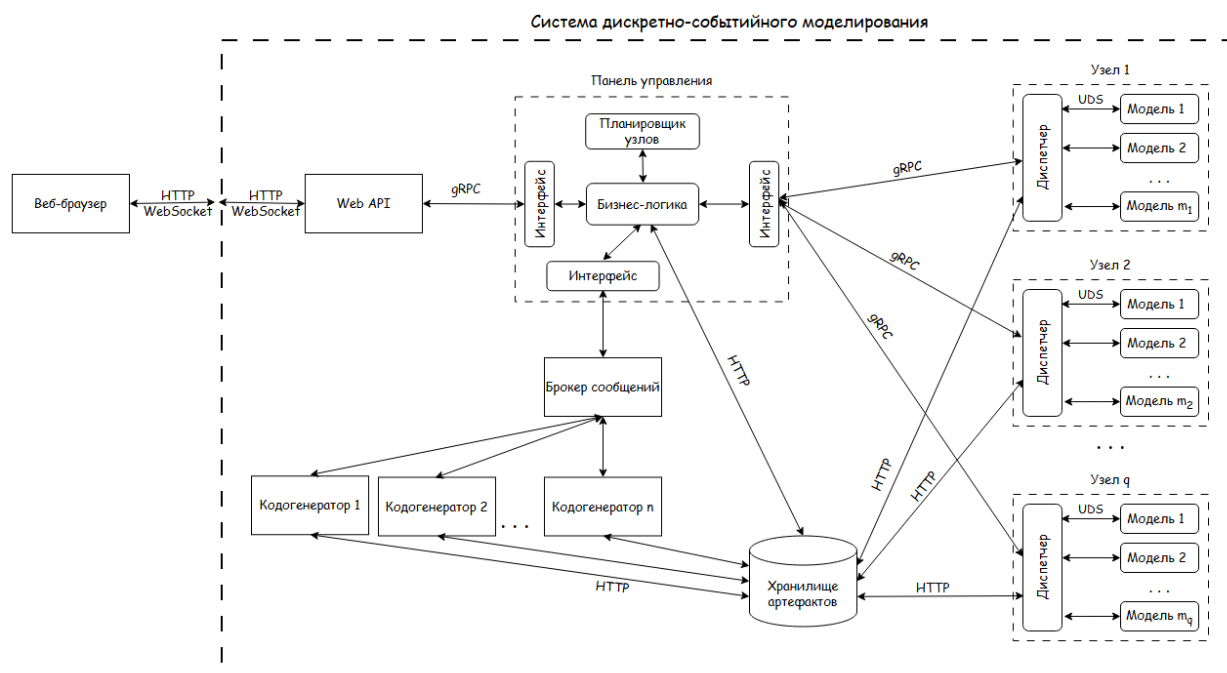


Рисунок 2 – Техническая архитектура облачной платформы

Микросервисная архитектура – это хороший выбор для работы в облачной инфраструктуре [8]. Она даёт возможность создать одну панель управления, которая будет выполнять всю логику планирования распределения моделей между узлами, обработки статистики и управления масштабированием узлов и кодогенераторов. Каждый экземпляр узла обладает программой-диспетчером, которая управляет запуском исполнения модели и следит за исполнением каждой модели, параллельно собирая статистику.

Особенностью данной архитектуры является наличие сервиса-кодогенератора, который будет генерировать исходный код модели на языке программирования Go из описания модели, полученного от пользователя, и компилировать его в исполняемую на узле программу. Данный подход позволяет под каждую модель создавать наиболее производительный код и даёт пользователю возможность создавать сложные правила реагирования на наступления различных событий.

ЛИТЕРАТУРА

1. Ходаева П. А., Зими́на Л. В. Роль имитационного моделирования в современном мире // Концепции и модели устойчивого инновационного развития общества: сборник статей Международной научно-практической конференции, Волгоград, 10 июня 2025 года. - Уфа: Общество с ограниченной ответственностью "Аэтерна", 2025. - С. 58-62.
2. Sabutkevich A.M., Vkhlyaev D.A., Kubov N.A., Nikiforov I.V., Samochadin A.V. Method of Generation of Alternatives of Multi-agent System Behavior Using Simulation System // Gibadullin, A. (eds) Digital and Information Technologies in Economics and Management. DITEM 2024. Lecture Notes in Networks and Systems, vol 1422. - Springer, Cham, 2025. - Pp. 55-68. - DOI: 10.1007/978-3-031-94273-0_5.
3. Sabutkevich A.M., Nikiforov I.V., Samochadin A.V. A method for modeling of individual agent behavior in the process-network paradigm of discrete-event simulation. // Computing, Telecommunications and Control. - 2025. - Vol. 18, No. 1. - Pp. - DOI: 23-35 10.18721/JCSTCS.18102.
4. Gabriele D'Angelo, Moreno Marzolla New trends in parallel and distributed simulation: From many-cores to Cloud Computing // Simulation Modelling Practice and Theory. - 2014. - Volume 49. - Pp. 320-335. - DOI: 10.1016/j.simpat.2014.06.007.
5. Вихляев Д. А., Сабуткевич А. М., Никифоров И. В., Самочадин А. В. Применение методов машинного обучения для построения моделей поведения многоагентных систем и их анализа // Современные технологии в теории и практике программирования: Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых. - Санкт-Петербург:

Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2023. - С. 122-124.

6. Zehe D., Cai W., Knoll A., Aydt H. Tutorial on a modeling and simulation cloud service // 2015 Winter Simulation Conference (WSC). - Huntington Beach, CA, USA: 2015. - Pp. 103-114. - DOI: 10.1109/WSC.2015.7408156.
7. Go for Cloud & Network Services // URL: <https://go.dev/solutions/cloud> (дата обращения: 08.03.2026).
8. Малыгин Д. С. Микросервисная архитектура в облачных системах: риски и возможности применения в 2024-2030 гг // Моделирование, оптимизация и информационные технологии. - 2024. - №2(45). - DOI: 10.26102/2310-6018/2024.45.2.029.

УДК 32.973

А.Д. Григорьев, Л.Н. Карпов (4 курс бакалавриата),
Е.А. Муравьев, к.т.н., доцент.

БИБЛИОТЕКА КОМПИЛЯТОРОВ СИНТАКСИЧЕСКИХ КОНСТРУКЦИЙ-РАСШИРЕНИЙ КОНТРОЛИРУЕМОГО ЕСТЕСТВЕННОГО ЯЗЫКА СПЕЦИФИКАЦИИ ТРЕБОВАНИЙ

В докладе 2025 г. на данной конференции[1] сообщалось о компиляторе с контролируемого естественного языка CNL_Lab. Грамматика языка получена ограничением грамматики русского языка. Язык предназначен для формальной спецификации требований при бизнес моделировании предметных областей, в системах интеллектуального поиска в нормативных документах, таких как стандарты, справочники, разного рода руководства и др. В 2025 году на отлично были защищены 4 выпускные квалификационные работы на квалификацию бакалавра, представившие макет системы автоматического семантического поиска требований в Российском Морском Регистре Судоходства (PMPC) [2].

Ограничения, накладываемые грамматикой CNL_Lab, требуют достаточно высокой квалификации разработчика спецификаций. Цель докладываемой сегодня работы состоит в снижении доли ручных операций при подготовке спецификаций требований на естественном языке. Идея решения задачи состоит в том, чтобы нарастить ядро ограниченного языка CNL расширяемой библиотекой компиляторов синтаксических конструкций (КСК), характерных для документов, разработанных на неограниченном естественном языке (ЕЯ) и предназначенных для перевода на ограниченный ЕЯ CNL.

CNL накладывает на синтаксис текста сильные ограничения. Примеры таких ограничений – главные члены предложения: подлежащее – субъект действия (sub), сказуемое – действие (pred), дополнение – объект действия (obj), должны присутствовать в предложении в единственном экземпляре. Не допускаются причастные обороты, есть и др. ограничения.

Предложение, содержащее синтаксическую конструкцию (СК), выходящую за границу, определяемую грамматикой CNL, декомпозируется в дискурс – множество предложений, связанных смысловым единством. Дискурс сохраняет семантику предложения, при этом каждое предложение дискурса удовлетворяет грамматике CNL.

Конструкция, например, множественного вхождения объекта действия (obj) в исходном предложении заменяется на переменную, соответствующую категории члена предложения, в данном примере это *‘объект’*. Для каждого вхождения объекта *x* в исходное предложение КСК строит распространяющее предложение дискурса, типа *‘объект это x’*. Алгоритмы обработки семантического фрейма – результата компиляции CNL предложения, дорабатываются до обработки дискурсов.

Ниже приводятся примеры СК и результатов их компиляции в дискурс. В примерах приведены исходные тексты требований PMPC, не отвечающие грамматике CNL. Каждый пример демонстрирует шаг применения КСК к фрагменту предложения, содержащему соответствующую СК, и приведение этого фрагмента в соответствие грамматике CNL.

Остальная часть исходного предложения может в приводимом примере не отвечать грамматике CNL, и требовать обработки другими КСК.

Пример 1 – множественность субъектов действия (sub)

Исходное предложение, множественные члены выделены курсивом:

Системы автоматизации, их элементы и устройства должны надежно работать при следующих температурах окружающей среды:

от 0 до + 45 °С в закрытых помещениях;

от -25 до + 45 °С на открытой палубе.

Дискурс – результат применения КСК:

Субъекты должны надежно работать при следующих температурах окружающей среды:

от 0 до + 45 °С в закрытых помещениях;

от -25 до + 45 °С на открытой палубе.

Первичный субъект это системы автоматизации.

Субъект это первичный субъект.

Субъект это элементы первичного субъекта.

Субъект это устройства первичного субъекта.

Пример 2 – множественность объектов действия (obj)

Исходное предложение:

Температура до +70 °С не должна вызывать повреждений *систем автоматизации, их элементов и устройств*.

Дискурс:

Температура до +70 °С не должна вызывать повреждений *объектов*.

Первичный объект это системы автоматизации.

Объект это первичный объект.

Объект это элементы первичного объекта.

Объект это устройства первичного объекта.

Пример 3 – множественность местоположений (loc)

Исходное предложение:

Электронные элементы и устройства, предназначенные для установки в *распределительные щиты, пульты или кожухи*, должны надежно работать при температуре окружающей среды до +55 °С.

Дискурс:

Электронные элементы и устройства, предназначенные для установки *локация* должны надежно работать при температуре окружающей среды до +55 °С.

Локация это в *распределительные щиты*.

Локация это в *пульты*.

Локация это в *кожухи*.

Мы предполагаем, что технические предметные области (ПрО) характеризуются сравнительно небольшим множеством типовых СК, применяемых в нормативных документах. Затратив сравнительно небольшие ресурсы труда на разработку библиотеки КСК удастся существенно снизить долю ручного труда в разработке CNL спецификаций, и практически решить проблему трудоемкости их разработки.

ЛИТЕРАТУРА

1. Е.Д. Доманцевич, Е.А. Муравьев, Обработка ошибок в компиляторе с контролируемого естественного языка спецификации требований, В сб. материалов научно-практической конференции студентов, аспирантов и молодых ученых “Современные технологии в теории и практике программирования”, СПб, 2025 г., Политех-пресс, стр. 202 – 204.
2. Российский морской регистр судоходства. Правила классификации и постройки морских судов. Часть XV Автоматизация. ФАУ «Российский морской регистр судоходства» 191186, Санкт-Петербург, Дворцовая набережная, 8 www.rs-class.org/ru/

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МНОГОЯЗЫЧНОГО ПРИЛОЖЕНИЯ РЕАЛЬНОГО ВРЕМЕНИ НА ОСНОВЕ ЛОГИКИ НЕПРЕРЫВНЫХ СЕССИЙ

В повседневной жизни языковой барьер редко возникает в спокойной и заранее подготовленной ситуации. Намного чаще он проявляется в движении, в поездке, на учебе, на работе, во время коротких инструкций, бытовых запросов или спонтанного разговора с незнакомым человеком. В условиях глобальной мобильности, международного туризма и постоянных межкультурных контактов задача быстрого и понятного межкультурного общения приобретает не только бытовое, но и прикладное значение. Особенно важны такие решения в тех ситуациях, где человеку необходимо сохранить естественный ритм разговора, быстро понять собеседника и получить перевод без длительных пауз и сложных действий в интерфейсе.

Техническая база для решения такой задачи уже существует. Apple предоставляет на Apple Watch приложение Translate для перевода текста и голоса [1]. Фреймворк Speech предназначен для распознавания устной речи в записанном и живом аудио [3], а WatchConnectivity обеспечивает обмен данными между приложением iPhone и сопряженным приложением watchOS [2]. Однако наличие отдельных технологических блоков еще не устраняет главный практический недостаток большинства существующих решений: пользователь по-прежнему вынужден постоянно вмешиваться в процесс, вручную запускать запись, подтверждать ввод, ждать результат и затем снова активировать следующий цикл. В живом разговоре это разрушает естественный ритм общения и заставляет человека думать не только о смысле реплики, но и об управлении интерфейсом.

В данной работе предлагается иной подход: не эпизодический перевод по отдельным командам, а непрерывная сессия, управляемая логикой Start/Stop. Проект VoxFlow представляет собой распределенное watch-first приложение, в котором Apple Watch выступает основной точкой быстрого взаимодействия, iPhone выполняет роль обязательного вычислительного и коммуникационного спутника, а серверная часть отвечает за учетные записи, синхронизацию и хранение истории. Такой подход выбирается не только из соображений удобства, но и в соответствии с рекомендациями Apple для коротких, glanceable и контекстных взаимодействий на watchOS [4]. После запуска сессии система должна автоматически проходить цикл «прослушивание — обработка — выдача результата» вплоть до явной остановки пользователем. Именно такая логика делает взаимодействие ближе к естественной работе персонального переводчика, а не к набору разрозненных кнопочных операций.

Функционально приложение включает три режима. Первый режим — это разговор-перевод $A \leftrightarrow B$, где реплики двух собеседников на разных языках обрабатываются последовательно, переводятся и при необходимости озвучиваются, а также сохраняются в виде полного транскрипта с временными метками. Второй режим — голосовой ИИ-ассистент, в котором пользователь взаимодействует с интеллектуальным модулем голосом и получает текстовый и аудиоответ. Третий режим — пассивный переводчик: приложение распознает внешнюю речь, показывает перевод в виде субтитров и, при активной опции аудио, воспроизводит перевод через доступный аудиоканал. Для третьего режима существенным является механизм подавления эха: при обнаружении новой внешней речи воспроизведение перевода должно прерываться, чтобы система не мешала восприятию следующей фразы и не ухудшала восприятие последующей речи.

Цель работы состоит в проектировании и реализации многоязычного приложения реального времени на основе логики непрерывных сессий. Для достижения этой цели необходимо решить несколько взаимосвязанных задач: проанализировать существующие мобильные и облачные решения перевода; сформулировать функциональные и нефункциональные требования; обосновать архитектуру Watch-iPhone-Backend; спроектировать устойчивую машину состояний для координации аудиопотока; определить модель данных для сущностей user, session, turn, transcript и preferences; реализовать прототип и подготовить методику его экспериментальной проверки. Важной особенностью проекта является то, что основная инженерная новизна лежит не в самом факте использования STT, машинного перевода или TTS, а в их согласованной оркестрации внутри одной непрерывной сессии.

Практическая и экономическая ценность предлагаемого решения связана прежде всего с повышением автономности пользователя. В типовом мобильном сценарии человеку не нужно каждый

раз доставать телефон, смотреть в экран, повторно запускать запись и тем самым терять нить разговора. Это снижает когнитивную нагрузку, экономит время и делает приложение полезным в ситуациях, где руки заняты или внимание ограничено: в дороге, на тренировке, во время коротких рабочих инструкций, при общении с клиентом, преподавателем или незнакомым собеседником. Кроме того, в простых бытовых и организационных сценариях такая система может уменьшить зависимость от посредника-переводчика, не претендуя при этом на замену профессионального перевода в медицинских, юридических и иных высокорисковых областях. С инженерной точки зрения проект может быть экономически целесообразным, поскольку опирается на уже существующую потребительскую экосистему Apple и не требует создания отдельного специализированного устройства.

С точки зрения программной инженерии работа охватывает сразу несколько значимых направлений: проектирование пользовательского интерфейса для носимого устройства, организацию устойчивого сетевого обмена между клиентскими компонентами и сервером, моделирование данных для хранения сессий и транскриптов, а также обеспечение конфиденциальности пользовательской информации. Теоретическую основу для речевых и переводческих компонентов составляют классические работы по Transformer-моделям и CTC [5, 6], которые остаются важными ориентирами при понимании современных систем распознавания речи и машинного перевода.

Ожидаемым результатом работы является функциональный прототип, который продемонстрирует осуществимость непрерывной многоязычной коммуникации в watch-first формате и подтвердит, что даже в условиях ограничений носимого устройства можно построить полезный, понятный и технически обоснованный инструмент. В дальнейшем проект может быть расширен за счет оптимизации энергопотребления, частичной офлайн-работы, улучшения качества сегментации речи и развития интеллектуальных сценариев обработки транскриптов. Таким образом, работа сочетает реальную человеческую потребность, ясную прикладную пользу и полноценную инженерную задачу уровня выпускной квалификационной работы.

ЛИТЕРАТУРА

1. Apple Support. Translate text and voice on Apple Watch. URL: <https://support.apple.com/guide/watch/translate-apd25ec90584/watchos>
2. Apple Developer Documentation. WatchConnectivity. URL: <https://developer.apple.com/documentation/watchconnectivity>
3. Apple Developer Documentation. Speech. URL: <https://developer.apple.com/documentation/speech>
4. Apple Developer. Designing for watchOS. URL: <https://developer.apple.com/design/human-interface-guidelines/designing-for-watchos>
5. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need. 2017. URL: <https://arxiv.org/abs/1706.03762>
6. Graves A., Fernandez S., Gomez F., Schmidhuber J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. ICML, 2006. URL: https://www.cs.toronto.edu/~graves/icml_2006.pdf

УДК 519.816

М.Р. Есаков, Д.А. Терещенко (2 курс магистратуры),
В.А. Пархоменко, ст. преподаватель

КОМБИНАЦИЯ ЭВРИСТИЧЕСКИХ АЛГОРИТМОВ NIVES, GA, ННО ДЛЯ ПОВЫШЕНИЯ СОВМЕСТИМОСТИ ЭКСПЕРТОВ В ГРУППОВОМ РАНЖИРОВАНИИ АЛЬТЕРНАТИВ

В задачах группового многокритериального принятия решений состав экспертов не является постоянным: участники могут выбывать, заменяться или подключаться на разных этапах проекта. Подобная ротация способна приводить к скачкообразным изменениям весов экспертов и, как следствие, к заметной перестройке ранжирования альтернатив даже при неизменности самой предметной области. Так, например, состав государственной экзаменационной комиссии по направлению может меняться из года в год, однако оценка

выпускных работ и подготовки студентов разных выпусков должна оставаться сопоставимой для работодателя. *Цель работы* – на основе матриц парных сравнений при замене экспертов (по одному на цикл групповой работы) сохранить и повысить уровень консенсуса в обновлённой группе. Под консенсусом (согласием) подразумевается минимальная совместимость с агрегированной групповой матрицей [1]. Минимальная совместимость понимается как наименьшая из попарных совместимостей между оценкой каждого эксперта и групповой агрегированной оценкой: $Cons = \min_{k \in [1, m]} (C_k)$, где C_k – совместимость каждого эксперта и агрегированной оценкой.

Увеличение консенсуса может быть достигнуто алгоритмом HIVES, а также с помощью использования таких эвристик, как Harris Hawk optimization (ННО) [2] или Генетический алгоритм (ГА). HIVES [3] выполняет иерархическую интеграцию оценок, назначая веса экспертам на основании характеристик распределения их оценок по каждому критерию. ННО и ГА направлены на корректировку весов экспертов, чтобы увеличить уровень консенсуса экспертов. Каждый из подходов применяется отдельно. Предлагаемый комбинированный алгоритм предполагает следующее: 1. Посчитать «сырые» веса экспертов. 2. Сформировать целевой вес нового эксперта. 3. Восстановить сумму весов. 4. Оптимизировать веса экспертов с помощью эволюционных методов. 5. Проверить уровень консенсуса экспертов.

При применении HIVES предлагается использовать механизм плавного переноса веса. Пусть p — ушедший эксперт, n — новый, s — критерий. После расчёта базовых весов $\tilde{w}_{s,c}$ (s это номер конкретного эксперта) вводится сглаживание веса нового эксперта как линейной комбинации с весом предшественника: $w_{n,c}^{final} = \alpha \cdot \tilde{w}_{n,c} + (1-\alpha) \cdot w_{p,c}^{old}$, $\alpha \in [0,1]$. При наличии истории участия нового эксперта используется расширенный вариант: $w_{n,c}^{final} = \alpha \cdot \tilde{w}_{n,c} + \beta \cdot \tilde{w}_{n,c}^{hist} + \gamma \cdot w_{p,c}^{old}$, $\alpha + \beta + \gamma = 1$. Далее выполняется восстановление нормировки для всех $s \neq n$: $w_{s,c}^{final} = \tilde{w}_{s,c} \cdot (1 - w_{n,c}^{final}) / (1 - \tilde{w}_{n,c})$. После проводится стандартная иерархическая агрегация HIVES и формируется итоговая оценка $F(A_i)$ и выбор победителя: $A^* = \arg \max_i F(A_i)$. Вместо полного пересчёта весов «с нуля» влияние нового эксперта учитывается постепенно на основе сочетания его текущих оценок с влиянием предыдущего эксперта.

Далее предлагается применение ГА и ННО для поиска такого вектора весов экспертов $W^* = (w_1, \dots, w_n)$, который максимизирует степень совместимости между индивидуальными оценками экспертов и агрегированным групповым решением. Целевая функция для максимизации: $C_{max}(W) = \max_{k \in [1, m]} (\min_{k \in [1, m]} C(R^{(k)}, R_{agg}(W)))$, где C — мера совместимости, $R^{(s)}$ — матрица оценок s -го эксперта, R_{agg} - агрегированная групповая матрица, рассчитанная с весами W . Архитектура разрабатываемого решения отображена на рисунке 1.

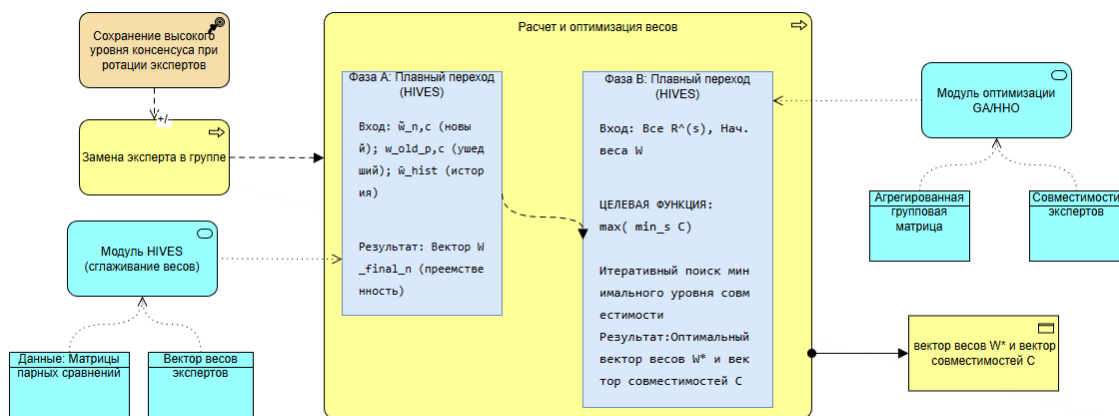


Рисунок 1 – Архитектура разработанного решения

Имплементация HIVES протестирована на результатах [3], а имплементация ННО и ГА – на результатах [1]. Решения сошлись. Эксперименты проводились с синтетическими данными, где было переменное число экспертов и критериев. Датасеты для 5 экспертов по 6 и 8 критериев, а также 7 экспертов по 6 и 8 критериев названы 5e6c, 5e8c, 7e6c, 7e8c,

соответственно. В эксперименте проводилась замена одного эксперта. Таким образом, имелось изначально 4 эксперта, которые участвуют в оценке, и один, который не участвует. После замены одного эксперта HIVES была проведена оценка совместимости и увеличение уровня консенсуса посредством ГА и ННО, обозначенных в таблице 1 под №1 и №2 соответственно. Под «уровнем ЭА» понимается уровень консенсуса после применения эвристических алгоритмов. Исходный уровень консенсуса рассчитывался после замены эксперта. Под приростом понимается количественная мера того, насколько метод улучшил результат по сравнению с исходным уровнем консенсуса.

Таблица 1 – Результаты изменения уровня консенсуса при 10 повторях

Тест	Метод	Исходный уровень	Уровень ЭА	Прирост (%)	Доверительный интервал (95%)
5e6c	1	0.6077	0.6674	9.83	[0.6674, 0.6675]
	2		0.6674	9.83	[0.6674, 0.6675]
5e8c	1	0.5755	0.5848	1.62	[0.5847, 0.5849]
	2		0.5841	1.5	[0.5835, 0.5847]
7e6c	1	0.5534	0.5591	1.03	[0.5584, 0.5598]
	2		0.5595	1.11	[0.5590, 0.5601]
7e8c	1	0.557	0.5647	1.38	[0.5637, 0.5657]
	2		0.5654	1.52	[0.565, 0.566]

Предложенные комбинированные подходы эффективно повышают минимальную совместимость в группе экспертов после замены одного из участников. Во всех тестовых конфигурациях наблюдалось улучшение исходного уровня консенсуса. Наиболее заметный прирост (около 9.83%) был достигнут в тесте 5e6c. Это указывает на высокую эффективность метода в условиях небольшой группы с умеренным числом критериев. Эффективность ГА и ННО сопоставима.

ЛИТЕРАТУРА

1. Calache, L. D. D. R., et al. (2022). A genetic algorithm based on dual hesitant fuzzy preference relations for consensus group decision making. *Applied Soft Computing*, 121, 108778.
2. He J., Xu L., Zhang Y. Harris hawks optimization: Algorithm and applications // *Future Generation Computer Systems*. — 2019. — Vol. 105. — P. 58–75.
3. Boix-Cots, D. A hierarchical integration method under social constraints to maximize satisfaction in multiple criteria group decision making systems / D. Boix-Cots, F. Pardo-Bosch, P. Pujadas // *Expert Systems with Applications*. — 2023. — Vol. 216. — Art. 119471. — P. 1–13. — DOI: 10.1016/j.eswa.2022.119471.
4. Esakov M.R., Tereschenko D.A., Parkhomenko V. A. Metod-HIVES-and-DHF // URL: <https://github.com/Karperash/Metod-HIVES-and-DHF> (дата обращения: 11.01.2026).

УДК 004.453

И.Р. Еникеев (4 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ СТАТИЧЕСКИХ АНАЛИЗАТОРОВ PVS-STUDIO И SONARQUBE

Статический анализ кода позволяет выявлять ошибки и уязвимости без выполнения программы, используя методы анализа синтаксического дерева, потоков данных, символьного выполнения и др. Однако на рынке существует множество анализаторов и сложно выбрать лучший.

Целью данной работы является сравнение возможностей и эффективности анализаторов PVS-Studio [1] и SonarQube [2], доступных в бесплатных версиях, при поиске ошибок в Java-проектах.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Изучить возможности статических анализаторов PVS-Studio и SonarQube.
2. Подобрать набор данных с реальными ошибками в программных проектах.
3. Разработать экспериментальный стенд для автоматического запуска анализа и сбора результатов.
4. Провести сравнительный анализ результатов работы анализаторов.

Для проведения эксперимента использовался датасет Bugs.jar, содержащий 1158 реальных ошибок из крупных проектов с открытым исходным кодом, таких как Accumulo, Camel, Commons Math, Flink, Maven и других [3]. Данный набор содержит различные виды ошибок, не все из которых могут быть обнаружены с помощью статического анализа, но точное распределение по видам ошибок неизвестно. Каждая ошибка в наборе представлена отдельной веткой git-репозитория, содержащей исходный код с ошибкой и соответствующее исправление.

Для автоматизации эксперимента разработан экспериментальный стенд на языке Java [4], выполняющий последовательный анализ веток репозитория. Пайплайн работы программы показан на рисунке 1. Программа получает список веток проекта, переключается между ними с помощью команды git checkout, определяет файлы с изменениями и запускает статический анализ для каждого из них.

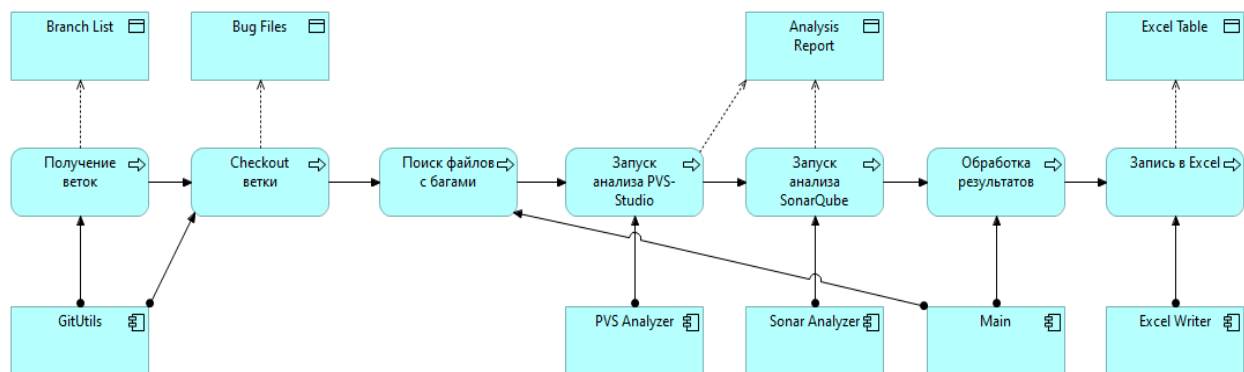


Рисунок 1 – Пайплайн работы программы

Стенд состоит из нескольких модулей:

- Analyzers используется для адаптации вывода различных анализаторов к общему формату для дальнейшего анализа.
- GitUtils используется для работы с git-репозиторием для переключения между ветками, получения diff веток.
- AnalyserComparator используется для сравнения полученных результатов с эталонными, фильтрации результатов и записи в файл.

Работа стенда выполнялась с использованием PVS-Studio и SonarQube, результаты работы анализаторов автоматически собираются и сохраняются в таблицу Excel для последующего анализа. Для работы с Excel используется библиотека Apache POI.

Excel файл имеет 4 колонки:

- Branch – название ветки с ошибкой.
- File – название файла в котором обнаружена ошибка.
- PVS – текст ошибки, состоящий из кода, и описания ошибки, обнаруженной PVS-studio.
- SonarQube – текст ошибки, состоящий из положения ошибки, и описания ошибки, обнаруженной SonarQube.

В ходе эксперимента анализаторы проверяли исходный код проектов и формировали предупреждения о возможных ошибках. После анализа результатов и их ручной проверки выявлены случаи, когда анализаторы обнаруживали предупреждения, соответствующие реальным ошибкам, которые впоследствии были исправлены разработчиками.

Таблица 1 – Обнаруженные анализаторами ошибки

Описание ошибки	Обнаружено PVS-Studio	Обнаружено SonarQube
Поимка слишком общего исключения	-	+
Выражение всегда ложно	+	-
Проверка на null при имеющейся проверке instanceof	-	+
безусловный оператор return внутри цикла	+	+
Неиспользуемые параметры конструктора	+	-
Потенциальное обращение к null	+	-
Ошибка точности вычисления	+	-
Дублирующее присваивание переменной	+	-
Ошибки приведения типов	+	+

По результатам, указанным в таблице 1, видно, что больше видов ошибок обнаружены PVS-Studio, меньше – SonarQube. Всего анализаторами обнаружено 20 ошибок, соответствующих реальным дефектам в проектах. При этом PVS-Studio вывел в отчет 14 предупреждений, связанных с исправленными ошибками, которые касаются потенциально опасных конструкций языка и низкоуровневых дефектов таких, как возможные обращения к null. SonarQube вывел в отчет 6 предупреждений, в основном связанных с качеством кода и нарушением практик разработки, например проверка на null при имеющейся проверке instanceof.

В открытом репозитории размещен стенд и артефакты экспериментов [4]. В дальнейшем авторы планируют продолжать эксперименты с бесплатно распространяемыми версиями статических анализаторов, расширяя как их число, так и число программ для тестирования, количество выявляемых типов ошибок.

ЛИТЕРАТУРА

1. Документация PVS-Studio. – URL: <https://pvs-studio.ru/ru/docs/> (дата обращения: 10.03.2026).
2. SonarQube Documentation. – URL: <https://docs.sonarsource.com/sonarqube/> (дата обращения: 10.03.2026).
3. Just R., Jalali D., Ernst M. Bugs.jar: A Large-Scale, Diverse Dataset of Real-World Java Bugs // Proceedings of the 15th International Conference on Mining Software Repositories. – 2018.
4. Еникеев И. Р., Пархоменко В.А. Стенд для анализа ошибок в проектах (Bug-search). – URL: <https://github.com/Gachikust/Bug-search> (дата обращения: 10.03.2026).

УДК 004.852

А.Р. Зиятдинова (4 курс бакалавриата),
Л.В. Уткин, д.т.н., профессор,
А.В. Константинов, к.ф.-м.н., ст. преподаватель

ОЦЕНКА ЭФФЕКТА ЛЕЧЕНИЯ В ЗАДАЧАХ АНАЛИЗА ВЫЖИВАЕМОСТИ С ИСПОЛЬЗОВАНИЕМ МЕТОДА ПЕРЕГРУППИРОВКИ ДАННЫХ

Задачи оценки эффекта лечения (Treatment Effect Estimation) играют важную роль в различных прикладных областях, таких как медицина, страхование и маркетинг [1]. В рамках подобных задач требуется определить, как конкретное воздействие влияет на показатели времени жизни объекта или пациента. Особый интерес представляет оценка индивидуального эффекта лечения, позволяющая определить, для каких объектов воздействие является наиболее эффективным.

На практике подобные исследования часто проводятся в условиях ограниченного объёма данных. Например, при разработке новых лекарственных препаратов на ранних этапах клинических испытаний может быть доступно лишь ограниченное количество доз препарата. В таких условиях лечение получают только некоторые пациенты, что создаёт дисбаланс между группами лечения и контроля. Подобные ограничения приводят к дисбалансу между группами лечения и контроля и усложняют корректную оценку эффекта воздействия [2].

Формально задача рассматривается в рамках причинно-следственного анализа: для каждого объекта есть два потенциальных исхода — при наличии лечения и без него. Однако в реальных данных наблюдается только один из этих вариантов, что приводит к проблеме отсутствия контрфактических наблюдений [1]. Для оценки индивидуального эффекта воздействия используется показатель Conditional Average Treatment Effect (CATE), характеризующий разницу между ожидаемыми исходами при лечении и без него для объекта с заданными признаками [3].

Сложность добавляет то, что исходом часто является время до наступления события (выздоровление или рецидив). Подобные задачи относятся к области анализа выживаемости (survival analysis) и предполагают наличие цензурированных наблюдений, для которых момент наступления события неизвестен [4]. Наличие цензурирования ограничивает применение стандартных методов оценки эффекта воздействия и требует использования специализированных моделей анализа времени до события.

В данной работе рассматривается задача оценки индивидуального эффекта лечения в условиях анализа выживаемости. В качестве базовой модели используется метод Random Survival Forest, позволяющий учитывать цензурированные наблюдения и оценивать функцию выживаемости для каждого объекта [5]. Помимо применения стандартных подходов оценки CATE, в работе предлагается метод перегруппировки обучающей выборки, направленный на повышение устойчивости оценки эффекта при ограниченном объёме наблюдений.

Предлагаемый подход основан на генерации искусственных наблюдений для объектов контрольной группы. Для каждого такого объекта формируется дополнительный экземпляр, интерпретируемый как наблюдение после лечения. При этом признак лечения изменяется с $T=0$ на $T=1$, а событие считается цензурированным. Время цензурирования для таких наблюдений генерируется случайным образом из заданного временного диапазона. Полученные синтетические объекты добавляются в группу лечения, формируя расширенный набор данных. Подобная перегруппировка позволяет имитировать возможные контрфактические наблюдения и увеличить перекрытие распределений между группами лечения и контроля.

Схема предложенного метода перегруппировки данных представлена на рисунке 1.

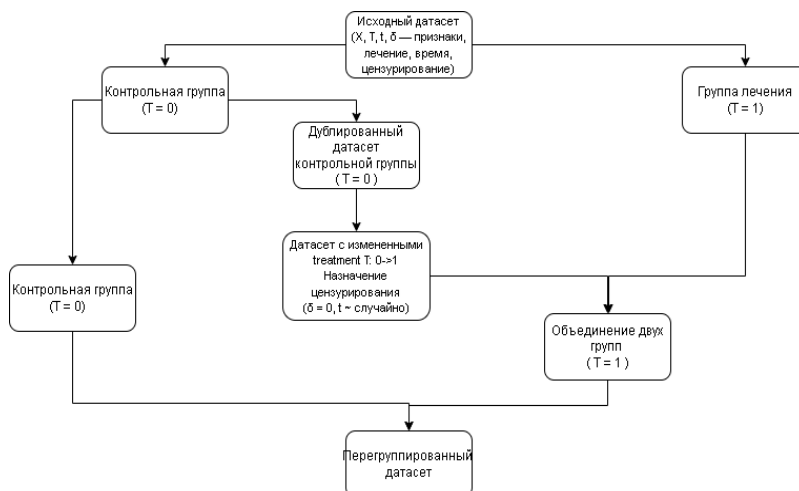


Рисунок 1 – Схема метода перегруппировки данных.

Экспериментальное исследование проводится на синтетических данных, что позволяет использовать известные значения истинного индивидуального эффекта лечения. Для оценки

качества моделей используются метрики PENE, MSE, а также смещение оценки среднего эффекта лечения (ATE bias). Проведение экспериментов позволяет проанализировать влияние предложенного метода перегруппировки на точность оценки эффекта лечения и устойчивость моделей при ограниченном количестве наблюдений в группе лечения.

Таким образом, предложенный подход к перегруппировке данных позволяет повысить устойчивость моделей оценки индивидуального эффекта лечения в задачах анализа времени до наступления событий. Использование искусственно сформированных цензурированных наблюдений позволяет частично компенсировать недостаток данных в группе лечения и улучшить условия обучения моделей причинно-следственного анализа. В дальнейшем планируется проведение экспериментов на реальных медицинских данных, а также исследование возможности применения более сложных моделей анализа выживаемости. Работа выполнена при финансовой поддержке Российского Научного Фонда (проект №25-11-00021).

ЛИТЕРАТУРА

1. Rubin D. B. Causal Inference Using Potential Outcomes // Journal of the American Statistical Association. – 2005. – URL: <https://www.jstor.org/stable/27590541> (дата обращения: 15.03.2026).
2. Künzel S., Sekhon J., Bickel P., Yu B. Meta-Learners for Estimating Heterogeneous Treatment Effects // Proceedings of the National Academy of Sciences. – 2019. – Vol. 116, No. 10. – P. 4156–4165. – DOI: 10.1073/pnas.1804597116.
3. Ishwaran H., Kogalur U., Blackstone E., Lauer M. Random Survival Forests // The Annals of Applied Statistics. – 2008. – Vol. 2, No. 3. – P. 841–860. – DOI: 10.1214/08-AOAS169.
4. Klein J. P., Moeschberger M. L. Survival Analysis: Techniques for Censored and Truncated Data. – New York: Springer, 2003.
5. Imbens G., Rubin D. Causal Inference for Statistics, Social and Biomedical Sciences. – Cambridge: Cambridge University Press, 2015.
6. Utkin L. V., Coolen F. P. A. Survival Analysis Based on Imprecise Probability // International Journal of Approximate Reasoning. – 2011. – Vol. 52, No. 8. – P. 1133–1147. – DOI: 10.1016/j.ijar.2011.06.003.
7. Utkin L. V., Konstantinov A. Counterfactual Explanations of Machine Learning Survival Models // Informatica. – 2021. – URL: <https://informatica.vu.lt/journal/INFORMATICA/article/1238> (дата обращения: 15.03.2026).

УДК 004.89

Э.М. Клеблеева (4 курс бакалавриата),
В.С. Тutyгин, к.т.н, доцент

АДАПТИВНАЯ ОЦЕНКА ШУМА ИЗМЕРЕНИЙ В ФИЛЬТРЕ КАЛМАНА С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ

Фильтр Калмана является одним из наиболее распространённых алгоритмов оценки состояния динамических систем на основе измерений, содержащих шумы или погрешности, изменяющиеся во времени. Он широко применяется в задачах навигации, робототехники, обработки сигналов и управления динамическими объектами. Классический фильтр Калмана обеспечивает оптимальную оценку состояния системы при условии, что статистические характеристики шумов процесса и измерений известны и не изменяются во времени [1,2].

На практике это предположение часто нарушается. В реальных системах уровень шума датчиков может изменяться вследствие внешних воздействий, изменения условий эксплуатации или деградации оборудования. В таких ситуациях использование фиксированных параметров шума приводит к некорректной настройке коэффициента усиления фильтра и снижению точности оценки состояния системы. Подобные проблемы адаптации фильтров Калмана активно исследуются в современных работах по алгоритмам нелинейной и адаптивной фильтрации [3].

Целью данной работы является разработка метода адаптивной настройки фильтра Калмана на основе нейронной сети, позволяющего оценивать текущий уровень шума измерений и автоматически корректировать параметры фильтра.

Рассматривается дискретная модель движения объекта:

$$x_{k+1} = x_k + aT k + w_k$$

где: x_k - вектор состояния системы в момент времени k , a - ускорение, T - шаг дискретизации, w_k - шум процесса;

Измерения датчика описываются выражением:

$$z_k = x_k + v_k$$

где: z_k - измерения датчика, на шаге k , v_k - шум измерения, на шаге k . Классический фильтр Калмана использует следующие основные уравнения:

$P_k^- = P_{k-1} + Q$ - априорная оценка дисперсии ошибки состояния

$K_k = \frac{P_k^-}{P_k^- + R}$ - коэффициент усиления фильтра Калмана

$x_k = x_k^- + K_k(z_k - x_k^-)$ - корректировка оценки состояния

где P_k^- - априорная дисперсия ошибки оценки состояния на шаге k , P_{k-1} - апостериорная дисперсия ошибки на предыдущем шаге, K_k - коэффициент усиления Калмана. x_k^- - априорная оценка состояния системы, полученная из модели движения, x_k - уточнённая (апостериорная) оценка состояния, Q - дисперсия шума процесса, R - дисперсия шума измерений.

Ключевым параметром алгоритма является дисперсия шума измерений R . При некорректной оценке этого параметра фильтр начинает либо чрезмерно доверять измерениям, либо игнорировать их, что приводит к ухудшению точности оценки. В современных исследованиях предлагается использовать методы машинного обучения для адаптивной оценки параметров фильтра Калмана и повышения устойчивости алгоритма к изменяющимся условиям наблюдений [4].

В данной работе предлагается использовать нейронную сеть для оценки текущего уровня шума измерений на основе статистики ошибок измерений (residual). Ошибка измерения определяется как:

$$r_k = z_k - x_k^-$$

В качестве входных признаков нейронной сети используются несколько последних значений residual, а также прогноз состояния системы. Нейронная сеть реализует отображение:

$$\sigma_{\eta,k} = f_{\theta}(r_k, r_{k-1}, \dots, r_{k-n}, x_k^-)$$

где f_{θ} - функция, реализуемая нейронной сетью. Результатом работы сети является оценка текущего стандартного отклонения шума измерений $\sigma_{\eta,k}$. Далее дисперсия шума измерений вычисляется как:

$$R_k = \sigma_{\eta,k}^2$$

и используется для вычисления коэффициента усиления фильтра.

Таким образом формируется адаптивный нейронный фильтр Калмана, который автоматически подстраивается под изменяющийся уровень шума. Подобные гибридные методы, объединяющие нейронные сети и алгоритмы фильтрации, активно развиваются в последние годы. Например, в алгоритме KalmanNet нейронная сеть используется для аппроксимации вычисления коэффициента усиления фильтра на основе наблюдений системы [5]. Схожие идеи применяются также в ряде современных работ по нейросетевой адаптации фильтров Калмана в задачах навигации и анализа временных рядов [6].

Для обучения модели использовались синтетические данные, полученные путём моделирования движения объекта и работы датчика. В экспериментах уровень шума измерений изменялся во времени, что позволило проверить способность модели адаптироваться к изменяющимся условиям.

Нейронная сеть реализована на основе полносвязной архитектуры и обучалась с использованием оптимизатора Adam. Функция потерь определялась как среднеквадратичная ошибка между истинным уровнем шума измерений и оценкой нейронной сети. Разработанная

модель была интегрирована в алгоритм фильтра Калмана и протестирована на серии численных экспериментов. В ходе экспериментов сравнивалась точность оценки состояния объекта для классического фильтра Калмана и предложенного нейросетевого подхода.

Результаты моделирования показали, что использование нейронной сети для адаптивной оценки шума измерений позволяет существенно повысить точность фильтрации. В проведённых экспериментах среднеквадратическая ошибка оценки состояния уменьшилась примерно на 60% по сравнению с классическим фильтром Калмана.

Полученные результаты подтверждают перспективность использования методов машинного обучения для адаптивной настройки параметров алгоритмов фильтрации. Предложенный подход сохраняет математическую структуру классического фильтра Калмана, но обеспечивает его адаптацию к изменяющимся характеристикам шума.

ЛИТЕРАТУРА

1. Simon D. Optimal State Estimation: Kalman, H_∞ , and Nonlinear Approaches. – Hoboken: Wiley, 2006.
2. Grewal M., Andrews A. Kalman Filtering: Theory and Practice Using MATLAB. – 4th ed. – Hoboken: Wiley, 2014.
3. Si J., Niu Y., Wang B. A Review of Nonlinear Filtering Algorithms in Integrated Navigation Systems // Sensors. – 2025.
4. Hao P., Karakus O., Achim A. RKFNet: A Neural Network Aided Robust Kalman Filter // Signal Processing. – 2024.
5. Revach G., Shlezinger N., Ni X. et al. KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics // IEEE Transactions on Signal Processing. – 2022.
6. Chen K., Zhang P., You L., Sun J. Kalman Filter Fusion Navigation Algorithm Assisted by CNN-LSTM Neural Network // Applied Sciences. – 2024.

УДК 004.453

Н.Ю. Каралюс (4 курс бакалавриата),
А.В. Сергеев, к.т.н., доцент

СИСТЕМА АВТОМАТИЧЕСКОЙ БЕСШОВНОЙ РОТАЦИИ КРИПТОГРАФИЧЕСКИХ КЛЮЧЕЙ АВТОРИЗАЦИИ В ВЫСОКОНАГРУЖЕННОЙ МИКРОСЕРВИСНОЙ ПЛАТФОРМЕ

Микросервисная архитектура позволяет независимо разворачивать компоненты системы, однако вместе с ростом числа сервисов расширяется и поверхность атаки [1]. Для межсервисной авторизации широко применяются JWT-токены [2], слабым местом которых является закрытый ключ подписи: его компрометация позволяет злоумышленнику формировать произвольные токены и получать неограниченный доступ ко всем ресурсам платформы [3]. Проблема обостряется, когда единый ключ разделяется между всеми микросервисами: компрометация любого из них равносильна компрометации всей системы авторизации. В платформе Mindbox [4] – экосистеме роста бизнеса, насчитывающей порядка 120 микросервисов в нескольких кластерах Kubernetes, – единый RSA-ключ хранился в HashiCorp Vault [6] и был доступен каждому сервису, а его замена требовала одновременного обновления всех контуров развёртывания, что на практике делало ротацию невозможной.

Цель работы – проектирование, реализация и внедрение системы автоматической бесшовной ротации криптографических ключей в высоконагруженной микросервисной платформе.

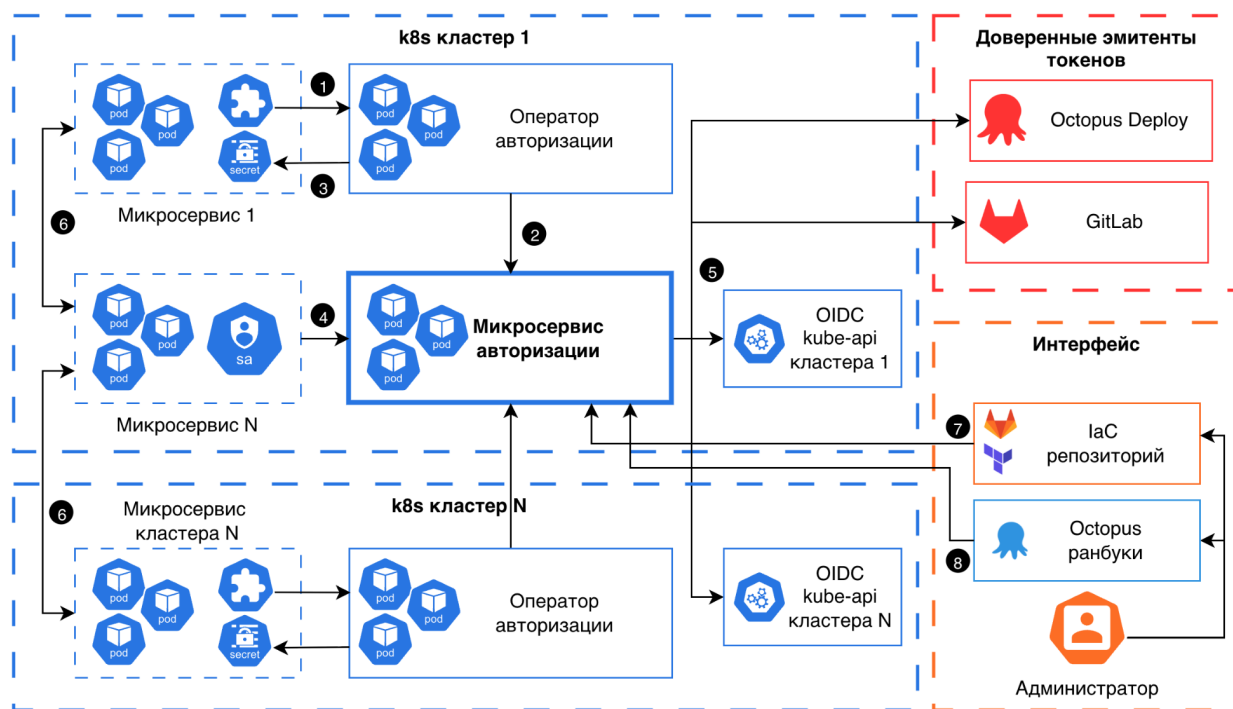
В основу решения положена архитектура с централизованным микросервисом авторизации – единственным компонентом, имеющим доступ к закрытому ключу подписи. Публичная часть ключа предоставляется остальным сервисам через API. В каждом Kubernetes-кластере развёрнут оператор авторизации – контроллер, который с периодичностью 30 секунд считывает спецификацию запрашиваемых токенов из Custom Resource, получает токены и

публичные ключи у сервиса авторизации и записывает их в секрет микросервиса. Благодаря данному push-механизму обновлённые криптографические данные доставляются ко всем микросервисам автоматически, без перезапуска подов.

Процесс ротации реализован в три фазы на основе пары ключей: основного (PRIMARY) и вспомогательного (SECONDARY). На первой фазе задание ротации генерирует новый RSA-ключ и сохраняет его как SECONDARY; токены продолжают подписываться основным ключом, однако при проверке подписи принимаются оба ключа. На второй фазе, по истечении 10 минут, роли ключей меняются: новый ключ становится основным, а прежний – вспомогательным. На третьей фазе, еще через 10 минут, прежний ключ удаляется. Наличие двух одновременно действующих ключей на протяжении переходного периода гарантирует, что ранее выпущенные токены остаются валидными на время распространения нового ключа по всем кластерам.

Архитектура решения представлена на рисунке 1. Для федерации кластеров микросервис авторизации получает публичные ключи через OIDC-интерфейс kube-api [5] каждого кластера и самостоятельно валидирует SA-токены. Конфигурация клиентов и политик управляется декларативно через IaC-репозиторий, а интеграция с внешними эмитентами токенов (Octopus Deploy, GitLab) реализована посредством механизма Token Exchange.

Система успешно апробирована в промышленной эксплуатации платформы Mindbox, обрабатывающей свыше 2 миллионов запросов в минуту. Ротация выполняется автоматически по расписанию без прерывания работы сервисов при сохранении показателя 95-го перцентиля времени выпуска токена не более 30 мс.



- ❶ Оператор раз в 30 секунд читает CR – спецификацию запрашиваемых токенов из микросервиса авторизации.
- ❷ Оператор на основе спецификации получает токены и публичные ключи у микросервиса авторизации.
- ❸ Оператор записывает полученные данные авторизации в секрет микросервиса.
- ❹ Микросервис получает токены от сервиса авторизации, если данные известны только в рантайме. Авторизация через SA токен.
- ❺ Для авторизации субъектов и возможностей федерации сервис авторизации запрашивает публ. ключи доверенных систем.
- ❻ Микросервисы отправляют и авторизуют запросы с помощью полученных данных. В том числе, между разными кластерами.
- ❼ Администратор управляет конфигурацией микросервиса авторизации (клиенты, политики, эмитенты) через IaC репозиторий.
- ❽ Octopus ранбуки дают доступ к ручному администрированию: принудительный запуск ротации, выпуск токенов для отладки.

Рисунок 1 – Архитектура решения

Таким образом, разработана и внедрена система автоматической бесшовной ротации криптографических ключей, обеспечивающая непрерывность обработки запросов и

существенно повышающая безопасность межсервисной авторизации в высоконагруженной микросервисной платформе.

ЛИТЕРАТУРА

1. Research Trends and Recommendations for Future Microservices Research / Z. Stojanov [и др.] // Trudy ISP RAN/Proc. ISP RAS. — 2024. — Т. 36, № 1. — С. 105—130. — DOI 10.15514/ISPRAS-2024-36(1)-7.
2. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT): RFC / Internet Engineering Task Force (IETF). — 2015. — № 7519. — URL: <https://www.rfc-editor.org/rfc/rfc7519.html>.
3. Enhancing Microservices Security with Token-Based Access Control Method / A. Venčkauskas [и др.] // Sensors. — 2023. — Т. 23, № 6. — С. 3363. — DOI 10.3390/s23063363.
4. ООО «Майндбокс». Mindbox — экосистема роста бизнеса. — URL: <https://mindbox.ru/> (дата обращения: 15.03.2026).
5. OpenID Foundation. OpenID Connect Core 1.0. — 2014. — URL: https://openid.net/specs/openid-connect-core-1_0.html (дата обращения: 15.03.2026).
6. Vault by HashiCorp. — URL: <https://www.vaultproject.io/> (дата обращения: 15.03.2026)

УДК 004.42

О.В. Коробицына (4 курс бакалавриата),
Е.В. Скуднева, к.т.н., доцент

РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ТРАЕКТОРИЙ ФАЗОВОГО ПОРТРЕТА СИСТЕМЫ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ НА ОСНОВЕ ТЕХНОЛОГИИ MPI

В теории динамических систем ключевым инструментом анализа является фазовое пространство — многомерное пространство, каждая точка которого соответствует полному состоянию системы в данный момент времени. Траектория в фазовом пространстве описывает эволюцию системы со временем, а изучение этих траекторий позволяет выявлять устойчивые режимы, переходные процессы и характер поведения системы. Фазовые портреты нужны для наглядного и качественного анализа динамических систем. Быстрое построение фазовых портретов дает мощный инструмент для качественного анализа, который лежит в основе понимания и управления динамическими процессами во многих областях науки и техники.

Актуальность работы по разработке параллельных алгоритмов решения систем обыкновенных дифференциальных уравнений (СОДУ) подтверждаются современными работами: диссертации Ващенко Г.В. [1], Д.Н. Бутусова [2], а также работы, посвященные параллельным алгоритмам решения СОДУ в различных областях [3-5].

Рассматривается динамическая система вида

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \quad (1)$$

с начальным условием

$$\mathbf{x}(0) = \mathbf{x}_0, \quad (2)$$

где $\mathbf{x} \in \mathbb{R}^n$, а $f(\mathbf{x}(t))$ — непрерывная функция, которая удовлетворяет локальному условию Липшица в $\mathbb{R}^n$, и для любого $\mathbf{x}_0 \in \mathbb{R}^n$ на конечном интервале времени существует единственное решение системы, которое непрерывно зависит от $\mathbf{x}_0$. Решением задачи Коши (1)-(2) является непрерывная вектор-функция $\mathbf{x}(t)$ такая, что в каждой точки $(t, \mathbf{x}(t))$ кривая $t \rightarrow \mathbf{x}(t)$ должна иметь касательный вектор $f(\mathbf{x})$. Фазовым портретом системы (1) называется совокупность решений задачи (1)-(2) при различных $\mathbf{x}_0$ в рассматриваемой области $G \subset \mathbb{R}^n$. Для вычисления траекторий фазового портрета необходимо воспользоваться численным методом решения систем ОДУ (например, методом Рунге–Кутты 4 порядка).

Для разработки эффективной программы для суперкомпьютера, необходимо учитывать высокую производительность процессоров и большие накладные расходы на коммуникации между вычислительными узлами. Требуется реализовать подход, который снизит

взаимодействие вычислителей при решении задачи. Необходимо разделить задачу на относительно независимые блоки, решение которых не требует интенсивного взаимодействия вычислительных узлов. Предлагается разбить область G на непересекающиеся области (клетки). В каждой клетке задать необходимое число начальных точек траекторий, в зависимости от требуемой плотности траекторий на фазовом портрете. Для каждой начальной точки вычислить точки траектории системы. Тогда получится необходимое количество траекторий, проходящих через заданную клетку. При этом вычисления в разных клетках совершенно независимы друг от друга, их можно производить на разных вычислителях без передачи дополнительных данных. Останется только передать найденные траектории на центральный узел вычислительной системы, где сохранить их в выходной файл.

Эффективность реализации программы была оценена на задаче, имеющей размерность $n=8$, функция правой части имеет линейный вид

$$f(x(t))=(f_1, f_2, \dots, f_n), \quad x(t)=(x_1, x_2, \dots, x_n), \quad f_j(x(t))=\sum_{i=1}^n a_{ij} x_i(t),$$

где a_{ij} — коэффициенты матрицы линейной системы ($i=1,2,\dots,n, j=1,2,\dots,n$).

Степень заполнения фазового пространства задается параметром p . Вдоль каждой оси координат фазовое пространство разбивается пополам p раз. При $p=6$ получим разбиение на $2^p=64$ отрезка по каждой оси. Тогда все фазовое пространство разбивается на 2^{p+n-1} клеток. При $n=8, p=6$ количество клеток составляет 8192. Через каждую клетку должна проходить хотя бы одна траектория. Чтобы это реализовать, достаточно построить траектории с начальными точками внутри каждой клетки. Однако нет необходимости брать начальную точку в той клетке, через которую прошла уже некоторая найденная траектория. В худшем случае потребуется вычислить 16384 траектории, по две на каждую начальную точку, в положительном и отрицательном направлении времени. При реализации программы использована технология MPI [6].

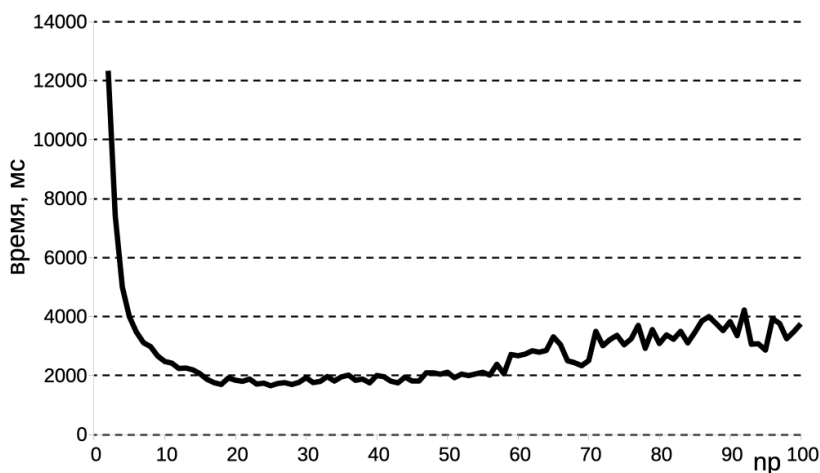


Рисунок 1 – График зависимости времени построения фазового портрета тестовой системы (в мс) от количества используемых вычислительных потоков n_p

В результате эксперимента установлено, что для исследуемой системы при заданной точности требуется вычислить приблизительно 890 000 точек траектории. Для расчета на ПК потребовалось 7,75 с. Также был проведен эксперимент на кластере с вычислительными узлами Intel® Xeon E5-2697 v3 (14 ядер, 2,6 ГГц), вычислительная сеть Mellanox InfiniBand FDR 56 Гбит/с. Было получено минимальное время 1,78 с. Это сократило время вычислений фазового портрета более чем в 4 раза. Из полученных результатов (рисунок 1) видно, что оптимальное значение количества потоков составляет $n_p=16$. Увеличение параметра не позволило сократить время вычисления. Дальнейшая работа предполагает поиск путей уменьшения времени вычислений за счет сокращения объема передаваемых данных между узлами и использования асинхронных вызовов.

Результаты работы получены с использованием вычислительных ресурсов суперкомпьютерного центра Санкт-Петербургского политехнического университета Петра Великого (www.spbstu.ru).

ЛИТЕРАТУРА

1. Ващенко Г. В. Разработка параллельных одношаговых методов и комплекса программ для решения задач химической кинетики и биологии: Дис. канд. физ.-мат. наук: 05.13.18 / Ващенко Г. В. Красноярск, 2012. – 132 с.
2. Бутусов Д. Н. Численные методы моделирования хаотической динамики: Дис. докт. физ.-мат. наук: 1.2.2 / Бутусов Д. Н. Санкт-Петербург, 2025. – 317 с.
3. Борисов Д. К., Яковлев А. Н., Скуднева Е. В. Реализация операций численного решения дифференциальных уравнений в ССУБД Катарсис // Современные технологии в теории и практике программирования: сборников материалов научно-практической конференции студентов, аспирантов и молодых ученых, 23-24 апреля 2025 г. – СПб.: ПОЛИТЕХ-ПРЕСС, 2025. – С. 23–25.
4. Ампилов Н. Б., Соловьев И. П., Кадомский А. А. О методах исследования динамических системы // Компьютерные инструменты в образовании, 2024. № 1. – С. 5–17.

УДК 004.021

Д.О. Королев (2 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент

ДИСКРЕТНО-СОБЫТИЙНОЕ МОДЕЛИРОВАНИЕ КАК ИНСТРУМЕНТ ПОВЫШЕНИЯ НАДЕЖНОСТИ ОБЛАЧНЫХ ИНФРАСТРУКТУР

В условиях активной цифровой трансформации экономики особую значимость приобретает вопрос обеспечения бесперебойного функционирования распределенных систем обработки данных, лежащих в основе облачных инфраструктур. Высокая сложность таких систем обуславливает существенные риски возникновения отказов, способных повлечь за собой нарушение соглашений об уровне обслуживания (SLA), финансовые издержки и снижение репутации сервис-провайдеров. Традиционные методы верификации и валидации надежности зачастую демонстрируют недостаточную эффективность в условиях современных облачных сред. В качестве альтернативного подхода в настоящей работе предлагается использование дискретно-событийного моделирования для целей анализа и последующего повышения надежности систем [1].

В рамках исследования выполнен сравнительный анализ четырех основных подходов моделирования облачных структур:

- физическое моделирование обеспечивает высокую степень достоверности выходных данных, однако сопряжено со значительными финансовыми затратами и сложностями при масштабировании, что ограничивает его применимость для анализа крупных распределенных систем обработки данных.

- системное моделирование предоставляет возможность исследования макропараметров и долгосрочных эффектов управленческих воздействий, но не позволяет проводить детальный анализ поведения отдельных компонентов и учитывать влияние стохастических факторов.

- агентное моделирование ориентировано на изучение взаимодействий множества автономных сущностей, включая применение вероятностных моделей для формализации процессов принятия решений. Несмотря на высокий потенциал, данный подход характеризуется значительными требованиями к вычислительным ресурсам и сложностью верификации ввиду стохастической природы поведения агентов [2].

- дискретно-событийное моделирование представляется наиболее перспективным инструментом, поскольку позволяет с высокой степенью детализации воспроизводить функционирование системы в виде последовательности событий, моделировать случайные события и степени интенсивности входящего потока запросов, а также осуществлять анализ

временных характеристик и тестирование сценариев восстановления без вмешательства в работу промышленной инфраструктуры [3].

Методология дискретно-событийного моделирования активно применяется для исследования надежности облачных систем распределенной архитектуры. В работе [4] представлен подход к разработке, ориентированный на использование моделей (Model-Driven Development, MDD), базирующийся на принципах дискретно-событийного моделирования. Применение формальных моделей в качестве фундамента для разработки упрощает процесс расширения системы и позволяет детально воспроизводить динамику облачной среды, включая обработку событий, управление вычислительными ресурсами и механизмы отказоустойчивости.

В исследовании [5] предложен метод повышения точности моделей отказоустойчивости путем симуляции активности агентов в рамках дискретно-событийной парадигмы. Каждый компонент системы (например, сервер) описывается конечным набором состояний (функционирование, сбой, исключение из пула) и правилами переходов между ними. Такой подход позволяет детально моделировать процессы деградации и последующего восстановления вычислительного кластера, что способствует повышению точности прогнозирования коэффициента готовности и других показателей надежности.

В ходе работы был проведен анализ инструментальных средств реализации указанного подхода, а именно AnyLogic, OMNeT++ и CloudSim. Платформа AnyLogic признана наиболее релевантной для реализации задач дискретно-событийного моделирования благодаря наличию встроенной поддержки как агентного, так и дискретно-событийного подходов. Инструмент предоставляет обширные возможности визуализации и проведения оптимизационных экспериментов, позволяя осуществлять настройку параметров и идентификацию узких мест системы. Несмотря на необходимость высокой квалификации разработчиков и существенные вычислительные затраты при моделировании масштабных инфраструктур, данные ограничения могут быть нивелированы за счет оптимизации программного кода и применения более производительных аппаратных платформ [6].

В качестве количественной метрики надежности в работе предлагается использование коэффициента готовности (Availability Factor, AF). Данный показатель рассчитывается по формуле:

$$AF = \frac{MTBF}{MTBF + MTTR} \quad (1)$$

где MTBF – среднее время между сбоями системы (Mean Time Between Failures);

MTTR – среднее время восстановления (Mean Time To Repair).

Эта метрика отражает реальную доступность системы и учитывает как частоту отказов, так и скорость восстановления [7].

В рамках исследования разработана имитационная модель в среде AnyLogic, включающая в себя следующие компоненты: источник запросов (реализованный как пуассоновский поток событий), балансировщик нагрузки (поддерживающий циклическую стратегию, стратегию наименьшего числа соединений и взвешенного распределения) и совокупность серверов, для которых предусмотрена возможность моделирования отказов и последующего восстановления. Каждый сервер представлен в виде системы массового обслуживания, функционирование которой описывается с использованием модели конечного автомата. Результаты моделирования позволили проследить динамику изменения коэффициента готовности и визуализировать полученные данные в форме гистограммы распределения.

Анализ полученной гистограммы свидетельствует о том, что моделируемая система в целом функционирует устойчиво. Вместе с тем, наличие значений коэффициента готовности, приближающихся к нулю, указывает на существование определенных проблем в обеспечении надежности. На основе проведенного анализа были систематизированы факторы, негативно влияющие на уровень надежности, и предложены меры по его повышению. К числу таких мер относятся: минимизация времени восстановления за счет реализации механизмов

автоматического переключения на резервные узлы; повышение уровня отказоустойчивости посредством микросервисной декомпозиции и репликации данных [8].

Проведенное исследование позволяет заключить, что дискретно-событийное моделирование является эффективным инструментом анализа надежности облачных систем и формирования практических рекомендаций, направленных на ее повышение. Дальнейшие научные изыскания могут быть сосредоточены на разработке адаптивных алгоритмов прогнозирования сбоев и создании гибридных моделей, интегрирующих различные подходы к моделированию.

ЛИТЕРАТУРА

1. Замятина О. М. Моделирование систем: Учебное пособие, 2009.
2. Sabutkevich A.M., Vikhlyaev D.A., Kubov N.A., Nikiforov I.V., Samochadin A.V. Method of Generation of Alternatives of Multi-agent System Behavior Using Simulation System // Gibadullin, A. (eds) Digital and Information Technologies in Economics and Management. DITEM 2024. Lecture Notes in Networks and Systems, vol 1422. - Springer, Cham, 2025. - Pp. 55-68. - DOI: 10.1007/978-3-031-94273-0_5.
3. Jefferson David R. Virtual time // ACM Transactions on Programming Languages and Systems, 1985, no 7(3), pp. 404–425.
4. Лонишин М.Р., Сабуткевич А.М., Никифоров И.В. Метод модели-ориентированной разработки на основе дискретно-событийного моделирования // Современные технологии в теории и практике программирования: Сборник материалов конференции, Санкт-Петербург, 24 апреля 2025 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2025, С. 284-285, EDN ACJXEE.
5. Sabutkevich A.M., Nikiforov I.V., Samochadin A.V. A method for modeling of individual agent behavior in the process-network paradigm of discrete-event simulation. Computing, Telecommunications and Control, 2025, vol. 18, no. 1, pp. 23–35. DOI: 10.18721/JCSTCS.18102.
6. Кузьмин Д.В. Разработка имитационной дискретно-событийной модели транспортной инфраструктуры с использованием инструментов оптимизации // T-Comm, 2023, Т. 17, № 2, С. 42–48.
7. Сухоплюев Д.И., Назаров А.Н. Мониторинг отказоустойчивости в распределенных системах // Computational nanotechnology, 2024, № 4.
8. Карпович М.Н. Особенности проектирования микросервисно-событийных архитектур для высоконагруженных распределенных систем обработки информации // Труды БГТУ. Серия 3: Физико-математические науки и информатика, 2023, № 1 (266), С. 89-95.

УДК 004.94

Н.А. Кубов (1 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент,
С.М. Устинов, д.т.н., профессор

СТАТИЧЕСКИЙ АНАЛИЗ ИМИТАЦИОННЫХ МОДЕЛЕЙ И СИНТЕЗ ПЛАНОВ ИХ ИСПОЛНЕНИЯ В РАСПРЕДЕЛЕННЫХ СРЕДАХ

По мере роста сложности имитационных моделей и повышения требований к скорости симуляций всё более заметными становятся ограничения наивных подходов к их распределённому исполнению [1]. Эффективность запуска в распределённой среде определяется не только доступными вычислительными ресурсами, но и структурой самой модели. Поэтому задача статического анализа модели и синтеза плана её исполнения до начала симуляции является актуальной как с теоретической, так и с практической точки зрения [2].

Переход от общего направления имитационного моделирования к формализму Parallel Discrete Event System Specification (PDEVS) в данной работе обусловлен необходимостью опоры на аппарат, который, с одной стороны, достаточно универсален для описания широкого класса дискретно-событийных систем, а с другой – предоставляет строго определённую

структуру для анализа. PDEVS позволяет описать модель через атомарные и связанные компоненты, порты, связи, иерархию и явные временные характеристики. Именно эти элементы непосредственно влияют на поведение модели при параллельном и распределённом исполнении и потому могут служить основой для статического анализа и синтеза плана исполнения.

Целью работы является сокращение времени распределенного исполнения PDEVS-моделей при использовании существующих программных средств имитационного моделирования за счет внедрения подхода, основанного на статическом анализе модели и компиляции плана исполнения, учитывающего структуру связей, временные ограничения и ожидаемую вычислительную активность компонентов ещё до запуска симуляции.

Предлагаемый подход начинается с построения канонического промежуточного представления PDEVS-модели. Такое представление явно фиксирует иерархию атомарных и связанных моделей, множество портов, связи между компонентами, правила разрешения событийных конфликтов и доступные временные характеристики. Далее над этим представлением выполняется статический анализ, ориентированный на свойства модели релевантные для исполнения. На этом уровне формируются: граф связанности компонентов и граф связей компонентов на уровне портов, вычисляется глубина иерархии модели, потенциальная стоимость ее декомпозиции, сильносвязные компоненты, признаки локальности взаимодействий и вероятные точки наибольших коммуникационных затрат.

Отдельный блок составляют априорные оценки характеристик исполнения. Поскольку реальные показатели исполнения до запуска неизвестны, планировщик не может напрямую оптимизировать количество откатов, время блокировки или максимальное потребляемое количество памяти. Вместо этого вводятся прокси-метрики: оценка коммуникационных затрат, прогнозируемая нагрузка логических процессов, оценка давления на память, оценка риска откатов и оценка риска блокировок. Эти оценки строятся на основе структуры модели и заданных ограничений.

На основе полученных признаков планировщик должен синтезировать план исполнения в трёх вариантах: наивном, основанном только на графовой структуре и учитывающем специфику PDEVS. Такое разделение позволит сравнить выигрыш, обеспечиваемый статическим разбиением как таковым, графовым разбиением и использованием информации, специфичной для PDEVS.

Для оптимистического распределённого исполнения целевая функция должна минимизировать трафик между логическими процессами, каскады откатов, объем антисообщений и накладные расходы на сохранение состояний. Для консервативного или гибридного исполнения приоритет смещается к увеличению доступного опережения (lookahead) [3], уменьшению блокировок и сокращению числа жёстких межкомпонентных зависимостей.

Валидацию предлагается проводить на нескольких инструментах, выполняющих разные роли. PythonPDEVS целесообразно использовать как основную распределённую вычислительную платформу, поскольку он поддерживает распределённое исполнение PDEVS и позволяет наблюдать статистику откатов и коммуникационные накладные расходы [4]. xDEVS удобно использовать как вторую платформу для проверки переносимости подхода построения плана исполнения и сравнения с современными оптимизациями для систем с общей памятью [5]. Cadmium целесообразен как эталонная типизированная платформа PDEVS и источник открытых моделей [6]. В качестве набора тестовых задач должны использоваться модели семейства DEVStone [7], коммуникационно-нагруженные сценарии и по меньшей мере один структурно сложный реальный пример.

Основными измеряемыми показателями после применения подхода синтеза плана исполнения моделей в распределенных средах являются: общее время исполнения, ускорение, коэффициенты масштабируемости, число и размер межпроцессных сообщений, фактический дисбаланс нагрузки, пиковое потребление памяти и накладные расходы на планирование. Для оптимистичных алгоритмов симуляции следует дополнительно учитывать число откатов, а

для консервативных время блокировок и трафик нулевых сообщений. Отдельно должны детерминированность результатов при повторных запусках и переносимость подхода между платформами. Таким образом, вклад работы состоит в построении внешнего слоя статического анализа и компиляции, который превращает PDEVs-модель в план её распределённого исполнения.

Ожидаемым результатом является разработка методики, объединяющей промежуточное представление PDEVs, структурный анализ, априорное оценивание характеристик исполнения и синтез плана выполнения, а также экспериментальное подтверждение того, что такие планы способны стабильно превосходить подходы, не учитывающие топологию модели. Практическая значимость работы определяется возможностью повышать эффективность распределённого исполнения существующих PDEVs-моделей без изменения их предметной логики и без необходимости переписывания самих симуляторов.

ЛИТЕРАТУРА

1. Кубов, Н. А. Применение метода горизонтального масштабирования для исполнения имитационных моделей в облачной среде / Н. А. Кубов, А. М. Сабуткевич, И. В. Никифоров // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 221–223. – EDN VMXBTW.
2. Wainer G., Govind S. 100 volumes of SIMULATION—20 years of DEVS research // SIMULATION. 2024. Vol. 100, No. 12. P. 1297–1318. DOI: 10.1177/00375497241291871.
3. Fernandez A. Comparative analysis of parallel discrete event simulation algorithms: Time Warp, Window Racer, and Null Messages / A. Fernandez, A. Gazizova, I. Nikiforov [et al.] // Fourth International Conference on Optics, Computer Applications, and Materials Science (CMSD-IV 2024) : Proceedings of SPIE - The International Society for Optical Engineering, Dushanbe, Tajikistan, 23–25 декабря 2024 года. Vol. 13651. – Bellingham, Washington: SPIE - International Society for Optics and Photonics, 2025. – P. 136510D. – DOI 10.1117/12.3060495. – EDN EWPPXL.
4. Van Tendeloo Y., Vangheluwe H. PythonPDEVs: a distributed parallel DEVS simulator // Proceedings of the 2015 Spring Simulation Multiconference, SpringSim (TMS-DEVS). 2015. P. 91–98.
5. Risco-Martín J. L., Mittal S., Henares K., Cárdenas R., Arroba P. xDEVs: A toolkit for interoperable modeling and simulation of formal discrete event systems // Software: Practice and Experience. 2023. Vol. 53, No. 3. P. 748–789. DOI: 10.1002/spe.3168.
6. Wainer G., Ruiz Martin C. Defining DEVS Models Using the Cadmium Toolkit // Proceedings of the 2022 Winter Simulation Conference. 2022. P. 1356–1370. DOI: 10.1109/WSC57314.2022.10015492.
7. Cárdenas R., Henares K., Arroba P., Risco-Martín J. L., Wainer G. The DEVStone Metric: Performance Analysis of DEVS Simulation Engines // ACM Transactions on Modeling and Computer Simulation. 2022. Vol. 32, No. 3. Article 21. DOI: 10.1145/3543849.

УДК 004.94

Н.А. Кубов (1 курс аспирантуры),
А.М. Сабуткевич (3 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент

АКТУАЛЬНОСТЬ МЕТОДОВ СТАТИЧЕСКОГО АНАЛИЗА, КОМПИЛЯЦИИ И УПРАВЛЕНИЯ ИСПОЛНЕНИЕМ ИМИТАЦИОННЫХ МОДЕЛЕЙ В РАСПРЕДЕЛЕННЫХ СРЕДАХ

Имитационное моделирование широко используется для исследования и проектирования сложных технических, организационно-технических и киберфизических систем. Вместе с тем рост масштаба моделей, повышение требований к их детализации и увеличение объёма вычислений приводят к существенному росту времени исполнения [1]. В современной литературе параллельное и распределённое исполнение имитационных моделей рассматривается как одно из ключевых направлений повышения производительности и

масштабируемости вычислительных экспериментов [2, 3]. Использование распределённых и облачных сред открывает возможности для ускорения исполнения моделей, однако требует специальных методов организации процесса моделирования [4, 5].

Одной из основных проблем при исполнении имитационных моделей в распределённых средах остаётся эффективное разбиение модели на логические процессы, их размещение на вычислительных узлах и последующее управление исполнением особенно в средах, где необходимы методы адаптивного распределения задач и балансировки нагрузки [6, 7].

В связи с этим актуальной научной задачей является разработка комплексных методов статического анализа, компиляции и управления исполнением имитационных моделей, позволяющих автоматически формировать и корректировать план исполнения, рационально использовать ресурсы распределённой среды и повышать производительность существующих платформ имитационного моделирования

Целью исследования является повышение производительности платформ имитационного моделирования при исполнении имитационных моделей в распределённых средах за счет внедрения методов статического анализа, компиляции и управления исполнением имитационных моделей, которые реализованы в программно-аппаратном комплексе.

Результатом работы предполагается появление следующих пунктов, выражающих научную новизну.

1. Систематизация и сравнительный анализ существующих методов статического анализа, компиляции и управления исполнением программных систем с выделением их ключевых особенностей, достоинств и ограничений применительно к решению задач повышения производительности исполнения имитационных моделей в распределённых средах.

2. Метод статического анализа имитационных моделей, основанный на исследовании структуры модели, причинно-временных связей между ее компонентами и оценке их предполагаемой вычислительной активности, для выявления тесно связанных компонентов модели, прогнозирования интенсивности обмена сообщениями, а также для выбора эффективного разбиения модели на логические процессы.

3. Метод компиляции имитационной модели, основанный на преобразовании результатов структурного, причинно-временного и стоимостного анализа модели для построения плана исполнения, включающего разбиение модели на логические процессы, их размещение на вычислительных узлах и формирование параметров запуска, необходимых для эффективного исполнения имитационной модели в распределённой среде на поддерживаемых программных платформах имитационного моделирования с целью повышения их производительности.

4. Метод управления исполнением имитационной модели в распределённых и облачных средах, основанный на контролируемом изменении плана исполнения имитационной модели по результатам мониторинга текущего состояния и характеристик вычислительной среды, для перераспределения нагрузки и обновления схемы размещения логических процессов.

Таким образом вытекают задачи, необходимые для достижения цели.

1. Проведение исследования и систематизации методов статического анализа, компиляции и управления исполнением программных систем, с целью выявления их применимости для области имитационного моделирования в распределённых средах. А также анализ применяемых инструментов и бэнчмарков для их оценки.

2. Предложение метода статического анализа имитационных моделей для выявления ключевых признаков, используемых для разбиения модели на отдельные независимые участки.

3. Предложение метода компиляции имитационных моделей для построения плана исполнения моделей с целью сокращения времени исполнения моделей в распределённых средах.

4. Предложение метода управления исполнением имитационных моделей, обеспечивающего мониторинг вычислительных узлов и ограниченную адаптацию распределения нагрузки в ходе симуляции.

5. Реализация предложенных методов в программно-аппаратном комплексе.

6. Проведение экспериментальных исследований на репрезентативных имитационных моделях и бэнчмарках с оценкой времени исполнения, ускорения, метрик межузлового обмена сообщениями, накладных расходов на синхронизацию при использовании предложенного программно-аппаратного комплекса по сравнению с существующими решениями.

Опираясь на вышеизложенные пункты актуальности, научной новизны и поставленных целей можно сделать вывод о необходимости разработки прототипа системы, реализующей методы статического анализа, компиляции, управления исполнением имитационных моделей в распределенных средах для существующих платформ имитационного моделирования с целью повышения их производительности. Сравнительный анализ существующих методов, архитектурную проектировку, процесс разработки прототипа и конечные результаты работы будут представлены в дальнейших исследованиях.

ЛИТЕРАТУРА

1. Sabutkevich, A. M. Method of Generation of Alternatives of Multi-agent System Behavior Using Simulation System / A. M. Sabutkevich, D. A. Vikhlyayev, N. A. Kubov [et al.] // Digital and Information Technologies in Economics and Management: Proceedings of the International Scientific and Practical Conference, Dushanbe, Tajikistan, 20–24 ноября 2024 года. – Cham, Switzerland: Springer Nature Switzerland AG, 2025. – P. 55–68. – DOI 10.1007/978-3-031-94273-0_5. – EDN DFPJHT.
2. Fujimoto, R. M. Research Challenges in Parallel and Distributed Simulation // ACM Transactions on Modeling and Computer Simulation. – 2016. – Vol. 26, No. 4. – Art. 22. – DOI 10.1145/2866577
3. Fujimoto, R. M. Development of the Parallel and Distributed Simulation Field // SIMULATION. – 2024. – Vol. 100, No. 12. – P. 1197–1223. – DOI 10.1177/00375497241261407.
4. Кубов, Н. А. Применение метода горизонтального масштабирования для исполнения имитационных моделей в облачной среде / Н. А. Кубов, А. М. Сабуткевич, И. В. Никифоров // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 221–223. – EDN VMXBTW.
5. Risco-Martin, J. L., Henares, K., Mittal, S., Almendras, L. F., Olcoz, K. A Unified Cloud-Enabled Discrete Event Parallel and Distributed Simulation Architecture // Simulation Modelling Practice and Theory. – 2022. – Vol. 118. – Art. 102539. – DOI 10.1016/j.simpat.2022.102539.
6. D'Angelo, G. The Simulation Model Partitioning Problem: An Adaptive Solution Based on Self-Clustering // Simulation Modelling Practice and Theory. – 2017. – Vol. 70. – P. 1–20. – DOI 10.1016/j.simpat.2016.10.001.
7. Devi, N., Dalal, S., Solanki, K. et al. A Systematic Literature Review for Load Balancing and Task Scheduling Techniques in Cloud Computing // Artificial Intelligence Review. – 2024. – Vol. 57. – Art. 276. – DOI 10.1007/s10462-024-10925-w.

УДК 004.934

А.В. Ларин (1 курс аспирантуры),
П.Д. Дробинцев, к.т.н., доцент

ОЦЕНКА ПРОСОДИЧЕСКОЙ ИНФОРМАЦИИ В САМООБУЧАЮЩИХСЯ ПРЕДСТАВЛЕНИЯХ РЕЧИ: СРАВНИТЕЛЬНЫЙ АНАЛИЗ HuBERT И WavLM

Задача синтеза естественной выразительной речи остаётся одной из ключевых в области обработки речевых сигналов. Просодия – совокупность супraseгментных характеристик речи, включающих интонацию, ритм, темп и паузацию, – организована иерархически: от фонемного уровня до дискурсного, и именно её адекватное воспроизведение в значительной мере определяет воспринимаемую естественность синтезированной речи. В последние годы

самообучающиеся модели речевых представлений (Self-Supervised Learning, SSL), такие как HuBERT [1] и WavLM [2], продемонстрировали высокую эффективность в задачах распознавания речи и верификации говорящего. Вместе с тем применимость SSL-представлений для моделирования просодии исследована существенно меньше [3], что формирует значимый научный пробел в контексте разработки современных систем синтеза речи.

Настоящая работа посвящена сравнительному анализу просодической информативности представлений, извлекаемых моделями HuBERT-Base и WavLM-Large, в сопоставлении с традиционным акустическим описанием на основе мел-спектрограмм. В качестве основного исследовательского вопроса рассматривается следующий: в какой мере SSL-представления кодируют просодические характеристики речи – высоту основного тона (F0), длительность и энергию – и насколько эти характеристики отделимы от фонетического содержания высказывания?

Для достижения поставленной цели решаются следующие задачи. Во-первых, проводится количественная оценка способности различных слоёв трансформерных энкодеров восстанавливать траекторию F0 посредством обучения линейного регрессора. Во-вторых, исследуется степень «запутанности» (entanglement) просодических и фонетических признаков в пространстве представлений через введение метрики просодическо-фонетической независимости (PCIS). В-третьих, проводится сравнительный анализ эффективности рассматриваемых архитектур применительно к задаче классификации эмоций как косвенного показателя качества просодического моделирования.

Экспериментальная часть основана на корпусах LibriTTS и ESD (Emotional Speech Dataset). В качестве метрик применяются коэффициент корреляции Пирсона для восстановления F0, F1-мера для классификации эмоций, а также предлагаемая метрика PCIS, основанная на оценке взаимной информации между просодическими и фонетическими признаками пространства представлений.

Предварительные результаты показывают, что SSL-модели уступают мел-спектрограммам в задаче восстановления F0 на 7-15%, однако WavLM, за счёт совместного обучения на задачах маскированного предсказания и шумоподавления [2], демонстрирует лучшие показатели среди SSL-архитектур. Анализ послойного поведения выявляет, что просодические признаки концентрируются преимущественно в нижних слоях трансформерного энкодера, тогда как верхние слои кодируют преимущественно семантический контент. Данная закономерность согласуется с лингвистической концепцией иерархической организации просодии и открывает возможность для адресного извлечения просодических представлений с целью их использования в системах нейронного синтеза речи [4].

Практическая значимость работы определяется возможностью применения разработанных метрик и полученных выводов в архитектурах иерархического моделирования просодии для голосовых нейросетевых систем, ориентированных на выразительный и управляемый синтез речи. Кроме того, результаты служат основой для разработки SSL-архитектур с явным разделением пространства представлений на просодические и фонетические подпространства [5].

ЛИТЕРАТУРА

1. Hsu W.-N., Bolte B., Tsai Y.-H. H., Lakhota K., Salakhutdinov R., Mohamed A. HuBERT: Self-supervised speech representation learning by masked prediction of hidden units // IEEE/ACM Transactions on Audio, Speech, and Language Processing. – 2021. – Vol. 29. – P. 3451–3460. – DOI: 10.1109/TASLP.2021.3122291.
2. Chen S., Wang C., Chen Z., Wu Y., Liu S., Chen Z., Li J., Kanda N., Yoshioka T., Xiao X. [и др.] WavLM: Large-scale self-supervised pre-training for full stack speech processing // IEEE Journal of Selected Topics in Signal Processing. – 2022. – Vol. 16, № 6. – P. 1505–1518. – DOI: 10.1109/JSTSP.2022.3188113.

3. Mohamed A., Lee H.-Y., Borgholt L., Havtorn J. D., Edin J., Igel C., Kirchhoff K., Li S.-W., Livescu K., Maaløe L., Sainath T. N., Watanabe S. Self-supervised speech representation learning: A review // IEEE Journal of Selected Topics in Signal Processing. – 2022. – Vol. 16, № 6. – P. 1179–1210. – DOI: 10.1109/JSTSP.2022.3207050.
4. Liu Z.-C., Chen L., Hu Y.-J., Ling Z.-H., Pan J. PE-wav2vec: A prosody-enhanced speech model for self-supervised prosody learning in TTS // IEEE/ACM Transactions on Audio, Speech, and Language Processing. – 2024. – Vol. 32. – P. 4199–4210. – DOI: 10.1109/TASLP.2024.3449148.
5. Polyak A., Adi Y., Copet J., Kharitonov E., Lakhota K., Hsu W.-N., Mohamed A., Dupoux E. Speech resynthesis from discrete disentangled self-supervised representations // Proceedings of Interspeech 2021. – ISCA, 2021. – P. 3615–3619.

УДК 004.94

М.Р. Лонишин (4 курс бакалавриата),
А.М. Сабуткевич (3 курс аспирантуры),
Н.А. Кубов (1 курс аспирантуры),
А.В. Самочадин, к.т.н., доцент

ОЦЕНКА КОММУНИКАЦИОННЫХ ИЗДЕРЖЕК ПРИ РАСПРЕДЕЛЕННОМ ОБУЧЕНИИ LLM НА ОСНОВЕ ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ

Большие языковые модели (LLM) активно применяются в различных наукоемких и прикладных областях, к примеру медицинская диагностика и анализ юридической документации [1]. Рост возможностей таких моделей тесно связан с увеличением количества их параметров, что, в свою очередь, ведет к существенному росту затрат на их обучение. Для преодоления ограничений одиночных вычислительных систем обучение современных моделей с десятками и сотнями миллиардов параметров требует использования распределенных вычислительных кластеров. В таких условиях часть времени обучения расходуется не на полезные вычисления, а на межузловые коммуникации, что делает задачу повышения эффективности коммуникаций крайне актуальной.

Целью работы является разработка подхода для оценки коммуникационных издержек при распределённом обучении LLM для снижения времени обучения LLM за счёт уменьшения времени коммуникаций между узлами.

В ходе анализа предметной области были изучены существующие решения для оценки производительности сетей в задачах машинного обучения, такие как Calculon [2] и DistSim [3]. Указанные аналоги либо базируются на высокоуровневых аналитических моделях, теряющих детализацию, либо требуют доступа к реальному GPU-кластеру для сбора трассировочных данных, что ограничивает возможность проведения экспериментов на ранних этапах проектирования. В связи с этим возникла необходимость разработки инструмента для оценки издержек на коммуникацию без доступа к физическому оборудованию, сохраняя высокую детализацию сетевых процессов.

Обучение современных моделей на промышленных масштабах требует использования тысяч ускорителей, что порождает необходимость применения стратегий параллелизма [4]: Data Parallelism (DP) – распараллеливание по данным; Tensor Parallelism (TP) – разделение матричных операций; Pipeline Parallelism (PP) – конвейеризация слоев модели; 3D Parallelism – комбинированный подход. Для обеспечения корректной работы параллельных стратегий используются коллективные операции, такие как AllReduce, AllGather и ReduceScatter [5]. Эффективность реализации этих операций критически зависит от выбора алгоритма передачи данных в условиях конкретной топологии сети. Например, алгоритмы Ring и Halving-Doubling и другие. Рисунок 1 демонстрирует пошаговое распространение данных между 4 узлами, где стрелками обозначены коммуникационные события, отслеживаемые симулятором.

Для создания среды моделирования был выбран метод дискретно-событийного моделирования (DES) [6]. Данный метод позволяет детально воспроизводить логику сетевых

взаимодействий и динамику коммуникационных событий, что является ключевым фактором для точной оценки временных затрат.

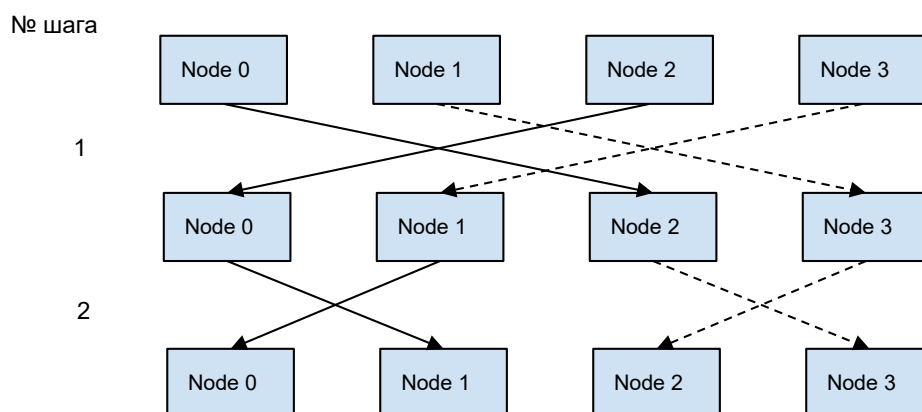


Рисунок 1 – Алгоритм Doubling для AllGather для 4 узлов

Разработанный программный комплекс реализован на языке Python и состоит из двух модулей. Первый модуль реализует ядро DES [6], а второй модуль расширяет его, добавляя логику специфических сетевых алгоритмов и протоколов обмена данными. В качестве входных данных пользователь предоставляет конфигурационные файлы (YAML или JSON), содержащие топологию сети, тип выбранного параллелизма (DP, TP, PP) и перечень используемых коллективных операций. В ходе выполнения программы заданные операции преобразуются в очередь событий, которые симулируются системой. На выходе формируется JSON-лог, содержащий детальную информацию о времени, затраченном на коммуникационные операции, что позволяет анализировать узкие места в системе.

В результате предложен и реализован инструмент, позволяющий быстро проводить вычислительные эксперименты с любыми заданными параметрами сети и алгоритмов без доступа к реальным GPU-ускорителям.

ЛИТЕРАТУРА

1. Top 20 Applications of Large Language Models in Real-Life – GeeksforGeeks [Электронный ресурс] – Режим доступа: <https://www.geeksforgeeks.org/artificial-intelligence/top-20-applications-of-large-language-models-in-real-life/>
2. Calculon [Электронный ресурс] – Режим доступа: <https://github.com/calculon-ai/calculon/tree/main>
3. Lu, G., Chen, R. (2023). DistSim: A performance model of large-scale hybrid distributed DNN training. arXiv preprint arXiv:2306.08423. [Электронный ресурс] – Режим доступа: <https://arxiv.org/abs/2306.08423>
4. Shoeybi, M., Patwary, M. [et al.]. (2019). Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv preprint arXiv:1909.08053. [Электронный ресурс] - Режим доступа: <https://arxiv.org/abs/1909.08053>
5. Collective Operations – NCCL 2.29.1 documentation [Электронный ресурс] – Режим доступа: <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html>
6. Лонишин М.Р., Сабуткевич А.М., Никифоров И. В. – Метод модели-ориентированной разработки на основе дискретно-событийного моделирования. // Современные технологии в теории и практике программирования: Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 284-285.

РЕАЛИЗАЦИЯ И ПРИМЕНЕНИЕ РАСШИРЕННОГО МЕТОДА TOPSIS ДЛЯ ГРУППОВОГО ПРИНЯТИЯ РЕШЕНИЙ НА ПРИМЕРЕ ВЫБОРА ОПТИМАЛЬНОГО ДАТАСЕТА ДЛЯ SQL-TO-TEXT

Актуальность работы объясняется необходимостью объективного агрегирования оценок экспертов при групповом принятии решения (ГПР) [1]. Ключевая проблема – разный уровень компетенции и знаний экспертов. В данной работе рассматривается расширенный метод TOPSIS [2] (eTOPSIS) для определения весов экспертов и ранжирования альтернатив. Целью данной работы является изучение, реализация на Python [3] и тестирование eTOPSIS для обоснованного определения весов экспертов в задачах ГПР.

Метод eTOPSIS отличается от классического тем, что в добавок к ранжированию альтернатив он упорядочивает и взвешивает приоритеты самих экспертов, а также имеет ряд отличий. В частности, отличается вычисление положительных (PIS) и отрицательных идеальных решений (NIS). PIS рассчитывается как усреднённая матрица группового решения (ГР), NIS делится на два отдельных: левое (L-NIS, матрица минимумов), и правое (R-NIS, матрица максимумов). Компетентность экспертов оценивается по близости их матрицы решения к PIS и дальности от обоих NIS. Архитектура и краткая последовательность работы реализованной программы представлена на рисунке 1.

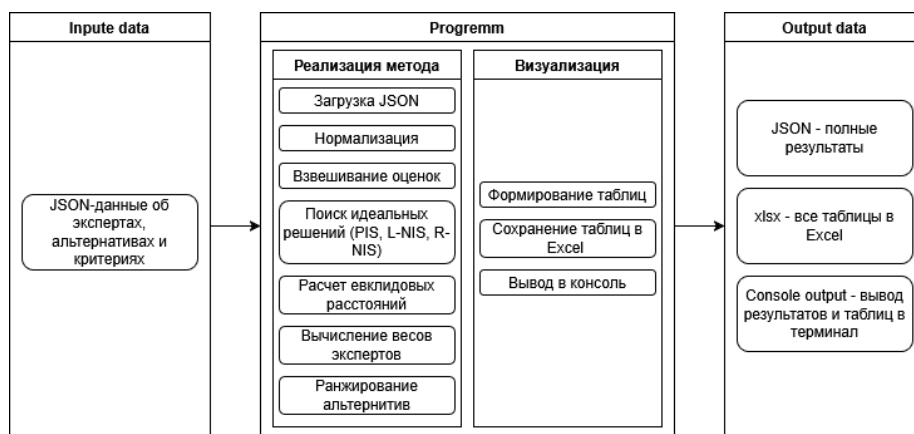


Рисунок 1 – Архитектура кода для метода eTOPSIS

Основной эксперимент посвящён решению прикладной задачи выбора датасета для создания русского аналога для решения задачи преобразование SQL-запросов в текстовое описание задачи на русском языке. Рассмотрены 6 популярных датасетов (Spider [4], WikiSQL [5], SParC [6], SEOSS-Queries [7], TableQA [8], Squall [9]), для оценивания которых построена трехуровневая иерархическая модель критериев. На верхнем уровне выделены три группы критериев по основным характеризующим аспектам: «Объем и разнообразие данных», «Качество и сложность данных», «Структура». Каждая группа критериев состоит из количественных показателей (таких как количество SQL-запросов, точность моделей, средняя длина запросов и др.). Все критерии являются описательными (количественными).

Задача предварительно решена в СППР «СВИРЬ» 10 методами оптимизации (Парето, лексикографический, лексиминный, аддитивные и мультипликативные свертки и др.), выступающими в роле «экспертов». Однако методы показали разрозненные результаты (размещены в [10] – «example3.json»): метод Парето показал низкую различающую способность (все методы в одном множестве Парето), аналогично лексиминный метод (3 ранга для 6 датасетов), а лексиграфический метод и методы скалярной оптимизации показали

противоположные результаты ранжирования. Для агрегации всех разнородных методов и получения взвешенного ГР был использован eTOPSIS.

Для согласования результатов экспертов ранги преобразованы в значения критериев, подлежащих максимизации. Метод eTOPSIS агрегировал мнения всех «экспертов» и определил их веса. Лучшие результаты показали наиболее сбалансированные методы (лучшим стала аддитивная свертка с расширением границ – вес 0,1), а худший результат – метод с жесткой расстановкой приоритетов (лексиграфическая оптимизация – 0,08).

В таблице 1 представлено итоговое ранжирование альтернатив: в колонке Candidates – наименования рассматриваемых датасетов, Total_Score – итоговая интегральная оценка, рассчитанная как взвешенная сумма оценок датасета по всем трём критериям верхнего уровня, Rank – итоговый ранг датасета, оцениваемый по убыванию значения Total_Score.

Таблица 1 – Итоговое ранжирование датасетов на основе описательных данных [4-9]

Candidate	Total_Score	Rank
Spider	0,527	1
SParC	0,418	2
TableQA	0,378	3
SEOSS-Queries	0,368	4
WikiSQL	0,315	5
Squall	0,289	6

Таким образом, eTOPSIS определил датасет Spider [4] как наилучший выбор среди альтернатив, что коррелирует с его широким применением в других работах, высокой детализацией схемы БД, объёмом данных, их качеством и сложностью. Поэтому авторы планируют создать русскоязычный аналог на его основе. Наихудшим признан датасет Squall [9]. Метод демонстрирует способность искать компромиссное решение на основе противоречивых оценок разных экспертов.

Результаты работы: открытая реализация [10] и тестирование eTOPSIS. Проверка разработанного алгоритма на тестовых данных статьи подтвердила корректность реализации, а практическое применение к проблеме выбора датасета показало его эффективность и полезность при работе с множеством альтернатив и экспертов. Разработанный прототип eTOPSIS [10] может быть использован для задач ГПР для разных предметных областей.

ЛИТЕРАТУРА

1. Introduction to Multiple Attribute Decision-making (MADM) Methods // Decision Making in the Manufacturing Environment. – London: Springer London, 2007. – P. 27–41. – DOI 10.1007/978-1-84628-819-7_3.
2. Yue Z. A method for group decision-making based on determining weights of decision makers using TOPSIS // Applied Mathematical Modelling. – 2011. – Vol. 35, no. 4. – P. 1926–1936. – DOI 10.1016/j.apm.2010.11.001.
3. Python Software Foundation. Python Language Reference, version 3.11. – URL: <https://www.python.org/> (дата обращения: 28.11.2025).
4. Yu T. et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. – 2018. – P. 3911–3921.
5. Zhong V., Xiong C., Socher R. Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning // arXiv preprint. – 2017. – arXiv:1709.00103.
6. Yu T. et al. SParC: Cross-Domain Semantic Parsing in Context // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. – 2019. – P. 4511–4523.
7. Tomova M., Hofmann M., Mader P. SEOSS-Queries - A Software Engineering Dataset for Text-to-SQL and Question Answering Tasks // Data in Brief. – 2022. – Vol. 42. – P. 108211.
8. Sun N., Yang X., Liu Y. TableQA: A Large-Scale Chinese Text-to-SQL Dataset for Table-Aware SQL Generation // arXiv preprint. – 2020. – arXiv:2006.06434.

9. Shi T. et al. On the Potential of Lexico-Logical Alignments for Semantic Parsing to SQL Queries // arXiv preprint. – 2020. – arXiv:2010.11246.
10. Noskova A.S., Sabinin O.Yu., Parkhomenko V.A. An eTOPSIS implementation. - URL: <https://github.com/Alexergy/eTOPSIS> (дата обращения: 29.12.2025).

УДК 519.816

В.Е. Петров, Г.А. Касьяненко, Р.Н. Никулина (2 курс магистратура),
В.А. Пархоменко, ст. преподаватель

МЕТОДЫ РЕШЕНИЯ ЗАДАЧ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ: ОТКРЫТАЯ КОМПЬЮТЕРНАЯ РЕАЛИЗАЦИЯ

Принятие решений (далее – ПрР) в условиях неопределенности является одной из насущных проблем в реальной жизни. В зависимости от полноты информации о факторах, влияющих на исход, выделяют задачи, решаемые в условиях риска (вероятностная неопределенность), полной и частичной неопределенности, далее – ВН, ПН и ЧН, соответственно [1]. Каждая задача применяет специализированный метод и критерии.

Целью данной работы является проектирование, реализация и тестирование открытой компьютерной системы, реализующей алгоритмы для поддержки ранжирования и выбора альтернатив в условиях неопределенности.

Классическая модель задачи ПрР в условиях неопределенности задается кортежем $M = \langle x, X, N, f, P \rangle$, где X – множество альтернатив, N – множество состояний внешней среды, $f(x_i, \lambda_j)$ – значение критерия, а P – информация о вероятностях состояний. В работе рассматривается три основных класса задач [1]: для ПН известно лишь множество возможных состояний среды, для ВН – распределение вероятностей на множестве состояний, а для ЧН – неполная информация о вероятностях, например, их качественное упорядочение.

Для ПН рассмотрены критерий Вальда (максимин), критерий максимакса, критерий Гурвица, критерий Сэвиджа (минимаксного сожаления), критерий Бернулли–Лапласа (принцип недостаточного основания), для ВН – вероятностное доминирование, мера риска, теория ожидаемой полезности, стохастическое доминирование, а для ЧН – метод Фишберна, метод Кирквуда и Сарина, обобщенный метод Кирквуда и Сарина.

Архитектура разработанной системы приведена на рисунке 1, где пользователю предоставляется возможность передать входные данные в виде однокритериальной или многокритериальной матрицы, выбрать метод ранжирования и выбрать режим работы. Тестирование осуществлено на задачах из учебника [1]. Система имеет два режима работы:

- Вычисления – прием на вход файл с матрицей (pxm) и название метода, возвращает ранжированный список альтернатив. В случае многокритериальной задачи необходимо записать функцию полезностей или выбрать ее из существующих.
- Тестирование – проверка корректности алгоритмов на контрольных данных.

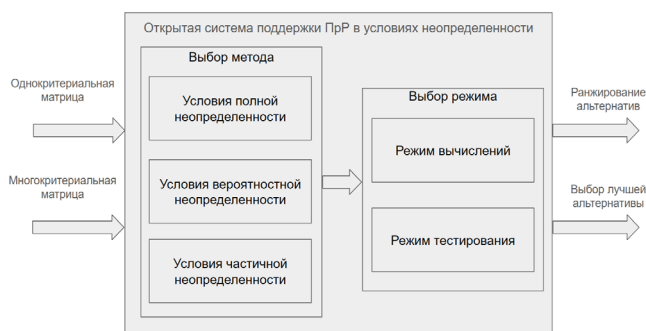


Рисунок 1 – Архитектура разработанного решения

Для демонстрации решения практико ориентированной задачи используется задача выбора лучшей акции на основе критериев, выражающихся в виде стратегий инвестиции в

условиях ПН и ВН [3]. Исходные данные задачи представлены в таблице 1 из [3]. Для выбора лучшей альтернативы в условиях ПН применим принципы, используемые в статье: Гурвица (f_μ), Вальда (f^W), Сэвиджа (f^S) и максимакса (f^O). Для применения принципа Гурвица требуется коэффициент пессимизма, в статье применяется коэффициент оптимизма. На этом основании примем следующее допущение – коэффициент $\mu = 0.3$. С учетом допущения полученный результат представлен в таблице 1 и файле *economarket.json* [2]. Лучшие альтернативы обозначены символом *, где W - для принципа Вальда, S - Сэвиджа, μ - Гурвица, O - максимина, соответственно.

Таблица 1 – Выбор лучшей альтернативы в условиях ПН

	f^W	f^O	f_μ	f^S	W	O	μ	S
x_1	3	12	9.3	4				*
x_2	-2	15	9.9	9		*	*	
x_3	7	7	7	8	*			

Сравнение результатов, приведенных в таблице 1 с результатами таблиц 2, 3, 4 и 5 из [3] демонстрирует полное соответствие, с учетом допущения.

Используем те же данные, но для условий ВН. Экспертная оценка дается при подборе наиболее близкого по эквивалентности решению к решению в таблице 9 статьи [3]. В статье применяется критерий Лапласа, модификация которого имеется в системе в виде критерия Бернулли-Лапласа. Однако применение данного критерия выполняется в условиях ЧН, что и будет являться допущением. При расчете критерия необходимы координаты вершин многогранника p_q , рассчитываемые по формуле: $p_1 = (1/k_1, 1/k_1, \dots, 1/k_1, 0, \dots, 0)$, $p_2 = (1/k_2, 1/k_2, \dots, 1/k_2, 0, \dots, 0)$, $p_q = (1/n, 1/n, \dots, 1/n)$ [1]. Примем $q = 1, k_1 = n$. Результат вычислений представлен в таблице 2 и файле *economarket_bernlap.json* [2]. В скобках таблицы 2 указаны координаты вершин многогранника p_q .

Таблица 2 – Выбор лучшей альтернативы в условиях ЧН

	K_1	K_2	K_4	K_4	Result
x_1	12(0.25)	8(0.25)	7(0.25)	3(0.25)	7.5*
x_2	15(0.25)	9(0.25)	5(0.25)	-2(0.25)	6.75
x_3	7(0.25)	7(0.25)	7(0.25)	7(0.25)	7.0

Несмотря на разные условия неопределенности, сравнение результатов, приведенных в таблице 2 с результатами таблицы 9 статьи [3] демонстрирует полное соответствие, с учетом допущения применяемого метода.

Таким образом, спроектировано и реализовано открытое консольное приложение [2] для ранжирования альтернатив и выбора лучшей альтернативы в разных условиях неопределенности, а также решена задача выбора стратегии инвестиции при неопределенности [3].

ЛИТЕРАТУРА

1. Подиновский, В. В. Многокритериальные задачи принятия решений: теория и методы анализа : учебник для вузов / В. В. Подиновский. — Москва : Издательство Юрайт, 2023. — 486 с.
2. Petrov V.E., Kasyanenko G.A, Nikulina R.N., Parkhomenko V.A. Methods of decision-making under conditions of uncertainty (complete, partial, stochastic) // GitHub – URL: <https://github.com/SciensialIdeas/DM-Stoch>
3. Taghavifard, M. T. Decision making under uncertain and risky situations / M. T. Taghavifard, K. Khalili Damghani, R. Tavakkoli Moghaddam // Society of Actuaries. – 2009. – P. 1-31.

РАЗРАБОТКА СЕРВИСА ДЛЯ КРИПТОГРАФИЧЕСКОЙ ПОДПИСИ PDF-ДОКУМЕНТОВ

В прикладных сценариях электронного документооборота (ЭДО) часто требуется подписать документ таким образом, чтобы сохранялась возможность криптографической проверки целостности и обеспечивались следующие факторы: управляемая выдача доступа к документу, подтверждение действия подписантом, воспроизводимое развертывание, возможность размещения в закрытой архитектуре.

Подписание документов вручную плохо масштабируется, учитывая их колоссальное количество (например, ордера или накладные), отсутствует единый контролируемый процесс, аудит, присутствует риск утечек. Существующие сервисы подписания электронных документов завязаны на облачных услугах или сложно интегрируются в закрытую архитектуру. Поэтому разработка сервиса для быстрого подписания PDF-документов актуальна.

Целью работы является разработка полноценного сервиса, обеспечивающего следующий пользовательский сценарий электронного подписания PDF: загрузка документа через API для последующего встраивания в системы ЭДО (1С, Битрикс24), формирование временных ссылок на просмотр, скачивание и подписание документов, подтверждение операции одноразовым кодом, получение подписанного документа, а также верификация подписанного документа. Дополнительно требуется обеспечить развертывание с применением контейнеризации, CI/CD и GitOps.

В рамках работы разработана микросервисная архитектура, включающая следующие сервисы:

1. Uploader, реализован на языке Go. Выполняет загрузку одного PDF-документа через браузер по протоколу tus [1], сохранение файла в S3 MiniO, генерацию токена для подписания, просмотра и загрузки, и добавление задачи на подпись в очередь сообщений.

2. Downloader (реализация на Go). Реализует просмотр и скачивание по токену подписанного файла.

3. Signer (реализация на Go). Управляет сессией подписания: создание OTP, проверка кода с ограничением числа попыток, интеграция с хранилищем, сохранение статуса в базу данных, верификация подписанного документа.

4. PdfSigner (реализация на Kotlin/Spring Boot). Отдельный сервис, отвечающий только за обработку PDF и криптографию. Добавляет на страницу визуальный штамп и формирует криптографическую подпись для PDF с использованием Apache PDFBox [2] и BouncyCastle. Выделение сервиса отдельно от Signer обусловлено тем, что для Kotlin есть мощные библиотеки для работы с PDF и криптографии.

5. Mailer (реализация на Go). Расширяемый сервис для отправки сообщений по электронной почте.

Инфраструктура также включает следующие компоненты: Redis для токенов и метаданных ссылок, очередь сообщений RabbitMQ, базу данных PostgreSQL. Развертывание выполнено через Kubernetes [3], доставка изменений через ArgoCD [4]. Контейнерные образы публикуются в GHCR через GitHub Actions.

Реализованная на данный момент версия полностью охватывает пользовательский сценарий: загрузка по API, отправка ссылки и одноразового кода по электронной почте или другими способами. Сервис создан с расчетом на расширение под конкретную задачу, подписание документа с введением одноразового кода и скачивание подписанного файла, который позже можно проверить. Временные ссылки и метаданные ограничены по времени в Redis, состояние подписания и результат фиксируется в базе данных. Подписанный документ хранится в Minio как отдельный объект. Сервис отказоустойчив, выводение из строя любого

компонента не приводит к необратимым изменениям, при перезапуске компонента работа продолжается в прежнем режиме. Корректность подписи проверяется по следующим параметрам: наличие подписи, ее валидность, отсутствие изменений в документе, подписание именно данным сервисом. Подпись может быть верифицирована не только через веб-интерфейс, но и с помощью API, что позволяет автоматизировать проверку.

ЛИТЕРАТУРА

1. tus. Resumable Upload Protocol 1.0.x [Электронный ресурс]. Режим доступа: <https://tus.io/protocols/resumable-upload> (дата обращения 15.03.2026)
2. Apache PDFBox. PDSignature (digital signatures): Javadoc / Apache PDFBox [Электронный ресурс]. Режим доступа: <https://pdfbox.apache.org/docs/2.0.5/javadocs/org/apache/pdfbox/pdmodel/interactive/digitalsignature/PDSignature.html> (дата обращения 15.03.2026)
3. Kubernetes. Ingress: concepts/services-networking [Электронный ресурс]. Режим доступа: <https://kubernetes.io/docs/concepts/services-networking/ingress/> (дата обращения 15.03.2026)
4. ARGO CD. ArgoCD Documentation (Declarative GitOps CD for Kubernetes) [Электронный ресурс]. Режим доступа: <https://argo-cd.readthedocs.io/en/stable/> (дата обращения 15.03.2026)

УДК 004.057.4

А.А. Смекалов (4 курс бакалавриата),
И.В. Шошмина, к.т.н., доцент

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕХАНИЗМОВ ЗАЩИТЫ ОТ MEV-АТАК НА ДЕЦЕНТРАЛИЗОВАННЫЕ БИРЖИ

Развитие децентрализованных финансов сопровождается ростом числа атак, направленных на эксплуатацию прозрачности блокчейн-транзакций. Одной из наиболее распространённых угроз являются MEV-атаки (maximal extractable value), при которых злоумышленник извлекает прибыль, манипулируя порядком обработки транзакций [1]. Особую опасность представляют sandwich-атаки, в ходе которых транзакции жертвы окружаются атакующими заявками с целью искусственного изменения цены актива [1]. Высокопроизводительный блокчейн Solana, несмотря на отсутствие публичного мемпула, не лишён данной уязвимости: валидаторы сохраняют влияние на порядок исполнения транзакций, а параметры торговых заявок остаются открытыми вплоть до момента их исполнения. Отсутствие комплексного сравнительного анализа существующих механизмов защиты в контексте архитектурных особенностей Solana определяет актуальность настоящего исследования.

В рамках исследования проводится сравнительный анализ подходов к защите от MEV-атак: приватной маршрутизации транзакций, пакетных аукционов, схемы commit-reveal, доказательств с нулевым разглашением и протоколов безопасных многосторонних вычислений. Сравнение осуществляется по следующим критериям: вычислительная стоимость транзакции (количество вычислительных единиц), размер транзакции в байтах, комиссия за исполнение, задержка подтверждения и количество транзакций, необходимых для совершения одной обменной операции. По совокупности критериев наиболее сбалансированным решением являются протоколы безопасных многосторонних вычислений. На основании результатов анализа разработана децентрализованная биржа на базе Solana с интеграцией MPC-протокола, в которой параметры торговой операции шифруются на стороне клиента и обрабатываются в распределённой сети узлов без возможности восстановления исходных данных ни одним из участников, включая валидаторов [2, 3]. Расшифровка происходит исключительно в момент исполнения смарт-контракта, что исключает возможность построения атакующих транзакций на этапе нахождения заявки в очереди [4]. Для экспериментальной оценки были проведены серии swap-операций через существующие децентрализованные биржи сети Solana с последующим анализом параметров транзакций.

Проведённый сравнительный анализ демонстрирует, что протокол безопасных многосторонних вычислений обеспечивает наиболее полную защиту от MEV-атак при приемлемых вычислительных затратах и минимальной зависимости от доверенных третьих сторон. Разработанная на его основе децентрализованная биржа позволит существенно снизить финансовые потери пользователей от sandwich-атак и повысить доверие к децентрализованным финансовым сервисам на блокчейне Solana. Перспективы исследования включают развитие функционала платформы и ее оптимизацию для высоконагруженных торговых сценариев.

ЛИТЕРАТУРА

1. Gramlich V., Jelito D., Sedlmeir J. Maximal extractable value: Current understanding, categorization, and open research questions // Electronic Markets. — 2024.
2. Lindell Y. Secure Multiparty Computation // Computations of the ACM. — 2021
3. Arcium Protocol Documentation. Confidential Computing Infrastructure [Электронный ресурс]. — 2024. — URL: <https://docs.arcium.com/> (Дата обращения: 10.01.2026).
4. Yakovenko A. Solana: A new architecture for a high performance blockchain [Электронный ресурс]. — 2018. — URL: <https://solana.com/solana-whitepaper.pdf> (Дата обращения: 08.01.2026).
5. Heimbach L., Wattenhofer R. Eliminating Sandwich Attacks with the Help of Game Theory // Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. — New York, 2022. — P. 153–167.

УДК 004.021

Р.А. Сафуанов (4 курс бакалавриата),
В.Э. Шмаков, к.т.н., доцент

ПОСТРОИТЕЛЬ ГРАФА ТРАНСПОРТНОЙ СЕТИ НА ОСНОВЕ МОДЕЛИ ТЕРО

Современные транспортные системы крупных городов сталкиваются с проблемой эффективного проектирования сети маршрутов, способной обслуживать растущие пассажиропотоки при ограниченных бюджетных ресурсах. Существующие подходы [1] к оптимизации транспортных сетей часто ориентированы на улучшение уже существующей инфраструктуры и не позволяют строить сеть «с нуля» с учётом реальных потребностей пассажиров. Бионические алгоритмы, в частности модель А. Теро [2], описывающая поведение слизевика *Physarum polycephalum*, демонстрируют способность к формированию эффективных сетевых структур, однако оригинальная модель обладает рядом ограничений, препятствующих её прямому применению в транспортном планировании.

В данном проекте рассматривается решение проблемы построения транспортной сети с использованием бионической модели поведения слизевика. Главной целью является разработка алгоритма для построения транспортных сетей, имитирующего принципы самоорганизации биологических систем, адаптированного для решения прикладных задач в условиях реальных пассажиропотоков и ограниченных ресурсов и соответствующих метрик [3]. Для использования алгоритма необходимо разработать соответствующее ПО на языке программирования Python [4].

В основе работы лежит математическая модель А. Теро, где поток Q_{ij} в ребре определяется через проводимость D_{ij} и разность давлений в вершинах, а эволюция проводимости зависит от величины потока. В предлагаемом алгоритме введены следующие нововведения:

– Множественные источники и стоки: каждая вершина является одновременно источником и стоком для других вершин.

– Бюджетное ограничение: суммарная «стоимость» сети ограничена, поэтому сумма произведений длин рёбер на их проводимости не может превосходить заданный бюджет.

– Гравитационная модель спроса [5]: спрос на пути из одной вершины в другую определяется весами вершин и их отношением друг к другу.

В программном средстве будет реализован следующий пользовательский функционал:

– Задание исходных данных – пользователь может указать координаты вершин, их веса (например, численность населения) и ограничение бюджета.

– Настройка параметров оптимизации – регулируются количество итераций, шаг обучения α , порог отображения рёбер.

– Визуализация графа в реальном времени – отображение сети с цветовой индикацией проводимости и стрелками, показывающими направление доминирующего потока.

– Мониторинг ключевых метрик – на графиках отображается эволюция шести показателей: суммарный поток, удовлетворение спроса, эффективность бюджета, коэффициент Джини, количество активных рёбер, асимметрия потоков.

– Сравнение начальной и финальной сети – автоматическое формирование таблицы с численными значениями метрик и их изменением в процессе оптимизации.

– Экспорт результатов – сохранение визуализации сети и её метрик во внешние файлы.

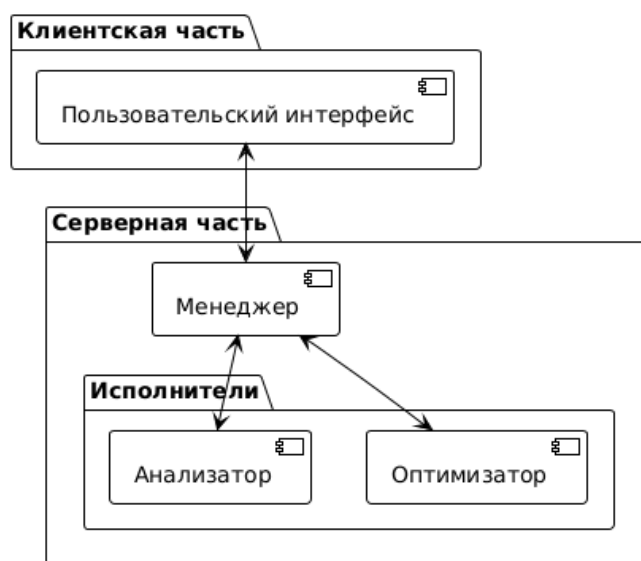


Рисунок 1 – Архитектура системы

Архитектура программного средства, представленная на рисунке 1, включает в себя три модуля:

– Оптимизатор – алгоритм, который по набору вершин с их координатами и весами строит оптимальную сеть под заданный бюджет.

– Анализатор – рассчитывает различные метрики для оценки полученной сети.

– Пользовательский интерфейс – обеспечивает удобное взаимодействие пользователя с моделью.

ЛИТЕРАТУРА

1. Oliveira C.A.S., Pardalos P.M. (eds.) Handbook of artificial intelligence and data sciences for routing problems. – Cham: Springer, 2025. – 257 p.
2. Tero A., Kobayashi R., Nakagaki T. A mathematical model for adaptive transport network in path finding by true slime mold // Journal of Theoretical Biology. – 2007. – Vol. 244, No. 4. – P. 553–564.
3. Newman M.E.J. Networks: An Introduction. – 2nd ed. – Oxford: Oxford University Press, 2018. – 800 p.
4. Steinkamp V. Python For Engineering and Scientific Computing: Practical Applications with NumPy, SciPy, Matplotlib, and More. – Birmingham: Packt Publishing, 2025. – 515 p.
5. Wilson A.G. A statistical theory of spatial distribution models // Transportation Research. – 1967. – Vol. 1, No. 3. – P. 253–269.

АВТОМАТИЗАЦИЯ НАСТРОЙКИ И ВЕРИФИКАЦИИ ТРЕБОВАНИЙ БЕЗОПАСНОСТИ ДЛЯ ОПЕРАЦИОННЫХ СИСТЕМ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Обеспечение информационной безопасности корпоративных информационных систем является одной из ключевых задач современной ИТ-инфраструктуры. При этом процесс настройки и проверки требований безопасности для операционных систем и программного обеспечения часто выполняется вручную и сопровождается рядом проблем:

- низкая степень автоматизации – многие процедуры настройки и проверки выполняются специалистами вручную, что требует значительных временных затрат;
- человеческий фактор – ручная конфигурация и аудит могут приводить к ошибкам и несоответствию установленным требованиям безопасности;
- сложность масштабирования – при увеличении количества серверов и сервисов возрастает трудоемкость обеспечения единых стандартов безопасности;
- трудности регулярной верификации – постоянный контроль соответствия систем требованиям безопасности требует проведения повторяющихся аудитов.

Современные технологии автоматизации инфраструктуры и управления конфигурациями позволяют существенно повысить эффективность обеспечения безопасности информационных систем. Использование средств автоматизации, контейнеризации и централизованного управления конфигурациями позволяет реализовать непрерывный процесс проверки соответствия систем установленным требованиям безопасности. Разработка специализированного программного комплекса для автоматизации настройки и верификации требований безопасности позволит повысить надежность и воспроизводимость процедур аудита, а также снизить нагрузку на специалистов осуществляющих настройку требований безопасности.

Цель работы – повышение эффективности настройки и верификации требований безопасности для операционных систем и программного обеспечения за счет метода сквозного аудита, реализованного в автоматизированном программном комплексе.

Для достижения поставленной цели были сформулированы следующие задачи.

1. Рассмотреть существующие подходы и инструменты автоматизации настройки и аудита требований безопасности для операционных систем и программного обеспечения.
2. Проанализировать проблемы, возникающие при ручной настройке и проверке требований безопасности в корпоративных информационных системах.
3. Изучить требования к системам автоматизации аудита безопасности, определить ключевые функциональные возможности, необходимые для внедрения в корпоративную инфраструктуру.
4. Проанализировать современные технологии автоматизации управления конфигурациями и оркестрации инфраструктуры, определить их применимость для решения поставленной задачи.
5. Разработать программный комплекс, обеспечивающий автоматизированную настройку и проверку требований безопасности операционных систем и программного обеспечения.
6. Провести тестирование демонстрационной версии системы и оценить эффективность предложенного решения.

В работе разрабатывается программный комплекс автоматизации аудита требований безопасности. Основу решения составляет стек технологий Python, Ansible, Docker, а также веб-интерфейс, реализованный на базе React и Flask.

Основные функциональные возможности разрабатываемого комплекса представлены ниже.

1. Автоматизированная настройка параметров безопасности – применение сценариев конфигурации для приведения операционных систем и программного обеспечения в соответствии с требованиями безопасности.

2. Автоматизированная верификация требований безопасности – проведение аудита систем на соответствие установленным политикам безопасности.

3. Контейнеризация компонентов системы – использование контейнерной инфраструктуры для упрощения развертывания и масштабирования решения.

4. Централизованный веб-интерфейс управления – предоставление пользователю удобного интерфейса для запуска проверок, просмотра результатов аудита и управления конфигурациями.

5. Формирование структурированных отчетов – результаты проверок автоматически сохраняются и могут использоваться для анализа состояния безопасности инфраструктуры.

Взаимодействие компонентов разрабатываемой системы можно представить в виде архитектурной схемы (рисунок 1), отражающей взаимодействие модулей автоматизации, системы управления конфигурациями и пользовательского веб-интерфейса.



Архитектура системы автоматизации аудита безопасности

Рисунок 1 – Архитектура системы автоматизации аудита безопасности

На текущий момент разработана демонстрационная версия системы, позволяющая выполнять автоматизированную проверку параметров безопасности операционных систем и управлять процессом аудита через веб-интерфейс. В дальнейшем планируется расширение набора проверок, повышение уровня автоматизации настройки и интеграция системы с существующими инструментами управления инфраструктурой.

ЛИТЕРАТУРА

1. Syed A. A. M. Automating Active Directory Management with Ansible: Case Studies and Efficiency Analysis // Journal of Recent Trends in Computer Science and Engineering. – 2022.
2. Saavedra N., Ferreira J. GLITCH: Automated Polyglot Security Smell Detection in Infrastructure as Code // arXiv. – 2022.
3. Sesto V. Practical Ansible: Configuration Management from Start to Finish. – Springer, 2022.
4. Choi B., Medina E. Introduction to Ansible Network Automation: A Practical Primer. – Apress, 2023.
5. Red Hat. Ansible Automation Platform – Security Automation Guide. – 2024.
6. Сафронов, Д. Автоматическая балансировка нагрузки между потоковой обработкой данных и внутренними задачами кластера с использованием Kubernetes / Д. Сафронов, К. М. Стоноженко, И. В. Никифоров // Современные технологии в теории и практике программирования : сборник материалов конференции, Санкт-Петербург, 23 апреля 2020 года / Санкт-Петербургский политехнический университет Петра Великого; Dell Technologies; EPAM Systems. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2020. – С. 165-167. – EDN VOXGIM.
7. Ивлев, В. А. Актуальность автоматизированной настройки инфраструктуры ит-проекта / В. А. Ивлев, И. В. Никифоров // Современные технологии в теории и практике программирования :

УДК 004.3

К.И. Трубач (4 курс бакалавриата),
А.А. Иванюк, д.т.н., профессор

СИНТЕЗ СПЕЦИАЛИЗИРОВАННОГО МИКРОПРОГРАММНОГО ЯДРА ДЛЯ УПРАВЛЕНИЯ ФИЗИЧЕСКИ НЕКЛОНИРУЕМЫМИ ФУНКЦИЯМИ

В современных сложных системах на кристалле (СнК) выполнение специфических задач зачастую требует гибкости, которую не обеспечивают традиционные аппаратные решения. В таких случаях целесообразно использование процессорных ядер [1, 2]. Однако применение полнофункциональных ядер с избыточным набором инструкций и полным набором регистров для подобных узкоспециализированных задач часто оказывается избыточным, а проектирование и верификация оригинальных платформ под каждую задачу сопряжены со значительными временными затратами.

Целью настоящей работы является формализация процесса извлечения набора ограничений (constraints) из исходных алгоритмов для последующего высокоуровневого синтеза их минимальной аппаратной реализации в рамках тракта данных (datapath) микропроцессорного ядра. Предлагаемый подход позволяет оптимизировать затраты на разработку и верификацию, сократить площадь кристалла в случае ASIC-решений или минимизировать использование ресурсов ПЛИС при софт-процессорной реализации. При этом базисом для применения извлеченных правил может выступать как ядро существующей архитектуры, так и обобщенный шаблон, структура которого адаптируется путем добавления или удаления элементов тракта данных в соответствии с выявленными ограничениями.

Первый этап извлечения таких правил — задание алгоритма любым доступным методом (в виде текстовых описаний, граф-схем, программного или машинного кода) и его отладка. Наиболее формализованные подходы (программный код) предпочтительны в силу возможности автоматизации ранних стадий проектирования.

Следующим этапом является компиляция — преобразование алгоритма в исполняемый машинный код. Для специализированных архитектур на этой стадии проводится эвристический анализ алгоритма с целью формирования оптимального набора инструкций «с нуля». Если же используется существующая базовая архитектура, компиляция сводится к выделению необходимого подмножества команд и ресурсов. В этом случае этап компиляции закладывает основу для оптимизации аппаратных затрат: поддержка неиспользуемых инструкций может быть исключена из аппаратной реализации без потери функциональности.

Результатом стадии компиляции является машинный код, выступающий предметом статического и динамического анализа. Статический анализ заключается в количественной оценке используемых ресурсов архитектуры: инструкций, регистров, оперативной памяти, операций над данными и т.д. Полученные данные позволяют существенно сократить перечень необходимых аппаратных компонентов, а также выявить повторяющиеся последовательности инструкций, которые могут быть оптимизированы или объединены.

Итогом анализа становится совокупность правил (ограничений), описывающих минимальный набор аппаратных ресурсов, строго необходимых для решения конкретной задачи. На заключительном этапе на основе этих правил генерируется синтезируемое описание (HDL-модель). Рассмотрим практический пример, иллюстрирующий потенциал сокращения аппаратных затрат при использовании статического анализа.

Одним из применений микропроцессорных ядер (в т.ч. и софт-процессорных) является управление физически неклонируемыми функциями (ФНФ) [3], а также расчёт их

характеристик. На рисунке 1 представлен алгоритм, заданный на псевдоязыке, для расчёта характеристики стабильности (STA) для ФНФ.

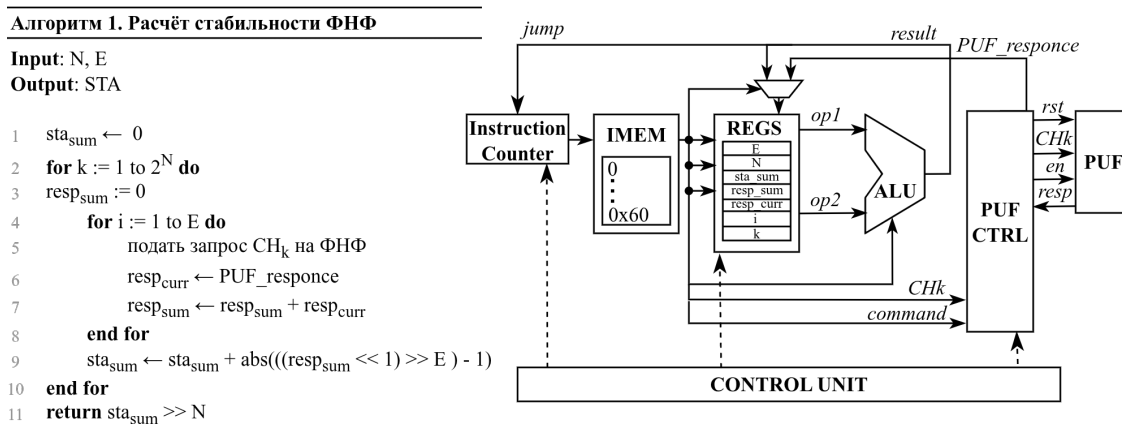


Рисунок 1 – Псевдокод для алгоритма расчёта стабильности ФНФ, а также соответствующая алгоритму аппаратура

Статический анализ данного кода показывает, что для его реализации необходима поддержка ограниченного набора операций: арифметических с константой и с регистрами («+», «-»), сдвигов на единицу и на значение регистра («<<», «>>»), модуля (abs), а также операций сравнения между регистрами («<», «>»). Для работы с циклами вводятся операции инкремента и декремента («+1», «-1»), механизмы ветвления (jump). Работа с ФНФ требует дополнительных инструкций, которые трансформируются в команды (command) для блока управления ФНФ (PUF CTRL): отправка запроса и получение ответа. Запрос (CHk) кодируется в инструкции. В алгоритме задействуется семь регистров. Меньшее количество приводит к неоптимальности кода, а большее — к избыточности ресурсов. Специфика предметной области позволяет заменить вычисления с плавающей запятой на арифметику с фиксированной запятой, а операции деления — сдвигами вправо. Разрядность регистрового файла устанавливается в зависимости от диапазонов значений регистров N и E. Сформированный по итогам такого анализа перечень параметров служит ограничениями для генерации специализированной аппаратуры. Для демонстрации эффективности метода при сокращении уже существующих архитектур приведем статистику использования ресурсов стандартного процессора архитектуры RISC-V при выполнении данной программы: алгоритм задействует лишь 31% доступных типов инструкций, 21% регистров и 25% разрядности счетчика команд (PC). Исключение избыточных ресурсов позволяет достичь значительного сокращения площади кристалла и уменьшения критического пути.

Таким образом, предложенный в работе метод устанавливает прямую связь между требованиями исходного алгоритма и конечной аппаратной реализацией. Формализация процесса извлечения аппаратных ограничений закладывает основу для создания полностью автоматизированных систем проектирования для генерирования минимально необходимого и высокоэффективного вычислительного ядра непосредственно из программного кода.

ЛИТЕРАТУРА

1. Inside the iPhone 5s [Электронный ресурс] — ChipWorks — Режим доступа: <https://web.archive.org/web/20140803201521/http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-iphone-5s/> — Дата доступа: 21.09.2025.
2. The Samsung Exynos 7420 Deep Dive - Inside A Modern 14nm SoC [Электронный ресурс] — AnandTech — Режим доступа: <https://web.archive.org/web/20150630074026/http://www.anandtech.com/show/9330/exynos-7420-deep-dive> — Дата доступа: 21.09.2025.
3. Иванюк А.А. Исследование физически неклонированной функции конфигурируемого кольцевого осциллятора. Информатика. 2025, т. 22, № 1, с. 73–89. DOI: <https://doi.org/10.37661/1816-0301-2025-22-1-73-89>.

УДК 004.42

К.П. Алымова, И.М. Савинцев (2 курс бакалавриата),
В.А. Касилов, ассистент

АВТОМАТИЗАЦИЯ СОЗДАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ СРЕДСТВ АНАЛИЗА КОДА С ПОМОЩЬЮ CLANG-TIDY И CLANG-QUERY

В современном мире вместе с ростом сложности программного обеспечения растут и требования к его качеству. Одним из способов повышения качества является использование статических анализаторов исходных кодов программы. Многие из статических анализаторов являются решениями с открытым исходным кодом, причем некоторые позволяют расширить встроенную функциональность. Это позволяет разработчику адаптировать готовые решения под уникальные требования своего проекта [1]. К таким расширяемым решениям относятся, например: Vera++ (C++) [3], ESLint (JavaScript) [4] и др.

Ручная проверка кода требует больших временных затрат на каждом этапе разработки и страдает от человеческого фактора. Разработка собственных расширений для готовых инструментов выполняется единоразово, более надежна, но все еще трудоемка. Поэтому целью данной работы является упрощение создания расширений для статического анализа кода с помощью автоматизации.

Компиляторы LLVM/clang предоставляют наиболее популярные инструменты для форматирования и статического анализа кода на языках программирования семейства C и, в частности, C++. Один из таких инструментов – clang-tidy [7], который проводит статический анализ с помощью обхода абстрактного синтаксического дерева (AST) исходного кода [6], сопоставляя его узлы с набором предикатов (сопоставителей). Для каждого найденного узла выводится предупреждение или реализуется автоматическое исправление (FixIt). Другой инструмент инфраструктуры LLVM/clang – clang-query [8] позволяет интерактивно тестировать сопоставители – отображает узлы, соответствующие заданному сопоставителю. Пример использования clang-query представлен на рисунке 1.

<pre>//main.cpp int add(int a, int b) { return a + b; } int sub(int a, int b) { return a - b; }</pre>	<pre>//terminal clang-query main.cpp clang-query> match functionDecl() //output Match #1: int add(int a, int b) Match #2: int sub(int a, int b)</pre>

Рисунок 1 – Пример использования clang-query

Clang-tidy имеет набор встроенных проверок (чекеров), каждый из которых состоит из сопоставителя и диагностики (предупреждение или FixIt). Также clang-tidy позволяет создавать пользовательские чекеры. Интерфейс clang-query подходит для автоматизации этого процесса. Предлагается объединить тестирование сопоставителей с помощью clang-query и генерацию на их основе чекеров. Пользователь получит возможность определить не только предупреждающее сообщение, но и FixIt блок для сгенерированного чекера. Это отличает наше решение от Query Based Custom Clang-Tidy Checks [6] – существующего в LLVM/clang инструмента для создания собственных проверок для clang-tidy.

Описанный подход реализуется на основе дополнительных команд для clang-query. Схема их использования представлена на рисунке 2. Команда для генерации проверки позволяет обработать сопоставитель, его имя и предупреждающее сообщение, вводимые пользователем [2], создать на их основе чекер. Команда для генерации автоматического исправления позволяет пользователю дополнить созданный чекер блоком FixIt, поддерживает решения, подходящие для большинства проверок. Результат обеих команд представляет из себя файл с кодом сгенерированного чекера на C++.

<code>//terminal</code>	<code>//terminal</code>
<code>clang-query create-checker</code>	<code>clang-query add-fixit</code>
<code>clang-query> name</code>	<code>clang-query> name</code>
<code>clang-query> matcher</code>	<code>clang-query> action (insert/erase/replace)</code>

Рисунок 2 – Создание чекера checker-name и добавление FixIt к чекеру checker-name

Стоит отметить, что такой подход не избавит разработчика полностью от необходимости вручную писать сложные проверки и редактировать сгенерированные, но автоматизирует создание простых.

Подход был реализован на языке C++, с использованием инструментов языка Python. Для его реализации были выполнены задачи, перечисленные ниже:

1. Реализация автоматической генерации чекеров на основе запросов clang-query.
2. Реализация автоматической генерации FixIt блоков на основе запросов clang-query.

3. Внедрение подхода в систему проверки студенческих работ по программированию на C++. Были реализованы чекеры, проверяющие стиль кода. Например, использование глаголов в именах функций и расположение секций доступа в классе в определенном порядке.

Команды осуществляют генерацию чекеров на основе запросов clang-query, поддерживают создание FixIt блоков, предоставляют отображение найденных для проверки или исправления локаций. Совместное использование реализованных команд, clang-tidy и clang-query позволило упростить настройку системы анализа кода. Самостоятельное написание чекера, требующее ручного создания всех компонентов, сократилась до использования одной команды в терминале – от часа разработки до десяти секунд автоматической генерации. Использование статического анализатора с расширениями, созданными предложенным методом, сократило время поиска ошибок более чем на 50%.

ЛИТЕРАТУРА

1. Голубев Д. Е., Шагалова П. А. Разработка программного решения для автоматического рефакторинга кода // Современные технологии в теории и практике программирования: сборник материалов конференции, Санкт-Петербург, 24-25 апреля 2024 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2024. – С. 271-272.
2. Полубелова М. И., Григорьев С. В. Лексический анализ динамически формируемых строковых выражений // Инструменты и методы анализа программ - 2015: материалы Международной научно-практической конференции, 12-14 ноября 2015 года. = Tools & Methods of Program Analysis (TMPA – 2015). – С. 27-37.
3. Vera++ [Электронный ресурс] Режим доступа: <https://github.com/verateam/vera>
4. ESLint [Электронный ресурс] Режим доступа: <https://eslint.org/>

5. Query Based Custom Clang-Tidy Checks. [Электронный ресурс] Режим доступа: <https://clang.llvm.org/extra/clang-tidy/QueryBasedCustomChecks.html>
6. Introduction to the Clang AST. [Электронный ресурс] Режим доступа: <https://clang.llvm.org/docs/IntroductionToTheClangAST.html>
7. Extra Clang Tools documentation clang-tidy. [Электронный ресурс] Режим доступа: <https://clang.llvm.org/extra/clang-tidy/>
8. AST Matcher Reference. [Электронный ресурс] Режим доступа: <https://clang.llvm.org/docs/LibASTMatchersReference.html>

УДК-004.4

А.В. Анисимова (4 курс бакалавриата),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ РАЗБОРА И АНАЛИЗА АРХИТЕКТУРНЫХ ДИАГРАММ ФОРМАТА DRAW.IO

В наши дни разработка программных систем требует создания и поддержки архитектурной документации. Необходимость ведения архитектурной документации обусловлена рядом факторов:

- современное программное обеспечение постоянно модернизируется и развивается для адаптации к растущим требованиям пользователей [1];
- программные системы пересматриваются и дорабатываются чаще, чем другие типы систем;
- как правило, сопровождение кода делегируется менее опытным младшим сотрудникам, которые не знакомы с кодом [2].

Одним из наиболее распространённых инструментов для создания диаграмм программной архитектуры является редактор Diagrams.net (Draw.io) [3], позволяющий формировать различные типы графических моделей, включая архитектурные диаграммы в нотации С4 [4]. При сохранении диаграмм используются форматы .xml [5] и .drawio, представляющие собой XML-документы, содержащие описание графических объектов, их атрибутов, связей между элементами и пространственного расположения на диаграмме. Однако структура данных файлов ориентирована преимущественно на визуальное отображение диаграммы в редакторе и не предназначена непосредственно для автоматизированного анализа архитектурных элементов.

В связи с этим возникает задача разработки программных средств, обеспечивающих автоматический разбор XML-представления диаграмм и извлечение из него структурной информации об архитектуре программной системы [6,7]. В рамках данной работы разрабатывается программный модуль на языке Python, предназначенный для анализа архитектурных диаграмм формата .drawio. Обработка XML-структуры осуществляется с использованием библиотеки lxml [8], предоставляющей эффективные средства для работы с XML-документами, включая навигацию по дереву элементов, извлечение атрибутов и обработку структурированных данных [9].

Парсер формирует структурированное представление архитектуры системы, включающее архитектурные сущности (например, системы, контейнеры, компоненты) и связи между ними. Дополнительно модуль выполняет анализ геометрических параметров элементов диаграммы, включая их координаты и размеры. На основе этих данных определяется вложенность объектов и иерархическая структура архитектурных элементов. Результатом обработки является промежуточное представление архитектуры, содержащее информацию о сущностях и связях между ними.

Одним из направлений применения разработанного модуля является сравнение архитектурных диаграмм С4 с фактической структурой директорий программного проекта. Такое сопоставление позволяет выявлять расхождения между архитектурной документацией

и реальной организацией проекта, что особенно важно при длительном развитии программных систем.

Кроме того, извлечённое представление может использоваться для сравнения диаграмм Draw.io с архитектурными диаграммами, представленными в других форматах, например в нотации PlantUML [10]. Сопоставление различных представлений архитектуры позволяет выявлять различия между ними, контролировать согласованность документации и упрощать процесс её сопровождения.

Полученные результаты демонстрируют возможность преобразования диаграмм Draw.io в формализованное промежуточное представление, пригодное для дальнейшего сравнения архитектур, анализа изменений и генерации альтернативных нотаций. Разработанное решение может быть интегрировано в системы сравнения и контроля архитектурной документации и инструменты анализа изменения программных систем.

ЛИТЕРАТУРА

1. B. Hongyu Pei, et al. "A systematic review of software architecture evolution research." *Info & Software Technology*, 54. 1, 2012, pp. 16–40.
2. B. Selic, "Agile Documentation, Anyone?," in *IEEE Software*, vol. 26, no. 6, pp. 11-12, Nov.-Dec. 2009, doi: 10.1109/MS.2009.167.
3. Draw.io diagrams [Электронный ресурс] Режим доступа: <https://www.drawio.com/> (дата обращения: 11.03.2026).
4. Richards, Mark; Ford, Neal (2019). *Fundamentals of software architecture: an engineering approach*. O'REILLY MEDIA. p. 293. ISBN 978-1-4920-4342-3. OCLC 1138515057
5. Extensible Markup Language (XML) 1.0 (Fifth Edition) [Электронный ресурс] Режим доступа: <https://www.w3.org/TR/xml/> (дата обращения: 11.03.2026).
6. Тронеv, В. В. Сравнение технических документаций сетевой инфраструктуры в форматах PlantUML и C4 / В. В. Тронеv, И. А. Селин // *Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года.* – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 344-345. – EDN WHMOOY.
7. Мамонова, А. С. Анализ архитектурной документации сравнение C4-диаграмм с каталогами проекта / А. С. Мамонова, В. В. Тронеv, И. А. Селин // *Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года.* – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 325-327. – EDN DGBUFH.
8. lxml - XML and HTML with Python. [Электронный ресурс] Режим доступа: <https://lxml.de/> (дата обращения: 11.03.2026).
9. Uzun E., Yerlikaya T., Kirat O. Comparison of python libraries used for web data extraction // *Journal of the Technical University-Sofia Plovdiv Branch, Bulgaria.* – 2018. – Т. 24. – С. 87-92.
10. Component Diagram [Электронный ресурс] // PlantUML at a Glance. – Режим доступа: <https://plantuml.com/component-diagram> (дата обращения 12.03.2026)

УДК 004.457

Е.М. Бзычкин, Д.А. Бордуков, Ю.В. Адодина (2 курс бакалавриата),
К.К. Смирнов, ст. преподаватель

РАЗЛИЧНЫЕ ПОДХОДЫ В ПРИМЕНЕНИИ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ В ПРИЛОЖЕНИИ AMPHIMIXIS

Приложение Amphimixis [3] создавалось с целью упрощения работы при исследовании пригодности проектов под архитектуру RISC-V и их оптимизации. Входными данными является файл конфигурации, в котором описаны все машины, — компьютеры, — для запуска сборки, кросс-сборки и профилирования, — адреса в сети и данные аутентификации, — компиляторы, флаги сборки и др. Сначала проект анализируется на качество

сопровождаемости: ищутся CI, тесты, системы сборки и др. Затем Amphimixis подключается к нужным машинам по SSH и выполняет соответствующие действия.

При разработке возникли естественные вопросы о том, как анализировать данные: репозиторий проекта и результаты профилирования. Чтобы учесть все возможные случаи, нужно написать много логической составляющей, но даже при этом нельзя быть уверенными, что применённые эвристики обрабатывают все интересующие случаи. Со стремительным развитием больших языковых моделей (LLM) появились возможности делегировать анализ на некоторую достаточно хорошо обученную модель, которая может делать выводы из данных на человеческом языке. Кроме того, задача Amphimixis и его работа хорошо описываются на естественном языке.

Целью данной работы является рассмотрение различных подходов использования LLM в разработке приложения. Для этого были поставлены следующие задачи:

- Реализовать анализ репозитория проектов с обращением к LLM.
- Реализовать анализ результатов профилирования и сравнение трасс с помощью LLM.
- Обратное применение: использовать разработанные для Amphimixis инструменты в чат-агенте, основанном на Opencode [4].

В Amphimixis анализом репозитория занимается модуль Анализатор: ищет зависимости проекта, пути до тестов, проверяет поддерживаемые системы сборки и др., — результаты его работы используются остальными модулями, например, на этапе сборки используются найденные системы сборки, если пользователем не указаны иные. Проекты могут сильно различаться по сценариям сборки, структуре репозитория, путям и названиям файлов, которые генерируются при сборке, например, пути до собранных тестов и бенчмарков, которые используются для профилирования. Такой анализ производится с помощью эвристик поиска в репозитории, основанного на опыте команды.

Для повышения эффективности работы Анализатора в него был внедрён ReAct-агент [5], который по шагам исследует репозиторий проекта: анализирует дерево каталогов, системные файлы CI, конфигурации сборки, документацию и файлы тестов, для извлечения более глубокой и разнородной информации о проекте. Практически ReAct-агент реализован как комбинация лёгкой LLM или локальной модельной подсистемы для рассуждений и набора безопасного API инструментов для LLM, передаваемого с помощью библиотеки LangChain: чтение файлов, обход репозитория проекта.

Собирает трассы работы приложения, — данные о работе процессора во время выполнения кода проекта, — модуль Профилировщик, используя perf stat и perf record для разных конфигураций на разных машинах. На этом этапе появляется задачи сравнения большого количества характеристик со многих различных трасс, анализ горячих мест, — инструкций, которые процессор обрабатывал большее количество тактов, — и предложения по оптимизациям, которые делегируются LLM [6].

Также предлагается рассмотреть иную стратегию использования LLM. До этого драйвером сценария работы было само приложение, которое обращалось к LLM за анализом, со строгой логикой, описанной в коде Amphimixis, но современные LLM с инструментами, хорошо справляются со многими задачами, а LLM-агенты предоставляют удобные среды для конфигурации собственных агентов. Так, на базе Opencode был сконфигурирован LLM-агент [7], выполняющий задачи Amphimixis, — LLM используется как драйвер процесса, оперируя инструментами, полученными в ходе разработки Amphimixis.

Opencode, с одной стороны, позиционируется как LLM-агент для разработчика, с другой стороны, позволяет расширять функциональность с помощью пользовательских инструментов, субагентов, навыков (Skills) и др. Поэтому было решено оформить процесс анализа проекта как навык для Opencode, добавить инструменты: анализа, сборки и профилирования, — ими упрощаются некоторые детали анализа, таким образом экономятся токены на чтение файлов, и контролируется запуск команд сборки и профилирования в системе, чтобы галлюцинации LLM не отразились в выполняемых командах. В системном запросе агента описан сценарий работы Amphimixis. Также планируется внедрить

дополнительные инструменты, например, использование глобального файла конфигурации Amphimixis, в котором хранятся именованные машины и тулчейны, — наборы инструментов для компиляции.

Как итог, внедрение ReAct-агента в Анализатор и LLM в Профилировщик повышает автоматизацию, реализуемую Amphimixis: агент позволяет получать больше деталей при анализе проекта, а использование LLM на этапе профилирования повышает качество сравнения трасс и предлагает возможные оптимизации горячих мест выполнения приложения. С другой стороны, среда Opencode позволила интегрировать Amphimixis в чат-агент, — контролируемо использовать LLM для интересных задач.

Работа выполнена в Лаборатории технологий программирования инфраструктурных решений СПбГУ.

ЛИТЕРАТУРА

1. Brendan Gregg, Linux Performance Analysis and Tools, 2021 [Электронный ресурс] Режим доступа: <https://www.brendangregg.com/linuxperf.html>
2. Rocha T., Neves N., Roma N., Tomás P., Sousa L. RVEBS: Event-Based Sampling on RISC-V // 2025 Design, Automation & Test in Europe Conference (DATE) [Электронный ресурс]. Режим доступа: https://web.tecnico.ulisboa.pt/~ist14359/wordpress/nfvr_pubs/date25.pdf
3. GitHub-репозиторий [Электронный ресурс]. Режим доступа: <https://github.com/ebzych/amphimixis>. [Дата обращения 17.03.2026]
4. Anomaly, сайт LLM-агента Opencode. Режим доступа: <https://opencode.ai>
5. Pull request с ReAct-агентом [Электронный ресурс]. Режим доступа: <https://github.com/ebzych/amphimixis/pull/67> [Дата обращения 17.03.2026]
6. Pull request с LLM в Профилировщике [Электронный ресурс]. Режим доступа: <https://github.com/ebzych/amphimixis/pull/66> [Дата обращения 17.03.2026]
7. Ветка с интеграцией с Opencode [Электронный ресурс]. Режим доступа: <https://github.com/ebzych/amphimixis/tree/opencode> [Дата обращения 17.03.2026]

УДК 004.021

А.М. Братенков (4 курс бакалавриата),
Н.О. Степина, ассистент,
О.А. Юсупова, ст. преподаватель,
И.В. Никифоров, к.т.н., доцент

ПРИМЕНЕНИЕ ЧИСЛЕННЫХ МЕТОДОВ ПОНИЖЕНИЯ РАЗРЯДНОСТИ ДАННЫХ ПРИ РЕШЕНИИ КОМБИНАТОРНЫХ ЗАДАЧ НА ПЛАТФОРМЕ RISC-V

Архитектура RISC-V, благодаря своей открытости и модульности, становится перспективной платформой для высокопроизводительных вычислений [1, 2]. Многие прикладные задачи в области логистики, планирования и машинного обучения сводятся к решению NP-трудных комбинаторных задач, для которых эффективно применение метаэвристик, в частности, метода имитации отжига в рамках модели Изинга [3, 4].

Ключевой проблемой при реализации алгоритмов на основе модели Изинга является работа с матрицей взаимодействий J , размер которой растет квадратично с увеличением числа спинов N . При N порядка 10^5 объем данных достигает десятков гигабайт, что создает высокую нагрузку на подсистему памяти и ограничивает производительность. Перспективным способом снижения этой нагрузки является квантование матрицы до целочисленных типов пониженной разрядности. Однако прямое округление вещественных коэффициентов ведет к искажению энергетического ландшафта и потере точности решений [5].

Целью данной работы является повышение эффективности решения комбинаторных оптимизационных задач на архитектуре RISC-V за счет применения усовершенствованных

методов округления матрицы взаимодействий модели Изинга. Для достижения поставленной цели необходимо решить следующие задачи.

1. Выполнить анализ архитектуры RISC-V и методов аппаратно-программной реализации алгоритмов отжига для модели Изинга.
2. Исследовать влияние ограниченной разрядности представления данных на производительность и точность решения задач комбинаторной оптимизации.
3. Разработать программный прототип на базе архитектуры RISC-V для реализации алгоритмов квантования матрицы взаимодействия и алгоритма имитации отжига.
4. Провести экспериментальное исследование влияния методов округления на точность и скорость получаемых решений на примере задач коммивояжера, о рюкзаке и о максимальном разрезе графа.

В рамках работы выбран подход чисто программной реализации на стандартном RISC-V процессоре, входящем в состав одноплатного компьютера Lichee Pi 4A. Данный выбор обусловлен необходимостью гибкого прототипирования и изолированного изучения влияния алгоритмов округления без усложнения задачи аппаратными ограничениями, при сохранении возможности последующего переноса результатов на архитектуры со специализированными расширениями или аппаратными ускорителями.

Алгоритм имитации отжига для модели Изинга и исследуемые методы квантования были реализованы на языке C++ в среде RISC-V. Помимо обычного и стохастического округления [6], были реализованы восемь алгоритмов диффузионного округления [7], различающихся схемой и коэффициентами распространения ошибки.

Схема эксперимента представлена на рисунке 1. Эксперимент проводился для трех классических NP-трудных задач: задачи коммивояжера, задачи о рюкзаке и задачи о максимальном разрезе графа. Для каждой задачи использовались стандартные тестовые наборы данных. К матрице связей применялись различные методы округления с битностью округления от 2 до 16 бит, затем сжатая матрица подавалась на вход алгоритму имитации отжига. Для каждой задачи вычислялись свои метрики качества решений на основе множества запусков.

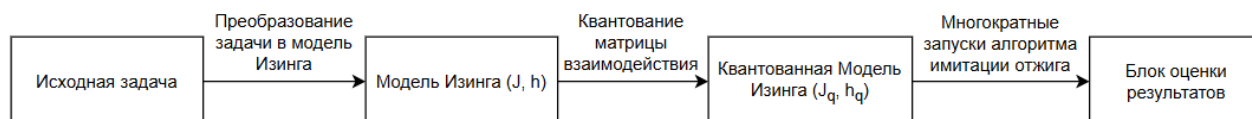


Рисунок 1 – Схема экспериментального исследования методов квантования

Для оценки качества решений применялись следующие метрики: среднее значение целевой функции и соотношение количества решений, удовлетворяющих ограничениям, ко всему количеству решений.

Результаты эксперимента показали, что эффективность методов диффузионного и стохастического округления зависит от рассматриваемой задачи.

Для задачи о рюкзаке диффузионные методы округления не продемонстрировали значимого преимущества, качество получаемых решений оказалось сопоставимым с результатами обычного округления при соответствующей битности. Стохастическое округление в среднем показывало несколько худшие результаты, однако в отдельных случаях наблюдалось незначительное улучшение.

В задаче коммивояжера характер влияния методов округления оказался более сложным. Стохастическое округление в целом показало результаты хуже обычного округления на всем диапазоне битностей. Диффузионные методы, напротив, продемонстрировали улучшение качества решений в области малых битностей (2–4 бита), однако при увеличении разрядности свыше 4 бит они уже уступали обычному округлению.

Наиболее интересные результаты получены для задачи о максимальном разрезе графа. Если стохастическое округление показало результаты, близкие к обычному округлению, то применение диффузионных методов позволило добиться стабильного улучшения качества решений. Их применение обеспечили прирост среднего значения целевой функции на 10% по сравнению с обычным и стохастическим округлением.

ЛИТЕРАТУРА

1. Cherepanov, N. I. Improving image analysis and processing performance on the RISC-V platform with Lichee Pi 4A / N. I. Cherepanov, N. O. Stepina, I. V. Nikiforov // *Proceedings of the Institute for System Programming of the RAS*. – 2025. – Vol. 37, No. 5. – P. 157-172. – DOI 10.15514/ISPRAS-2025-37(5)-12. – EDN UDVIEH.
2. Шемаев, К. Е. Векторизация методов решения обыкновенных дифференциальных уравнений на процессорной архитектуре RISC-V / К. Е. Шемаев, П. А. Ребдев, Н. О. Степина // *Техника средств связи*. – 2025. – № 4(172). – С. 41-50. – DOI 10.24412/2782-2141-2025-4-41-50. – EDN KUNERJ.
3. Lucas A. Ising formulations of many NP problems / A. Lucas // *Frontiers in Physics*. – 2014. – Vol. 2, article 5. – 27 p.
4. Kirkpatrick S. Optimization by simulated annealing / S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi // *Science*. – 1983. – Vol. 220, No. 4598. – P. 671–680.
5. Cui D. Proposal and Validation of Annealing-processor-calculation Method Using Error-diffusion Rounded Interaction Matrix / D. Cui, T. Kawahara // *2024 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. – Taipei, Taiwan, 2024. – P. 50–54.
6. Deep learning with limited numerical precision / S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan // *Proceedings of the 32nd ICML*. – 2015. – P. 1737–1746.
7. Floyd, R. W. An adaptive algorithm for spatial grey scale / R. W. Floyd, L. Steinberg // *Journal of the Society for Information Display*. – 1976. – Vol. 17. – P. 75–77.

УДК 004.451.34

А.С. Бугаев (3 курс бакалавриата),
Г.К. Сичкар (4 курс бакалавриата),
А.И. Васенина, инженер-исследователь

АДАПТАЦИЯ АЛГОРИТМА LZ4 ДЛЯ ОБРАБОТКИ РАЗРЫВНЫХ БУФЕРОВ В БЛОЧНОМ УРОВНЕ ЯДРА LINUX

Сжатие без потерь применяется в файловых системах и системах хранения данных для сокращения расходов дискового пространства. Одним из наиболее известных алгоритмов сжатия без потерь является LZ4, используемый, например, в ZFS [1]. Благодаря высокой скорости сжатия алгоритм LZ4 остается актуальным для сжатия в реальном времени, несмотря на более низкую степень сжатия по сравнению с другими алгоритмами [8].

Блочный уровень ядра Linux [2] — это подсистема, отвечающая за управление операциями чтения/записи между файловой системой и блочными устройствами. Библиотека LZ4 включена в ядро Linux [3], что делает её доступной для использования сторонними модулями, в том числе и обработки операций чтения/записи на блочном уровне.

Операции чтения/записи могут содержать большие объёмы данных, поэтому вместо непрерывных буферов на блочном уровне используются разрывные (scatter-gather) буферы, представляющие собой последовательность непрерывных кусков памяти, называемых векторами. Вектор данных блочного запроса представляется в ядре Linux структурой `bio_vec` [4], а разрывный буфер можно представить как массив таких векторов. Разрывный буфер служит носителем данных для блочного запроса, представленного в ядре Linux структурой `bio`.

Стандартная реализация LZ4 в ядре Linux поддерживает работу только с непрерывными буферами [5], так как основана на арифметике указателей [6]. Для работы с разрывными буферами в рамках существующей реализации можно выделить следующие возможные решения:

- копирование в дополнительный буфер — требует дополнительных затрат по времени и памяти, но обеспечивает оптимальную степень сжатия;
- сжатие каждого из непрерывных буферов с использованием предыдущего в качестве словаря — не требует дополнительной памяти, но малый словарь снизит степень сжатия.

Целью данной работы является адаптация адресной арифметики в функции сжатия библиотеки LZ4 для работы с разрывными буферами блочных запросов в ядре Linux.

Для достижения данной цели были поставлены следующие задачи.

1. Реализовать тестовый драйвер блочного устройства, включающий компрессию LZ4 в процесс обработки блочных запросов.
2. Спроектировать расширение алгоритмов компрессии/декомпрессии LZ4 для поддержки scatter-gather буферов.

Реализованное блочное устройство представляет собой виртуальное блочное устройство поверх существующего, заданного через параметры sysfs. При обработке операций чтения/записи данные сжимаются и разжимаются обратно, что позволит проверить корректность модифицированного алгоритма.

API функций компрессии и декомпрессии было модифицировано с помощью замены непрерывных буферов на массив структур `bio_vec`. Расширенные функции также принимают итераторы по разрывному буферу (`struct bvec_iter` [7]) в качестве начальных позиций, они же содержат и размеры входа/выхода. Сигнатура расширенной функции компрессии представлена на рисунке 1. Предложенное решение заключается в замене использования арифметики указателей на работу с `bvec_iter`.

```
/*
 * src: source buffer as a list of bio_vec's
 * dst: destination buffer as a list of bio_vec's
 * srcIter: iterator into 'src' at the start of data to compress
 * dstIter: iterator into 'dst' starting from which the bytes are written
 * wrkmem: pointer to the allocated working memory
 * returns: number of bytes written to 'dst', or 0 if compression fails
 */
int LZ4E_compress_default(const struct bio_vec *src, struct bio_vec *dst,
                          struct bvec_iter *srcIter, struct bvec_iter *dstIter, void *wrkmem);
```

Рисунок 1 – Сигнатура расширенной функции компрессии LZ4

В качестве основного языка разработки использовался C, для разработки модулей использовался API ядра Linux. Сборка осуществляется с помощью make и Kbuild, тесты и вспомогательные скрипты написаны на bash. Из используемых инструментов можно также выделить bear, clang-format, clang-tidy, разработка велась на виртуальной машине.

ЛИТЕРАТУРА

1. Параметры компрессии в ZFS. [Электронный ресурс] Режим доступа: <https://openzfs.github.io/openzfs-docs/Performance%20and%20Tuning/Workload%20Tuning.html#compression> (дата обращения: 17.03.2026)
2. A block layer introduction part 1: the bio layer. [Электронный ресурс] Режим доступа: <https://lwn.net/Articles/736534/> (дата обращения: 17.03.2026)
3. Исходный код LZ4 в ядре Linux. [Электронный ресурс] Режим доступа: <https://elixir.bootlin.com/linux/v6.19.8/source/lib/lz4> (дата обращения: 17.03.2026)

4. Структура bio_vec в ядре Linux. [Электронный ресурс] Режим доступа: <https://elixir.bootlin.com/linux/v6.19.8/source/include/linux/bvec.h#L19> (дата обращения: 17.03.2026)
5. Сигнатура одной из функций сжатия LZ4 в ядре Linux. [Электронный ресурс] Режим доступа: <https://elixir.bootlin.com/linux/v6.19.8/source/include/linux/lz4.h#L175> (дата обращения: 17.03.2026)
6. Pointer Arithmetic. [Электронный ресурс] Режим доступа: https://www.gnu.org/software/c-intro-and-ref/manual/html_node/Pointer-Arithmetic.html (дата обращения: 17.03.2026)
7. Структура bvec_iter в Linux Kernel. [Электронный ресурс] Режим доступа: <https://elixir.bootlin.com/linux/v6.19.8/source/include/linux/bvec.h#L77> (дата обращения: 17.03.2026)
8. Maulidina A. P. et al. Comparative study of data compression algorithms: Zstandard, zlib & lz4 // International Conference on Science, Engineering Management and Information Technology. – Cham: Springer Nature Switzerland, 2023. – С. 394-406. (дата обращения: 17.03.2026)

УДК 004.421

Е.В. Бутаковский, К.А. Демешкина (2 курс бакалавриата),
В.В. Амосов, к.т.н., доцент

ПРОГРАММНАЯ И АППАРАТНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ФЛОЙДА-УОРШЕЛЛА

Современный этап развития вычислительной техники характеризуется активным внедрением высокопроизводительных систем на кристалле и программируемых логических интегральных схем (ПЛИС) [1, 2]. Использование подобной элементной базы открывает возможности для аппаратного ускорения ресурсоемких процедур, что критично для бортовых интеллектуальных систем независимых программно-аппаратных комплексов, функционирующих в режиме реального времени [2, 3].

Целью работы является проектирование программно-аппаратной реализации алгоритма Флойда-Уоршелла для поиска кратчайших путей между всеми парами вершин графа [3, 4]. Выбор данного алгоритма обусловлен его способностью корректно обрабатывать ребра с отрицательным весом [3], что позволяет эффективно моделировать энергетические характеристики независимых программно-аппаратных комплексов [4]. В рамках модели приняты следующие допущения:

- положительный вес соответствует расходу энергии при активном продвижении;
- нулевой вес соответствует режиму планирования или зависания;
- отрицательный вес соответствует восполнению энергии.

В ходе исследования разработан программный комплекс, включающий реализацию алгоритма на языке С и аппаратный модуль на языке описания аппаратуры VHDL [1, 2]. Выполнено конфигурирование схем на уровне регистровых передач и технологического отображения [2]. Алгоритм также реализован в системе команд 32-разрядного RISC-процессора DP32 [2].

Для верификации разработанных модулей применялись специализированные тестовые окружения [2]. Программная часть реализована в среде Visual Studio Code, аппаратная — в САПР Quartus II и Active-VHDL [1, 2]. Проведенная ко-симуляция на поведенческой и структурной моделях подтвердила корректность работы алгоритма [2]. Разработанная система представляет собой эффективное решение для независимых программно-аппаратных комплексов, позволяющих в реальном времени оптимизировать маршруты с учетом динамического энергобаланса [3, 4].

ЛИТЕРАТУРА

1. Хвищук А. Ю., Григорьев А. В. Современные средства генерирования VHDL описаний аппаратуры при проектировании программируемых схем FPGA // Микроэлектроника и

- информатика : тез. 25-й Всероссийская межвузовская научно-техническая конференция студентов и аспирантов. – М. : МИЭТ, 2018. – С. 316
2. Амосов В. В. Схемотехника и средства проектирования цифровых устройств. – СПб.: БХВ-Петербург, 2007. – 560 с.
 3. Бодра Д., Хайрнар С. Ускорение и анализ производительности алгоритмов поиска кратчайшего пути на GPU с использованием платформы CUDA: алгоритмы Беллмана-Форда, Дейкстры и Флойда-Уоршелла // Научно-технический вестник информационных технологий, механики и оптики. – 2025. – № 8. – С. 866-875
 4. Зяблова Е. Р. Модификация алгоритма поиска кратчайших путей GH-графа для анализа сложных технических систем // Программные продукты и системы. – 2024. Т. 37. – № 3. – С. 354-363

УДК 004.421

П.В. Гаврилова, А.К. Карнаухова, С.Р. Киселёв (2 курс бакалавриата),
В.В. Амосов, к.т.н., доцент

РЕАЛИЗАЦИЯ АЛГОРИТМА DFS (ОБХОДА В ГЛУБИНУ) ДЛЯ ПРОСТЕЙШЕГО МИКРОПРОЦЕССОРНОГО НАБОРА (DP32) И ПЛИС

В связи с развитием технологий программируемых логических интегральных схем (ПЛИС) и систем на кристалле (SoC), сочетающих на одном микрочипе центральный процессор, память и программируемую логику, актуальной задачей является разработка гибридных программно-аппаратных решений [2]. Особую значимость данный вопрос приобретает в контексте растущих требований к энергоэффективности и быстродействию встраиваемых вычислительных систем. Создание специализированной вычислительной логики для конкретных алгоритмов на базе ПЛИС позволяет значительно ускорить выполнение задач по сравнению с классической программной реализацией на универсальных процессорах.

Целью данной работы является разработка программно-аппаратной реализации алгоритма обхода графа в глубину (Depth-First Search, DFS). Данный алгоритм является фундаментальным для решения широкого круга задач, включая поиск связности, топологическую сортировку, поиск циклов и решение лабиринтов [4], что делает его востребованным в системах автоматизации проектирования, навигации и анализа данных.

В ходе проекта разработан комплексный программный продукт, состоящий из следующих элементов:

1. Программная реализация алгоритма DFS на языке программирования C++ для проверки корректности аппаратных версий.
2. Аппаратная реализация алгоритма DFS на языке описания аппаратуры VHDL [1][3], ориентированная на синтез в ПЛИС. Ключевой особенностью данной реализации является использование итеративного подхода с явным управлением стеком вершин, что позволяет избежать рекурсивных вызовов и обеспечить детерминированное время выполнения.
3. Конфигурирование Register-Transfer Level (RTL) и Technology Map схем алгоритма в среде САПР Quartus II [2].
4. Реализация алгоритма DFS в виде последовательности команд для 32-разрядного RISC-процессора с Гарвардской архитектурой — DP32.
5. Проведение ко-симуляции (совместного моделирования) поведенческой и структурной моделей процессора DP32 для проверки корректности выполнения кода. Тестовые сценарии включали обход графов с различным количеством вершин и рёбер, а также обработку граничных случаев (пустой граф, граф с циклами).

Для тестирования разработанных решений были созданы соответствующие Test Bench окружения. Программная часть отлаживалась в средах Visual Studio Code и Active VHDL. Аппаратная реализация проверялась на синтезируемость и функциональную корректность с использованием пакета САПР Quartus II [2].

Разработанная система представляет собой комплексное решение, демонстрирующее эффективность переноса базовых алгоритмов обработки графов на аппаратный уровень. Учитывая возможности современных ПЛИС, предложенную реализацию можно масштабировать и применять в широком спектре промышленных и научных задач, требующих быстрого анализа связей в графовых структурах[5].

ЛИТЕРАТУРА

1. Kenneth L. Short, “VHDL for Engineers”, International Journal of Computer Science, vol. 12, no. 3, pp. 195-209, 2014.
2. В.В.Амосов, “Схемотехника и средства проектирования цифровых устройств” – СПб.: БХВ-Петербург, 2007 – 560 с.
3. Вахитов Р.М. Язык VHDL. Практическое применение. — М.: ДМК Пресс, 2020
4. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. — MIT Press, 2009.
5. Bohua Yang, Dong Wen, Lu Qin, Ying Zhang, Xubo Wang, Xuemin Lin. Fully Dynamic Depth-First Search in Directed Graphs — Proceedings of the VLDB, 2019.
6. CIS 1210: Data Structures and Algorithms Course Lecture Notes — University of Pennsylvania, 2023

УДК 621.396.96

В.А. Гарькуша, Е.А. Ганин, С.А. Игнатушин, С.В. Сатаев (4 курс бакалавриата),
О.Н. Корелин, к.т.н., доцент,
Д.А. Кобляков, ст. преподаватель

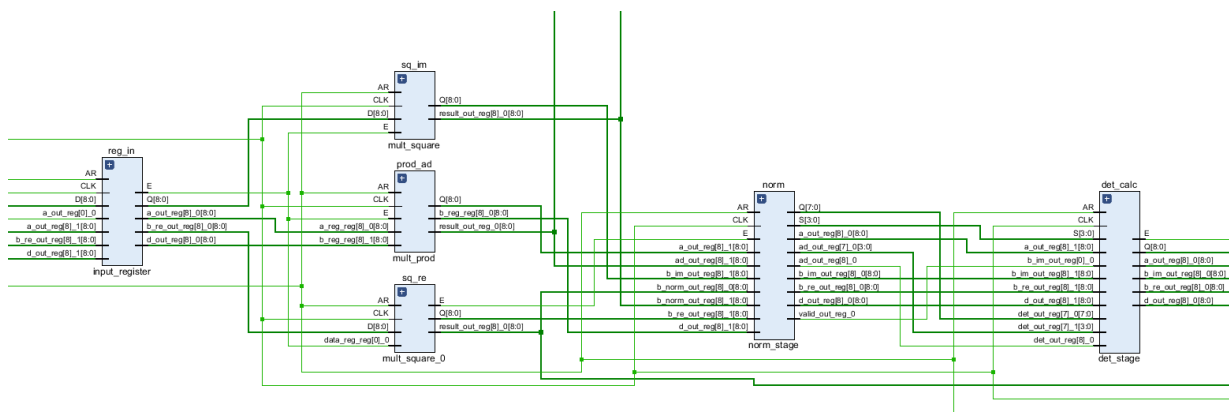
РЕАЛИЗАЦИЯ МАТЕМАТИЧЕСКИХ ОПЕРАЦИЙ НАД МАТРИЦАМИ НА FPGA

Современные вычислительные задачи требуют высокой производительности при обработке больших объемов данных в реальном времени. Особенно актуально это для цифровой обработки сигналов и матричных вычислений в ММО, где операции над матрицами (умножение, сложение, обращение) являются узким местом алгоритмов. Задача заключается в реализации обращения комплексных матриц 2×2 с использованием ресурсов FPGA. Для решения рассматриваются несколько подходов: LU-разложение, метод Холецкого и метод алгебраических дополнений. В нашем исследовании выбран метод алгебраических дополнений, поскольку для матриц 2×2 он обеспечивает минимальное число операций при вычислении детерминанта и гарантирует наибольшую скорость выполнения для получения обратной матрицы. На практике используется метод наименьших квадратов по формуле $(N \cdot N^H)^{-1} \cdot N^H \cdot y$, что позволяет эффективно решать задачи цифровой обработки сигналов.

Для экспериментов использовались отладочные платы на базе SoC Zynq 7000: Bajie Board XC7Z010 [1] и PYNQ-Z2. Характеристики плат: Bajie Board XC7Z010 оснащена SoC Zynq-7010 с 1 ядром ARM Cortex-A9 (максимальная частота 667 МГц), поддержкой DDR3 памяти до 512 МБ, интерфейсами USB, HDMI-in, miniDP для подключения монитора, а также доступом к GPIO и AXI-шинам для взаимодействия с PL частью. PYNQ-Z2 построена на Zynq-7020, содержит 2 ядра ARM Cortex-A9 (667 МГц), 512 МБ DDR3, 512 МБ QSPI Flash, поддерживает USB, Ethernet, HDMI-out, PMOD и Arduino совместимые разъемы. Выбор Xilinx обусловлен доступностью плат на кафедре, обширной учебной документацией и специализированным ПО — САПР Vivado и Vitis [3], обеспечивающим проектирование и синтез аппаратной логики, а также встроенные средства для работы с DSP-блоками и конвейеризацией операций.

Для разработки применялся High-Level Synthesis (HLS) в среде Vitis. HLS позволяет писать алгоритмы на C++ с поддержкой фиксированной и плавающей точек, синтезировать их в RTL и оптимизировать с помощью pragma-директив. Для алгоритма Холецкого имелись готовые примеры, однако для метода алгебраических дополнений HLS показал ограниченную

Матричные операции, такие как умножение, сложение и вычитание, реализованы с использованием встроенных DSP-блоков (25×18) FPGA, что обеспечивает значительное ускорение вычислений. Совокупность конвейерной архитектуры и DSP-блоков позволяет обрабатывать большое количество матриц с высокой скоростью и низкой задержкой, обеспечивая высокую производительность и эффективность при реализации алгоритмов цифровой обработки сигналов на FPGA.



SIMULATION - Behavioral Simulation - Functional - sim_1-tb_all

tb_matrix_inv_pipeline_behav.wcfg x tb_all v

Name	Value
clk	0
rst	0
valid_in	1
a_in[8]	-0.4453125
d_in[8]	0.34375
b_re_in[8]	0.5625
b_im_in[8]	0.05078125
mul_a_d[17]	-21.328125
mul_re[17]	3.515625
mul_im[17]	2.8475625
b_re_sq_out[8]	0.01171875
b_im_sq_out[8]	0.0078125
ad_prod_out[8]	-0.0859375
b_norm_s3d[8]	0.0125
inv_det_s5d[8]	-0.97265625
valid_out	1
det[8]	-0.390625
inv_a11[8]	-0.24609375
inv_a22[8]	0.296875
inv_b_re[8]	-0.0703125
inv_b_im[8]	0.11328125

363

ЛИТЕРАТУРА

1. QMTECH ZYNQ7000 Bajie board user manual (hardware) - [https://esys.ir/images/img_Item/2401/Files/Bajie_Board_User_Manual\(Hardware\)_V01.pdf](https://esys.ir/images/img_Item/2401/Files/Bajie_Board_User_Manual(Hardware)_V01.pdf)
2. Харрис Д. М., Харрис С. Л. Цифровая схемотехника и архитектура компьютера : учебник / Д. М. Харрис, С. Л. Харрис. – Москва : ДМК Пресс, 2018. – 791 с. – ISBN 978-5-97060-522-6.
3. Xilinx. Vivado Design Suite User Guide: Getting Started (UG910) [Электронный ресурс] : Getting Started Tutorial / Xilinx. – Верс. 2025.1. – 2025. – Режим доступа: <https://docs.amd.com> (дата обращения: 15.03.2026). – Загл. с экрана.

УДК 004.4

Э.Ф. Денисов-Элерс, А.О. Шевченко (2 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ KVADRAOS

Операционная система KvadraOS является программной платформой, разрабатываемой компанией YADRO для планшетных устройств KVADRA_T. Система включает пользовательский интерфейс, а также набор различных приложений и сервисов. По мере развития системы регулярно добавляется новый функционал и вносятся изменения в кодовую базу.

Активная разработка влечет за собой риск нарушения пользовательского опыта и регрессии функциональности системы из-за возможных ошибок. Процесс тестирования способствует выявлению дефектов на ранних этапах разработки и повышает общее качество программного продукта. В связи с этим актуальной является работа по развитию автотестов KvadraOS для обеспечения стабильности и надежности системы.

В рамках работы применяется автоматизированное сквозное (end-to-end) тестирование пользовательского интерфейса. Тестовые сценарии имитируют реальные действия пользователя при работе с планшетом: навигацию по интерфейсу системы и приложений, взаимодействие с элементами управления и использование физических клавиш устройства. Автоматизация позволяет заменить часть ранее выполнявшихся вручную тест-кейсов, повысить повторяемость проверок и ускорить процесс регрессионного тестирования.

Тесты создаются на языке Python (версия 3.10.6) с использованием фреймворка pytest (версия 9.0.1) [1] и собственного фреймворка, включающего функционал имитации действий и проверки результатов. Взаимодействие с интерфейсом системы осуществляется при помощи инструмента автоматизации тестирования Android и Web приложений Appium (версия 3.1.1) [2].

Запуск тестирования осуществляется с помощью системы непрерывной интеграции Jenkins [3]. На рисунке 1 представлен автоматизированный пайплайн, включающий сборку системы, загрузку ее на тестовые устройства, запуск UI-тестов и формирование отчетов о результатах. Информация о результатах тестирования автоматически передается командам разработки и ручного тестирования.

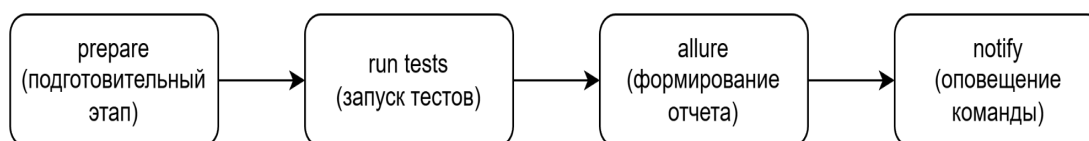
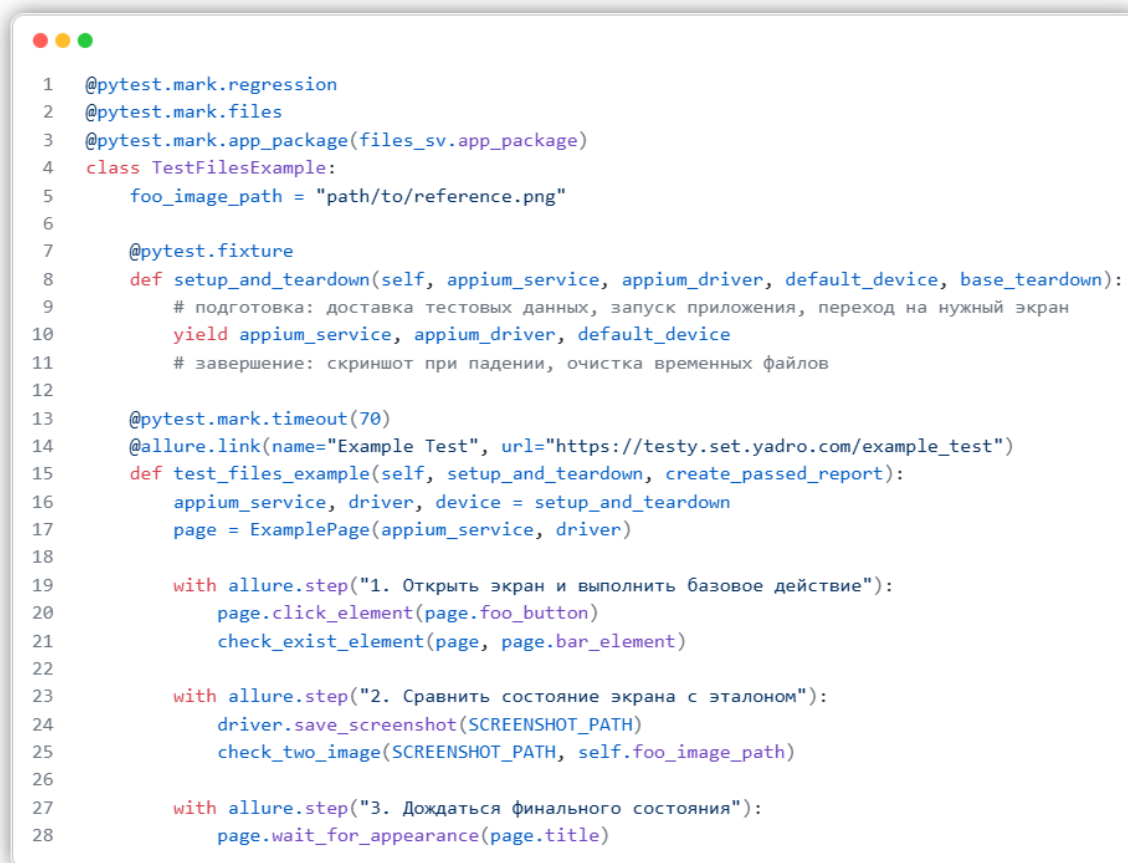


Рисунок 1 – Схема автоматизированного сценария тестирования Jenkins

Инфраструктура тестирования включает программные и аппаратные компоненты. Помимо упомянутых языка Python и фреймворка pytest используется система Allure (версия 2.35.0) [4], позволяющая представлять результаты выполнения в удобном для диагностики и

анализа виде. Тестирование производится на выделенном стенде из четырех планшетов KVADRA_T, что позволяет запускать сценарии параллельно и сокращает общее время проверки.

Каждый тест представляет собой Python-класс и состоит из трех основных этапов: setup, test и teardown (рисунок 2). На этапе setup подготавливается тестовое окружение и состояние устройства. В блоке test выполняется основная логика проверки, включающая имитацию действий пользователя и проверку корректности результатов выполнения действий. На этапе teardown выполняется очистка окружения и восстановление исходного состояния устройства. Шаги тестов описываются с использованием фреймворка Allure и соответствуют сценариям ручного тестирования.



```
1 @pytest.mark.regression
2 @pytest.mark.files
3 @pytest.mark.app_package(files_sv.app_package)
4 class TestFilesExample:
5     foo_image_path = "path/to/reference.png"
6
7     @pytest.fixture
8     def setup_and_teardown(self, appium_service, appium_driver, default_device, base_teardown):
9         # подготовка: доставка тестовых данных, запуск приложения, переход на нужный экран
10        yield appium_service, appium_driver, default_device
11        # завершение: скриншот при падении, очистка временных файлов
12
13    @pytest.mark.timeout(70)
14    @allure.link(name="Example Test", url="https://testy.set.yadro.com/example_test")
15    def test_files_example(self, setup_and_teardown, create_passed_report):
16        appium_service, driver, device = setup_and_teardown
17        page = ExamplePage(appium_service, driver)
18
19        with allure.step("1. Открыть экран и выполнить базовое действие"):
20            page.click_element(page.foo_button)
21            check_exist_element(page, page.bar_element)
22
23        with allure.step("2. Сравнить состояние экрана с эталоном"):
24            driver.save_screenshot(SCREENSHOT_PATH)
25            check_two_image(SCREENSHOT_PATH, self.foo_image_path)
26
27        with allure.step("3. Дождаться финального состояния"):
28            page.wait_for_appearance(page.title)
```

Рисунок 2 – Упрощённый пример структуры теста (демонстрационный код)

На данный момент в системе тестирования используется 579 тестов, из которых 306 относятся к smoke-набору (53%), а 370 входят в регрессионное тестирование (64%). В ходе разработки и выполнения автоматизированных тестов был выявлен ряд дефектов функциональности системы и пользовательского интерфейса. Среди них – избыточная перерисовка графика использования батареи в настройках системы, некорректная работа кнопки изменения размера после масштабирования и лишний перенос строки в заголовках в приложении Notes, проблемы с элементами плеера и распаковкой архивов в приложении Files.

Работы выполнялись в рамках лаборатории «Технологии программирования YADRO-Политех». В процессе разработки было создано более 30 автоматизированных UI-тестов и выявлено пять функциональных и интерфейсных дефектов. Разработка велась непосредственно в основном репозитории компании YADRO с доступом к рабочей инфраструктуре. Основной фокус автоматизации был направлен на системные приложения Файлы, Галерея и Часы. Автоматизация данных сценариев позволила снизить объем ручного регрессионного тестирования и повысить стабильность проверок ежедневных (nightly) сборок системы.

Дальнейшее развитие системы тестирования предполагает расширение тестового покрытия KvaadraOS, продолжение перевода ручных тестов в автоматические, внедрение метрик для оценки стабильности системы, увеличение функциональности фреймворка и оптимизацию объема регрессионного тестирования.

ЛИТЕРАТУРА

1. pytest Documentation [Электронный ресурс]. – URL: <https://docs.pytest.org/en/stable/> (дата обращения: 05.03.2026).
2. Appium Documentation [Электронный ресурс]. – URL: <https://appium.io/docs/en/latest/reference/> (дата обращения: 05.03.2026).
3. Jenkins Documentation [Электронный ресурс]. – URL: <https://www.jenkins.io/doc/> (дата обращения: 05.03.2026).
4. Allure Framework Documentation [Электронный ресурс]. – URL: <https://allurereport.org/docs/> (дата обращения: 05.03.2026).
5. Майерс Г., Сандлер К., Баджетт Т. Искусство тестирования программ. –3-е изд. – М.: Вильямс, 2013. – 272 с.
6. Чолак, Э. А. Интеграция DevOps и тестирования: как обеспечить качество по на каждом этапе / Э. А. Чолак, А. В. Петров // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 286-287. – EDN GJCCYU.

УДК 004.41

А.Н. Гринько (2 курс бакалавриата),
О.В. Александрова, ст. преподаватель

РАЗРАБОТКА ПОДХОДА К АВТОМАТИЗАЦИИ СБОРКИ И ДИСТРИБУЦИИ СЕРВЕРНОГО ЭМУЛЯТОРА API SWORDFISH

Swordfish [1] – это стандарт управления системами хранения данных, разработанный компанией SNIA [2], задачей которого является упрощение работы по администрированию и масштабированию инфраструктуры для IT-специалистов. Является преемником спецификации Redfish [3] компании DMTF [4].

Для тестирования данного стандарта был выпущен эмулятор на языке Python [5]. В рамках развития проекта и повышения эффективности был разработан полностью совместимый легковесный эмулятор на языке Go [6], который обеспечивает более высокую производительность и улучшенную архитектуру.

В существующем процессе развертывания эмулятора выявлен критический недостаток: необходимость ручной сборки из исходного кода в тестовых окружениях, что существенно замедляет подготовку инфраструктуры. В условиях активной разработки это приводит к снижению эффективности и дополнительным временным затратам.

Целью данной работы ставится поиск подхода к автоматизации сборки и дистрибуции с использованием средств непрерывной интеграции.

Для достижения этой цели были сформулированы следующие задачи:

- Оптимизация процесса сборки эмулятора;
- Реализация эффективного кэширования зависимостей;
- Организация мультиархитектурной поставки;
- Автоматизация формирования дистрибутивных архивов;
- Настройка автоматической публикации релизов.

Конфигурация пайплайна непрерывной интеграции реализована посредством платформы GitLab CI/CD. При каждом коммите в основной ветке автоматически выполняются проверки стиля кода, модульные тесты и первичная сборка бинарных файлов.

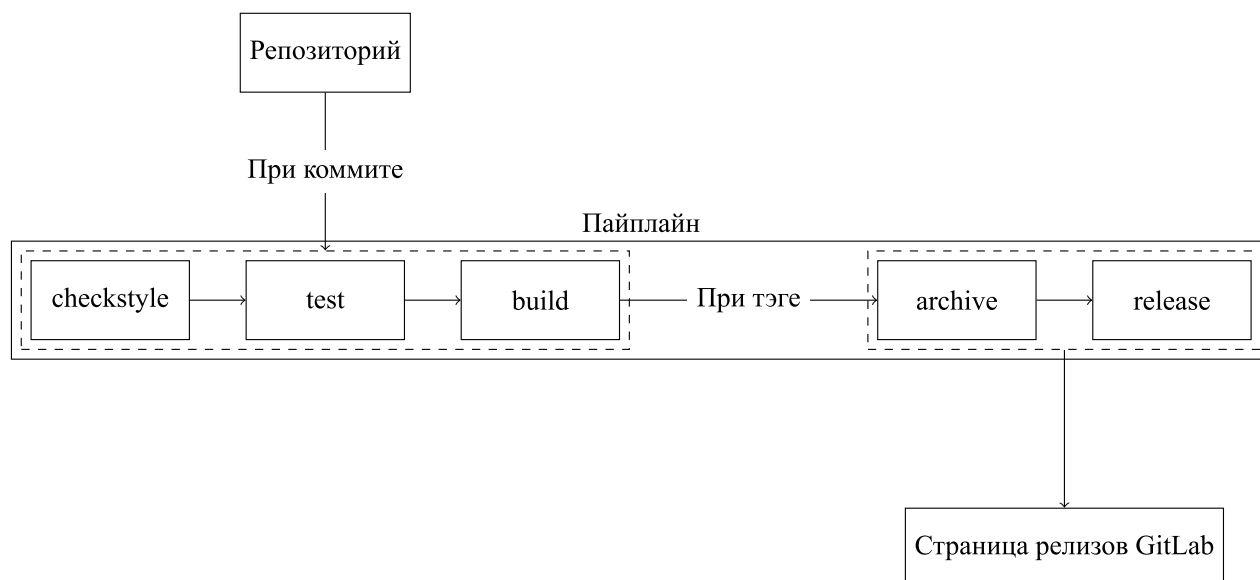


Рисунок 1 – Пайплайн

На рисунке 1 показаны шаги пайплайна непрерывной интеграции, отражающие полный цикл сборки эмулятора в контейнере на базе легковесного дистрибутива Linux – Alpine.

Для всех шагов ключевым элементом усовершенствования стало корректное кэширование пакетов-зависимостей языка Go. Все внешние библиотеки и вспомогательные инструменты размещаются в едином каталоге, путь к которому определяется через переменные окружения контейнера. Кэш этого каталога сохраняется между шагами, подтягиваясь из более ранних. Благодаря этому, время прохода по пайплайну сократилось в 4 раза – с 8 минут до 2.

В шаге build, предназначенном для сборки артефактов эмулятора, были применены ключи для компилятора, убирающие отладочную информацию и таблицы символов, что сократило размеры исполняемого файла. Дополнительно, этот шаг был разбит на два подшага, в которых задаётся переменная окружения GOARCH, обеспечивающая мультиархитектурную сборку.

Шаг archive подготавливает архивы с артефактами, базовой конфигурацией и дополнительными скриптами для всех поддерживаемых архитектур. Далее, подготовленные архивы передаются в шаг release, где применяется специально подготовленный платформой GitLab контейнер с инструментом release-cli [7], предназначенным для публикации на странице релизов GitLab.

Примененные изменения позволили сократить время, повысить удобство в развёртывании эмулятора, обеспечить автоматическую сборку сервера-эмулятора, а также получить опыт с системами непрерывной интеграции. В дальнейшем планируется автоматическое формирование контейнерных образов с настроенным эмулятором и публикация дистрибутива с помощью нового инструмента платформы GitLab – GLab [8].

ЛИТЕРАТУРА

1. SNIA Swordfish. URL: <https://www.snia.org/forums/smi/swordfish>.
2. SNIA. URL: <https://www.snia.org>.
3. Redfish. URL: <https://www.dmtf.org/standards/redfish>.
4. DMTF. URL: <https://www.dmtf.org>.
5. Swordfish-API-Emulator. URL: <https://github.com/SNIA/Swordfish-API-Emulator>.
6. Swordfish Emulator Go. URL: <https://gitlab.com/IgorNikiforov/swordfish-emulator-go>.
7. GitLab Release command-line tool. URL: <https://gitlab.com/gitlab-org/release-cli>.
8. GLab. URL: <https://gitlab.com/gitlab-org/cli>.

9. Автоматизация управления системами хранения данных в Kubernetes использованием CSI-драйвера и спецификации Swordfish / Соболева А. Е., Кладковой М. Д., Соломонов М. С., Никифоров И. В. // Современные технологии в теории и практике программирования: Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2025. – С. 338-339. – EDN EDN: XXXKREB
10. Реализация модуля управления доступом на основе ролей для сервер-эмулятора Swordfish / Д. А. Фоломкин, И. В. Никифоров, О. В. Александрова, Л. П. Котлярова // Современные технологии в теории и практике программирования: Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 24–25 апреля 2024 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. – С. 301-302. – EDN UIGVOW.

УДК 004.4'2

Э.Э. Истаев (1 курс магистратуры),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА ЕДИНОГО ПРОМЕЖУТОЧНОГО ПРЕДСТАВЛЕНИЯ ДЛЯ АНАЛИЗА АРХИТЕКТУРНОЙ ДОКУМЕНТАЦИИ

По мере роста сложности программных систем сопровождение архитектурной документации становится нетривиальной задачей. Архитектурные артефакты — C4-диаграммы, PlantUML-диаграммы, исходный код на Go, proto-файлы, структура директорий и ER-диаграммы — создаются и обновляются независимо, что неизбежно приводит к их рассогласованию [1, 2]. Явление расхождения реализации с задокументированной архитектурой принято называть архитектурным дрейфом (architectural drift) [3]. Для его своевременного обнаружения необходим автоматизированный инструментарий сравнения перечисленных артефактов [4].

Работы проводятся в рамках проекта по разработке инструментария анализа архитектурной документации систем управления сетью, решающего задачу сравнения четырёх пар представлений: C4-диаграммы со структурой директорий [5], C4-диаграммы с PlantUML-диаграммами [6], PlantUML-диаграмм с proto-файлами [7], PlantUML-диаграмм с исходным кодом на Go [8]. На начальном этапе каждая утилита имела собственный парсер и собственную модель данных, что порождало дублирование кода и несовместимость результатов.

Цель текущего этапа развития проекта — унификация кодовой базы посредством введения единого промежуточного представления (Intermediate Representation, IR).

Промежуточное представление — это структура данных, в которую каждый парсер преобразует свой артефакт, после чего единый компаратор сравнивает два IR и формирует отчёт о несоответствиях. Данный подход широко применяется в компиляторных технологиях и средствах статического анализа [9]. Перенос идеи в область архитектурного анализа позволяет поддерживать произвольное число форматов входных данных, добавляя только новый парсер, без изменения логики сравнения.

Разработанная модель IR реализована на языке Python с использованием библиотеки Rpydantic v2 [9] и включает три основных компонента. Сущность (*Entity*) описывает любой элемент архитектуры (сервис, таблицу базы данных, класс, директорию) и содержит нормализованный идентификатор, тип из фиксированного перечня, человекочитаемое имя и пространство имён. Связь (*Relationship*) описывает направленное отношение между двумя сущностями с указанием его характера: использование, вложенность, наследование и др. Контейнер (*IntermediateRepresentation*) объединяет списки сущностей и связей, хранит метаданные источника и автоматически проверяет целостность данных при создании объекта. Нормализацию идентификаторов обеспечивает отдельная утилита: имена сущностей из

разных источников приводятся к единому виду, что позволяет корректно сопоставлять их при сравнении. Для работы с диаграммами разных уровней детализации предусмотрена возможность исключения отдельных элементов (очереди, интерфейсов-посредников) из процесса сравнения.

Описанная структура данных используется парсерами четырёх форматов (PlantUML, C4/Draw.io, структура директорий, Go), реализованными участниками команды параллельно. Единый компаратор принимает два объекта IR и формирует унифицированный JSON-отчёт, содержащий раздел сводной статистики, списки совпавших и несовпавших сущностей, а также анализ связей с разделением на архитектурные и функциональные.

Внедрение IR исключило дублирование кода [11] парсеров (ранее только PlantUML-парсер существовал в трёх независимых копиях) и обеспечило единый контракт между компонентами проекта. Сравнение выполнялось на тестовых данных проекта: C4-диаграммы в формате Draw.io [12], PlantUML-диаграммы (компонентные и классовые) [13], структура директорий реального микросервисного продукта. Результаты подтвердили корректность идентификации совпадений и расхождений. Дальнейшее развитие модели предполагает добавление механизма нечёткого поиска для обнаружения семантически близких, но не идентичных имён сущностей.

ЛИТЕРАТУРА

1. Industrial software verification and testing technology/ P. D. Drobintsev, V. P. Kotlyarov, A. A. Letichevsky, I. A. Selin // CEUR Workshop Proceedings, Moscow, 14–16 нояб. 2017 г. – Moscow, 2017. – P. 221–229. – EDN RXUWDJ.
2. Technology and tools for developing industrial software test suites based on formal models and implementing scalable testing process on supercomputer / V. Kotlyarov, P. Drobintsev, N. Voinov [et al.] // Communications in Computer and Information Science. – 2018. – Vol. 779. – P. 51–63. – DOI 10.1007/978-3-319-71734-0_5. – EDN XXHJWX.
3. Navigate application architecture drift and organizational alignment [Электронный ресурс] // Gitlab. – URL: <https://about.gitlab.com/blog/navigate-application-architecture-drift-and-organizational-alignment/> (Дата обращения: 11.03.2026).
4. Conversion of abstract behavioral scenarios into scenarios applicable for testing / P. Drobintsev, V. Kotlyarov, I. Nikiforov [et al.] // Proceedings of the Institute for System Programming of the RAS. – 2016. – Vol. 28, No. 3. – P. 145–160. – DOI 10.15514/ISPRAS-2016-28(3)-9. – EDN WKFGVF.
5. Анализ архитектурной документации: сравнение C4-диаграмм с каталогами проекта / А. С. Мамонова, В. В. Тронеv, И. А. Селин // Современные технологии в теории и практике программирования: сборник материалов конференции. – СПб., 2025.
6. Сравнение технических документаций сетевой инфраструктуры в форматах PlantUML и C4 / В. В. Тронеv, И. А. Селин // Современные технологии в теории и практике программирования: сборник материалов конференции. – СПб., 2025.
7. Разработка программы для сравнения классовых диаграмм PlantUML и protobuf файлов / Лонишин М.Р., Богданов Н.Р., Селин И.А. // Современные технологии в теории и практике программирования: сборник материалов конференции. – СПб., 2025.
8. Анализ архитектурной документации: сравнение PlantUML-диаграмм с исходными файлами на языке Golang / Э. М. Клеблеева, Д. М. Вайндрах, И. А. Селин // Современные технологии в теории и практике программирования: сборник материалов конференции. – 2025. – EDN XXYNUD. (Дата обращения: 23.02.2025).
9. Drobintsev, P. D. Model oriented approach for industrial software development / P. D. Drobintsev, V. P. Kotlyarov, N. Voinov // Modeling and Analysis of Information Systems. – 2015. – Vol. 22, No. 6. – P. 773–782.
10. Pydantic v2 documentation [Электронный ресурс] // Pydantic. – URL: <https://docs.pydantic.dev/latest/> (Дата обращения: 11.03.2026).
11. DRY (Don't Repeat Yourself) в инженерии данных [Электронный ресурс] // Bi consult. – URL: <https://datafinder.ru/products/dry-dont-repeat-yourself-v-inzhenerii-dannyh>

12. С. Браун. Модель C4 для визуализации архитектуры программного обеспечения [Электронный ресурс] // C4model.com. – URL: <https://c4model.com/> (Дата обращения: 11.03.2026).
13. Построение диаграмм UML с использованием PlantUML [Электронный ресурс] // plantuml.net — URL: https://pdf.plantuml.net/PlantUML_Language_Reference_Guide_ru.pdf (Дата обращения: 11.03.2026).

УДК 004.932.2

Р.С. Кузьмин (3 курс бакалавриата),
П.А. Шагалова, к.т.н., доцент

РАЗРАБОТКА И ОПТИМИЗАЦИЯ АЛГОРИТМА РАЗМЫТИЯ ПО ГАУССУ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА MLIR

Алгоритмы компьютерного зрения широко применяются в системах реального времени, где критически важна высокая скорость обработки изображений. Одной из базовых операций обработки изображений является размытие по Гауссу, используемое для уменьшения шума и выделения деталей.

Обычно такие алгоритмы реализуются с помощью оптимизированных библиотек, например OpenCV [1], однако современные компиляторные технологии, такие как MLIR [2], открывают возможности для автоматической и ручной оптимизации на уровне промежуточного представления.

Целью данной работы являлась реализация алгоритма размытия по Гауссу ядром 5 на 5 на языке C++ с использованием библиотеки OpenCV и его оптимизированной версии с применением фреймворка MLIR [3], а также сравнение производительности полученных реализаций.

На первом этапе была разработана эталонная реализация алгоритма на C++ с использованием OpenCV. Алгоритм включал генерацию нормализованного ядра Гаусса, обработку границ методом отражения пикселей и свертку по трем каналам изображения.

На втором этапе алгоритм был написан на фреймворке MLIR с использованием диалектов `scf` (структурные циклы), `arith` (арифметические операции) и `memref` (работа с памятью). Входное изображение, ядро и выходные изображение представлялись как `memref`. Внутренние циклы свертки реализованы с помощью `scf.for` и включали явную проверку границ через `arith.maxsi/arith.minsi`. Ядро передавалось в качестве отдельного аргумента, что обеспечило гибкость и соответствие исходной логике.

Для генерации исполняемого кода применялись стандартные пассы MLIR: `--convert-scf-to-cf`, `--convert-cf-to-llvm`, `--convert-func-to-llvm`, `--convert-to-llvm` и `--reconcile-unrealized-casts`. Полученный MLIR IR компилировался с помощью `lsc` и линковался драйвером, который обеспечивал загрузку изображения и преобразование данных.

Измерение времени выполнения проводилось на изображении размером 357 на 354 пикселя (3 канала) с использованием `std::chrono`. Каждая реализация запускалась 10 раз. Усредненные результаты показывают, что реализация на MLIR работает на 10-15% быстрее, чем реализация алгоритма на OpenCV.

Полученные результаты показывают, что MLIR-версия алгоритма превосходит по скорости реализацию алгоритма на OpenCV. Это объясняется эффективным низкоуровневым представлением и возможностью применять оптимизации на различных уровнях абстракции.

Таким образом, MLIR является перспективным инструментом для разработки высокопроизводительных алгоритмов компьютерного зрения.

ЛИТЕРАТУРА

1. Bradski, G., Kaehler, A. Learning OpenCV: Computer Vision with the OpenCV Library. – O'Reilly Media, 2008. – 580 p.

2. Lattner, C. et al. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation // 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO). – 2021. – P. 2–14.
3. MLIR Documentation. Dialects. – <https://mlir.llvm.org/docs/Dialects/>

УДК 004.4

К.Л. Лузин (2 курс бакалавриата),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА ПАРСЕРА ДИРЕКТОРИЙ ДЛЯ АНАЛИЗА АРХИТЕКТУРНОЙ ДОКУМЕНТАЦИИ С ПРИМЕНЕНИЕМ ПРЕПРОЦЕССИНГА

Для разработки и сопровождения крупных программных проектов требуется систематическое ведение архитектурной документации [1-2]. Такая документация может храниться в различных форматах, включая C4-диаграммы и структуры директорий [3]. Использование нескольких форматов обусловлено разнообразием задач документирования, а также применяемыми инструментами разработки. С увеличением объёма документации и количества источников архитектурных описаний возникает проблема поддержания согласованности между ними. Изменения, внесённые в одном представлении архитектуры, могут не отражаться в других источниках, что приводит к рассинхронизации документации и повышает риск возникновения архитектурных несоответствий.

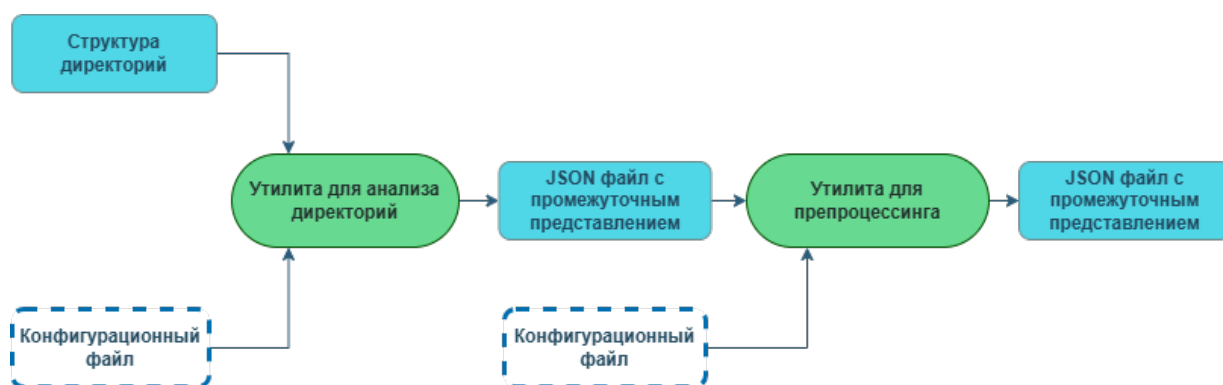


Рисунок 1 – Архитектурное решение

Для решения данной проблемы предлагается подход, обеспечивающий унифицированный анализ дерева директорий с обходом ограничений в названиях элементов файловой системы. Архитектура предлагаемого решения представлена на рисунке 1.

В рамках предложенного решения директории обрабатываются специализированным парсером. Разработанная утилита извлекает структурную информацию из каталогов в виде дерева. После идет преобразование в унифицированное промежуточное представление, сохраняются названия директорий и их связи. Результирующий объект записывается в json файл для дальнейшего сравнения с результатами утилит обработки других видов хранения документации.

Названия директорий могут частично отличаться от наименований элементов диаграмм и не должны содержать специальные символы \ / : * " < > |. В связи с указанными ограничениями после формирования промежуточного представления может потребоваться этап предварительной обработки данных, препроцессинг. Данный этап включает преобразование и подготовку информации для последующего сравнения. Для обеспечения гибкости утилита препроцессинга использует механизм регулярных выражений [4-5], позволяющий формализовать правила преобразования и нормализации имен элементов.

Парсер и препроцессор используют конфигурационные файлы для задания параметров обработки. Применение конфигурационных файлов повышает удобство использования

разработанных инструментов, поскольку анализ архитектурной документации часто выполняется повторно при незначительных изменениях проекта. В подобных случаях большинство входных параметров остаётся неизменным, что делает целесообразным сохранение ранее использованных настроек запуска.

Для реализации утилит выбран язык Python. Это обусловлено простотой и возможностями его стандартных библиотек. При помощи модуля `os` решаются задачи навигации по файловой системе и сбора сведений о структуре каталогов. Модуль `re` для регулярных выражений задействуется в операциях с текстовыми данными, включая модификацию и поиск подстрок.

Таким образом, предложенное решение позволяет повысить уровень согласованности архитектурной документации и снизить вероятность возникновения архитектурных несоответствий при сопровождении и развитии проектов.

ЛИТЕРАТУРА

1. Современный подход к программной архитектуре: сложные компромиссы / Н. Форд, М. Ричардс, П. Садаладж, Ж. Дехгани. — Санкт-Петербург: Питер, 2023. — 480 с. — ISBN 978-5-4461 2024-6.
2. Белугин, Д. С. Анализ архитектурных подходов к разработке веб-приложений / Д. С. Белугин // Наукосфера. — 2023. — № 5-2. — С. 198-202. — EDN MBHPDG.
3. А. С. Мамонова, Анализ архитектурной документации сравнение C4-диаграмм с каталогами проекта / А. С. Мамонова, В. В. Тронеv, И. А. Селин // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. — Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. — С. 325-327. — EDN DGBUFH.
4. Козлов, С. В. Применение регулярных выражений для обработки текстовых данных / С. В. Козлов, А. В. Светлаков // International Journal of Open Information Technologies. — 2022. — Т. 10, № 9. — С. 82-89. — EDN ZOSEER.
5. Багаев, А. В. Регулярные выражения в языке программирования python 3.5 / А. В. Багаев // Информационные технологии в образовании "ИТО-Саратов-2016" : Материалы VIII Международной научно-практической конференции, Саратов, 02–03 ноября 2016 года. — Саратов: ООО "Издательский центр "Наука", 2016. — С. 15-17. — EDN XGKBQX.

УДК 004.4

В.В. Малая (4 курс бакалавриата),
Б.М. Медведев, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ СЖАТИЯ ЗАГОЛОВКОВ ПАКЕТОВ ДАННЫХ В БАЗОВОЙ СТАНЦИИ 4G

Наиболее используемое поколение стандартов сотовой связи в мире – 4G или LTE-A (Long-Term Evolution Advanced). Этот стандарт обеспечивает передачу данных на скоростях до 100 Мбит/с и более. Кроме того, с приходом LTE появилась новая технология передачи голоса под названием VoLTE (Voice over LTE) [1], которая позволяет осуществлять голосовые вызовы «поверх» IP протокола, что обеспечивает уменьшение стоимости трафика.

Голосовой трафик требует минимальных задержек при передаче данных, поэтому пакеты данных формируются часто, из-за чего «полезных» пользовательских данных в них мало по сравнению со служебными заголовками IP/UDP/RTP протоколов. На 20-30 байт пользовательских данных приходится до 60 байт служебных [2]. При этом служебные заголовки содержат избыточную информацию, повторяющуюся во всех заголовках на протяжении сеанса передачи. Для сокращения избыточности в передаваемых сообщениях был разработан и стандартизирован алгоритм RoHC (Robust header compression) [3],[4]. Он позволяет эффективно сжимать заголовки пакетов данных, в лучшем случае превращая 60 байт заголовка в 1 байт. Сжатие основано на сохранении контекстов между передающей

сущностью (компрессор) и принимающей (декомпрессор). В начале сеанса передачи данных компрессор работает в режиме инициализации (IR – initialization and refresh) [5] и сообщает декомпрессору контекст, передавая несжатые заголовки. После чего компрессор начинает работать в одном из режимов отправки изменений, т.е. передаются исключительно значения изменяемых полей, сжатые особым образом. Спецификация содержит 3 режима работы RoHC, однако не приведены критерии переходов между этими режимами, что создает дополнительные сложности в процессе реализации.

Целью работы является разработка программных средств сжатия заголовков пакетов VoLTE с использованием алгоритма RoHC для базовой станции сотовой сети.

Задачи работы:

1. Разработка алгоритмов переключения режимов компрессора и декомпрессора.
2. Разработка архитектуры программных средств сжатия заголовков.
3. Разработка программных средства сжатия, включая алгоритмы переключения между состояниями и режимами, сохранения контекста, сжатия изменяемых полей.
4. Оценка эффективности сжатия полученной программной реализации RoHC.

В качестве алгоритмов переключения между режимами было рассмотрено несколько вариантов. Суть первого заключается в подборе условий переключения в зависимости от качества связи (учитывается абсолютное количество потерь). Другой алгоритм представляет собой усовершенствованную версию первого, а именно учитывает не только абсолютное количество k потерь из n последних пакетов, но и относительное количество за весь сеанс передачи. Третий алгоритм предлагает выбирать условия переключения в зависимости от стабильности потока данных (количество изменений в контексте). Наиболее эффективным на практике оказался второй вариант.

В рамках работы мною были также разработаны и протестированы программы, осуществляющие сжатие и обработку сжатых пакетов по алгоритмам RoHC. Структура компонентов изображена на рисунке 1. В состав программной реализации компрессора входит ряд сущностей, отвечающих за прием входящих пакетов и создание/поиск подходящего контекста (Channel Compressor), анализ входящего потока и сравнение содержимого полей с имеющимся контекстом (Flow Matcher), подбор подходящего типа сжатого пакета с наименьшей длиной и отправку сжатого пакета (Profile Encoder), изменение внутренних состояний (Flow Compressor) и др. Процессы со стороны декомпрессора похожи, однако выполняются в обратном порядке.

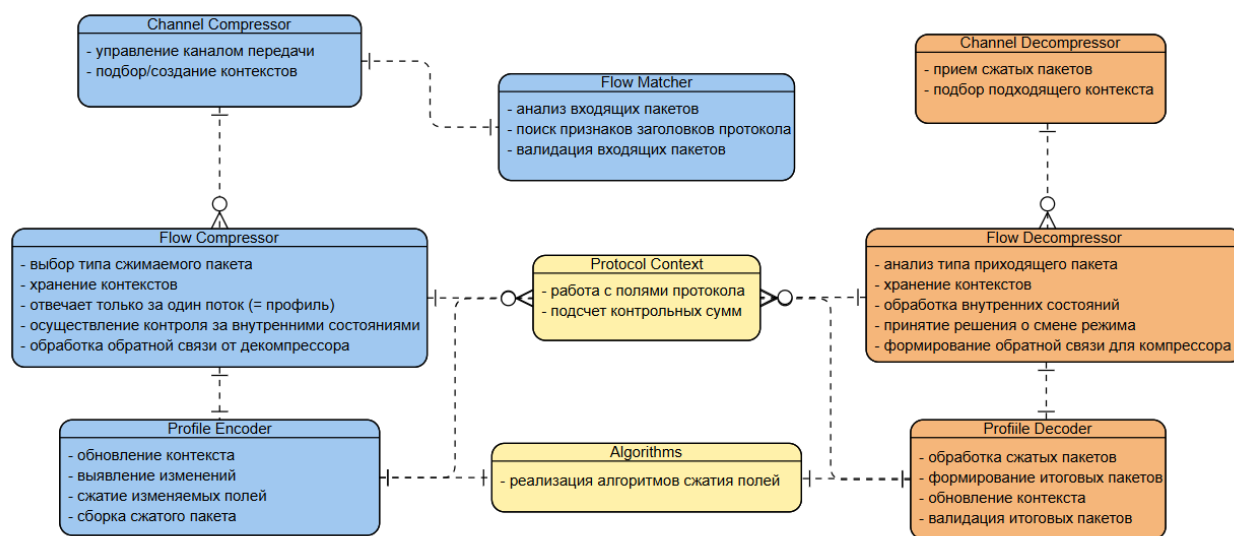


Рисунок 1 – Структура программной реализации средств сжатия заголовков

В тестовых сценариях для оценки производительности передающая и принимающая сущности, обмениваются пакетами данных. Входящий поток пакетов эмулирует трафик, по своему содержанию близкий к передаваемому в реальных системах. В тестовом потоке

изменяемые поля меняются предсказуемо (RTP SN и IP ID на +1, Timestamp на фиксированное значение, соответствующее интервалу отсчета или на значение, кратное ему, эмулируя тишину), остальные поля, помимо контрольных сумм, не изменяются. В таблице 1 представлены результаты оценки эффективности сжатия заголовков для различных потоков данных и используемых профилей RoHC. Оценка сжатия включает усредненные значения по всему сеансу передачи, включая передачу несжатых пакетов при инициализации.

Таблица 1 – Результаты оценки эффективности сжатия заголовков программной реализации RoHC

Профиль	1 (сжатие IP/UDP/RTP)		2 (сжатие IP/UDP)	
Версия IP	IPv4	IPv6	IPv4	IPv6
% сжатия заголовка	92	94,6	85,6	91,5
% сжатия всего пакета	50,5	61,3	33,2	47,7

В результате работы была разработана программная реализация сжатия заголовков, используемая для оптимизации передачи данных и увеличения пропускной способности базовой станции LTE. Алгоритм показал хорошие результаты при использовании потока данных, близкого к реальному (до 94,6% сжатия заголовка).

Разработанное программное обеспечение стало частью отечественной базовой станции, разрабатываемой компанией YADRO и будет использовано для VoLTE звонков.

ЛИТЕРАТУРА

1. Jasson Casey, Srivatsan Rajagopalan, Muxi Yan, Graham Booker, Alex Sprintson, and Walt Magnussen, “Supporting Voice over LTE: Solutions, Architectures, and Protocols”, Computer Communications and Networks (ICCCN), 2013
2. H. Wang and K.G. Seah, “An Analytical Model for the ROHC RTP Profile”, IEEE Communications Society, 2004
3. RFC 3095. RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed [Электронный ресурс] – <https://datatracker.ietf.org/doc/html/rfc3095>
4. RFC 4815. RObust Header Compression (ROHC): Corrections and Clarifications to RFC 3095 [Электронный ресурс] – <https://datatracker.ietf.org/doc/html/rfc4815>
5. Xiuliang Chen, Fuliang Guo, Ping Dang, Libing Wu “A Survey of ROHC Header Compression Schemes”, 2nd International Conference on Computer Science and Network Technology, 2012

УДК 004.75

Е.А. Милорадов, С.И. Чувашевский (2 курс бакалавриата),
О.В. Александрова, ст. преподаватель

ДОБАВЛЕНИЕ НОВЫХ ТИПОВ ХРАНИЛИЩ В СЕРВЕР-ЭМУЛЯТОР API-ИНТЕРФЕЙСА SWORDFISH ДЛЯ ТЕСТИРОВАНИЯ CSI-ИНСТРУМЕНТА KUBERNETES

Целью создания сервера-эмулятора является формирование реалистичного полигона для испытаний, поддерживающего множество протоколов персистентного хранения. Поскольку взаимодействие оркестратора Kubernetes с внешними сторадж-системами в современных условиях опосредовано драйверами Container Storage Interface (CSI), возникает потребность в инструменте, способном симулировать работу гетерогенных сред хранения данных. Такой подход позволяет проводить комплексное тестирование CSI-драйверов Kubernetes, исключая необходимость применения дорогостоящих аппаратных дисковых массивов на этапе валидации и разработки. Как показано на рисунке 1, сервер-эмулятор Swordfish API предоставляет набор ресурсов Volume с различными протоколами доступа (iSCSI, LBS, NFS, NVMe-oF), что позволяет имитировать работу реальной системы хранения данных при тестировании CSI-драйвера в среде Kubernetes.

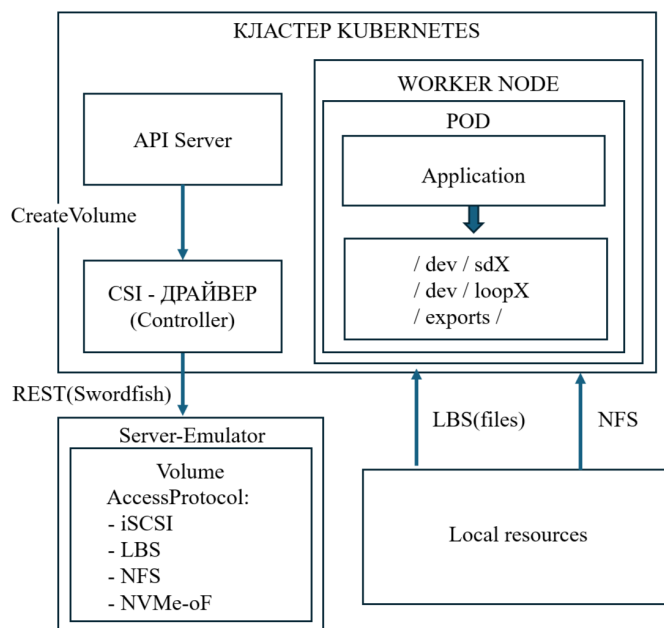


Рисунок 1 – Архитектура взаимодействия компонентов при тестировании CSI-драйверов

При этом существующий официальный эмулятор Swordfish, предоставляемый организацией SNIA, ориентирован преимущественно на демонстрацию соответствия REST API спецификации и не обеспечивает полноценной поддержки сценариев тестирования CSI-драйверов, включая эмуляцию различных типов блочного доступа.

Для восполнения данного ограничения в качестве начальных типов были выбраны: iSCSI — протокол блочного доступа к данным по сети и LBS — локальное блочное хранилище. Управление этими ресурсами осуществляется через API-интерфейс Swordfish — современный стандарт управления хранилищами, построенный на базе Redfish.

Целью работы является добавление поддержки протоколов iSCSI и LBS в существующий сервер-эмулятор Swordfish для расширения возможностей тестирования CSI-драйверов Kubernetes.

Для её выполнения были поставлены следующие задачи:

- расширение модели данных эмулятора;
- реализация создания логики iSCSI-тома и LBS-тома;
- интеграция с CSI-драйвером.

Для добавления новых типов Storage были внедрены:

- поле AccessProtocol в модели Volume с возможными значениями "iSCSI" и "Local",
- атрибуты TargetIqn, TargetPortal, Lun для iSCSI
- DevicePath для LBS.

В рамках реализации iSCSI-протокола для каждого тома генерируется уникальное имя iSCSI-квалификатора (TargetIqn) по шаблону, отражающему пространство имён разработанного эмулятора (например, iqn.2025-11.com.example:swordfish:vol-{id}). Параметры сетевого доступа (TargetPortal) задаются номинально и не предполагают установления фактического соединения, поскольку на текущем этапе данные атрибуты используются исключительно для формирования контекста тома в CSI-драйвере. Дополнительно специфицируется логический номер устройства (Lun).

Для локального блочного хранилища используется параметр DevicePath, который указывает на виртуальный путь вида: “/tmp/swordfish-local-vol-{uuid}.img”, по которому создаётся sparse-файл требуемого размера. В случае выполнения на Linux-системе и наличии необходимых привилегий данный файл привязывается к loop-устройству, после чего в DevicePath сохраняется путь к созданному блочному устройству, например: “/dev/loopX”.

Дальнейшее развитие сервера-эмулятора предполагает расширение перечня поддерживаемых протоколов с учётом современных тенденций в области

высокопроизводительных систем хранения данных. Приоритетным направлением является реализация поддержки NVMe-over-Fabrics (NVMe-oF) — протокола, обеспечивающего удалённый доступ к твердотельным накопителям с минимальными задержками и высокой степенью параллелизма за счёт использования моделей взаимодействия NVMe поверх сетевых транспортных протоколов (TCP, RDMA). Внедрение NVMe-oF в эмулятор позволит моделировать поведение современных All-Flash массивов и проводить тестирование CSI-драйверов, ориентированных на работу с высоконагруженными средами виртуализации и баз данных, без необходимости использования дорогостоящего оборудования, поддерживающего данный интерфейс.

ЛИТЕРАТУРА

1. Архитектура CSI-драйвера для Kubernetes с поддержкой спецификации Swordfish / Г. Дюльдин, И. Ф. Райкевич, М. С. Соломонов // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 309-311. – EDN HVLEZZ.
2. Kubernetes CSI Documentation // Kubernetes CSI. URL: <https://kubernetes-csi.github.io/docs/> (дата обращения: 20.02.2026).
3. Различные CSI драйверы. // Kubernetes CSI. URL: <https://kubernetescsi.github.io/docs/drivers.html> (дата обращения: 20.02.2026).
4. SNIA Swordfish. URL: <https://www.snia.org/forums/smi/swordfish> (дата обращения: 20.02.2026)
5. Samsung Electronics. nvmFabTest: NVMe-oF Compliance Test Suite. GitHub, 2024. URL: <https://github.com/SamsungDS/nvmFabTest> (дата обращения: 20.02.2026)

УДК 004.4

В.А. Ромашов (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент,
С.М. Устинов, д.т.н., профессор

КОНВЕЙЕР ДАННЫХ: ПАРСИНГ, ИНДЕКСАЦИЯ И СИНХРОНИЗАЦИЯ В КОРПОРАТИВНОЙ ИНЖЕНЕРНОЙ СРЕДЕ

Современные инженерные порталы и базы знаний, такие как Backstage, Confluence и Minerva играют ключевую роль в управлении документацией, задачами и проектами в ИТ-инфраструктуре [1]. Они агрегируют (автоматически или вручную) в себе информацию от различных источников и представляют их в унифицированном интерфейсе. Основной проблемой существующих решений-агрегаторов является временная задержка между моментом изменения данных в источнике и их отображением в клиентском интерфейсе инженерного портала-агрегатора. Эта задержка возникает из-за неэффективных механизмов синхронизации, чаще всего основанных на периодическом опросе источника, который не учитывает интенсивность изменений. Использование устаревших данных снижает скорость разработки прикладных систем и уменьшает лояльность пользователей к инженерным порталам. Таким образом, разработка алгоритма, обеспечивающего низкую задержку и высокую надежность обновления данных между источниками и унифицированным инженерным порталом, является актуальной задачей.

Целью работы является уменьшение задержки между изменением и предоставлением данных о конфигурациях сущностей в инженерном портале, за счет высокопроизводительного алгоритма обновления, индексации и парсинга данных [2], отличительной особенностью которого является использование системы контроля версий для отказоустойчивого хранения конфигураций. Для обеспечения качества алгоритма в работе используется модульное и интеграционное тестирование [3].

Предлагаемый алгоритм изменения данных представлен на рисунке 1.

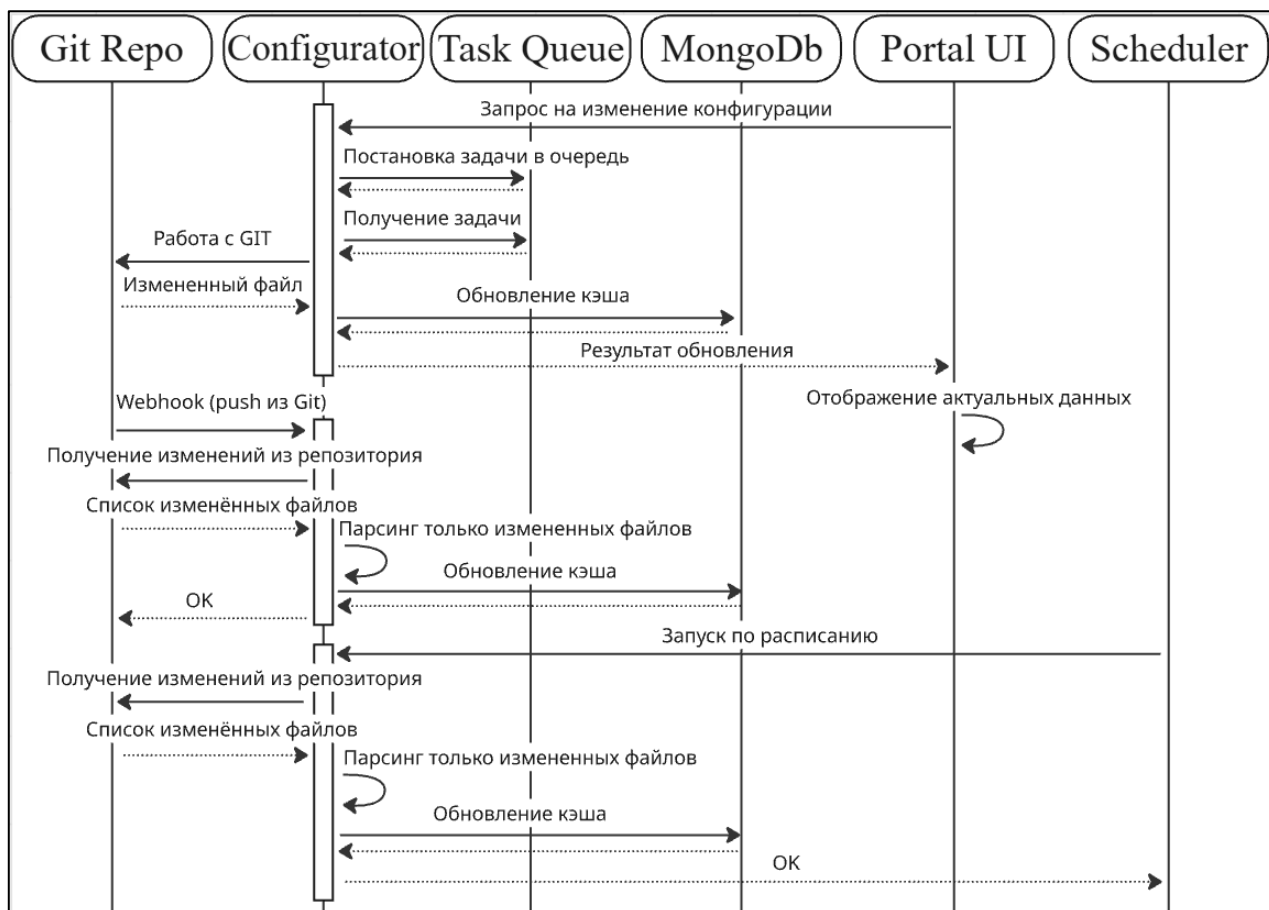


Рисунок 1 – Алгоритм обновления, индексации и парсинга данных

Решение комбинирует два механизма: первичный событийно-ориентированный канал (вебхуки, изменения через веб-интерфейс портала) и фоновый процесс периодической синхронизации, выполняющий роль «страховочной сети» [4].

Алгоритм работы описывается ниже.

1. Формирование запроса на изменение данных с WEB-интерфейса портала. Пользователь ожидает выполнение запроса и получение новых данных. Имитируем работу классического сервиса с реляционной базой данных.

2. Обработка запроса (нормализация, выбор сервиса конфигураций) на стороне сервера портала.

3. Постановка задачи в очередь для выполнения изменений (выполняется транзакционно).

4. Конфигуратор обрабатывает задачи в очереди последовательно, это гарантирует согласованность данных. В данном месте мы имитируем работу с Git [5].

5. Сервис возвращает измененный файл конфигурации данных.

6. Обновляется внутренний кэш данных в сервисе MongoDB. Это позволяет иметь актуальные данные здесь и сейчас [6].

7. Портал получает новые данные и отображает их на клиенте.

Также в алгоритм работы встроены фоновые процессы обновления состояния сервиса.

1. При первом запуске сервиса, выкачиваем репозиторий git и сохраняем его локально, для доступа к файлам и возможности сохранять изменения.

2. При пуше изменений в Git-репозиторий настроенный вебхук отправляет POST-запрос в эндпоинт `/webhook/git`. Сервис выполняет `git fetch` и `git diff`, определяя список измененных файлов конфигурации (JSON). Парсер обрабатывает только изменившиеся данные.

3. По расписанию (Cron) запускается процесс, проверяющий состояние всех зарегистрированных репозиториев. Это гарантирует согласованность данных даже при сбоях сети.

Благодаря такому подходу сокращается средняя задержка обновления данных по сравнению с традиционным периодическим опросом, снижая нагрузку на инфраструктуру и обеспечивая высокую доступность информации. Алгоритм можно использовать для изменения данных любого стороннего сервиса (источника), не опираясь на архитектуру инженерного портала [7].

Таким образом, в работе был представлен гибридный алгоритм синхронизации, индексации и парсинга данных, использующий Git как сервис отказоустойчивого хранения конфигураций данных. Предложенное решение для инженерного портала должно ускорить обновление и актуализацию данных, предоставляемых инженерам.

ЛИТЕРАТУРА

1. Ключкова, Л. И. Интеграция инструмента Grafana в инженерный портал компании / Л. И. Ключкова, В. А. Ромашов, И. В. Никифоров // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 24–25 апреля 2024 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. – С. 278-280. – EDN BNXLFT.
2. Калытюк, И. С. Обзор систем сбора данных методом парсинга / И. С. Калытюк // Наука. Технологии. Инновации: Сборник научных трудов XVI Всероссийской научной конференции молодых ученых. В 11-и частях, Новосибирск, 05–08 декабря 2022 года / Под редакцией А.С. Казьминой. Том Часть 1. – Новосибирск: Новосибирский государственный технический университет, 2022. – С. 83-86. – EDN EHZGUT.
3. Инкрементальный подход к технологии создания тестов для индустриальных проектов / П. Д. Дробинцев, В. П. Котляров, И. В. Никифоров, А. А. Летичевский // Моделирование и анализ информационных систем. – 2014. – Т. 21, № 6. – С. 144-154. – EDN TCCAKB.
4. Синхронизация и обновление PWA в фоновом режиме / Microsoft Learn. — 2025. — URL: learn.microsoft.com/ru-ru/microsoft-edge/progressive-web-apps/how-to/background-syncs (дата обращения: 14.03.2026).
5. Шакон Скотт. Pro Git -- М.: Apress, 2014. -- 289 с.
6. Резниченко, О.С. Алгоритмы и инструменты обработки данных о публикациях научных организаций, размещенных на портале eLIBRARY.RU / О.С. Резниченко // Экономика. Информатика. – 2025. – Т. 52, № 1. – С. 181-193. – DOI: 10.52575/2687-0932-2025-52-1-181-193. – EDN: FVUGLX.
7. Фаулер, М. Шаблоны корпоративных приложений / М. Фаулер. – М.: Вильямс, 2015. – 544 с.

УДК 004.7

А.Д. Николаев, Я.М. Парфенов (2 курс бакалавриата),
О.В. Александрова, ст. преподаватель

АРХИТЕКТУРНЫЕ РЕШЕНИЯ ИНТЕГРАЦИИ РАЗНОРОДНЫХ СИСТЕМ ХРАНЕНИЯ ДАННЫХ В CSI-ДРАЙВЕРАХ

Активное внедрение контейнеризации и использование Kubernetes для оркестрации приложений в разнородных системах хранения данных (СХД) выдвигает требование к использованию гибкого инструмента работы с ними. В рамках проводимых исследований разрабатывается CSI-драйвер, который обеспечивает взаимодействие с Swordfish API-эмулятором – системой тестирования СХД.

Текущая реализация CSI-драйвера ориентирована на работу с кластерами СХД, удовлетворяющими спецификации Swordfish [1], что приводит к жесткой привязке логики драйвера к конкретной модели API. Данный подход ограничивает круг поддерживаемых систем хранения и усложняет подключение СХД, не соответствующих данной спецификации, поскольку требует внесения изменений в основную логику драйвера. В связи с этим

актуальной является разработка архитектурного решения, обеспечивающего отделение логики CSI-драйвера от особенностей конкретных API систем хранения данных.

Целью работы является разработка архитектурного решения CSI-драйвера [2], обеспечивающего интеграцию разнородных систем хранения данных за счет введения унифицированного API и адаптерного слоя, позволяющих исключить зависимость логики драйвера от конкретной спецификации СХД [3]. Для достижения поставленной цели в работе предложено архитектурное разделение логики CSI-драйвера и механизмов взаимодействия с СХД, а также установлены следующие задачи: определить единый интерфейс для управления СХД, разработать архитектуру компонентов и адаптеров, реализующих преобразование унифицированных вызовов в API конкретных СХД, и провести интеграцию с сервером-эмулятором.

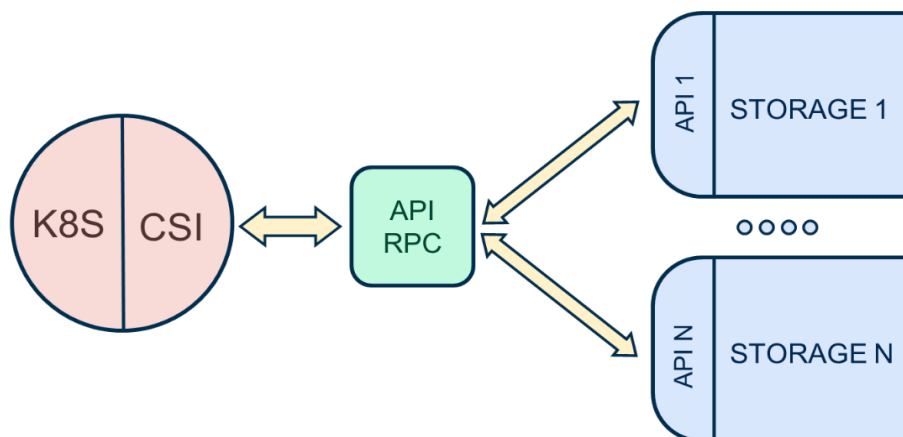


Рисунок 1 – Предлагаемая архитектура проекта

На рисунке 1 показана возможная реализация данного решения, которая представляет собой использование абстрактного слоя для интеграции, который скрывает от драйвера взаимодействие с API СХД и предоставляет CSI-драйверу лишь единый интерфейс для выполнения трансляции и маршрутизации команд управления ресурсами. Это позволит описать логику работы CSI-драйвера один раз и добавлять новые СХД посредством обновления API. Такое архитектурное разделение упрощает расширение системы и снижает трудоемкость поддержки CSI-драйвера при подключении разнородных СХД.

Новый CSI-драйвер разрабатывается с учётом принципа строгого разделения уровней ответственности и устранением монолитной связности между компонентами системы. В отличие от текущей реализации, где логика взаимодействия с конкретным API СХД встроена непосредственно в код используемого CSI-драйвера, разработанный подход предполагает использование промежуточного абстрактного слоя. Драйвер реализует исключительно стандартный контракт Container Storage Interface и оперирует унифицированными операциями управления ресурсами: созданием и удалением томов, управлением распределением емкостей подключенных носителей, размещением томов на узлах, работой с копиями и получением состояния ресурсов [4].

Все вызовы, инициируемые Kubernetes [5], обрабатываются драйвером на уровне CSI-методов, после чего преобразуются в обращения к сервисному интерфейсу (узлу API/RPC), не зависящему от конкретной спецификации API. Таким образом, драйвер не содержит кода, ориентированного на Swordfish или другую конкретную модель взаимодействия СХД и CSI-драйвера. Он работает с абстрактными сущностями хранения, а преобразование этих сущностей в реальные вызовы API выполняется за пределами его логики.

Блок API/RPC выполняет функцию промежуточного слоя между CSI-драйвером и конкретными СХД. Он принимает унифицированные запросы от драйвера, выполняет их валидацию, определяет целевую систему хранения и инициирует выполнение определенной операции, взаимодействуя с API СХД. Внутри компонента определяется единая модель

доменных операций. Каждая операция описывается входными и выходными структурами, не зависящими от конкретной реализации СХД. Далее запрос направляется к СХД.

Таким образом, предложенная архитектура обеспечивает изоляцию различных протоколов и форматов данных систем хранения внутри адаптеров, что позволяет расширять функциональность CSI-драйвера без внесения изменений в его код. Использование унифицированной модели доменных операций и адаптерного взаимодействия повышает тестируемость решения за счёт возможности применения имитационных моделей систем хранения. Это ускоряет процесс разработки, снижает риски регрессий и повышает устойчивость архитектуры CSI-драйвера к изменениям стандартов и версий API [6].

ЛИТЕРАТУРА

1. Архитектура CSI-драйвера для Kubernetes с поддержкой спецификации Swordfish / Г. Дюльдин, И. Ф. Райкевич, М. С. Соломонов // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 309-311. – EDN HVLEZZ.
2. Kubernetes CSI Documentation // Kubernetes CSI. URL: <https://kubernetes-csi.github.io/docs/> (дата обращения: 18.02.2026).
3. Различные CSI драйверы. // Kubernetes CSI. URL: <https://kubernetescsi.github.io/docs/drivers.html> (дата обращения: 18.02.2026).
4. Core Kubernetes. / Vyas J., Love C. // – Shelter Island: Manning Publications, 2022. – 450 p. – ISBN 978-1-61729-703-8
5. Документация Kubernetes // Kubernetes. URL: <https://kubernetes.io/> (дата обращения: 20.02.2026).
6. Приемы объектно-ориентированного проектирования. Паттерны проектирования. /Э. Гамма, [и др.] // СПб.: Питер, 2001. - 368 с.
7. Интеграция CSI драйвера YADRO в OrionSoft. URL: <https://wiki.orionsoft.ru/knowledgebase/nova/storage/csi-yadro-integration> (дата обращения: 21.02.2026)

УДК 004.42

Н.С. Онуфриенко (2 курс магистратуры),
Н.В. Богач, к.т.н., доцент

РАЗРАБОТКА ВСТРАИВАЕМОГО РЕДАКТОРА ЛОГИЧЕСКИХ СХЕМ

На текущий момент решения для редактирования логических схем представлены либо универсальными векторными редакторами (Visio, Drawio [1]), либо тяжелыми монолитными средами проектирования (KiCad [2], Altium Designer). Данные решения трудно интегрировать в другие проекты либо из-за невозможности получить информацию о построенной схеме, либо из-за их избыточности и узкой специализации.

В рамках данной работы предлагается простой, легковесный и расширяемый редактор логических схем, реализованный на языке программирования Rust. Данный редактор может быть интегрирован в различные среды проектирования или веб-приложения, а также быть использован для создания иллюстраций в технической документации. Данный редактор позволяет как получить сериализованную информацию о схеме в формате JSON, так и её графическое представление в формате SVG.

Т.к. основной целью при разработке было сделать легкое встраиваемое решение, в качестве GUI фреймворка был выбран «egui» [3], отличительными особенностями которого являются легковесность, кроссплатформенность и простота встраивания. Для минимизации размера бинарного файла было принято решение реализовывать процедурную графику для

отрисовки компонентов схемы и нестандартных элементов интерфейса. Пример графического интерфейса программы приведен на рисунке 1.

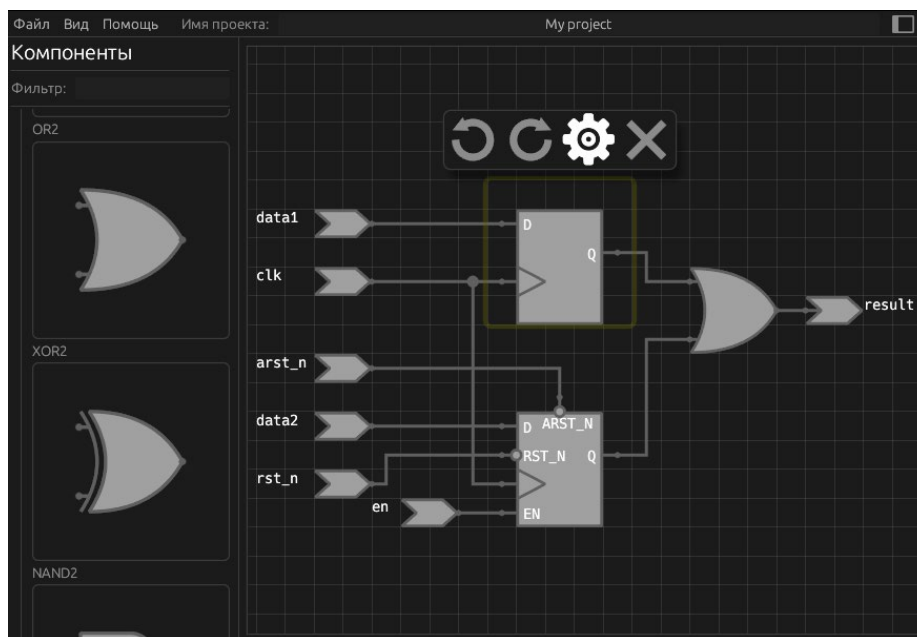


Рисунок 1 – Интерфейс программы

Для каждого компонента описываются функции, определяющие набор полигонов в виде списка контурных точек, положения точек соединения и их привязку к сетке, а также ломаные линии и текстовые метки. Данные функции вызываются как при отрисовке схемы, так и при её экспорте в формат SVG. При отрисовке для тесселяции полигонов используется библиотека «Lyon» [4]. Сетка («Mesh»), полученная в результате тесселяции, кэшируется в памяти и отображается средствами фреймворка «egui». При экспорте в формат SVG для каждого компонента схемы по заданным функциям генерируется соответствующий набор тегов. Такой подход позволил существенно упростить внедрение новых компонентов и избежать дублирования кода. В дальнейшем предполагается внедрить возможность загрузки пользовательских примитивов из json-файла с заранее заданной геометрией.

Также у разных классов компонентов есть свои конфигурируемые параметры: так, для примитивов логических гейтов «AND», «OR» и т. д. существует один численный параметр, задающий количество входов. По этому параметра вычисляется размер компонента и его геометрия. Для более сложных компонентов, к примеру, описывающих D-триггеры, задается наличие портов включения, синхронного и асинхронного сброса, а также их полярность. Пример процесса конфигурации компонентов схемы приведен на рисунке 2.

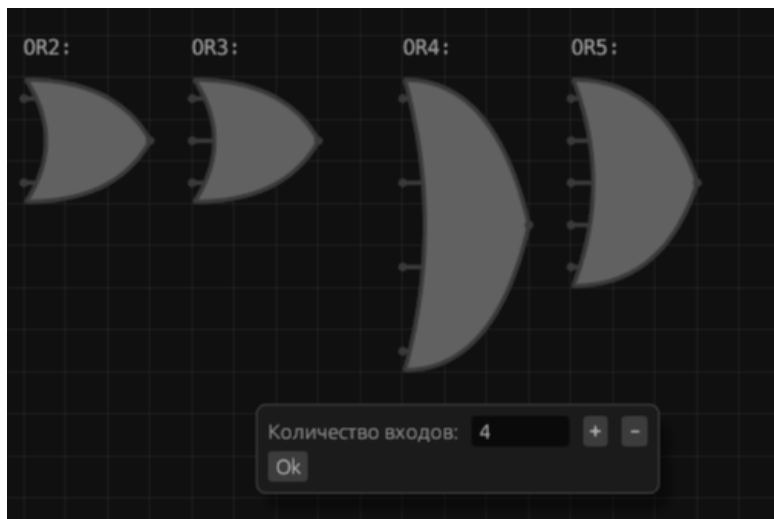


Рисунок 2 – Процесс конфигурации компонентов

Работа разработанного приложения была протестирована на таких платформах, как «WASM» в браузере Firefox и в браузерах на базе движка Chromium, «Windows» на архитектуре «x86-64», Linux на архитектурах «x86-64» и «riscv64gc». Исходный код программы был опубликован в GitHub-репозитории проекта [5] под открытой лицензией «MIT». Процесс сборки под все вышеперечисленные платформы был автоматизирован в GitHub Actions. Также на GitHub Pages была развернута демонстрационная страница с веб-версией приложения. В дальнейшем возможен перенос разработанной программы на мобильные платформы.

ЛИТЕРАТУРА

1. Редактор диаграмм Drawio [Электронный ресурс]. Режим доступа: <https://www.drawio.com/> (дата обращения: 26.02.2025).
2. Среда проектирования KiCad. [Электронный ресурс] Режим доступа: <https://www.kicad.org/> (дата обращения: 26.02.2025).
3. GUI фреймворк «egui». [Электронный ресурс] Режим доступа: <https://github.com/emilk/egui> (дата обращения: 26.02.2025).
4. Библиотека «Lyon». [Электронный ресурс] Режим доступа: <https://github.com/nical/lyon> (дата обращения: 26.02.2025).
5. Репозиторий проекта [Электронный ресурс]. Режим доступа: <https://github.com/nikonufrienko/editor> (дата обращения: 04.02.2025).

УДК 004.42

И.М. Савинцев, К.П. Алымова (2 курс бакалавриата),
В.А. Касилов, ассистент

ПРОЕКТИРОВАНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОЙ ОЦЕНКИ СТУДЕНЧЕСКИХ РАБОТ ИЗ УДАЛЕННОГО РЕПОЗИТОРИЯ

Современное обучение дисциплинам из области компьютерных наук и программной инженерии требует внедрения профессиональных инструментов разработки. Использование этих инструментов не только упрощает и автоматизирует классические процессы обучения, но и знакомит студентов с реальной практикой, с которой им предстоит столкнуться в будущем. Вместо составления и проверки формальных отчетов, предлагается делать упор на развитие практических навыков у студентов. Избыточная отчетность нередко приводит к тому, что студенты прибегают к помощи ИИ-инструментов для генерации текстов, что сводит на нет образовательный эффект.

Эффективнее, чем требование формальных отчетов, организовать учебный процесс с помощью инструментов, которые непосредственно используются в профессиональной практике. Например, использование единого соглашения об оформлении кода подкрепляется применением статических анализаторов, что приучает студентов к дисциплине написания кода. Применение систем контроля версий и удаленных репозиторий развивает навык последовательного решения задачи. Автоматизация с помощью CI/CD инфраструктуры или покрытие кода тестами позволяет проверять работу на соответствие функциональным требованиям. Внедрение таких инструментов открывает возможности и для автоматизации сопутствующих задач, в частности, для создания механизма выставления оценок. Это позволяет исключить ручной сбор и перенос результатов, снижая нагрузку на преподавателя.

Таким образом, *цель* данной работы – исключить ручной сбор данных и расчёт оценок при проверке работ, автоматизировав эти процессы.

Важно отметить: внедрение профессиональных практик и автоматизация оценивания не заменяют содержательную проверку знаний. Устные собеседования, контрольные и защита лабораторных работ остаются неотъемлемой частью учебного процесса. Автоматизация

рутинных операций освобождает время преподавателя для такого взаимодействия со студентами.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработать модуль для сбора информации о Pull Request'ах через API платформы (GitHub, GitLab и др.) с возможностью извлечения необходимых метаданных.
2. Реализовать механизм применения заданных правил проверки и расчета итоговой оценки для каждого студента.
3. Обеспечить выгрузку результатов в удобный для дальнейшего использования формат и публикацию по постоянной ссылке для последующего импорта в электронные таблицы.

Для решения поставленной задачи была разработана система (рисунок 1) на языке Go [1]. Архитектура реализована с расчетом на расширяемость: взаимодействие с Git-хостингом вынесено в интерфейс, что позволяет добавить поддержку GitLab или Bitbucket. В текущей версии используется GitHub GraphQL API [2]. Запрос формируется с указанием владельца и имени репозитория, ответ обрабатывается JQ-фильтром [3], извлекая для каждого Pull Request'a набор метаданных: автор, заголовок (идентификатор работы), временные метки создания и слияния, историю меток (labels). Выполняется расчет итоговых оценок на основе заданных преподавателем правил. Результаты публикуются в GitHub Gist (или аналогичных сервисах) с возможностью получения прямой ссылки для импорта в таблицы.

Пользовательское взаимодействие реализовано через Telegram Bot API [4, 5], что снижает трудоемкость взаимодействия по сравнению с работой через командную строку.

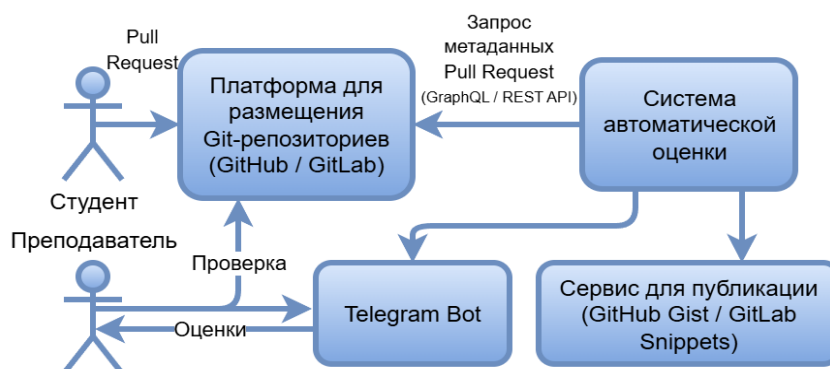


Рисунок 1 – Архитектура системы

Внедрение разработанной утилиты в учебный процесс позволило кардинально снизить трудоемкость выставления оценок, что особенно ощутимо на большом потоке студентов. По оценкам, для четырех групп, выполняющих несколько лабораторных работ за семестр, ручной перенос оценок занимал до 20 часов, теперь занимает 10 секунд.

ЛИТЕРАТУРА

1. Введение в язык программирования Go : учебное пособие / И. В. Никифоров, А. А. Андрианов, О. В. Александрова [и др.]. – Санкт-Петербург : Санкт-Петербургский политехнический университет Петра Великого, 2025. – 377 с. – ISBN 978-5-7422-9011-7. – DOI 10.18720/SPBPU/2/id25-34. – EDN SACKZQ.
2. Брайловский, А. В. Методология сбора данных для анализа активности проектов с открытым исходным кодом на платформе GITHUB / А. В. Брайловский // Парадигма. – 2025. – № 6-1. – С. 47-57. – EDN LWCXPI.
3. jq 1.8 Manual. [Электронный ресурс] Режим доступа: <https://jqlang.org/manual/> (дата обращения: 16.03.2026)
4. Go Packages – tgbotapi package – Golang bindings for the Telegram Bot API. [Электронный ресурс] Режим доступа: <https://pkg.go.dev/github.com/go-telegram-bot-api/telegram-bot-api/v5> (дата обращения: 13.03.2026)
5. Яровикова, А. С. Клиент-серверное приложение "Электронный дневник преподавателей и студентов" / А. С. Яровикова, С. С. Беякова, И. Э. Вишняков // Теоретическая информатика и

компьютерные технологии : Материалы I Научно-практической конференции, Москва, 01 ноября 2023 года. – Москва: Издательство МГТУ им. Н.Э. Баумана, 2024. – С. 85-87. – EDN VQYSUI.

6. Петров, А. С. Сервис автоматизированной проверки лабораторных работ: разработка и внедрение / А. С. Петров, А. А. Фалеев // Актуальные аспекты развития науки и общества в эпоху цифровой трансформации (шифр - МКАА) : Сборник материалов XXII Международной научно-практической конференции, Москва, 30 мая 2025 года. – Москва: АНО ДПО «Университет ИТБО», 2025. – С. 143-154. – EDN LCWVNQ.

УДК 004.4

А.А. Пономарев (4 курс бакалавриата),
И.В. Никифоров, к.т.н., доцент

СРАВНЕНИЕ СОБЫТИЙНО- И СЕРВИС-ОРИЕНТИРОВАННОЙ АРХИТЕКТУР В СИСТЕМАХ ОАМ И ВЛИЯНИЕ АРХИТЕКТУРНОГО ПОДХОДА НА ТРУДОЁМКОСТЬ РАЗВИТИЯ СИСТЕМЫ

Системы управления и автоматизации (ОАМ – Operations, Administration and Maintenance) [1] являются важной частью телекоммуникационной инфраструктуры, обеспечивая мониторинг, управление и настройку сетевых устройств [2]. В системах ОАМ, реализованных на основе сервис-ориентированной архитектуры (SOA – service oriented architecture), взаимодействие компонентов часто строится на синхронных вызовах сервисов [3]. По мере роста функциональности это приводит к увеличению числа межсервисных зависимостей, что усложняет модификацию системы и повышает стоимость внесения изменений [4]. Событийно-ориентированная архитектура (EDA – event-driven architecture) позволяет решать эти проблемы за счет асинхронной обработки событий и уменьшения зависимостей между компонентами системы [5].

В работе исследуется, как переход от сервис-ориентированной архитектуры ОАМ к событийно-ориентированной влияет на трудоемкость поставки новых функциональностей и изменение метрик системы. Для обеспечения сопоставимости результатов построены две архитектурные модели (SOA и EDA) в виде диаграмм компонентов. Системы реализованы согласно предложенным моделям на языке Golang и C++. Программная реализация происходила последовательно, сначала был реализован SOA вариант системы, который представлен на рисунке 1.

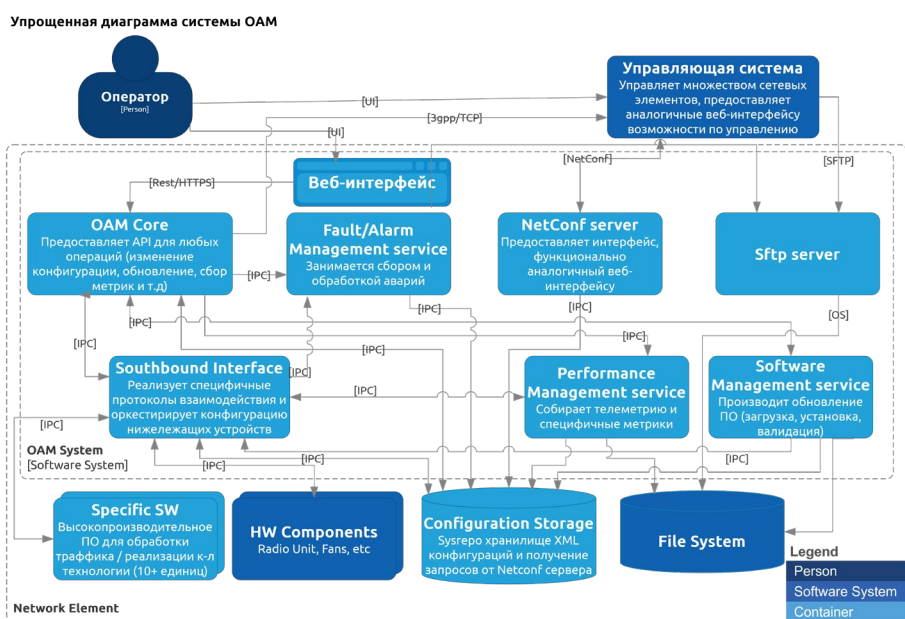


Рисунок 1 – Диаграмма компонент системы ОАМ на основе SOA

Затем, осуществлен переходе на Event-Driven архитектуру, как показано на рисунке 2, в результате чего центральным компонентом системы является “Event Dispatcher”, который выполняет роль брокера событий и их обработчика согласно формальной метамодели. Метамодель позволяет обрабатывать сложные зависимости между событиями, а также производить корреляцию между ними для подавления избыточных событий в системе.

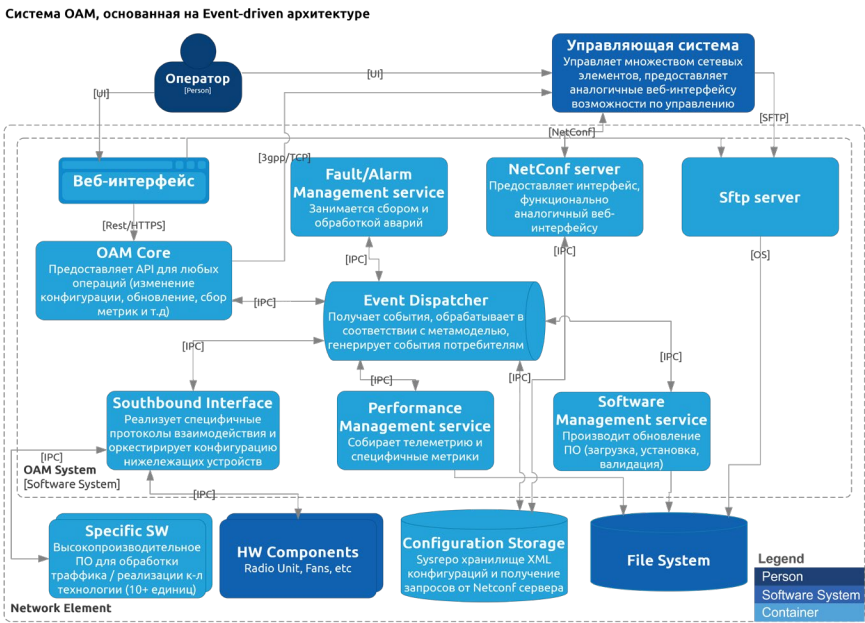


Рисунок 2 – Диаграмма компонент системы OAM на основе EDA

Для сравнения двух вариантов системы были выбраны следующие характеристики и соответствующие им метрики: связанность компонентов – Coupling Between Objects (CBO); модифицируемость – Propagation Cost (PC) [6]; масштабируемость – Scalability Index (SI); отказоустойчивость – Cascading Failure Probability (CFP) [7, 8]. В результате подсчета метрик для двух систем были получены значения, отображенные в таблице 1, которые показывают, что переход к событийно-ориентированной архитектуре приводит к снижению связанности компонентов и уменьшению стоимости распространения изменений в системе примерно в 2 раза.

Таблица 1 – Сравнение метрик двух вариантов системы

Архитектура\Метрика	CBO	PC	SI	CFP
SOA	2,9	41%	0.5	0.83
EDA	1,4	20%	0.7	0.65

Проведен эксперимент, результаты которого отображены на рисунке 3, по внесению однотипных изменений на уровне продукта – базовой станции телекоммуникационной сети, и компонента OAM. Для оценки трудозатрат использовалась метрика Story Point (SP).

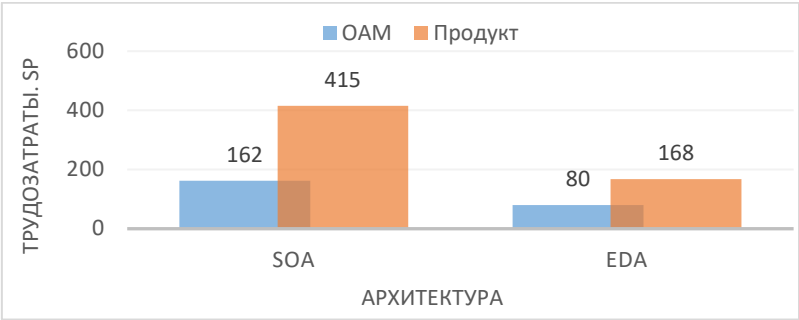


Рисунок 3 – Затраты Story Point на однотипные изменения для двух вариантов системы

Проведенное сравнение архитектурных моделей коррелируют с результатами эксперимента, который подтверждает снижение трудозатрат в 2 раза. Полученные результаты

подтверждают целесообразность применения событийно-ориентированной архитектуры при проектировании систем управления и администрирования телекоммуникационных сетей.

ЛИТЕРАТУРА

1. RFC 7276: "An Overview of Operations, Administration, and Maintenance (OAM) Tools" // IETF Datatracker URL: <https://datatracker.ietf.org/doc/html/rfc7276> (дата обращения: 01.03.2026).
2. 3GPP TS: "Telecommunication management; Principles and high level requirements" // 3GPP Portal URL: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1838> (дата обращения: 01.03.2025).
3. Карпович М. Н. Особенности проектирования микросервисно-событийных архитектур для высоконагруженных распределенных систем обработки информации // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2023. № 1 (266). С. 89–95. DOI: 10.52065/2520-6141-2023-266-1-15.
4. Valipour M. H. et al. A brief survey of software architecture concepts and service oriented architecture //2009 2nd IEEE International Conference on Computer Science and Information Technology. – IEEE, 2009. – С. 34-38.
5. Clark T., Barn B. S. Event driven architecture modelling and simulation //Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE). – IEEE, 2011. – С. 43-54.
6. Nord R. L. et al. Variations on using propagation cost to measure architecture modifiability properties //2013 IEEE International Conference on Software Maintenance. – IEEE, 2013. – С. 400-403.
7. Shereshevsky M. et al. Information theoretic metrics for software architectures //25th Annual International Computer Software and Applications Conference. COMPSAC 2001. – IEEE, 2001. – С. 151-157.
8. Varlamov, D. A. Algorithm for monitoring and improving the stability of the IT infrastructure based on availability and reliability metrics / D. A. Varlamov, I. V. Nikiforov, S. M. Ustinov // Computing, Telecommunications and Control. – 2024. – Vol. 17, No. 2. – P. 24-37. – DOI 10.18721/JCSTCS.17203. – EDN ANXOZH.

УДК 004.4

Л.Е. Смирникова (4 курс бакалавриата),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА КОМПАРАТОРА ДЛЯ АНАЛИЗА АРХИТЕКТУРНОЙ ДОКУМЕНТАЦИИ ПРОГРАММНЫХ СИСТЕМ

В процессе разработки и сопровождения программных систем архитектурная документация играет ключевую роль, отражая структуру системы, взаимодействие компонентов, принятые проектные решения и архитектурные ограничения. На практике она создаётся в разнообразных форматах и нотациях (C4-диаграммы, PlantUML, ER-диаграммы, Protobuf, структура каталогов проекта, элементы исходного кода). Такое разнообразие делает задачу сравнения и проверки соответствия трудоёмкой и слабо формализуемой [1].

Автоматизация процесса сравнения архитектурных документов является одним из решений указанной проблемы. Ранее в рамках проекта были реализованы специализированные утилиты для парного сравнения: диаграмм PlantUML с кодом на языке Golang [2], диаграмм PlantUML с описаниями структур в Protobuf [3], C4-диаграмм со структурой каталогов проекта [4], PlantUML с диаграммами C4 [5]. Эти инструменты подтвердили практическую ценность автоматизации, однако выявили ряд ограничений: дублирование кода парсеров и логики сравнения, а также сложность добавления поддержки новых нотаций.

Целью работы является снижение трудоёмкости верификации архитектурной документации программных систем путём разработки универсального компаратора на основе единого промежуточного представления.

Ключевая идея решения заключается в переходе от попарного сравнения к единому подходу: все разнородные входные данные (файлы Draw.io, PlantUML, директории проекта) приводятся к единому унифицированному промежуточному представлению (Intermediate Representation, IR). Это позволяет выполнять сравнение не между исходными форматами, а между их IR-представлениями, делая логику компаратора независимой от источника данных.

Разработанный компаратор представляет собой модульную систему, реализованную на языке Python [6] с использованием принципов объектно-ориентированного проектирования. Каждый тип входных данных обрабатывается специализированным парсером, формирующим унифицированное промежуточное представление архитектуры (IR). IR представляет собой JSON-модель, описывающую архитектурные сущности (компоненты, контейнеры, модули, классы и др.) и отношения между ними. Алгоритм сравнения архитектурных моделей включает два основных этапа: сопоставление сущностей и анализ отношений между ними.

На этапе сопоставления сущностей выполняется индексация элементов целевой модели по их нормализованным именам. Далее для каждой сущности исходной модели осуществляется поиск соответствия в построенном индексе. При обнаружении совпадения дополнительно проверяется совместимость типов архитектурных элементов на основе заданных правил соответствия. В случае успешной проверки фиксируется пара сопоставленных сущностей, которая используется при последующем анализе связей.

Анализ структуры связей выполняется на основе представления архитектуры в виде ориентированного графа [7]. Из множества отношений выделяются связи, отражающие иерархическую структуру системы (вложенность и композицию). На основе этих отношений для каждой модели строится ориентированный граф, в котором вершины соответствуют сущностям, а рёбра отражают отношения типа «родитель–потомок». Анализ таких графовых структур позволяет выявлять несоответствия в иерархии компонентов. Поскольку полное сравнение графов с учётом всех типов связей является вычислительно сложной задачей, связи, не относящиеся к иерархическим (например, зависимости, использование и ассоциации), анализируются отдельно. При их сопоставлении учитываются наличие соответствующих сущностей в обеих моделях, совместимость типов связей и их направленность. Такой подход позволяет выявлять как отсутствующие связи, так и расхождения в структуре взаимодействий компонентов системы.

Результаты работы модуля сравнения агрегируются в структурированном виде, включающем:

- списки сопоставленных сущностей и связей с указанием соответствующих пар;
- перечни сущностей и связей, не нашедших соответствия в другом представлении;
- статистические метрики, отражающие степень соответствия между сравниваемыми архитектурами.

Для оценки корректности работы компаратора проведено регрессионное тестирование на наборах данных, ранее использовавшихся при проверке специализированных утилит [4, 5]. Тестовый набор включал C4-диаграммы с соответствующими структурами каталогов, а также классовые PlantUML-диаграммы и C4-диаграммы. Сравнение результатов показало полное совпадение выявленных соответствий и расхождений с результатами, полученными специализированными утилитами, что подтверждает корректность рефакторинга.

В настоящее время реализована поддержка трёх типов источников: C4-диаграмм (формат Draw.io), PlantUML-диаграмм и структур каталогов проектов. Предложенная архитектура компаратора упрощает расширение системы и позволяет добавлять поддержку новых форматов архитектурной документации за счёт разработки дополнительных модулей преобразования.

ЛИТЕРАТУРА

1. Uzun, B. Detecting deviations in the code using architecture view-based drift analysis / B. Uzun, B. Tekinerdogan // Computer Standards & Interfaces. – 2024. – Vol. 87. – P. 103774. – DOI 10.1016/j.csi.2023.103774. – EDN WXPWUW.

2. Клеблеева, Э. М. Анализ архитектурной документации сравнение PlantUML-диаграмм с исходными файлами на языке Golang / Э. М. Клеблеева, Д. М. Вайндрах, И. А. Селин // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 316-318. – EDN SROAMJ.
3. Лонишин, М. Р. Разработка программы для сравнения классовых диаграмм plantUML и ProtoBuf файлов / М. Р. Лонишин, Н. Р. Богданов, И. А. Селин // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 321-322. – EDN XEQFRP.
4. Мамонова, А. С. Анализ архитектурной документации сравнение C4-диаграмм с каталогами проекта / А. С. Мамонова, В. В. Тронеv, И. А. Селин // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 325-327. – EDN DGBUFH.
5. Тронеv, В. В. Сравнение технических документаций сетевой инфраструктуры в форматах PlantUML и C4 / В. В. Тронеv, И. А. Селин // Современные технологии в теории и практике программирования : Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, Санкт-Петербург, 23–24 апреля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 344-345. – EDN WHMOOY.
6. Python 3 Documentation : официальная документация. – URL: <https://docs.python.org/3/> (дата обращения: 10.03.2026).
7. Mitchell B. S., Mancoridis S. On the automatic modularization of software systems using the bunch tool //IEEE Transactions on Software Engineering. – 2006. – Т. 32. – №. 3. – С. 193-208.

УДК 004.421

М.Ю. Тимофеев, А.С. Смирнов, Д.М. Климашов (2 курс бакалавриата),
В.В. Амосов, к.т.н., доцент

ПРОГРАММНАЯ И АППАРАТНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА СОРТИРОВКИ РАСЧЁСКОЙ

С появлением современных ПЛИС (CPLD) и систем на кристалле (SoC), объединяющих процессор, память и программируемую логику, открываются широкие возможности для ускорения вычислений путём реализации алгоритмов непосредственно в аппаратуре. В связи с этим актуальной задачей является разработка аппаратно-программных решений для конкретных алгоритмов обработки данных на современной элементной базе.

Целью работы является разработка программно-аппаратной реализации алгоритма сортировки расчёской (comb sort). Данный алгоритм относится к классу алгоритмов обменной сортировки и решает задачу упорядочивания массивов данных, обеспечивая повышение эффективности по сравнению с пузырьковой сортировкой за счёт использования переменного шага сравнения элементов[1].

В ходе проекта разработан комплексный программный продукт, состоящий из следующих элементов:

1. Программная реализация алгоритма сортировки расчёской на языке программирования C++[1, 4];
2. Аппаратная реализация алгоритма сортировки расчёской на языке программирования VHDL[3];
3. Конфигурирование Register-Transfer Level (RTL) и Technology Map схем алгоритма[2];
4. Реализация алгоритма сортировки расчёской в командах 32-разрядного RISC-процессора с Гарвардской архитектурой – DP32[2];
5. Выполнение ко-симуляции на поведенческой и структурной модели DP32[2].

В рамках работы для тестирования программной и аппаратной реализаций алгоритма были разработаны соответствующие Test Bench. При программной реализации использовались среды программирования Microsoft Visual Studio и Active VHDL. При аппаратной реализации для проверки на синтезируемость и для тестирования – САПР Quartus II.

Благодаря использованию современных ПЛИС, разработанная система реализует алгоритм сортировки расчёской в виде комплексного аппаратно-программного решения. Это позволяет применять её в различных промышленных областях, где требуется эффективная обработка данных в реальном времени, включая задачи организации баз данных, цифровой обработки сигналов и встраиваемых систем.

ЛИТЕРАТУРА

1. Dobosiewicz, W. "An efficient variation of bubble sort". *Information Processing Letters*, vol. 11, no. 1, pp. 5–6, 1980.
2. В.В.Амосов, “Схемотехника и средства проектирования цифровых устройств” – СПб.: БХВ-Петербург, 2007 – 560 с.
3. Вахитов Р.М. Язык VHDL. Практическое применение. — М.: ДМК Пресс, 2020
4. Е.Н. Каширская, М.А. Макаров, С.Е. Харьковский, “Программирование алгоритмов сортировки учебное пособие”, pp. 48-49, 2021

УДК 004.272.32, 272.44, 4'418

С.А. Флусова (4 курс бакалавриата),
И.В. Шошмина, к.т.н., доцент

АВТОНАСТРОЙКА ПАРАМЕТРОВ GPU-ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ ВЕРИФИКАЦИИ НА ОСНОВЕ TLA+

Графические процессоры (GPU) широко применяются для решения задач, требующих интенсивных параллельных вычислений. Архитектура GPU ориентирована на выполнение большого числа потоков, работающих по принципу SIMT (Single Instruction Multiple Threads). Для программирования таких вычислений используется стандарт OpenCL [4], в котором параллельная программа состоит из хост-части, исполняемой на центральном процессоре, и вычислительного ядра, выполняемого множеством рабочих элементов на GPU [1].

Производительность OpenCL-программ существенно зависит от параметров конфигурации выполнения. К таким параметрам относятся, например, размер рабочей группы (WG — work-group size) и количество элементов данных, обрабатываемых рабочей группой за одну итерацию (TS — tile size). Подбор оптимальных значений этих параметров называется автонастройкой (autotuning). Из-за сложной архитектуры GPU и большого числа взаимодействующих вычислительных элементов влияние параметров конфигурации на время выполнения программы трудно предсказать аналитически, поэтому в большинстве систем автонастройки оптимальные значения подбираются экспериментально, что требует значительных вычислительных затрат.

В данной работе рассматривается подход к автонастройке параметров GPU-программы, основанный на методе проверки моделей (model checking) [2]. Этот метод позволяет формально описать поведение системы и проверить корректность выполнения всех возможных сценариев. Если проверяемое свойство не выполняется, средство верификации строит контрпример — последовательность состояний системы, приводящую к нарушению свойства. Такой контрпример используется в работе для получения конфигурации параметров, обеспечивающей более эффективное выполнение программы.

Целью работы является моделирование вычислительного конвейера операций графического процессора и подбор оптимальных параметров автонастройки GPU-программы с использованием средств формальной верификации на основе логики действий TLA+ (Temporal Logic of Actions) [5].

Для достижения поставленной цели были сформулированы следующие задачи:

1. Проанализировать архитектуру графического процессора и структуру выполнения программ на языке OpenCL (рисунок 1).
2. Построить формальную модель выполнения OpenCL-программы на GPU.
3. Смоделировать основные процессы вычислительного конвейера GPU.
4. Выполнить трансляцию модели параллельной программы из языка Promela в формальную спецификацию на языках PlusCal и TLA+.
5. Провести верификацию модели и выполнить подбор параметров автонастройки.

В разработанной модели выполнение программы на GPU представляется в виде системы взаимодействующих процессов как показано на рисунке 1. Моделируются процессы main, host, device и clock, отвечающие за управление выполнением программы и отслеживание времени, а также процессы, описывающие выполнение вычислений на GPU и синхронизацию потоков. Инструкции OpenCL моделируются как атомарные операции, а механизм синхронизации реализуется с использованием барьерной синхронизации.

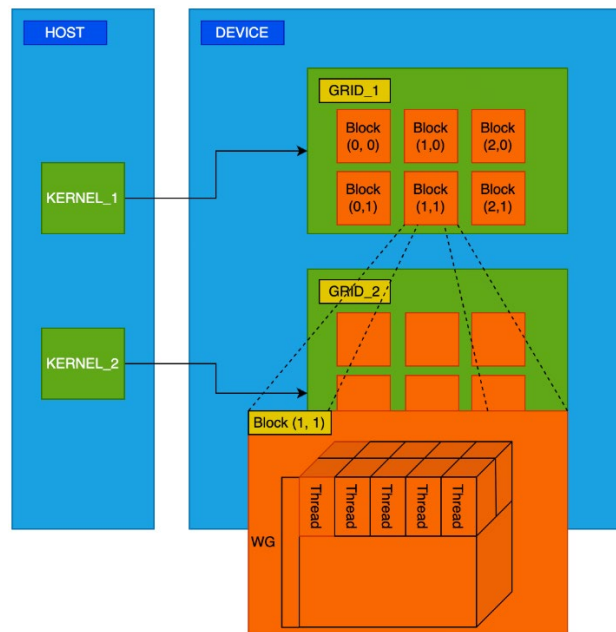


Рисунок 1 – Структура GPU-программы на языке OpenCL

Для поиска оптимальных параметров используется проверка свойства безопасности на языке LTL (линейной темпоральной логики):

$$G(\text{final} \rightarrow \text{time} > T),$$

которое утверждает, что программа не может завершиться быстрее заданного времени T . Если верификатор находит контрпример, это означает, что существует сценарий выполнения, при котором программа завершается за меньшее время. Из такого контрпримера извлекаются значения параметров конфигурации (WG и TS), при которых достигается более эффективное выполнение программы.

Формальная модель описывается на языке PlusCal [6], который позволяет задавать алгоритмы в процедурной форме и автоматически преобразуется в спецификацию TLA+. Для анализа модели используются средства верификации TLA+ и инструмент Apalache. В отличие от классических средств проверки моделей, основанных на явном переборе состояний (например, SPIN), современные инструменты анализа TLA+ используют символьные методы, основанные на SMT-решателях. Такой подход позволяет исследовать пространство состояний модели на значительно большей глубине и анализировать большее число возможных конфигураций системы. Для задачи автонастройки параметров GPU-программы это особенно важно, поскольку увеличение глубины анализа позволяет получать более точные сведения о времени выполнения программы и более надёжно выявлять эффективные значения

параметров конфигурации. Для анализа модели используются средства верификации TLA+ и инструмент Apalache [7].

В работе также рассматривается задача трансляции описания модели параллельной программы из языка Promela, использованного в предыдущих исследованиях [3], в язык PlusCal. Такая трансляция позволяет применять средства формальной верификации на основе TLA+ для анализа моделей GPU-вычислений. Данный подход предполагает возможность разработки специализированного транслятора Promela $\rightarrow$ PlusCal, который позволит автоматически преобразовывать модели параллельных вычислений для последующей формальной верификации.

ЛИТЕРАТУРА

1. Hennessy J., Patterson D. Computer Architecture: A Quantitative Approach. 5th ed. Morgan Kaufmann, 2017.
2. Garanina N., Gorlatch S. Autotuning parallel programs by model checking. Моделирование и анализ информационных систем. 2021. Т. 28. №4. С. 338–355.
3. Gorlatch S., Garanina N., Staroletov S. Auto-Tuning High-Performance Programs Using Model Checking in Promela. 2023.
4. Khronos Group. The OpenCL Specification. Version 3.0. <https://www.khronos.org/opencl/>
5. Lamport L. Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.
6. Lamport L. PlusCal Algorithm Language. <https://lamport.azurewebsites.net/tla/pluscal.html>
7. Apalache model checker for TLA+. <https://github.com/apalache-mc/apalache>

УДК 004.89

Н.И. Черепанов (1 курс магистратуры),
А.Е. Марфина (2 курс бакалавриата),
Н.О. Степина, ассистент,
И.В. Никифоров, к.т.н., доцент

ПРИМЕНЕНИЕ ГИБРИДНЫХ ОПЕРАТОРОВ TVM И РАСШИРЕНИЯ RVV ДЛЯ ОПТИМИЗАЦИИ ВЫЧИСЛЕНИЙ НА АРХИТЕКТУРЕ RISC-V

Современные системы компьютерного зрения и машинного обучения активно применяются на периферийных устройствах с ограниченными вычислительными ресурсами. Одной из перспективных аппаратных платформ для таких систем является архитектура RISC-V, представляющая собой открытую систему команд с возможностью расширения и адаптации под конкретные вычислительные задачи [1].

При выполнении алгоритмов машинного обучения на подобных устройствах существенным ограничением является производительность операций обработки данных. Одним из способов повышения эффективности вычислений является использование векторных инструкций [2]. Дополнительно повысить производительность позволяют специализированные компиляторы машинного обучения, такие как Apache TVM, которые оптимизируют вычисления на уровне промежуточного представления и автоматически генерируют машинный код для целевой архитектуры [3].

Целью данной работы является исследование возможности оптимизации алгоритмов машинного обучения на архитектуре RISC-V с использованием фреймворка Apache TVM и векторных расширений архитектуры.

В рамках исследования решаются следующие задачи:

- развертывание и настройка среды Apache TVM;
- реализация вычислительных ядер алгоритмов машинного обучения с использованием TVM Hybrid Script;

- генерация векторных инструкций RISC-V при помощи LLVM;
- разработка пользовательских инструкций для ускорения вычислений;
- экспериментальная оценка производительности различных реализаций алгоритмов.

Для оценки эффективности предложенных решений был реализован набор тестов для нескольких алгоритмов машинного обучения, включая функции активации и постобработки результатов. В экспериментальный набор вошли алгоритмы ReLU, применяемый в качестве функции активации в глубоких нейронных сетях [4], LayerNorm, используемый для нормализации признаков в архитектурах глубокого обучения [5], Softmax, применяемый для преобразования выходных значений модели в распределение вероятностей [5] и Non-Maximum Suppression (NMS), используемый на этапе постобработки результатов детекции объектов [6]. Для каждого алгоритма были реализованы четыре варианта исполнения:

- базовая реализация на NumPy;
- реализация TVM без векторизации;
- реализация TVM с использованием векторного расширения RVV;
- реализация TVM с использованием пользовательских инструкций.

Экспериментальная оценка проводилась путем сравнения времени выполнения различных реализаций алгоритмов машинного обучения. Для каждого алгоритма были реализованы четыре варианта, а именно базовая реализация на NumPy, скалярная реализация TVM, реализация TVM с использованием векторного расширения RVV и версия с использованием пользовательских инструкций.

Измерения выполнялись при фиксированных размерах входных данных. Параметр B обозначает размер батча, а N размер вектора признаков или количество элементов, обрабатываемых в одном вычислительном ядре. Тестирование выполнялось для нескольких комбинаций размеров входных данных. В таблице 1 представлены результаты для одной из конфигураций параметров, выбранных в качестве примера.

Таблица 1 – Результаты сравнения производительности различных реализаций алгоритмов.

Алгоритм	NumPy (мс)	TVM scalar (мс)	TVM RVV (мс)	TVM custom (мс)
ReLU (B=256, N=16384)	53.03	92.95	83.82	61.51
LayerNorm (B=128, N=4096)	17.29	17.02	9.55	8.08
Softmax (B=128, N=4096)	35.99	38.23	35.17	34.65
NMS (N=2048)	249.60	100.98	81.85	839.44

Эксперименты показали, что применение векторных инструкций позволяет ускорить вычисления. Наибольший эффект был получен для алгоритмов ReLU и LayerNorm, для которых наблюдается ускорение до 2.1 раза по сравнению со скалярной реализацией.

Для алгоритма Softmax прирост производительности оказался меньше, увеличилось на 10%, что связано с использованием функции экспоненты, плохо поддающейся векторизации.

Алгоритм NMS демонстрирует еще меньший выигрыш из-за большого количества ветвлений и нерегулярных обращений к памяти, которые затрудняют использование векторных инструкций.

В результате проведенного исследования было показано, что использование Apache TVM позволяет автоматически генерировать векторные инструкции для архитектуры RISC-V и применять их при реализации алгоритмов машинного обучения. Полученные результаты подтверждают перспективность использования TVM и векторных расширений архитектуры RISC-V для ускорения вычислений на периферийных устройствах.

ЛИТЕРАТУРА

1. Cherepanov, N. I. Improving image analysis and processing performance on the RISC-V platform with Lichee Pi 4A / N. I. Cherepanov, N. O. Stepina, I. V. Nikiforov // *Proceedings of the Institute for System Programming of the RAS.* – 2025. – Vol. 37, No. 5. – P. 157-172. – DOI 10.15514/ISPRAS-2025-37(5)-12. – EDN UDVIEH.
2. Шемаев, К. Е. Векторизация методов решения обыкновенных дифференциальных уравнений на процессорной архитектуре RISC-V / К. Е. Шемаев, П. А. Реблев, Н. О. Степина // *Техника средств связи.* – 2025. – № 4(172). – С. 41-50. – DOI 10.24412/2782-2141-2025-4-41-50. – EDN KUNERJ.
3. Yu, M.-S. Case Study: Optimization Methods with TVM Hybrid-OP on RISC-V Packed SIMD / M.-S. Yu, C.-Y. Yuan, T.-L. Chen, J.-K. Lee // *IEEE Access.* – 2024. – Vol. 12. – P. 1–11. – DOI 10.1109/ACCESS.2024.3397195.
4. Zhu, M. Constructing a smoothed Leaky ReLU using a linear combination of the smoothed ReLU and identity function / M. Zhu, W. Min, J. Li [et al.] // *Neural Computing & Applications.* – 2025. – Vol. 37, No. 9. – P. 6465–6478. – DOI 10.1007/s00521-024-10935-3. – EDN XMOYYK.
5. Kim, R. Hardware Accelerator for Approximation-Based Softmax and Layer Normalization in Transformers / R. Kim, D. Lee, J. Kim [et al.] // *Electronics.* – 2025. – Vol. 14, No. 12. – P. 2337. – DOI 10.3390/electronics14122337. – EDN BAFRZG.
6. Symeonidis, Ch. Neural Attention-Driven Non-Maximum Suppression for Person Detection / Ch. Symeonidis, I. Mademlis, I. Pitas, N. Nikolaidis // *IEEE Transactions on Image Processing.* – 2023. – Vol. 32. – P. 2454–2467. – DOI 10.1109/TIP.2023.3268561. – EDN SGCWVO.

УДК 004.457

В.С. Чурикова, М.А. Новик, М.О. Мишин (2 курс бакалавриата),
К.К. Смирнов, ст. преподаватель

AMPHIMIXIS: РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ АВТОМАТИЗАЦИИ СБОРКИ И ПРОФИЛИРОВАНИЯ ПРИЛОЖЕНИЙ ДЛЯ РАЗЛИЧНЫХ АРХИТЕКТУР

Современная разработка программного обеспечения требует проверки работоспособности и производительности приложений на разных платформах. Эта задача приобрела особую актуальность в ходе летней школы СПбГУ, в рамках проекта “Миграция на RISC-V”, ручная сборка и профилирование проектов для различных архитектур оказались трудоёмкими.

Чтобы оценить готовность проекта к миграции на новую архитектуру (например, RISC-V), необходимо выполнить довольно много шагов: убедиться, что проект собирается под целевую архитектуру, тесты проходят успешно, и что производительность собранного приложения соответствует ожиданиям. Каждый из этих шагов можно автоматизировать в рамках отдельного инструмента (например, фреймворк тестирования), однако сделать это в общем виде сложно. В результате разработчик тратит время на то, чтобы разобраться, как собрать проект, как запустить тесты и как измерить производительность для конкретной архитектуры. Таким образом, ручная сборка и профилирование проектов для различных архитектур замедляет исследование.

Для решения этой проблемы предложено разработать инструмент, который получил название Amphimixis [1], выполняющий процесс исследования проекта “на пригодность к миграции на различных архитектурах” автоматизировано, без ручного вмешательства.

Целью данной работы является проектирование архитектуры и реализация ключевых модулей Amphimixis, обеспечивающих автоматизацию процессов локальной и удалённой сборки и профилирования.

Для достижения цели необходимо решить следующие задачи:

- Разработать архитектуру приложения, позволяющую выполнять отдельные этапы пайплайна независимо.
- Реализовать модули, отвечающие за анализ проекта, сборку и профилирование.

- Создание CLI для взаимодействия пользователя с инструментом.
- Проведение тестирования на конкретных проектах.

Архитектура приложения и общий пайплайн проекта представлены на рисунке 1.

Amphimixis представляет собой консольное приложение на языке Python. Пользователь описывает варианты сборки в локальном конфигурационном файле (формат YAML), указывая целевые машины (адрес, архитектура), рецепты сборки (параметры компилятора, флаги) и используемые тулчейны. Пайплайн работы инструмента обобщает действия разработчика при исследовании проектов и включает в себя четыре основных модуля: анализ проекта путём проверки наличия CI, тестов и бенчмарков, а также определение поддерживаемых систем сборки (Analyzer), разбор локального конфига и передача данных для следующих модулей (Configurator), сборка с указанными флагами компиляции (Builder), сбор и сравнение трасс полученных сборок с использованием утилит perf stat и perf record [2] (Profiler). Дополнительно для взаимодействия с удалёнными машинами разработан модуль Shell, инкапсулирующий работу с SSH. Кроме того, реализован модуль LaboratoryAssistant, управляющий глобальным конфигурационным файлом, в котором пользователь может сохранять часто используемые машины и обращаться к ним по именам.

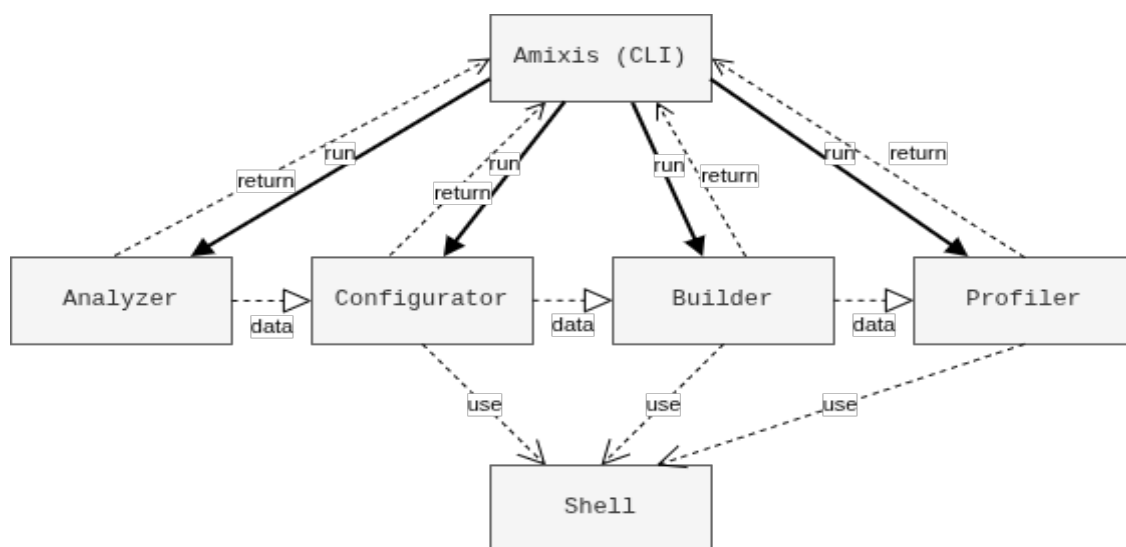


Рисунок 1 – Архитектура приложения

Разделение на модули позволяет запускать пайплайн как целиком, так и отдельными этапами, что полезно при повторном исследовании уже собранных проектов.

Было проведено ручное тестирование на проектах с открытым исходным кодом: tinysql2, yaml-cpp и nlohmann/json. Разработаны тесты, подтверждающие корректность локального и удалённого запуска, взаимодействия между модулями и работоспособность на архитектуре RISC-V.

В ходе работы спроектирована архитектура инструмента, реализованы ключевые модули Analyzer, Builder, Configurator и Profiler, а также интерфейс командной строки для взаимодействия с пользователем. Подтверждена работоспособность инструмента на тестовых проектах с системой сборки CMake. Разработанный инструмент позволяет автоматизировать рутинные операции по сборке и профилированию приложений для архитектур x86, RISC-V и направлен на сокращение временных затрат разработчика. Работа выполнена в Лаборатории технологий программирования инфраструктурных решений СПбГУ.

ЛИТЕРАТУРА

1. GitHub-репозиторий [Электронный ресурс]. Режим доступа: <https://github.com/ebzych/amphimixis>. [Дата обращения 17.03.2026]
2. Brendan Gregg, Linux Performance Analysis and Tools, 2021 [Электронный ресурс] Режим доступа: <https://www.brendangregg.com/linuxperf.html>

А.В. Парфенов (4 курс бакалавриата),
Н.С. Самодуров (2 курс бакалавриата),
Т.С. Фролов (3 курс бакалавриата),
Я.А. Кириленко, ст. преподаватель

РАЗРАБОТКА ЛИНТЕРА VERIHOGG ДЛЯ SYSTEMVERILOG

Современное проектирование цифровых устройств всё чаще использует язык SystemVerilog (IEEE 1800–2017) [1] не только для синтезируемого RTL-кода, но и для верификации. При разработке программного обеспечения широко применяется класс быстрых инструментов статического анализа кода, называемых в профессиональной среде «линтерами». Существует множество реализаций линтеров для всех распространённых языков программирования. Однако анализ открытых инструментов для SystemVerilog (Verible, Slang, svlint, Verilator, Surelog), проведённый в рамках данного исследования, показывает критический разрыв: большинство существующих решений с открытым исходным кодом либо не поддерживают стандарт в полном объёме, либо ограничены синтезируемым подмножеством, либо не обеспечивают анализ верификационных конструкций. Коммерческие инструменты, такие как DVT Verissimo и другие, обладают необходимой полнотой, но недоступны для открытых проектов, не всегда доступны для использования в преподавании и академических исследованиях.

Целью работы является создание конкурентного решения с открытым исходным кодом для обнаружения дефектов в коде SystemVerilog (IEEE 1800–2017).

Для достижения этой цели были поставлены задачи, перечисленные ниже.

- Анализ существующих парсеров и компиляторов SystemVerilog на предмет пригодности для создания линтера.
- Выбор инструментальной основы, обеспечивающей полную поддержку стандарта и доступ к AST и семантическим связям.
- Реализация набора правил статического анализа для выявления типовых ошибок в верификационных конструкциях.

В результате сравнительного анализа (Verible, Slang, Surelog, svlint, Verilator) наиболее полную семантическую модель UHDM (Universal Hardware Data Model) и максимальное соответствие IEEE 1800–2017 предоставляет компилятор Surelog.

В качестве ориентира для формирования набора реализуемых правил был выбран коммерческий инструмент Verissimo (DVT) [2].

В ходе практической реализации создан линтер Verihogg, исходный код которого доступен в публичном репозитории [3]. На данный момент в рамках проекта реализовано более 50 правил статического анализа, охватывающих следующие категории: работа с классами (включая проверки циклического наследования, дублирования классов и конструкторов), корректность описаний covergroup и coverpoint, проверки событий и перечислений, а также общие синтаксические и семантические ограничения языка. Полный список реализованных правил доступен в репозитории инструмента.

Для подтверждения эффективности Verihogg был протестирован на наборе открытых репозиторий, где продемонстрировал стабильное время анализа и низкий уровень ложных срабатываний, успешно выявляя семантические дефекты, пропускаемые существующими решениями с открытым исходным кодом. В ближайшей перспективе планируется интеграция линтера в типовые CI/CD-конвейеры, расширение набора правил до 100+ позиций и адаптация инструмента под образовательные курсы по верификации, что позволит повысить качество обучения разработке на SystemVerilog и укрепить экосистему открытых средств анализа HDL-кода. Инструмент готовится к представлению инженерному сообществу.

Работа выполнена в Лаборатории технологий программирования инфраструктурных решений СПбГУ. Выражаем благодарность Ангелине Васильевне Медеян за вклад в данное исследование.

ЛИТЕРАТУРА

1. IEEE Std 1800-2017. IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language. – New York: IEEE, 2018. – 1315 p. – DOI 10.1109/IEEESTD.2018.8299595.
2. Verissimo SystemVerilog Linter User Guide – Compilation Rules. [Электронный ресурс] Режим доступа: <https://eda.amiq.com/documentation/verissimo/toc/rulesets/compilation-rules.html> (дата обращения: 16.03.2026)
3. Verihogg Linter. [Электронный ресурс] Режим доступа: <https://github.com/verihogg/verihogg-lint> (дата обращения: 16.03.2026)

УДК 004.272.44

Ш.Р. Шаехмитов (2 курс бакалавриата),
С.В. Григорьев, к. ф.-м. н., доцент,
В.А. Кутев, инженер-исследователь

УЛУЧШЕНИЕ СОВМЕСТИМОСТИ RISC-V GPGPU VORTEX С БИБЛИОТЕКОЙ РАЗРЕЖЕННОЙ ЛИНЕЙНОЙ АЛГЕБРЫ

Разреженная линейная алгебра – основа многих алгоритмов [1] анализа графов, стандарт GraphBLAS [2] формализует примитивы и операции разреженной линейной алгебры для этих алгоритмов. Графические ускорители используются для достижения высокой степени параллелизма в данных задачах [3]. Платформа Vortex GPGPU [4, 5], основанная на архитектуре RISC-V и ориентированная на выполнение вычислений с использованием OpenCL 1.2, представляет интерес как открытое и конфигурируемое решение для высокопроизводительных вычислений. SPLA [6, 7] – библиотека разреженной линейной алгебры, в которой реализуются алгоритмы анализа графов для использования на GPU. В ходе продолжения предыдущей работы [8] над запуском SPLA на Vortex были обнаружены следующие проблемы: неправильный подсчет смещений для распределения задач на ядрах, гонки данных в коде для запуска ядер, некорректная работа с локальной памятью, отсутствие операции копирования буфера.

Целью работы является устранение ошибок совместимости Vortex с SPLA. Для достижения поставленной цели решаются перечисленные ниже задачи.

1. Добавление в среду исполнения операции копирования буфера.
2. Исправление схемы расчета для распределения задач между ядрами.
3. Исправление ошибок работы устройства с локальной памятью.
4. Устранение гонки данных в коде запуска ядер.

Операция копирования буфера используется в SPLA для эффективного копирования данных внутри устройства. В среде исполнения Vortex отсутствовала данная операция, которая является стандартной для среды исполнения OpenCL, поэтому изначальным решением была полная замена операций копирования в SPLA на последовательные чтение и запись. Однако это не было лучшим решением, так как приходилось менять большое количество кода. Поэтому в среду исполнения для всех симуляторов в Vortex была добавлена операция копирования буфера.

В ходе работы с запуском библиотеки было обнаружено, что на некоторых конфигурациях ядра пересчитывали данные друг друга, что приводило к некорректным результатам. Изучив код, который отвечает за распределение задач между ядрами, было обнаружено, что формула подсчёта распределения задач на ядра была некорректна, поэтому было предложено исправление, решающее эту проблему.

Ещё одной выявленной проблемой стала некорректная работа с локальной памятью. Среда исполнения RoCL содержала код, который использовал неправильный размер указателей для смещений внутри локальной памяти. Также в среде исполнения смещение для локальных аргументов не изменялось и всегда указывало на первый аргумент, поэтому всегда в качестве локальных аргументов брался первый переданный аргумент. Поэтому было предложено исправление, которое решает эту проблему.

Следующей проблемой стала гонка данных в секции BSS (Block Started by Symbol), которая является памятью для хранения неинициализированных или инициализированных нулём глобальных и статических переменных. Каждое ядро перед запуском очищало секции BSS, но некоторые ядра доходили до исполнения кода быстрее, чем другие заканчивали чистить BSS. SPLA содержит циклы, в которых значение после каждой итерации увеличивается на `global_size` (количество рабочих элементов), который находится в BSS, так как значение вычисляется во время исполнения. Поэтому данные, которые лежали в BSS, обращались в ноль, что приводило при запуске SPLA к бесконечным циклам, так как значение в цикле увеличивалось на ноль. Чтобы исправить эту проблему заполнение BSS нулями было перенесено на сторону хоста, который при запуске ядер заполняет эту память нулями. Это решение исправило гонку данных для конфигураций с несколькими ядрами.

Проведенная работа показала, что Vortex требует ещё большой доработки, чтобы корректно работать на любых конфигурациях. Возникающие проблемы связаны главным образом не с исполнением инструкций, а с кодом, отвечающим за подготовку запуска устройства и за среду исполнения. Поэтому для использования Vortex как конфигурируемого GPGPU требуется доработка и проведение экспериментов на разных библиотеках, использующих OpenCL, чтобы понять пригодность устройства для высокопроизводительных задач. Полученные результаты расширяют использование этого устройства на многоядерных конфигурациях, что может послужить основой для исследований или расширения функциональности этого устройства.

ЛИТЕРАТУРА

1. Davis T. A. Algorithm 1037: Suitesparse: graphblas: Parallel graph algorithms in the language of sparse linear algebra //ACM Transactions on Mathematical Software. – 2023. – Т. 49. – №. 3. – С. 1-30.
2. Brock B. et al. Introduction to graphblas 2.0 //2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). – IEEE, 2021. – С. 253-262.
3. Yang C., Buluç A., Owens J. D. GraphBLAST: A high-performance linear algebra-based graph framework on the GPU //ACM Transactions on Mathematical Software (TOMS). – 2022. – Т. 48. – №. 1. – С. 1-51.
4. Blaise T. et al. Vortex: An Open Source Reconfigurable RISC-V GPGPU Accelerator for Architecture Research //Hot Chips. – 2020. – Т. 32.
5. Vortex RISC-V GPGPU.[Электронный ресурс] Режим доступа: <https://github.com/vortexgpgpu/vortex> (дата обращения: 11.03.2026)
6. Библиотека анализа графов SPLA. [Электронный ресурс] Режим доступа: <https://github.com/SparseLinearAlgebra/spla> (дата обращения: 11.03.2026)
7. Орачев Е.С. Generalized sparse linear algebra library with vendor-agnostic GPUs acceleration. [Электронный ресурс] Режим доступа: <https://dspace.spbu.ru/items/2aa43d43-60f5-465d-981d-a8078c69297c> (дата обращения: 15.03.2026)
8. Шаехмитов Ш.Р. Исследование поведения RISC-V GPGPU VORTEX при запуске библиотеки разреженной линейной алгебры // Математическое моделирование и суперкомпьютерные технологии: Труды XXV Международной конференции. Нижний Новгород, 2025. – Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского, Нижний Новгород, 2025. – С. 207-208.

РАЗБОР PLANTUML-ДИАГРАММ ДЛЯ АНАЛИЗА АРХИТЕКТУРНОЙ ДОКУМЕНТАЦИИ

Современное программное обеспечение (сокр.: ПО), как правило, представляет собой сложную систему, состоящую из множества взаимосвязанных компонентов. Структура такой системы и принципы взаимодействия её частей описываются в архитектурной документации. Однако на практике архитектура редко фиксируется в одном виде, в большинстве случаев она представляет собой совокупность различных представлений: формальные модели, диаграммы, структуры каталогов проекта, исходный код, API-контракты и др. [1]

Одной из проблем, возникающих в ходе разработки ПО, является расхождение элементов архитектурной документации, когда изменение одного из артефактов не всегда сопровождается актуализацией остальных. Расхождения усложняют сопровождение ПО и мешают пониманию устройства системы, тем самым повышая материальные и временные издержки разработки [2], [3].

Одним из распространённых представлений архитектуры являются PlantUML-диаграммы [4]. Данный вид диаграмм часто используется для формализованного описания структуры системы и взаимодействия её элементов, а также для согласования архитектурных решений между бизнес-заказчиком и разработчиком.

Данная работа является продолжением исследования, посвящённого проблеме расхождения архитектурной документации [5]. Цель настоящей работы – решить проблему путём разработки парсера PlantUML-диаграмм в рамках единой системы анализа элементов архитектуры ПО.

Разработанный парсер предназначен для разбора структурных PlantUML-диаграмм в текстовом виде; работа осуществляется с файлами с расширениями .plantuml, .puml и .uml. Общую схему работы парсера можно наблюдать на рисунке 1.

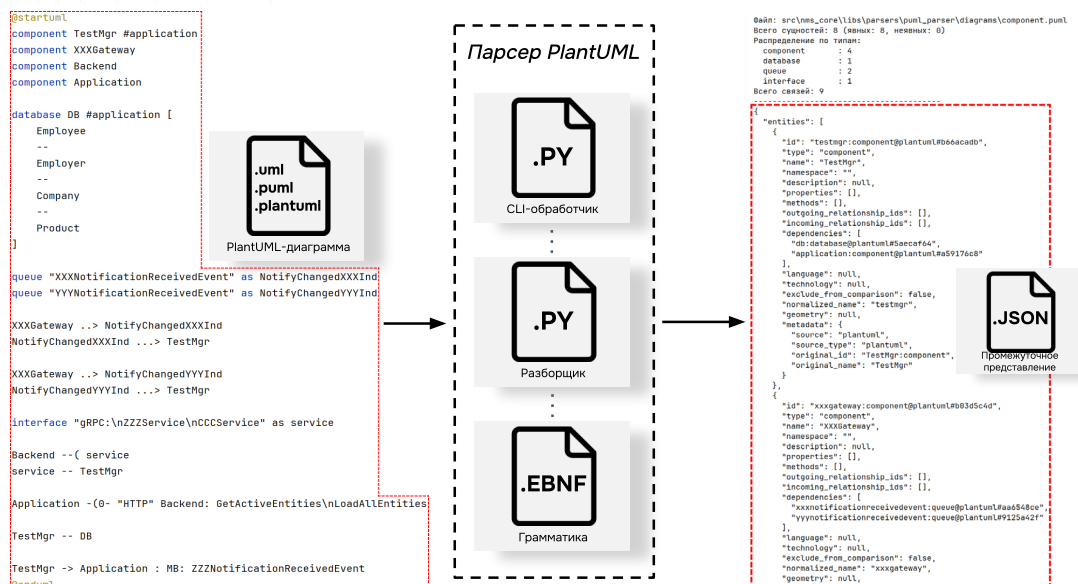


Рисунок 1 – Общая схема работы парсера

Для каждого обрабатываемого файла применяется следующий алгоритм:

1. Поиск внутри файла директив @startuml и @enduml с выделением текста между ними для последующей работы. Всё, что находится вне этого блока, в дальнейший анализ не попадает.

2. Разбиение полученного UML-блока на отдельные строки. Каждая строка обрабатывается отдельно, пустые строки игнорируются.

3. Разбор полученной строки на основе EBNF-грамматики [6] на основные сущности системы: package, class, interface, enum, component, node, artifact, database и др. [4].

4. Выделение и фиксация основных характеристик для каждой найденной сущности: название, вид элемента и положение в общей структуре диаграммы.

5. Разбор связей между сущностями и фиксация характера их отношения.

6. Приведение полученных данных к единому виду. Наименования, типы элементов и сведения об их связях упорядочиваются таким образом, чтобы исключить неоднозначность интерпретации и обеспечить единообразное представление результатов разбора.

7. Формирование целостного промежуточного представления диаграммы (Intermediate Representation; сокр.: IR) в виде JSON-файла, содержащего сведения о выделенных сущностях и связях между ними.

Разработка парсера проводилась на языке программирования Python ввиду наличия у последнего библиотечной поддержки синтаксического анализа, развитых средств обработки текстов и встроенных механизмов работы со структурированными данными. Для реализации синтаксического разбора была использована внешняя библиотека Lark [7], а синтаксис обрабатываемого подмножества PlantUML формализован с помощью грамматики в нотации EBNF.

Парсер состоит из трёх основных частей:

1. Разборщик. Основная часть парсера, ответственная за реализацию полного цикла обработки диаграммы. В нём выполняются предварительная обработка входного текста, лексико-синтаксический разбор строк на основе заданной грамматики, выделение сущностей и связей, разрешение вложенности и псевдонимов, а также формирование единого промежуточного представления.

2. CLI-обработчик. Необходим для обработки команд, вводимых в консоль. Он обеспечивает приём входных путей, поиск файлов диаграмм, вызов разборщика для каждого входного файла, вывод краткой информации о результатах обработки, а также экспорт промежуточного представления в формате JSON в консоль или в отдельные файлы.

3. Грамматика. Содержит грамматику для Lark [7] в нотации EBNF, описывающую поддерживаемое подмножество синтаксиса PlantUML. Грамматика задаёт правила разбора основных конструкций диаграммы, включая сущности, пакеты, вложенные области и связи, и используется синтаксическим анализатором для формального распознавания входного текста.

В результате работы реализован парсер структурных PlantUML-диаграмм, формирующий промежуточное представление в формате JSON и поддерживающий последующее построение читаемого отчёта. Это делает процесс анализа архитектурной документации более удобным и наглядным. Применение разработанного решения позволяет выявлять расхождения между различными представлениями архитектуры ПО, что способствует повышению их согласованности и упрощает сопровождение системы.

ЛИТЕРАТУРА

1. ISO/IEC/IEEE 42010:2022. Software, systems and enterprise — Architecture description. — Текст : электронный // ISO. — URL: <https://www.iso.org/standard/74393.html> (дата обращения: 13.03.2026).
2. Industrial software verification and testing technology / P. D. Drobintsev, V. P. Kotlyarov, A. A. Letichevsky, I. A. Selin // CEUR Workshop Proceedings : 5, Moscow, 14–16 ноября 2017 года. – Moscow, 2017. – P. 221-229. – EDN RXUWDJ.
3. Keim, J. Detecting inconsistencies in software architecture documentation using traceability link recovery / J. Keim, S. Corallo, D. Fuchß, A. Koziolok. — Текст : непосредственный // 2023 IEEE 20th International Conference on Software Architecture (ICSA). — 2023. — P. 141–152. — DOI 10.1109/ICSA56044.2023.00021.
4. PlantUML Language Reference Guide. — Текст : электронный // PlantUML. — URL: <https://plantuml.com/guide> (дата обращения: 12.03.2026).

5. Клеблеева, Э. М. Анализ архитектурной документации: сравнение PlantUML-диаграмм с исходными файлами на языке Golang / Э. М. Клеблеева, Д. М. Вайндрах, И. А. Селин. — Текст : непосредственный // Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, 23–24 апреля 2025 г. — СПб. : ПОЛИТЕХ-ПРЕСС, 2025. — С. 316–318. — EDN SROAMJ.
6. ISO/IEC 14977:1996. Information technology — Syntactic metalanguage — Extended BNF. — Текст : электронный // ISO. — URL: <https://www.iso.org/standard/26153.html> (дата обращения: 13.03.2026).
7. Grammar Reference. — Текст : электронный // Lark documentation. — URL: <https://lark-parser.readthedocs.io/en/latest/grammar.html> (дата обращения: 13.03.2026).

СОДЕРЖАНИЕ

Приветствие от компании YADRO	3
Тезисы докладов конкурса-конференции.....	4
Секция Прикладная программная инженерия: продукты, сервисы и системы.....	4
<i>Авдеева Н.А., Самочадин А.В.</i> Разработка интеллектуального агента для помощи в планировании путешествий.....	4
<i>Авдышева П.Г., Котлярова Л.П.</i> Разработка мобильного приложения на платформе Android для коррекции цветовосприятия визуального контента у пользователей с нарушениями цветового зрения.....	5
<i>Алексеев Е.В., Самочадин А.В.</i> Разработка веб-приложения системы вопросов и ответов с поддержкой программного кода и математических формул	7
<i>Андреев В.И., Мамедова Г., Зайцев И.В.</i> Автоматизированная система формирования расписаний образовательных учреждений	9
<i>Антонов А.А., Ковалев А.Д.</i> Метод выбора стратегии рендеринга Web-приложений	11
<i>Асылбашева Ш.Б., Лыченко Н.М.</i> Разработка программных средств для анализа и прогнозирования инцидентов на основе заявок HelpDesk	13
<i>Богданова А.В., Зеленчук И.В.</i> Создание мобильного помощника для написания стихов	14
<i>Божко Д.А., Зайцев И.В.</i> Разработка веб-приложения для курса по языку SQL на основе СУБД Postgresql	16
<i>Бавыкин А.А., Воинов Н.В.</i> Разработка AI-агента для трекинга питания с использованием гибридного векторного поиска.....	18
<i>Бонилья Тельес Ж.А., Ковалев А.Д.</i> Разработка веб-редактора почтовых сообщений с поддержкой drag-and-drop и API для пользовательских виджетов	19
<i>Брютова С.Д., Леонтьева Т.В.</i> Проектирование масштабируемой серверной архитектуры приложения для индустрии красоты	21
<i>Быкова К.А., Прокофьев О.В.</i> Разработка обучающего сайта-симулятора инвестиций для начинающих пользователей	23
<i>Ван Ю., Амосов В.В.</i> Онлайн-чат на основе Spring Boot и Vue C возможностью изменений для нужд пользователя.....	25
<i>Васин Н.А., Шмаков В.Э.</i> Система автоматизированного взаимодействия с заказчиком в управлении проектом	26
<i>Вешикова Д.А., Касьянова И.А., Комарова Е. В.</i> Разработка алгоритма семантической сегментации топографических карт на основе методов глубокого обучения	28
<i>Вяземский А.А., Леонтьева Т.В.</i> Поиск оптимальных расположений модели для 3D-печати	30
<i>Вяземский А.А., Онуфриенко Н.С., Леонтьева Т.В.</i> Приложение для автоматизации расстановки мебели по набору правил для BIM среды проектирования	32
<i>Гончаров Р.Р., Загидулин Р.Р., Маслаков А.П.</i> Разработка клиент-серверного приложения для подсчёта труб	34
<i>Домбровская В.И., Селин И.А.</i> Разработка Android-приложения для ведения пародонтологического лечения пациента	35
<i>Домингес Молина Р.Ю., Селин И.А.</i> Разработка прототипа системы электронной коммерции на базе микросервисной архитектуры	37
<i>Дубинин Б.С., Петров О.Н.</i> Разработка приложения анализа учебных планов нового поколения.....	39

<i>Дубинин Б.С., Петров О.Н.</i> Разработка приложения визуализации учебных планов нового поколения	41
<i>Думаев Р.И., Молодяков С.А.</i> Разработка веб-приложения объяснимой системы поддержки принятия решений для анализа КТ-изображений	43
<i>Дьяконов О.А., Пушин М.В., Болсуновская М.В.</i> Гибридная интеллектуальная система рекомендации стилей одежды STUDQR: архитектура и первые результаты.....	45
<i>Еремина К.И., Воинов Н.В.</i> Разработка модуля управления документами для AI-ассистента на основе LLM и RAG.....	47
<i>Ерофеев Д.А., Горавнева Т.С.</i> Разработка приложения для автоматической генерации блок-схем	49
<i>Ефремов А.Д., Шошмина И.В.</i> Разработка программного комплекса «Мобильный контролер» для газовой компании	50
<i>Инчина К.А., Верзунов С.Н.</i> Применение нейросетевых моделей для краткосрочного прогнозирования скорости ветра	52
<i>Загидулин Р.Р., Зайцев И.В.</i> Разработка системы анализа достижений студентов.....	54
<i>Иванов М.К., Юсупова О.А.</i> Разработка CRM-системы для производства	56
<i>Касьянова И.А., Вещикова Д.А., Комарова Е.В.</i> Разработка модели для распознавание зубочелюстных дефектов на изображениях в формате DICOM	57
<i>Кириллова Д. Д., Никифоров И. В.</i> Автоматизация преобразования неструктурированного текста презентаций в формат Markdown для интеграции в корпоративную базу знаний.....	58
<i>Кашмак С.В., Шемякин И.А.</i> Реализация серверной части высоконагруженного веб-приложения	61
<i>Коба А.Ю., Воинов Н.В.</i> Универсальный механизм просмотра файлов в интерфейсе 1С:Предприятие.....	62
<i>Косминская О.А., Маслаков А.П.</i> Нейросетевая обработка показаний приборов в реальном времени.....	64
<i>Костюков Б.Н., Юсупова О.А.</i> Разработка модуля отображения тегов на основе AI-анализа отзывов	66
<i>Ключко Е.А., Хахина А.М.</i> Клиент-серверное приложение для определения позиции и траектории мяча: на примере волейбола.....	68
<i>Краснов В.В., Александрова О.В.</i> Разработка программного модуля краткосрочного прогнозирования гипогликемии на основе временных рядов непрерывного мониторинга глюкозы (CGM)	70
<i>Крючков С.А., Юсупова О.А.</i> Разработка веб-приложения для управления и визуализации логистических данных	72
<i>Костюченко Т.Р., Гончаров Р.Р., Спиваков М.Б., Малеев О.Г.</i> Разработка веб-платформы для визуализации и анализа результатов распознавания действий человека в офисной среде.....	73
<i>Котова Я.С., Петров А.В.</i> Создание платформы отраслевого профилирования для подготовки инженеров	74
<i>Ларионенко А.А., Орлов Е.С.</i> Микросервисная платформа цифровых университетских сервисов с централизованной аутентификацией и модулем контроля посещаемости	75
<i>Кулагин И.А., Прокофьев О.В.</i> Разработка корпоративной информационно-аналитической системы для оценки вероятности геологического успеха (GPOS)	77
<i>Кумачев А.А., Баилачева Е.А., Гончаров А.В.</i> Применение Istio для сбора информации об активности пользователей в микросервисном приложении и организации service mesh	79

<i>Лаврова Л.А., Туральчук К.А.</i> Исследование влияния ROI-ограничения на качество интерпретации моделей машинного обучения для МРТ головного мозга	81
<i>Лисовская Е.А., Яковлев А.Н.</i> Разработка системы интерактивной 3D-визуализации результатов сейсмического моделирования на суперкомпьютерных системах	83
<i>Литвин С.Р., Медведев Б.М.</i> Микросервис для автоматической модерации пользовательского контента на основе нейронных сетей	84
<i>Лысков Р.Ю., Шмаков В.Э.</i> Разработка мессенджера с микросервисной архитектурой для малого бизнеса	86
<i>Месропян А.В., Маслаков А.П.</i> Разработка программного обеспечения для автономного аудиорегистратора с функцией подавления шумов	88
<i>Макарова А.А., Малыхина Г.Ф.</i> Интерпретируемая классификация диабетической ретинопатии	90
<i>Мишанина Е.А., Шелковников Д.С., Амбражей А.Н.</i> Программный комплекс управления роботизированной линией Industry 4.0 на платформе 1С	92
<i>Морозов Н.Д., Александрова О.В.</i> Программная реализация системы прогнозирования гипогликемии на основе искусственного интеллекта для мобильных устройств	94
<i>Муратов А.Р., Шемякин И.А.</i> Разработка качественного клиентского интерфейса для социальной сети	95
<i>Немчинов И.М., Шмаков В.Э.</i> Разработка сервиса по аренде водного транспорта	97
<i>Неянин В.Ю., Никифоров И.В.</i> Анализ логистических данных транспортно-экспедиторской компании для автоматизации расчёта коммерческих предложений	99
<i>Никитин А.Э., Семенова-Тян-Шанская В.А.</i> Разработка приложения для формирования базы данных по судам-прототипам	101
<i>Николаев М.Ю., Воинов Н.В.</i> Разработка метода персонализации интерфейса мобильного приложения на основе поведения пользователя	103
<i>Олейникова А.Д., Самочадин А.В., Самочадина Т.Н.</i> Сервис управления отсутствиями в микросервисной корпоративной платформе	105
<i>Осокин Д.А., Воинов Н.В.</i> Разработка Android-приложения для позиционирования IoT-устройств без использования GPS	106
<i>Овсянников К.А., Малеев О.Г.</i> Построение поведенческого портрета клиента на основе транзакционных данных для задач персонализации в розничной сети	108
<i>Панкратова К.С., Жаранова А.О.</i> Автоматизированное средство организации процесса вязания изделий с модулем распознавания петель на схемах по вязанию крючком	110
<i>Потапова М.А., Круглов С.К.</i> Разработка интеллектуальной системы мониторинга и управления микроклиматом комнатных растений	112
<i>Скрипкарь А.Ф., Шошмина И.В.</i> Разработка мобильного приложения для персонализированного подбора тренировочных программ на основе анализа параметров тела пользователя	113
<i>Рожнев Н.А., Родимов Г.Р., Комарова Е.В.</i> Архитектурные особенности и границы применимости протокола WebSocket в системах реального времени	115
<i>Рышкова С.Ю., Малеев О.Г.</i> Интеллектуальный модуль планирования питания на основе анализа продуктовых запасов и предпочтений для мобильного приложения	117
<i>Селиверстова А.К., Туральчук К.А.</i> Разработка модуля для проверки учебных знаний на основе LLM	119
<i>Сидоркина М.А., Хахина А.М.</i> Разработка веб-приложения с адаптивным интерфейсом для чтения, ориентированного на пользователей с синдромом дефицита внимания и гиперактивности	121

<i>Скорина К.А., Молодяков С.А.</i> Разработка редактора медиафайлов с интегрированным NLP-модулем для интеллектуального взаимодействия.....	123
<i>Спирчина В.А., Дробинцев П.Д.</i> Разработка iOS-супераппа для интеграции медиа-сервисов	125
<i>Спирidonов В.А., Маслаков А.П.</i> Разработка модуля ERP-системы на языке Scala с использованием фреймворка Global.....	126
<i>Стародубцева К.А., Малеев О.Г.</i> Разработка архитектуры интегрируемого ядра AI-помощника для фитнес-тренировок с контролем техники выполнения упражнений.....	128
<i>Струк А.В., Самочадин А.В.</i> ИИ-помощник для отельеров и специалистов клиентского отдела.....	130
<i>Сусликов В.А., Александрова О.В.</i> Разработка программного модуля для стратификации рисков осложнений сахарного диабета на основе клинико-лабораторных данных	131
<i>Талай Ю.Н., Воинов Н.В.</i> Разработка приложения для обмена сообщениями и онлайн-встреч..	133
<i>Торопов А.М., Беляев А.А.</i> Разработка системы автоматизации продаж игровых предметов и учёта капитала на базе торговой площадки market CS:GO.....	135
<i>Титова Е.Д., Григорьев С.В., Чистякова А.А.</i> Сравнение нейронных сетей, управляющих символьной машиной.....	136
<i>Ушакова М.Г., Малеев О.Г.</i> Мобильное приложение анализа пищевых привычек по покупаемым продуктам и по сокращению их отходов	138
<i>Фомин Н.А., Халиуллин А.Л., Павлов Е.А.</i> Гибридная двухуровневая система подбора партнеров для дейтинг-приложения с астрологическим уклоном.....	140
<i>Цай Ц., Юшков С.Е., Пархоменко В.А.</i> Исследование результатов решения методом анализа иерархий при различных правилах комбинирования свидетельств	141
<i>Царев М.Е., Пархоменко В.А.</i> Изучение влияния выбора функции дефаззификации нечётких треугольных чисел на выявление предвзятых экспертов	143
<i>Чемодуров А.А., Медведев Б.М.</i> Разработка офлайн-режима для мобильного и веб-приложения организации совместного досуга	145
<i>Халиуллин А.Л., Фомин Н.А., Павлов Е.А.</i> Контекстно-зависимая рекомендательная система контента для астрологического приложения	147
<i>Чубаров Д.А., Прокофьев О.В.</i> Разработка платформы для репетиторов с использованием методов машинного обучения.....	148
<i>Чурина М.А., Зайцев И.В.</i> Разработка внешнего модуля тестирования для LMS Moodle на основе стандарта LTI 1.3	150
<i>Шлякова А.С., Ковалев А.Д.</i> Программный сервис few-shot обучения для автоматизированного построения вероятностного психологического профиля пользователя по текстам ВКонтакте ..	152
<i>Шалыгин И.Д., Барахнин В.Б.</i> Использование алгоритмов компьютерного зрения в приложении для автоматизации обработки деталей на лазерных станках	154
<i>Шемякина В.П., Горавнева Т.С.</i> Разработка приложения для формирования порядка посещения врачей в медицинском центре	155
<i>Шматков А.А., Комарова Е.В.</i> Разработка фреймворка для цифровой обработки музыки с использованием нейронных сетей	157
<i>Щукин Н.А., Туральчук К.А.</i> Разработка торгового бота для алгоритмической торговли с использованием алгоритмов машинного обучения.....	159
<i>Шушин Д.С., Дробинцев П.Д.</i> Разработка аналитических витрин для агрегации и анализа неунифицированных производственных данных в нефтегазовом секторе.....	161
<i>Щеглов А.И., Семенова-Тян-Шанская В.А.</i> Разработка справочника терминов и определений по судостроению на основе нормализации разнородных данных	163

<i>Ян Кай, Медведев Б.М.</i> Разработка системы анализа и прогнозирования цен на сельскохозяйственную продукцию на основе модели SARIMA	165
<i>Балтобаев А.Р., Хмельва И.В.</i> Иерархическая систематизация генерации целеориентированных процедурных анимаций	167
Секция Инструментальные средства и технологии проектирования и разработки ПО	169
<i>Айнализев Н.З., Комарова Е.В.</i> Развертывание SPA-приложений в объектном хранилище как альтернатива классическому веб-серверу в DevOps-платформе	169
<i>Арбаева А.Ф., Леонтьева Т.В.</i> Разработка программного каркаса для построения серверно-управляемого пользовательского интерфейса на языке Dart и оценка его влияния на скорость внедрения функциональности, сопровождаемость и производительность клиентских приложений	171
<i>Арбаева А.Ф., Леонтьева Т.В.</i> Разработка системы удалённой настройки и управления функциональными возможностями для приложений на языке Dart	173
<i>Ахмедов Р.Э., Павлов Е.А.</i> Разработка высокоинтерактивного Honeypot с использованием обучения с подкреплением для защиты веб-приложений	174
<i>Бабин Е.А., Туральчук К.А.</i> Разработка no-code конструктора кроссплатформенных приложений с runtime-загрузкой контента на базе Unity	176
<i>Булатов Д.Е., Леонтьева Т.В.</i> Разработка серверного приложения системы «умный дом» с интеллектуальной подсистемой классификации событий на основе машинного обучения.....	178
<i>Баландин Д.А., Никифоров И.В.</i> Разработка и автоматизация развертывания отказоустойчивого кластера DeckHouse.....	180
<i>Булыкина Е.С., Щукин А.В.</i> Исследование способности LLM к восстановлению эллипсисов....	182
<i>Вяткин А.А., Воинов Н.В.</i> Повышение масштабируемости CI/CD-инфраструктуры путем контейнеризации Gitlab Runner в Kubernetes	184
<i>Гайнетдинов И.Р., Юсупова О.А.</i> Декларативный фреймворк для генерации ETL-пайплайнов на основе YAML-конфигураций.....	186
<i>Ганиуллин Р.Р., Прокофьев О.В.</i> Оптимизация конфигурации PostgreSQL на основе алгоритма SMAC.....	187
<i>Грибкова Е.М., Прокофьев О.В.</i> Исследование и верификация механизмов оптимизации хранения структуры данных jsonb в СУБД PostgreSQL 18	189
<i>Григорьева М.К., Молодяков С.А.</i> Разработка программного инструмента для мониторинга соблюдения соглашений об уровне сервиса в копоративных процессах.....	190
<i>Дьяченко М.А., Воинов Н.В.</i> Разработка методов автоматизированной генерации тест-кейсов для кастомизируемых API-запросов.....	192
<i>Гудков В.О., Сараджишвили С.Э.</i> Разработка утилиты с графическим интерфейсом для ручных запросов к RPC микросервисов.....	193
<i>Гудкова П.О., Никифоров И.В.</i> Автоматизированная система мониторинга и анализа загрузки серверов центров обработки данных.....	195
<i>Денисова Ю.В., Бородянский Ю.М.</i> Разработка матрицы классификации задач SDLC для селективного внедрения генеративного искусственного интеллекта в процесс разработки ПО	197
<i>Джичко А.О., Шемякин И.А.</i> Проектирование отказоустойчивой архитектуры нагруженного веб-сервиса	199

<i>Дюков Н.А., Болсуновская М.В.</i> Интеграция журналов аудита системы управления секретами OpenVault с централизованным стеком мониторинга на основе Loki и Grafana в Kubernetes-кластере	201
<i>Евсеева В.В., Андрианова Е.Е.</i> Сравнение производительности СУБД PostgreSQL и ClickHouse в качестве целевых хранилищ в ETL-пайплайне для обработки временных рядов	203
<i>Егоров А.С., Яковлев А.Н.</i> Разработка системы векторизации и семантического поиска с SQL-интерфейсом и масштабированием на несколько графических процессоров	205
<i>Емишанова П.И., Прокофьев О.В.</i> Разработка клиентской части no-code веб-сервиса для визуального проектирования ETL-процессов	206
<i>Ершов С.С., Черноруцкий И.Г.</i> Обучение нейронных сетей на основе методов с экспоненциальной релаксацией	208
<i>Зезина К.М., Болсуновская М.В.</i> Аутентификация и авторизация в распределенных Realtime-системах	209
<i>Злобин П., Андрианова Е.Е.</i> Сравнительный анализ эффективности генеративного искусственного интеллекта в процессах проектирования схем реляционных баз данных	211
<i>Иглаков Т.Ю., Прокофьев О.В.</i> Разработка серверной части No-Code веб-сервиса для автоматизации ETL-процессов	213
<i>Иванов Р.А., Александрова О.В.</i> Разработка системы аутентификации, авторизации и аудита для REST API	215
<i>Илюшкин А.Д., Селин И.А.</i> Корпоративное хранилище данных и поставка верифицированных данных с помощью систем «IC:Аналитика» и «ETL»	216
<i>Казаков В.Е., Юсупова О.А.</i> Интеллектуальный помощник с использованием больших языковых моделей с подходом RAG	218
<i>Карасев В.И., Самочадин А.В.</i> Средства генерация UML-документации для унаследованного кода	220
<i>Козяков Д.А., Болсуновская М.В.</i> Повышение надёжности инфраструктурного кода посредством статического анализа YAML-файлов Helm-чартов	221
<i>Кондрюков Д.С., Пархоменко В.А.</i> Разработка системы генерации модульных и API тестов на Java с использованием LLM	223
<i>Коновалов А.С., Леонтьева Т.В.</i> Веб-сервис обработки больших данных на основе декларативного описания трансформаций	225
<i>Кочетков И.А., Семенова-Тян-Шанская В.А.</i> Разработка и обеспечение безопасного функционирования веб-приложения	227
<i>Коноплёв Н.В., Ковалев А.Д.</i> Разработка программного комплекса для обнаружения голосового фишинга в режиме реального времени на основе методов машинного обучения	230
<i>Котельникова К.А., Мордвинов Д.А.</i> Автоматический вывод графов взаимодействий в микросервисных архитектурах на основе фреймворка Spring	232
<i>Кудрявцев А.К., Прокофьев О.В.</i> Разработка инструмента автоматизации UI-тестов по словесному описанию кейсов с помощью AI	233
<i>Кушнарева С.В., Амосов В.В.</i> Набор средств для разработки социальной сети нескольких мобильных приложений	235
<i>Литвак А.И., Хитров Е.Г.</i> Методы управления контекстом в системах на основе LLM	237
<i>Мирзоев Х.Б., Пархоменко В.А.</i> Исследование и экспериментальное сравнение средств проверки валидности JSON-моделей на Python	239

<i>Мартюшова Д.Н., Самочадин А.В.</i> Разработка платформы Quxito для управления жизненным циклом микросервисных приложений	241
<i>Меркадо Оудалова Д.А., Никифоров И.В.</i> Формализованный подход к выбору метода Model Checking для BPMN-моделей	243
<i>Митрошин Д.В., Петров А.В.</i> Исследование эффективности различных стратегий организации Worker Pool при обработке конкурентных задач на языке Go	245
<i>Курташ Н.С. Молодяков С.А.</i> Программные сервисы для автоматизированной верификации систем компьютерного зрения с использованием симуляционных данных	247
<i>Мурзич О.А., Пархоменко В.А.</i> Исследование и экспериментальное сравнение constraint-based средств тестирования на Python	249
<i>Набережнов Д.А., Юсупова О.А.</i> Разработка плагина Database User Federation для интеграции Keycloak с существующими базами данных пользователей	251
<i>Недорезов Д.Д., Самочадин А.В., Самочадина Т.Н.</i> Разработка средств генерации автоматизированных тестов для различных платформ	252
<i>Новиковский К.В., Преображенский А.П.</i> Метод интеллектуального распределения вычислительных задач в программных системах на основе машинного обучения	254
<i>Оржеховский А.Н., Пархоменко В.А., Щукин А.В.</i> Сопоставление фреймворков контроля качества данных и мониторинга при прогнозировании временных рядов	256
<i>Петрова Ю.Д., Жаранова А.О.</i> Автоматизированное средство планирования астрономических наблюдений с прогнозированием локальной видимости полярных сияний на основе машинного обучения	258
<i>Плискин Н.В., Туральчук К.А.</i> Разработка инструмента автоматического анализа архитектурных особенностей программных систем на C#	260
<i>Попов Е., Коликова Т.В.</i> Разработка инструмента универсального мониторинга производительности приложений	262
<i>Поливанова А.А., Дробинцев П.Д.</i> Повышение эффективности процесса внесения изменений в ИТ-инфраструктуру на основе аналитических записок	263
<i>Примаков А.В., Пшеничная К.В., Капонен В.А.</i> Методы машинного обучения в задаче определения остаточного сопротивления маломерного судна	265
<i>Прокофьев А.А., Герасимов А.С.</i> Разработка программного менеджера локально развёртываемых больших языковых моделей	267
<i>Руденко М.А., Пархоменко В.А.</i> Сравнение методов приоритизации регрессионных тестов	269
<i>Потехин А.В., Андрианова Е.Е.</i> Разработка системы мониторинга и оптимизации производительности реляционных баз данных	270
<i>Скарлыгина М.Д., Яковлев А.Н.</i> Система оркестрации фаззинг-тестирования на основе суперкомпьютерных технологий	272
<i>Смирнова И.С., Коган О.А., Слатик А.И., Советус Н.В.</i> Разработка программного инструмента анализа зеркалированного сетевого трафика для тестирования промышленных Ethernet-коммутаторов	274
<i>Семенов С.А., Воинов Н.В.</i> Реализация RAID-массивов с тремя контрольными суммами для восстановления данных	276
<i>Спирчина Я.А., Маслаков А.П.</i> Разработка модуля анализа SDK для обработчика apk-файлов RuStore	277
<i>Спиваков М.Б., Костюченко Т.Р., Маслаков А.П.</i> Разработка серверной части генератора автоматизированных тестов на основе ИИ	279

<i>Солодовников С.Ф., Никифоров И.В., Устинов С.М. Методика автоматизации тестирования облачных сервисов в распределенных инфраструктурах</i>	280
<i>Сырбу Г.В., Федорова С.К., Малеев О.Г. Платформа контекстного чат-бота на основе GraphRAG для работы с большими объемами документов</i>	282
<i>Старунская Е.И., Никифоров И.В. Соломонов М.С. Разработка приложения для анализа активности репозиторий GitHub</i>	284
<i>Устинов И.В., Никифоров И.В. Методика миграции телеметрических данных из строковой реляционной СУБД MySQL в колоночное хранилище ClickHouse</i>	287
<i>Чернецов А.М., Гончаров А.В. Современные способы взаимодействия с API без явного декларативного управления намерениями</i>	290
<i>Чумакова О.Р., Ковалев А.Д. Разработка программного обеспечения для технического и UX аудита веб-сайтов с помощью методов машинного обучения</i>	292
<i>Финогенова А.А., Дробинцев П.Д. ИИ Агент для автоматического заполнения и распознавания корректности текстового документа на основе пользовательского запроса</i>	294
<i>Шевцов А.В., Малеев О.Г. Система анализа посещаемости торговых залов на основе видео с камер наблюдения</i>	296
<i>Шожат А.А., Леонтьева Т.В. Применение искусственного интеллекта для персонализации тренировочного процесса</i>	297
<i>Янбеков Р.Р., Никифоров И.В. Алгебраические модели согласованности распределённых систем хранения и обработки данных</i>	299
<i>Яровой В.Д., Шошмина И.В. Разработка NFS-сервера на языке Rust в пользовательском пространстве Linux</i>	301
<i>Секция Методы, алгоритмы и подходы в теории программной инженерии</i>	303
<i>Авдышева П.Г., Зиятдинова А.Р., Яковлев А.Н. Исследование эффективности нейросетевого ансамблевого алгоритма Random Forest на архитектуре GPU (Tesla K40) с использованием суперкомпьютерных систем</i>	303
<i>Абубакурова З.М., Лыченко Н.М. Прогнозирование классов индекса качества воздуха на основе графовых нейронных сетей</i>	304
<i>Азаров И.И., Пархоменко В.А. Исследование чувствительности параметров алгоритма АЕМ-СОМ при разрешении конфликтов в групповом принятии решений</i>	306
<i>Багрова А.В., Черноруцкий И.Г. Реализация алгоритма роевой оптимизации для систем логистики множества независимых программно-аппаратных комплексов</i>	308
<i>Балышев А.М., Косарев Д.С. Разработка компилятора для защиты от атак на цепочки поставок</i>	310
<i>Гуляев И.В., Никифоров И.В. Архитектура облачной платформы дискретно-событийного моделирования на языке программирования Go</i>	311
<i>Григорьев А.Д., Карпов Л.Н., Муравьев Е.А. Библиотека компиляторов синтаксических конструкций-расширений контролируемого естественного языка спецификации</i>	314
<i>Дорзея Л., Прокофьев О.В. Проектирование и реализация многоязычного приложения реального времени на основе логики непрерывных сессий</i>	316
<i>Есаков М.Р., Терещенко Д.А., Пархоменко В.А. Комбинация эвристических алгоритмов NIVES, GA, ННО для повышения совместимости экспертов в групповом ранжировании альтернатив</i>	317
<i>Еникеев И.Р., Пархоменко В.А. Экспериментальное сравнение статических анализаторов PVS-Studio и SonarQube</i>	319

<i>Зиятдинова А.Р., Уткин Л.В., Константинов А.В.</i> Оценка эффекта лечения в задачах анализа выживаемости с использованием метода перегруппировки данных.....	321
<i>Клеблеева Э.М., Тутыгин В.С.</i> Адаптивная оценка шума измерений в фильтре Калмана с использованием нейронных сетей	323
<i>Каралюс Н.Ю., Сергеев А.В.</i> Система автоматической бесшовной ротации криптографических ключей авторизации в высоконагруженной микросервисной платформе.....	325
<i>Коробицына О.В., Скуднева Е.В.</i> Реализация параллельных вычислений траекторий фазового портрета системы обыкновенных дифференциальных уравнений на основе технологии MPI... ..	327
<i>Королев Д.О., Никифоров И.В.</i> Дискретно-событийное моделирование как инструмент повышения надежности облачных инфраструктур	329
<i>Кубов Н.А., Никифоров И.В., Устинов С.М.</i> Статический анализ имитационных моделей и синтез планов их исполнения в распределенных средах.....	331
<i>Кубов Н.А., Сабуткевич А.М., Никифоров И.В.</i> Актуальность методов статического анализа, компиляции и управления исполнением имитационных моделей в распределенных средах	333
<i>Ларин А.В., Дробинцев П.Д.</i> Оценка просодической информации в самообучающихся представлениях речи: сравнительный анализ HuBERT И WavLM	335
<i>Лонишин М.Р., Сабуткевич А.М., Кубов Н.А., Самочадин А.В.</i> Оценка коммуникационных издержек при распределенном обучении LLM на основе дискретно-событийного моделирования	337
<i>Носкова А.С., Сабинин О.Ю., Пархоменко В.А.</i> Реализация и применение расширенного метода TOPSIS для группового принятия решений на примере выбора оптимального датасета для SQL-to-Text	339
<i>Петров В.Е., Касьяненко Г.А., Никулина Р.Н., Пархоменко В.А.</i> Методы решения задач в условиях неопределенности: открытая компьютерная реализация	341
<i>Павлов Е.С., Воинов Н.В.</i> Разработка сервиса для криптографической подписи PDF-документов	343
<i>Смекалов А.А., Шошмина И.В.</i> Сравнительный анализ механизмов защиты от MEV-атак на децентрализованные биржи.....	344
<i>Сафуанов Р.А., Шмаков В.Э.</i> Построитель графа транспортной сети на основе модели Тетро ...	345
<i>Соколов И.А., Никифоров И.В.</i> Автоматизация настройки и верификации требований безопасности для операционных систем и программного обеспечения	347
<i>Трубач К.И., Иванюк А.А.</i> Синтез специализированного микропрограммного ядра для управления физически неклонируемыми функциями	349
Секция Современная программная инженерия в практике компании YADRO ..	351
<i>Алымова К.П., Савинцев И.М., Касилов В.А.</i> Автоматизация создания пользовательских средств анализа кода с помощью clang-tidy и clang-query.....	351
<i>Анисимова А.В., Селин И.А.</i> Разработка программного модуля разбора и анализа архитектурных диаграмм формата Draw.io	353
<i>Бзычкин Е.М., Бордуков Д.А., Адодина Ю.В., Смирнов К.К.</i> Различные подходы в применении больших языковых моделей в приложении Amphimixis	354
<i>Братенков А.М., Степина Н.О., Юсупова О.А., Никифоров И.В.</i> Применение численных методов понижения разрядности данных при решении комбинаторных задач на платформе RISC-V	356
<i>Бугаев А.С., Сичкар Г.К., Васенина А.И.</i> Адаптация алгоритма LZ4 для обработки разрывных буферов в блочном уровне ядра Linux	358

<i>Бутаковский Е.В., Демешикина К.А., Амосов В.В.</i> Программная и аппаратная реализация алгоритма Флойда-Уоршелла.....	360
<i>Гаврилова П.В., Карнаухова А.К., Киселёв С.Р., Амосов В.В.</i> Реализация алгоритма DFS (обхода в глубину) для простейшего микропроцессорного набора (DP32) и ПЛИС.....	361
<i>Гарькуша В.А., Ганин Е.А., Игнатушин С.А., Сатаев С.В., Корелин О.Н., Кобляков Д.А</i> Реализация математических операций над матрицами на FPGA	362
<i>Денисов-Элерс Э.Ф., Шевченко А.О., Маслаков А.П.</i> Автоматизация тестирования операционной системы KvadraOS	364
<i>Гринько А.Н., Александрова О.В.</i> Разработка подхода к автоматизации сборки и дистрибуции серверного эмулятора API Swordfish.....	366
<i>Истаев Э.Э., Селин И.А.</i> Разработка единого промежуточного представления для анализа архитектурной документации	368
<i>Кузьмин Р.С., Шагалова П.А.</i> Разработка и оптимизация алгоритма размытия по Гауссу с использованием фреймворка MLIR.....	370
<i>Лузин К.Л., Селин И.А.</i> Разработка парсера директорий для анализа архитектурной документации с применением препроцессинга	371
<i>Малая В.В., Медведев Б.М.</i> Разработка программных средств сжатия заголовков пакетов данных в базовой станции 4G	372
<i>Милорадов Е.А., Чувашевский С.И., Александрова О.В.</i> Добавление новых типов хранилищ в сервер-эмулятор API-интерфейса Swordfish для тестирования CSI-инструмента Kubernetes.....	374
<i>Ромашов В.А., Никифоров И.В., Устинов С.М.</i> Конвейер данных: парсинг, индексация и синхронизация в корпоративной инженерной среде	376
<i>Николаев А.Д., Парфенов Я.М., Александрова О.В.</i> Архитектурные решения интеграции разнородных систем хранения данных в CSI-драйверах.....	378
<i>Онуфриенко Н.С., Богач Н.В.</i> Разработка встраиваемого редактора логических схем	380
<i>Савинцев И.М., Алымова К.П., Касилов В.А.</i> Проектирование системы автоматической оценки студенческих работ из удаленного репозитория	382
<i>Пономарев А.А., Никифоров И.В.</i> Сравнение событийно- и сервис-ориентированной архитектур в системах ОАМ и влияние архитектурного подхода на трудоёмкость развития системы.....	384
<i>Смирникова Л.Е., Селин И.А.</i> Разработка компаратора для анализа архитектурной документации программных систем	386
<i>Тимофеев М.Ю., Смирнов А.С., Климашов Д.М., Амосов В.В.</i> Программная и аппаратная реализация алгоритма сортировки расчёской.....	388
<i>Флусова С.А., Шошмина И.В.</i> Автонастройка параметров GPU-программ с использованием средств верификации на основе TLA+	389
<i>Черепанов Н.И., Марфина А.Е., Степина Н.О., Никифоров И.В.</i> Применение гибридных операторов TVM и расширения RVV для оптимизации вычислений на архитектуре RISC-V ...	391
<i>Чурикова В.С., Новик М.А., Мишин М.О., Смирнов К.К.</i> Amphimixis: разработка инструмента для автоматизации сборки и профилирования приложений для различных архитектур.....	393
<i>Парфенов А.В., Самодуров Н.С., Фролов Т.С., Кириленко Я.А.</i> Разработка линтера Verihogg для SystemVerilog	395
<i>Шаехмитов Ш.Р., Григорьев С.В., Кутев В.А.</i> Улучшение совместимости RISC-V GPGPU Vortex с библиотекой разреженной линейной алгебры	396
<i>Яцуненко М.И., Селин И.А.</i> Разбор PlantUML-диаграмм для анализа архитектурной документации	398

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

Сборник материалов
научно-практической конференции студентов,
аспирантов и молодых ученых

22–23 апреля 2026 года

Подписано в печать 10.04.2026. Формат 60×84/16. Печать цифровая.

Усл. печ. л. 25,75. Тираж 200. Заказ 2060.

Отпечатано с готового оригинал-макета,
предоставленного редакционной коллегией,
в Издательско-полиграфическом центре Политехнического университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел.: (812) 552-77-17; 550-40-14.