



ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

Институт
компьютерных наук
и кибербезопасности



СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

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

23–24 апреля 2025 года



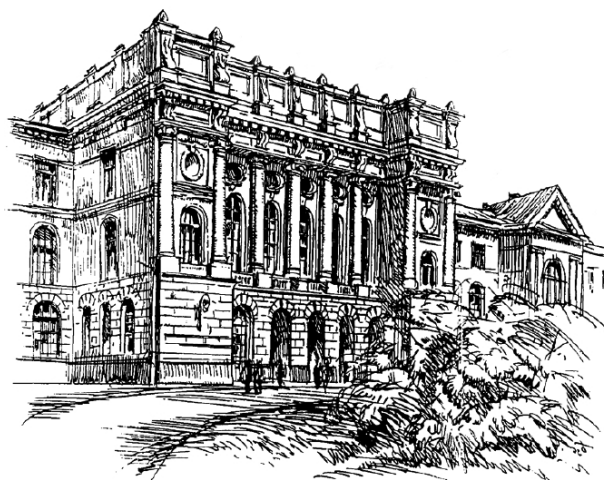
ПОЛИТЕХ-ПРЕСС
Санкт-Петербургский
политехнический университет
Петра Великого

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

23–24 апреля 2025 года



ПОЛИТЕХ-ПРЕСС

Санкт-Петербургский
политехнический университет
Петра Великого

Санкт-Петербург
2025

УДК 004
ББК 32.973
С56

Современные технологии в теории и практике программирования :
сборник материалов научно-практической конференции студентов, аспиран-
тов и молодых ученых, 23–24 апреля 2025 г. – СПб. : ПОЛИТЕХ-ПРЕСС,
2025. – 372 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 охватывают актуальные вопросы реализации программных проектов различного назначения, что делает его полезным как для теоретиков, так и для практиков.

Редакционная коллегия:

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

ISBN 978-5-7422-8965-4

© Санкт-Петербургский политехнический
университет Петра Великого, 2025

Приветствие от компании YADRO

Уважаемые 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».

Проект конференции реализуется уже многие годы. Он стал возможен благодаря вашему интересу к исследованию, стремлению к знаниям и желанию делиться своими мыслями и достижениями. YADRO уже не первый год является партнёром конференции. Мы считаем важным поддерживать молодых учёных и их научные исследования, так как такая поддержка способствует дальнейшему развитию инженерных направлений, в которых работает наша компания.

Каждая статья, представленная в этом сборнике, — это уникальный взгляд на актуальные проблемы и вопросы, которые волнуют молодое поколение. Выражаем благодарность всем авторам за их труд и вдохновение, которые они вложили в свои работы.

Мы уверены, что ваши труды, изложенные на этих страницах, станут важным шагом в вашей дальнейшей карьере.

Евгений Викторович Максимов

Директор по развитию экосистемы и образовательных инициатив YADRO

УДК 004.773

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

РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ЧАТА НА ВЕБ-СОКЕТАХ

В условиях интенсивного развития цифровых коммуникаций возрастают требования к качеству и функциональности онлайн-сервисов. В частности, чат-приложения играют ключевую роль в обеспечении мгновенного обмена информацией между пользователями. Основные требования к таким системам включают высокую скорость работы, отказоустойчивость, удобный интерфейс и расширенные возможности, такие как поддержка мультимедийных вложений и видеосвязи.

Данное исследование направлено на разработку веб-приложения чата в реальном времени, обеспечивающего текстовое общение, передачу файлов, аудио и видеозвонки, а также управление пользователями и возможности масштабирования.

Целью проекта является создание кроссплатформенного веб-приложения, обладающего следующими характеристиками:

1. Чат в реальном времени: мгновенный обмен сообщениями, групповые чаты, уведомления.
2. Обмен файлами: загрузка и отправка изображений, документов с ограничением по размеру и возможностью предпросмотра.
3. Аудио- и видеозвонки: интеграция WebRTC для звонков, поддержка групповых вызовов.
4. Управление ролями и модерация: гибкая система ролей (админ, модератор, пользователь), инструменты для управления контентом и пользователями.
5. Масштабируемость и производительность: архитектура для поддержки большого числа пользователей, оптимизация базы данных и кэширование.

Целью архитектуры системы является обеспечение эффективного взаимодействия между клиентской и серверной частями, обладающими следующими характеристиками:

Серверная часть: Node.js с Express.js для HTTP-запросов и Socket.io для реального времени [1][2]. Клиентская часть: React [3], Next.js, TailwindCSS, Zustand. База данных: MongoDB [4], аутентификация: JWT. Для звонков: LiveKit и WebRTC, для оптимизации запросов Tanstack Query. Основные функции: обмен сообщениями, файлами, редактирование сообщений, видеозвонки, управление пользователями. Приложение оптимизировано для высокой производительности и кроссплатформенности.

Основной функционал системы включает в себя следующие возможности:

1. Обмен сообщениями – мгновенная отправка и получение текстовых сообщений.
2. Поддержка вложений – загрузка изображений, видео и документов.
3. Редактирование и удаление сообщений – обновление сообщений в реальном времени.
4. Аудио- и видеозвонки – использование LiveKit для групповых и личных звонков.
5. Гибкая система управления пользователями – настройка ролей (гость, модератор, администратор), возможность исключения участников.
6. Создание и управление серверами – пользователи могут создавать собственные комнаты для общения с настраиваемыми параметрами.

Для повышения производительности внедрены:

1. Бесконечная подгрузка сообщений через Tanstack Query для плавной работы чата.
2. Оптимизация запросов к БД с использованием индексов и кэширования в MongoDB.
3. WebRTC и LiveKit для видеозвонков с минимальной задержкой. При реализации видеозвонков использовались принципы сетевой передачи данных, описанные в работе Tanenbaum A. [5], что позволило минимизировать задержки и повысить стабильность соединения.
4. Кросс-платформенность благодаря React и Next.js, поддерживающим различные устройства и браузеры.

Дальнейшее развитие проекта предполагает улучшение системы безопасности для обеспечения надёжной защиты данных [6], внедрение персонализированного интерфейса для повышения удобства пользователей, а также интеграцию с внешними сервисами. Важным направлением является расширение функциональности за счёт разработки кроссплатформенного мобильного приложения, что позволит сделать систему более доступной и универсальной.

ЛИТЕРАТУРА

1. McGrath M. JavaScript для начинающих. – Москва: Бомбора, 2024.
2. Crockford D. JavaScript: The Good Parts. – O'Reilly Media, 2008.
3. React Official Documentation. [Электронный ресурс] Режим доступа: <https://reactjs.org/docs/getting-started.html>
4. MongoDB Official Documentation. [Электронный ресурс] Режим доступа: <https://www.mongodb.com/docs/>
5. Tanenbaum A. Computer Networks. 5th ed. – Pearson Education, 2010.
6. OWASP Foundation. OWASP Top Ten Web Application Security Risks. – 2021. – [Электронный ресурс] Режим доступа: <https://owasp.org/www-project-top-ten/>

УДК 004.4

В.Е. Архипов (4 курс бакалавриата),
Н.В. Воинов, к.т.н., доцент

РАЗРАБОТКА TELEGRAM-БОТА ДЛЯ СКАЧИВАНИЯ И ОБРАБОТКИ КОНТЕНТА С YOUTUBE

С развитием цифровых технологий пользователи все чаще сталкиваются с необходимостью скачивания видео контента с платформы YouTube. Однако доступные инструменты могут иметь некоторые ограничения на количество загружаемого контента и его качество либо требуют установки различных плагинов для открытия таких функций, но и с ними не все так просто - может оказаться, что установленный плагин старой версии и он не подходит для работы с YouTube под новые правила.

Telegram активно развивается как мессенджер, добавляя возможность автоматизации некоторых задач с помощью ботов. В связи с этим можно создать удобный инструмент,

который позволит пользователям загружать видео и аудиофайлы с YouTube без использования сторонних плагинов для браузера. Основная цель проекта – разработка Telegram-бота, обеспечивающего скачивание видео по ссылке в различных форматах и упрощенное взаимодействие с интерфейсом для пользователя.

Разработанный Telegram-бот предоставляет возможность скачивания контента с YouTube в различных форматах: видео в исходном качестве, аудиодорожка в формате MP3 и аудиофайл в формате голосового сообщения (OGG). Пользователь отправляет ссылку на видео, после этого выбирает необходимую функцию (видео, аудио, голосовое сообщение), после чего бот загружает контент и отправляет его в чат.

На данный момент тестирование прототипа бота на различных видеороликах показало следующие результаты (возможны погрешности, зависящие от скорости предоставленного интернет-соединения):

- среднее время скачивания пятиминутного видео – 9-12 секунд;
- среднее время обработки аудиофайла – 5-7 секунд;
- минимальные задержки при отправке контента пользователям.

Используемые технологии [1-4]:

- Язык программирования: Python.
- Среда разработки: PyCharm.
- Фреймворк для Telegram-ботов: aiogram.
- Библиотека для скачивания видео: yt-dlp.
- Форматирование и конвертация файлов: ffmpeg (конвертация аудио в MP3, OGG).

Существуют различные инструменты для скачивания видео с YouTube. В данном проекте были рассмотрены две основные библиотеки:

- «yt-dlp» обладает высокой скоростью работы, стабильностью, поддерживает множество форматов и позволяет обходить ограничения YouTube.
- «pytube» — более простой в использовании, но уступает по функционалу из-за отсутствия ключевых обновлений.

В рамках проекта выбор сделан в пользу «yt-dlp», так как он оказался более эффективен для работы с обновленными правилами платформы YouTube.

Разработанный Telegram-бот предоставляет удобный и простой в использовании инструмент для получения видео контента и аудиодорожек с YouTube без лишних проблем с нахождением нужной программы или плагина для браузера с возможностью обхода ограничений на скачивание контента с YouTube в нормальном качестве и без платных подписок.

ЛИТЕРАТУРА

1. Документация yt-dlp. URL: <https://github.com/yt-dlp/yt-dlp>
2. Официальный сайт Telegram API. URL: <https://core.telegram.org/bots/api>
3. Документация aiogram. URL: <https://docs.aiogram.dev/en/latest/>
4. Официальная документация FFmpeg. URL: <https://ffmpeg.org/documentation.html>

УДК 004.65

Я.А. Алексеев (1 курс магистратуры),
Е.Е. Андрианова, ст. преподаватель

МЕТОД НАСТРОЙКИ ЛОГИЧЕСКОЙ ДВУНАПРАВЛЕННОЙ РЕПЛИКАЦИИ В БАЗЕ ДАННЫХ PostgreSQL

PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций. Благодаря свободной лицензии PostgreSQL разрешается бесплатно использовать, изменять и распространять всем и для любых целей — личных, коммерческих или учебных [1].

Логическая репликация в PostgreSQL позволяет передавать изменения на уровне строк, используя модель «публикация-подписка», где один сервер публикует данные, а другие подписываются на них. В отличие от физической репликации, она обеспечивает гибкость, позволяя реплицировать только нужные таблицы и даже фильтровать данные [2].

Поддержка логической репликации появилась начиная с 9 версии PostgreSQL, а начиная с 10 появилась возможность настраивать между различными версиями серверов, что значительно упростило настройку логической репликации [3]. Возможность настройки двунаправленной репликации было представлено в виде сторонних разрешений или программ.

Цель данной работы реализовать метод настройки двунаправленной репликации для операций INSERT и UPDATE, используя встроенные возможности PostgreSQL. Метод будет помогать синхронизировать данные таблиц, расположенных на разных серверах, позволяя ускорить работу между приложениями и создавая единую базу данных. Метод будет актуален для приложений, использующих старую версию PostgreSQL или для операционных систем где доступны только старые версии PostgreSQL, а также для компаний, которые хотят централизовать данные между приложениями.

Для достижения этой цели необходимо решить следующие задачи:

1. Рассмотреть существующие методы для настройки двунаправленной репликации.
2. Реализовать метод для работы двунаправленной репликации.
3. Провести тестирование работы репликации в различных ситуациях.
4. Продемонстрировать работоспособности метода.

PostgreSQL имеет возможность создавать публикацию и подписку при помощи синтаксиса “CREATE PUBLICATION...” [4] и “CREATE SUBSCRIPTION...” [5]. При помощи данной конструкции можно настроить передачу данных с одного сервера на второй и в обратную сторону.

После настройки репликации могут возникнуть такие ситуации как циклы или конфликт данных, что приведет к падению слотов репликации и остановку её работы. Для решения данной проблемы будут использоваться триггеры на таблицах, которые будут перехватывать только запросы, отправленные по репликации при помощи конструкции “ENABLE REPLICA TRIGGER ...”.

Для решения конфликтов вставки данных с одинаковыми первичными ключами для запросов вставки данных триггер будет приостанавливать вставку и не допустит ошибки. А для решения проблемы циклов при обновлении, будет добавлен столбец с типом timestamp, который будет описывать время изменения, в случае если изменения по времени равно или меньше, чем нынешнее значение столбца, то такое изменения не пройдет и заикливание не возникнет.

Настройка синхронизации данных осуществляется при помощи скриптов, использующих расширение dblink, которое позволяет исполнять запросы на удаленном сервере PostgreSQL [7].

В ближайшее время планируется автоматизировать данный процесс, для того чтобы уменьшить время для настройки логической двунаправленной репликации.

Результат работы уже реализован и протестирован на базах данных судов в различных портах. После настройки двунаправленной репликации можно регистрировать новые суда в базе данных, а также изменять их параметры, результаты отображаются на всех настроенных серверах, позволяя ускорить работу операторов, снизить риски расхождения данных судов, а также централизовать данные всех серверов.

ЛИТЕРАТУРА

1. PostgreSQL documentation [Электронный ресурс] Режим доступа: <https://postgrespro.ru/docs/postgresql/17/intro-what-is>
2. PostgreSQL replication [Электронный ресурс] Режим доступа: <https://postgrespro.ru/docs/postgresql/10/logical-replication?>
3. PostgreSQL replication [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/slurm/articles/457512/>

4. PostgreSQL publication [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/sql-createpublication.html>
5. PostgreSQL subscription [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/sql-createsubscription.html>
6. PostgreSQL alter table [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/sql-altertable.html>
7. PostgreSQL dblink [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/contrib-dblink-function.html>

УДК 04.383.3

А.А. Алпеева (2 курс магистратуры),
О.Н. Корелин, к.т.н., доцент,
Д.А. Кобляков, ст. преподаватель

ВЗАИМОДЕЙСТВИЕ ОПЕРАЦИОННЫХ СИСТЕМ PL И PS ЧАСТЕЙ SOC ZYNQ

Цифровая обработка сигналов (ЦОС) играет все более важную роль в самых разных областях, от связи и обработки изображений до радаров и медицинской диагностики. Для эффективной реализации алгоритмов ЦОС требуются высокопроизводительные вычислительные платформы, способные обрабатывать большие объемы данных в реальном времени. Программируемые логические интегральные схемы (FPGA) представляют собой привлекательное решение для этих задач, предлагая гибкость, параллелизм и высокую производительность, которые не всегда доступны в традиционных процессорных системах.

В исследованиях была использована плата Edge Board FZ5: 4 ядра ARM Cortex-A53 (максимальная частота работы – 1,19 MHz, минимальная – 0,29 MHz), Zynq UltraSCALE+, доступно подключение периферийных по USB, мультимедийной камеры по разъему HDMI in, подключение к монитору через DP-miniDP. Плата поставляется с PaddleLiteDemo – демонстративная версия open-source фреймворка для машинного обучения, с нее мы начали изучение взаимодействия ARM и ПЛИС. PaddleLite представлен в версиях на Python и на C++ (уже готовый скомпилированный код, а также исходный код), обе реализации выполняют одинаковые задачи: классификацию, детектирование, сегментацию. Для решения каждой пользователю предоставляется выбор нескольких моделей машинного обучения, на плате доступны изображения-примеры, которые были использованы в качестве входных данных, в файлах конфигурации для каждой задачи создан файл со списком выделяемых объектов. Параметр `use_fpga_preprocess` в PaddleLiteDemo предназначен для включения или отключения предварительной обработки данных на FPGA. Если он установлен в `true`, предварительная обработка изображения (например, изменение размера, нормализация) будет выполняться непосредственно на FPGA, что может значительно ускорить процесс обработки, особенно при работе с большим количеством изображений. Если установлен в `false`, предварительная обработка будет выполняться на CPU или другом процессоре, что может быть медленнее. В наших исследованиях мы использовали обе реализации фреймворка, изменяли способ получения входных данных: конфигурировали json файл на использование USB камеры. На рисунке 1 показаны результаты использования модели детектирования.

Следующим этапом в исследованиях использования связки ARM и Zynq UltraSCALE+ была разработка аппаратных блоков и программируемой логики. Был выбран путь, что и цикле статей [1] – разработка Hard и Soft-CPU (взаимодействие по AXI-шинам) с использованием таких инструментов разработки как Vivado, Vitis (xsct и microblaze-linux-gnu для кросс-компиляции), Petalinux, VM Linux Ubuntu 22.04, рекомендуется сразу производить разработку на Linux системе версии LTS, так как для создания проекта petalinux необходим Linux [2]. Проект для Zynq UltraSCALE+ сконфигурирован с блоком MicroBlaze, как показано на рисунке 2. В работе использовался классический блок, однако сейчас доступен к

использованию RISC-V MicroBlaze. Собрана и запущен проект на ARM и Zynq UltraSCALE+ с одновременно работающими операционными системами на PS и PL частях, которые загружаются посредством скрипта при запуске платы по двум адресам в оперативной памяти. Доступ к консоли Linux на стороне PL части осуществляется через UART интерфейс PS части. После загрузки образов запускаем MicroBlaze сменой значения по адресу, на который отражается AXI-GPIO проекта Vivado, далее подключаемся в PL часть через screen или minicom в PS части.

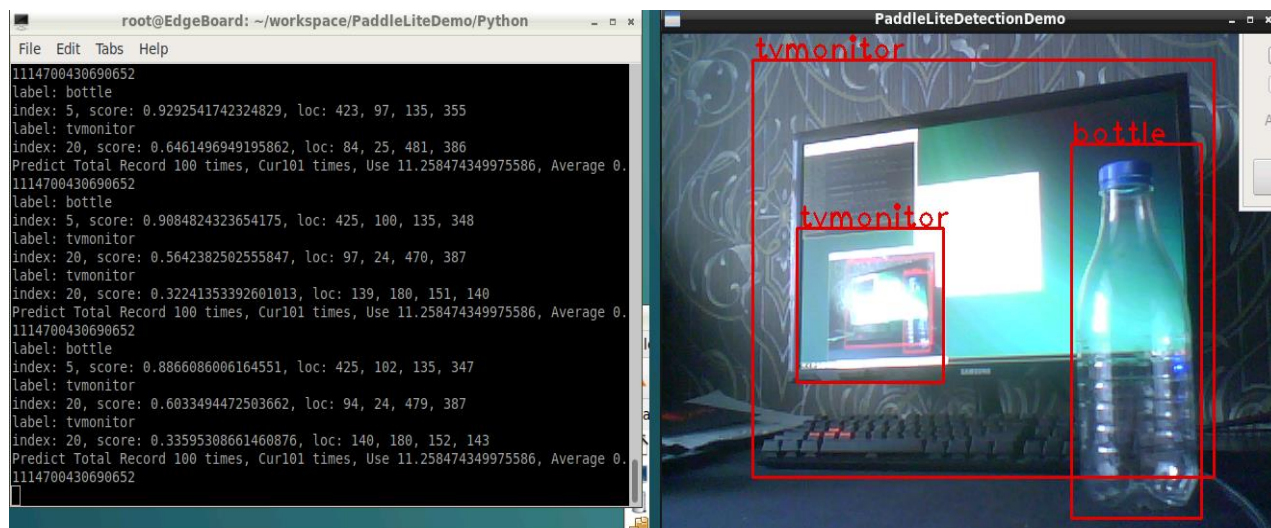


Рисунок 1 – Результат работы модели детектирования.

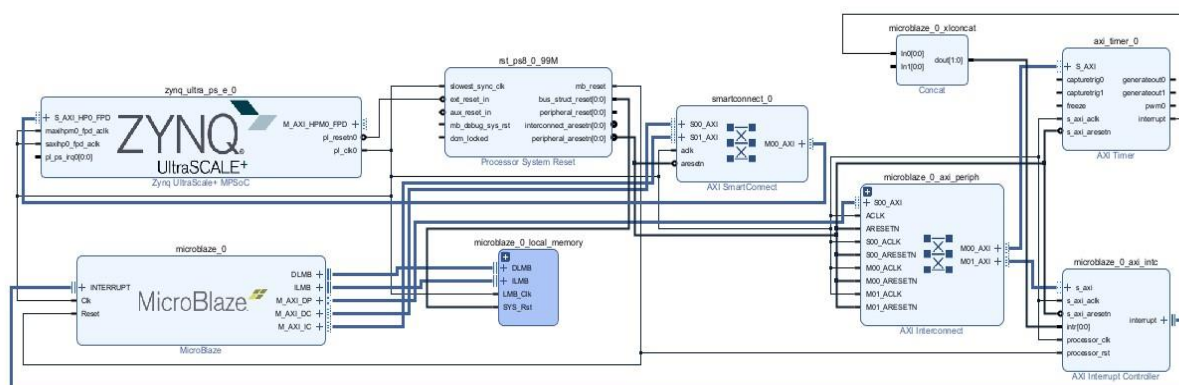


Рисунок 2 – block design проекта.

ЛИТЕРАТУРА

1. Запускаем Embedded Linux на Hard- и Soft-CPU Xilinx Zynq: проект программируемой логики. [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/yadro/articles/845714/>
2. Запускаем Embedded Linux на Hard- и Soft-CPU Xilinx Zynq: сборка операционной системы. [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/yadro/articles/852780/>
3. The Userspace I/O HOWTO – The Linux Kernel Documentation. [Электронный ресурс] Режим доступа: <https://www.kernel.org/doc/html/latest/driver-api/uio-howto.html>
4. Калачев А.В. Интеграция программируемой логики и процессорной системы zynq в ос xillinux // Высокопроизводительные вычислительные системы и технологии. - Алтайский государственный университет, 2023. - С.31-37.

РАЗРАБОТКА ПРОГРАММЫ ОЦЕНКИ ЗАБРАСЫВАЕМОГО ВЕСА СТУПЕНИ РАКЕТЫ

В условиях продолжающейся и активно развивающейся космической гонки вопрос оценки конкурентоспособности ракетно-космических носителей (РКН) РФ становится особенно актуальным. Конкуренция между государствами в области космических технологий требует достоверной информации о характеристиках РКН, что позволит оценить их технический уровень и соответствие мировым стандартам. Разработка методов и программ для анализа этих характеристик способствует объективной оценке возможностей отечественных РКН и их сопоставимости с разработками других стран, что имеет важное значение для принятия стратегических решений в космической отрасли.

Целью данной работы является разработка программы для оценки массы забрасываемого веса ступени ракеты. В рамках работы предполагается использование открытых данных о характеристиках ракеты, а также данных измерений, полученных, например, с радиолокатора. На основе этих данных планируется восстановить траекторию движения ракеты, принимая для упрощения допущение об отсутствии ошибок в измерениях. Разработанная программа позволит проверить достоверность информации о характеристиках ракеты, доступной в открытых источниках, и оценить соответствие фактических параметров заявленным.

Математическая модель, описывающая траекторию движения ступени ракеты, имеет следующий вид:

$$\begin{aligned} \frac{dx}{dt} &= v_x \\ \frac{dy}{dt} &= v_y \\ \frac{dz}{dt} &= v_z \\ \frac{dv_x}{dt} &= g_0 \frac{R_3^2}{(x^2 + y^2 + z^2)} + \frac{\dot{m} g_0 p_{уд}}{m_{кон} + \dot{m}(t_f - t)} \frac{v_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}} \\ \frac{dv_y}{dt} &= g_0 \frac{R_3^2}{(x^2 + y^2 + z^2)} + \frac{\dot{m} g_0 p_{уд}}{m_{кон} + \dot{m}(t_f - t)} \frac{v_y}{\sqrt{v_x^2 + v_y^2 + v_z^2}} \\ \frac{dv_z}{dt} &= g_0 \frac{R_3^2}{(x^2 + y^2 + z^2)} + \frac{\dot{m} g_0 p_{уд}}{m_{кон} + \dot{m}(t_f - t)} \frac{v_z}{\sqrt{v_x^2 + v_y^2 + v_z^2}} \end{aligned} \quad (1)$$

Где x, y, z – это координаты точки, v_x, v_y, v_z – координаты проекции скорости, заданными параметрами являются $p_{уд}$ – удельный импульс, $\dot{m}$ – массовый расход, g_0 – ускорение свободного падения, R_3 – радиус Земли, t_f – конечное время.

Из постановки задачи известен интервал времени и начальные условия:

$t_s \leq t \leq t_f$, где $t_s, t_f, x(t_s), y(t_s), z(t_s), v_x(t_s), v_y(t_s), v_z(t_s)$ – заданные величины.

Требуется найти значение параметра $m_{кон}$ такое, при котором разница между известной траекторией и полученной в результате решения системы уравнений была бы минимальной.

Вычисление разницы отображено в данной функции, для которой и производится поиск минимума:

$$F(m_{кон}) = \frac{1}{n} \sum_{i=1}^n \left(x(t_i) - x_{Tp}(t_i) \right)^2 + \left(y(t_i) - y_{Tp}(t_i) \right)^2 + \left(z(t_i) - z_{Tp}(t_i) \right)^2 \quad (2)$$

Где x, y, z , – это координаты, рассчитываемые из системы, x_{Tp}, y_{Tp}, z_{Tp} – известные координаты траектории.

Разработка программы осуществлена на языке программирования Python с использованием библиотеки SciPy, в которой доступны используемые в данной работе

математические методы для решения систем дифференциальных уравнений и минимизации функции одной переменной.

ЛИТЕРАТУРА

1. Устинов С.М., Зимницкий В.А. Вычислительная математика, Санкт-Петербург : БХВ-Петербург, 2008. - 336 с. - ISBN 978-5-9775-0318-1.
2. Дмитриевский А. А., Лысенко Л. Н. Внешняя баллистика, Москва : Машиностроение, 2005. - 608 с. - ISBN 5-217-03252-9.
3. Matplotlib: Visualization with Python. [Электронный ресурс] Режим доступа: <https://matplotlib.org/>.
4. Fundamental algorithms for scientific computing in Python. [Электронный ресурс] Режим доступа: <https://scipy.org/>.
5. Python 3.13.2 documentation. [Электронный ресурс] Режим доступа: <https://docs.python.org/3/>

УДК 004.4

А.А. Бавыкин, А.С. Савраненко (4 курс бакалавриата),
О.В. Александрова, ст. преподаватель

ОПТИМИЗАЦИЯ СКЛАДСКИХ БИЗНЕС-ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМА НАЗНАЧЕНИЯ АДРЕСОВ

В современном мире растёт потребность в оптимизации бизнес-процессов, связанных с логистикой и складированием. Одним из ключевых факторов, влияющих на эффективность работы склада, является рациональное распределение товаров и снижение зависимости от человеческого фактора. В данной работе рассматривается разработка алгоритма автоматического назначения адресов для хранения товаров на складе, что позволит сократить потери времени на поиск, минимизировать ошибки при размещении и повысить общую продуктивность работы.

Целью данной работы является разработка и реализация алгоритма адресного хранения на складе с учетом особенностей его работы и хранимых товаров.

Для достижения этой цели необходимо решить следующие задачи:

1. Исследование предметной области, включая анализ текущих складских процессов.
2. Анализ существующих систем автоматизации складских бизнес-процессов и их применение.
3. Обдумывание деталей решения и разработка алгоритма, адаптированного к специфике конкретного склада.
4. Реализация прототипа системы и тестирование его эффективности.
5. Внедрение реализованного прототипа в работу склада.

Рассматриваемый склад является специфичным по ряду признаков, при этом в нем отсутствует адресное размещение. В связи с этим разрабатывается алгоритм адресного размещения товаров, и инструмент автоматизированного назначения адресов, учитывающий параметры товаров, особенности правил хранения и требования склада.

Система включает следующие компоненты (рисунок 1):

- Пользовательский интерфейс (GUI – предоставляет возможность передачи необходимых параметров о товаре (вес, габариты, востребованность, дата отгрузки) в автоматизированную систему генерации адресов хранения, а также возможность передачи итогового результата непосредственно в базу данных.
- Алгоритм назначения адресов – рассчитывает наиболее выгодное место для хранения товара, основываясь на введенных параметрах, особенностях правил хранения и требованиях склада.
- База данных – хранит информацию о товарах и их размещении.
- Модульность – разработанный инструмент автоматического назначения адресов может быть использован в приложении как локально, так и интегрирован в другие

системы. На рассматриваемом складе используется 1С КА, поэтому, в частности, можно подключить его туда через расширение, позволяющее вызывать скрипты Python.

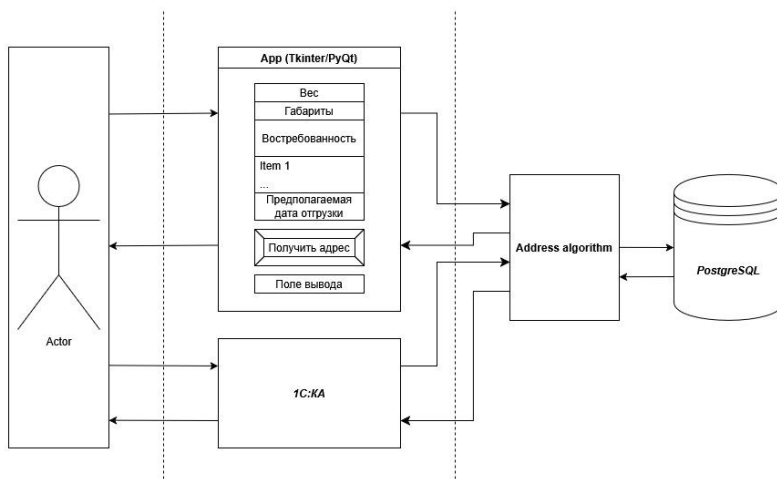


Рисунок 1 – Архитектура системы

Подход будет реализован как Desktop-приложение с помощью языка Python на Tkinter в связке с PostgreSQL.

ЛИТЕРАТУРА

1. Ильичева, А. М. Складская логистика предприятия: актуальность автоматизации системы контроля движения и размещения запасов / А. М. Ильичева, С. А. Радкевич // Аллея науки. – 2019. – Т. 3, № 5(32). – С. 427-431. – EDN PGAZGZ.
2. Савчанчик, А. С. Штрихкодирование: оптимизация учета запасов на складе / А. С. Савчанчик // Научная деятельность в условиях цифровизации: теоретический и практический аспекты : Сборник статей Международной научно-практической конференции, Екатеринбург, 01 июня 2024 года. – Уфа: ООО "Аэтерна", 2024. – С. 124-127. – EDN QSJQWF.
3. Ларман, К. Применение UML 2.0 и шаблонов проектирования / К. Ларман. – 2-е изд. – СПб.: Питер, 2022. – 736 с.
4. Марценко, М. А. Оптимизация работы склада с помощью системы адресного хранения / М. А. Марценко, А. С. Рыжова // Транспорт и логистика: актуальные вопросы, проектные решения и инновационные достижения : Материалы Всероссийской научно-практической конференции, Красноярск, 23 октября 2020 года. – Красноярск: Федеральное государственное бюджетное образовательное учреждение высшего образования "Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева", 2020. – С. 101-103. – EDN GACCBM.
5. 4logist. Метод ABC анализа в логистике. [Электронный ресурс] Режим доступа: <https://www.4logist.com/novosti/metod-abc-analiza-v-logistike/>

УДК 004.4

Е.Е. Балунина (4 курс бакалавриата),
Е.В. Комарова, ст. преподаватель

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОЗДАНИЯ ПЕРСОНАЛИЗИРОВАННОГО МЕНЮ И СПИСКА ПОКУПОК НА ОСНОВЕ РЕЦЕПТОВ, ДОБАВЛЕННЫХ МЕТОДОМ ВЕБ-СКРАПИНГА ПО ССЫЛКЕ

В современном обществе, ориентированном на рациональное потребление ресурсов и тайм-менеджмент, многие люди сталкиваются с проблемой неэффективного расходования времени и денежных средств. Так, согласно исследованию [1], около 12% опрошенных семей признают, что не планируют покупку продуктов, а еще 8% отмечают, что покупают излишки еды из-за недостатка планирования, что приводит к выбрасыванию неиспользованных

продуктов и нерациональной трате семейного бюджета. Кроме того, спонтанные покупки без планирования увеличивают время, затрачиваемое на посещение магазинов.

Одним из способов решения этих проблем можно назвать составление списка покупок, что влечет за собой составление меню. Такое планирование позволит избежать импульсивных покупок и минимизировать количество ненужных продуктов, которые будут выброшены, что экономит деньги и время, затраченное на посещение магазинов.

Таким образом, *целью* данной работы является разработка веб-приложения для создания персонализированного меню и списка покупок. Данное веб-приложение позволит пользователю разумно подойти к вопросу своего питания: собрать используемые рецепты в одном месте, составить из них меню и на его основе автоматически сгенерировать интерактивный список покупок, который позволит сократить растрату времени и денег.

Для достижения поставленной цели необходимо решить следующие *задачи*:

1. Исследовать аналоги веб-приложений для создания персонализированного меню и списка покупок, включая анализ их функциональности, сравнение пользовательских интерфейсов и выявление их преимуществ и недостатков.

2. Сделать обзор и выбор технологий для разработки веб-приложения и веб-скрапера.

3. Спроектировать архитектуру веб-приложения.

4. Разработать веб-скрапер рецептов по ссылке, предоставляемой пользователем.

5. Реализовать пользовательский интерфейс и функциональность создания меню и на его основе генерацию списка покупок.

6. Разработать и реализовать серверную часть веб-приложения.

7. Интегрировать разработанных веб-скрапер в веб-приложение.

В ходе работы были проанализированы аналоги разрабатываемого веб-приложения Mealime [2], Plan to Eat [3], EasyMenu [4], а также проведен сравнительный анализ их функционала и на его основе выделен общий функционал аналогов веб-приложения для создания персонализированного меню и списка покупок.

С учетом выявленных возможностей и недостатков был сформирован список функциональных требований для разрабатываемого веб-приложения. Во-первых, веб-приложение должно поддерживать русский язык, так как это обеспечит его доступность для русскоязычных пользователей, которые являются целевой аудиторией веб-приложения. Во-вторых, необходимо обеспечить возможность добавления рецептов как вручную, так и по ссылке на рецепт со стороннего сайта. Добавление рецептов по ссылке позволит пользователям в один клик интегрировать рецепты из интернета в свою коллекцию рецептов, что значительно быстрее и удобнее, чем вводить их руками или копировать по частям.

Кроме того, важно предусмотреть возможность установки на рецепты пользовательских тегов. Это позволит организовать рецепты с учетом индивидуальных потребностей пользователя, что сделает поиск нужного рецепта более быстрым и эффективным. Также важным требованием можно считать возможность добавления рецептов и составленных меню в избранное, что сделает поиск нужной информации еще более практичным.

Одним из основных требований является возможность заполнения меню, разграниченного по дням и приемам пищи, сохраненными рецептами пользователя или повторения старого меню, добавленного в избранное. На основе текущего меню будет автоматически сгенерирован интерактивный список покупок, в котором можно будет отмечать уже купленные продукты. Это значительно упростит процесс покупки продуктов, позволяя пользователям минимизировать покупку избыточных продуктов, избежать лишних трат и уменьшить время, проведенное в магазине. Дополнительным требованием является возможность распечатать рецепт, меню и список покупок, что позволит взять с собой бумажный список или готовить, используя распечатанные материалы, не отвлекаясь на экран телефона и не рискуя загрязнить телефон в условиях процесса приготовления.

После выявления основных функциональных требований был осуществлен выбор технологий для реализации разрабатываемого веб-приложения.

Клиентская часть веб-приложения реализована с помощью фреймворка Vue.js. Он был выбран среди фреймворков React, Angular и Vue.js на основе сравнительного исследования [5], так как он в большей степени, чем остальные фреймворки, соответствует поставленным требованиям – в отличие от Angular, у него хорошая производительность на небольших проектах, высокая скорость разработки, а также минимальная необходимость в подключении сторонних библиотек, что делает его привлекательнее React.

Для разработки серверной части веб-приложения среди рассмотренных технологий был выбран фреймворк Express.js. Он, в отличие от более тяжеловесных фреймворков, таких как Spring Boot или Laravel, обладает минималистичной архитектурой и высокой производительностью на небольших приложениях. При сравнении с другими фреймворками в статье [6] он показал, что обеспечивает лучшую гибкость, скорость работы и простоту, что особенно важно для разрабатываемого веб-приложения.

Функционал веб-скрапера реализован с помощью библиотеки Puppeteer из-за сочетания её возможности работать с динамическим контентом, относительно высокой скорости работы и простоты установки и настройки, как отмечено в сравнительном анализе [7], что делает его лучшим выбором для разрабатываемого веб-приложения.

ЛИТЕРАТУРА

1. Romani S. et al. Domestic food practices: A study of food management behaviors and the role of food preparation planning in reducing waste // *Appetite*. – 2018. – Т. 121. – С. 215-227.
2. Mealime. – URL: <https://my.mealime.com/>. – (Дата обращения: 10.01.2025).
3. Meal Planner, Recipe Organizer, and Automatic Grocery Lists - Plan to Eat. – URL: <https://www.plantoeat.com/welcome/>. – (Дата обращения: 15.01.2025).
4. EasyMenu. – URL: <https://easy-menu.io/ru/>. – (Дата обращения: 15.01.2025).
5. Karić, A. Comparison of JavaScript Frontend Frameworks – Angular, React, and Vue / A. Karić, N. Durmić // *International Journal of Innovative Science and Research Technology (IJISRT)*. – 2024. – P. 1383-1390. – DOI 10.38124/ijisrt/ijisrt24jun600. – EDN IGADIU.
6. Node.js or PHP? Determining the better website server backend scripting language // *Issues in Information Systems*. – 2023. – DOI 10.48009/1_iis_2023_128. – EDN PQHCQE
7. Уколов, А. Н. Веб-скрапинг: сравнительный анализ популярных библиотек и фреймворков / А. Н. Уколов // *Молодой ученый*. – 2024. – № 30(529). – С. 31-33. – EDN BVAWRT.

УДК 004.891.3

С.А. Бендрышев (4 курс бакалавриата),
И.А. Селин, к.т.н., доцент

ПРЕДСКАЗАНИЕ САХАРНОГО ДИАБЕТА I ТИПА НА ОСНОВЕ МЕДИЦИНСКИХ ПОКАЗАТЕЛЕЙ

Сахарный диабет I типа – аутоиммунное заболевание, при котором происходит разрушение бета-клеток поджелудочной железы, ответственных за выработку инсулина. В результате этого процесса уровень инсулина в организме значительно снижается, что приводит к повышению уровня глюкозы в крови. Раннее выявление и управление этим заболеванием критически важны для предотвращения серьезных осложнений, таких как диабетическая кетоацидоз, сердечно-сосудистые заболевания и хроническая почечная недостаточность. В последние годы машинное обучение стало мощным инструментом в медицинской диагностике, позволяя улучшить точность прогнозирования и найти персонализированный подход к лечению пациента [1].

Для реализации системы предсказания сахарного диабета I типа необходимо собрать данные из различных источников. Это могут быть открытые наборы данных, доступные в

интернете, например, Diabetes Data Set [2] из репозитория UCI Machine Learning Repository или данные из проектов, таких как T1D Exchange [3].

Собранные данные должны включать информацию о демографических характеристиках пациентов, результатах лабораторных исследований, истории болезни и других факторах риска. Эти данные необходимо предварительно обработать: заполнить пропуски, нормализовать значения и закодировать категориальные переменные.

Для построения предсказательных моделей можно использовать различные алгоритмы машинного обучения – деревья решений, случайный лес и градиентный бустинг (например, XGBoost [4]). В ходе экспериментов классификатор на основе XGBoost показал точность 0.68, что подтверждает его эффективность в задачах классификации. Другие модели, такие как случайный лес, продемонстрировали точность около 0.65, что указывает на их сопоставимую, но менее эффективную производительность по сравнению с XGBoost.

Важным шагом является использование методов кросс-валидации для оценки производительности моделей и выбора оптимальных гиперпараметров с целью получить наиболее точные результаты.

Таким образом, целью данной работы является разработка и внедрение алгоритмов машинного обучения для прогнозирования вероятности развития сахарного диабета I типа на основе анализа клинических, генетических и демографических данных пациентов.

Методы исследований в области машинного обучения включают несколько ключевых этапов:

1. Предобработка данных: обработка пропусков, кодирование категориальных переменных, нормализация и стандартизация.
2. Выбор и обучение моделей: кросс-валидация, оптимизация гиперпараметров.
3. Интерпретация моделей: использование методов SHAP для объяснения предсказаний.

Результаты исследований могут быть представлены в виде следующих положений:

1. Сравнительный анализ производительности различных моделей машинного обучения.

Классификатор на основе XGBoost показал наилучшие результаты с точностью 0.68, в то время как случайный лес достиг точности 0.65. Это подтверждает, что градиентный бустинг является более подходящим методом для данной задачи.

2. Идентификация факторов, наиболее значимых для прогнозирования риска развития заболевания.

Анализ важности признаков выявил, что ключевыми факторами являются возраст пациента, уровень глюкозы в крови, индекс массы тела (ИМТ) и семейная история заболевания. Эти факторы играют критическую роль в оценке риска и должны быть внимательно отслеживаемыми.

3. Интерпретация предсказаний.

Использование методов SHAP [7] дало возможность медицинским работникам лучше понять, какие именно факторы влияют на предсказания модели, что может помочь в принятии более обоснованных клинических решений.

Разработка и внедрение алгоритмов машинного обучения для прогнозирования сахарного диабета I типа может значительно улучшить раннюю диагностику, управление заболеванием и качество жизни пациентов. Это исследование открывает новые перспективы для применения технологий машинного обучения в медицине и требует дальнейших исследований для улучшения точности и интерпретируемости моделей.

ЛИТЕРАТУРА

1. Донцов, А. Д. Анализ медицинских данных с помощью методов машинного обучения / А. Д. Донцов, И. А. Селин // 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. 114-116. – EDN IWUZKK.

2. Diabetes Data Set. Набор данных, который используется для исследования и анализа факторов, связанных с диабетом в Индии [Электронный ресурс]. URL: <https://www.kaggle.com/datasets/ankushpanday1/diabetes-prediction-in-india-dataset> (дата обращения: 16.03.2025).
3. T1D Exchange: Платформа, предоставляющая данные о пациентах с диабетом I типа для исследований [Электронный ресурс]. URL: <https://t1dexchange.org/> (дата обращения: 16.03.2025).
4. XGBoost. XGBoost Documentation. [Электронный ресурс]. – URL: <https://xgboost.readthedocs.io/en/latest/> (дата обращения: 16.03.2025).
5. TensorFlow Documentation. Открытая библиотека для машинного обучения, позволяющая строить и обучать нейронные сети [Электронный ресурс]. URL: <https://www.tensorflow.org/learn?hl=ru> (дата обращения: 16.03.2025).
6. Scikit-learn Documentation. Библиотека для машинного обучения на Python, предоставляющая множество алгоритмов для классификации, регрессии и кластеризации [Электронный ресурс]. URL: <https://scikit-learn.org/stable/index.html> (дата обращения: 16.03.2025).
7. Lundberg, Lee. A Unified Approach to Interpreting Model Predictions / NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. 04 December 2017 / Long Beach, USA / ISBN: 9781510860964 / pp. 4768 - 4777

УДК 004.416.6

М.А. Беркоз (2 курс магистратуры),
Н.В. Воинов, к.т.н., доцент

РАСШИРЕНИЕ МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ В СИСТЕМЕ ХРАНЕНИЯ ДАННЫХ «ЗАКРОМА.ХРАНЕНИЕ»

Эпоха больших данных сформировала перед организациями различных отраслей как новые возможности, так и значительные вызовы. По мере экспоненциального роста объёмов информации критически важным становится её эффективное управление, обеспечивающее высокое качество данных, их безопасность и соответствие нормативным требованиям. Современные системы хранения данных должны учитывать вопросы надёжности, контроля доступа и структуризации информации, что требует применения передовых технологических решений и адаптации корпоративных IT-стратегий к изменяющимся условиям цифровой среды [1].

В условиях импортозамещения отечественные решения, такие как «Закрома.Хранение», становятся приоритетными для российских предприятий. «Закрома.Хранение» — это гибридное объектное хранилище, совместимое со стандартом S3, обеспечивающее масштабируемое и безопасное хранение данных [2].

Анализ функциональных возможностей системы показал, что текущая модель управления доступом обладает ограниченной гибкостью. В частности, действующий механизм разграничения прав не позволяет предоставлять избирательный доступ к операциям просмотра настроек прав доступа к бакетам и объектам и их изменения без назначения полного административного контроля. Это не соответствует современным корпоративным стандартам информационной безопасности и управления доступом, что создаёт потенциальные риски для защиты данных.

Таким образом, целью данной работы является увеличение безопасности управления доступом в объектной системе хранения Zakroma.Хранение за счёт детализации модели прав доступа и внедрения поддержки отдельных уровней доступа без необходимости предоставления полного контроля.

Для достижения этой цели необходимо решение следующих задач:

1. Провести обзор существующих моделей управления доступом в объектных хранилищах.

2. Провести сравнительный анализ подходов к разграничению прав в аналогичных системах.

3. Разработать методику детализации управления доступом в Zakroma.Хранение.

4. Реализовать предложенный подход в системе.

5. Оценить влияние модификаций на безопасность и удобство управления доступом.

Для эффективного управления доступом в различных информационных системах и хранилищах данных применяются различные модели и подходы [3,4]. Сравнение популярных подходов поможет выбрать наиболее подходящий для системы метод разграничения прав (таблица 1) [5].

Таблица 1 – Сравнительный анализ моделей управления доступом

Характеристика	RBAC	ACL	ABAC	DAC
Уровень управления	Роли пользователей	Индивидуальные пользователи	Атрибуты пользователей и объектов	Решение владельца объекта
Масштабируемость	Хорошая масштабируемость в крупных системах	Ограниченная при большом числе пользователей	Очень высокая, позволяет гибко адаптировать систему	Масштабируемость ограничена контролем владельцев
Сложность управления	Низкая, права назначаются ролям	Высокая, требуется настройка прав для каждого объекта	Средняя, требует настройки атрибутов	Низкая, контролируется владельцем объекта
Контроль доступа	Управление доступом через роли	Управление доступом через списки прав	Управление доступом на основе атрибутов	Управление доступом решается владельцем объекта
Применение	Организации с заранее определёнными ролями и обязанностями пользователей	Системы с детализированным разграничением прав доступа	Системах с переменными условиями доступа	Системы, где владельцы объектов контролируют доступ к своим данным

Для расширения модели управления доступом была выбрана модель RBAC, дополненная S3 политиками доступа. Для администрирования системы хранения данных используется управление доступом на основе ролей – Owner, Maintainer, Viewer. Пользовательские права управления регулируются с помощью политик доступа S3. Такое сочетание подходов способствует повышению производительности системы. Политики S3 обеспечивают быстрый и гибкий контроль доступа, а RBAC способствует упрощению управления правами доступа, устраняя возможность конфликтов и избыточных настроек, что позволяет более эффективно организовать процессы хранения и обработки данных [6].

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. ЗАКРОМА [Electronic resource]. URL: <https://zakroma.ru/products/storage> (accessed: 08.03.2025).
3. Alshehri A., Sandhu R. Access control models for virtual object communication in cloud-enabled IoT // Proc. - 2017 IEEE Int. Conf. Inf. Reuse Integr. IRI 2017. 2017. Vol. 2017-Janua. P. 16–25.

4. Penelova M. Access Control Models // Cybern. Inf. Technol. 2021. Vol. 21, № 4. P. 77–104.
5. Mehra T. The Critical Role of Role-Based Access Control (RBAC) in securing backup , recovery , and storage systems The Critical Role of Role-Based Access Control (RBAC) in securing backup , recovery , and storage systems. 2024. № December.
6. Bhosale S., Daund K., Kalambe R. International Journal of Research Publication and Reviews Efficient Data Storage and Retrieval Using Amazon Web Services S3. 2024. Vol. 5, № 6. P. 3417–3421.

УДК 004.41

Р.А. Береснев (4 курс бакалавриата),
Ю.В. Литвинов, к.т.н., ст. преподаватель

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ДЕТЕКЦИИ ДЕФЕКТОВ ЛЕНТЫ ПРОМЫШЛЕННОГО КОНВЕЙЕРА

Промышленные конвейеры используются для автоматизации процесса транспортировки руды. В результате длительной и интенсивной работы конвейеров на поверхности их лент появляются дефекты, приводящие к их разрыву и простоя конвейера. Даже недолгие простои могут обойтись предприятиям в миллионы рублей упущенной прибыли и потере репутации. Для минимизации риска появления и прогрессирования дефектов работниками предприятий вручную проводятся ежедневные осмотры, но это не позволяет обнаружить все неисправности и отследить процесс их развития.

Для автоматизации детекции дефектов ленты конвейера применяются такие методы обнаружения, как построение и анализ 3D-модели поверхности [1], применение датчиков на основе принципа магнитной индукции [2], рентгеновское излучение [2,3], обработка изображений и классификация 2D-изображений с помощью алгоритмов компьютерного зрения [2, 4].

Компанией ООО «Системы Компьютерного Зрения» была реализована библиотека, обнаруживающая и классифицирующая дефекты. Для создания полноценной удобной для использования системы в рамках данной работы была поставлена *цель* в виде *реализации веб-приложения к имеющейся библиотеке компьютерного зрения*.

Для достижения цели необходимо решение следующих *задач*.

1. Обзор возможных дефектов ленты конвейера и методов их обнаружения.
2. Проведение анализа существующих аналогов, использующихся для дефектоскопии, и выявление их сильных сторон для сбора требований к собственному решению.
3. Проектирование и реализация архитектуры API веб-приложения.
4. Создание пользовательского интерфейса.
5. Написание модульных тестов и проведение апробации.
6. Выполнение развертывания веб-приложения и по возможности его внедрение на предприятие.

Для анализа функциональности и сбора требований к собственному решению были отобраны уже существующие системы-аналоги. Выявленные сильные стороны в функциональности и пользовательском интерфейсе можно объединить в следующие.

1. Просмотр ленты конвейера в реальном времени (либо просмотр фотографий).
2. Отображение общего состояния ленты и ее толщины.
3. Просмотр истории возникших дефектов в виде таблиц или списков.
4. Подробная информация об отдельном дефекте (его вид, критичность, фотография, время возникновения, положение на ленте).
5. Система оповещений (посредством Telegram, e-mail или уведомлений в приложении).
6. Выгрузка отчетов о найденных дефектах и общем состоянии конвейера (в форматах pdf, xls или docx).
7. Настройки параметров критичности для дефектов.
8. Пользовательский интерфейс с информацией, разделенной по тематическим вкладкам.

Для разработки серверной составляющей веб-приложения был выбран язык программирования Python. Основной фреймворк для разработки API — FastAPI, библиотека для работы с СУБД PostgreSQL посредством механизма ORM — SQLAlchemy. Для реализации сервиса оповещений были использованы Gmail API и библиотека python-telegram-bot, а для создания pdf-документов в сервисе загрузки отчетов — библиотека reportlab.

API имеет пять сервисов.

1. Сервис отправки оповещений — реализует два способа отправки оповещений: через сообщение от Telegram-бота и через письмо со специального Gmail-аккаунта.
2. Сервис для работы с дефектами — позволяет получать отфильтрованные по различным параметрам дефекты с подробной информацией из базы данных на сервере, менять их критичность или удалять их.
3. Сервис контроля состояния ленты конвейера — предоставляет информацию об общем статусе конвейера и его параметрах.
4. Сервис логирования — отвечает за создание записей логов всех действий, происходящих в приложении, а также позволяет работать с ними.
5. Сервис для загрузки отчетов — используется для формирования отчетов (в формате pdf/csv) по всем дефектам или конкретному дефекту, а также для создания отчетов с информацией о параметрах и актуальном статусе всего конвейера.

На рисунке 1 приведена *архитектура* API в виде диаграммы компонентов для вышеупомянутых сервисов.

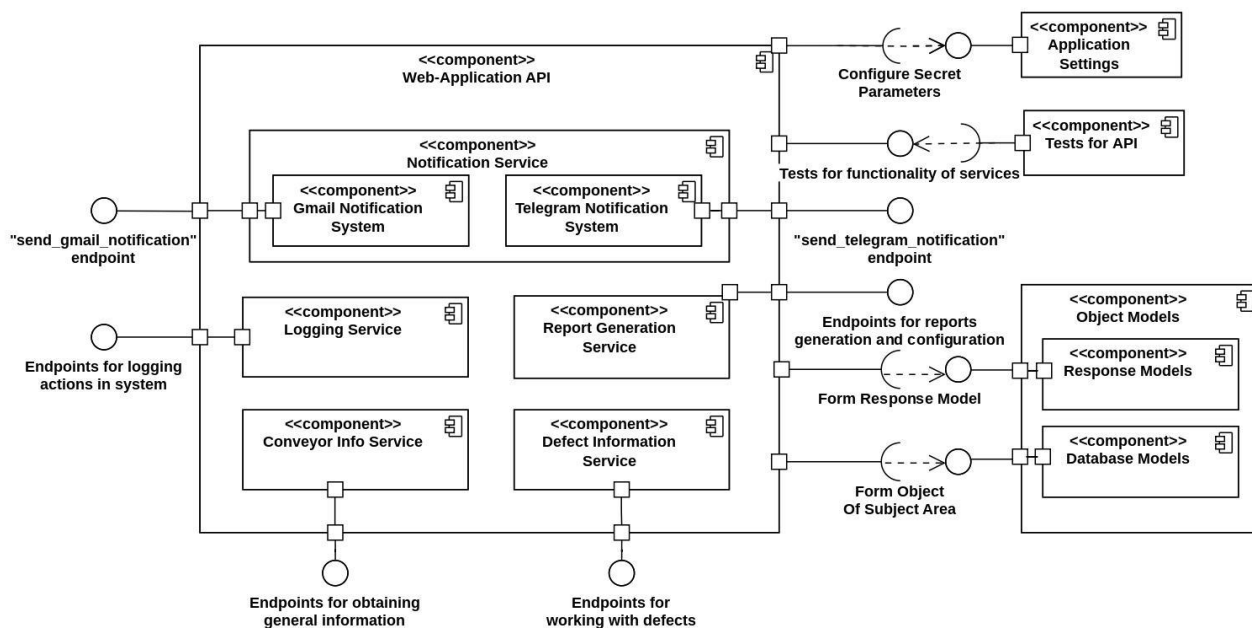


Рисунок 1 – Диаграмма компонентов API веб-приложения

В результате *был реализован* API веб-приложения для автоматизации дефектоскопии [5]. В будущем планируется реализовать пользовательский интерфейс веб-приложения (т.е. клиентская часть). После этого предстоит процесс развертывания веб-приложения в облаке, а также внедрение на предприятие.

ЛИТЕРАТУРА

1. Roxon HX270 for critical conveyors // Roxon : [Электронный ресурс]. URL: https://qsgroup.com/infusions/page_documents/documents/roxon_hx270_for_critical_conveyors.pdf
2. Xiaoqiang G. [et al.]. Belt Tear Detection for Coal Mining Conveyors // Micromachines. 2022. V. 13(3): 449.
3. Changyun M. [et al.]. Study on nondestructive detection system based on X-ray for wire ropes conveyor belt // Proc. SPIE - Int. Soc. Opt. Eng. 2008. V. 6622.

4. Yang Y. [et al.]. Longitudinal tear early-warning method for conveyor belt based on infrared vision // Measurement. 2019. V. 147: 106817.
5. Веб-приложение на базе FastAPI с использованием СУБД PostgreSQL для системы мониторинга промышленных конвейерных лент // GitHub : [Электронная платформа]. URL: <https://github.com/RuslanBeresnev/Conveyor-Belt-Monitoring-Web-Application> (дата обращения: 16.03.2025).

УДК 004.453

Я.А. Боков (4 курс бакалавриата),
А.Н. Яковлев, ст. преподаватель,
Е.В. Скуднева, к.т.н., доцент

РЕАЛИЗАЦИЯ ОПЕРАЦИИ ИСЧИСЛЕНИЯ ПОДОБИЯ ТЕКСТОВ В РАМКАХ ССУБД КАТАРСИС

Современные системы управления базами данных (СУБД) позволяют хранить большие объемы данных и совершать различные операции над ними [1]. Однако их функциональность в области анализа текстов остается ограниченной.

ССУБД Катарсис предназначена для эффективного использования вычислительных ресурсов суперкомпьютера при решении задач, связанных с обработкой и хранением больших данных. Архитектура ССУБД поддерживает параллельную обработку и оптимальное использование аппаратных ресурсов, как при исчислении классических реляционных операторов, так и при выполнении сложных прикладных вычислений.

В рамках настоящей работы разрабатывается расширение для языка SQL, позволяющее эффективно решать задачи классификации документов и обработки естественных языков непосредственно в СУБД, без необходимости извлечения данных во внешнюю вычислительную среду, что может быть чрезвычайно востребовано с учетом современных объемов текстовой информации.

Алгоритмической основой работы является введение нового типа данных для представления в ССУБД значений вектора характеристики содержимого документа, а также реализация операции вычисления меры сходства для векторов (вещественное число от 0 (разные) до 1 (идентичные)), как общие механизмы эффективного хранения и вычисления векторов, учитывающие возможности распределенной обработки данных и параллельных вычислений [2].

В рамках семантического разбора на уровне ядра ССУБД Катарсис должны быть переопределены стандартные SQL-операторы сравнения (равенства-неравенства, больше, меньше), существующие в нотации SQL-92 для элементарных типов данных, а также строк, таким образом, чтобы их поведение было корректно и применительно к векторам похожести.

Исчисление степени похожести предполагает реализацию новой операции, применяемой к двум векторам, осуществляющую численное вычисление меры их сходства.

Для вычисления вектора похожести планируется предоставить возможность выбора алгоритма вычисления степени схожести: евклидово расстояние, коэффициент Жаккара, расстояние Левенштейна и другие [3].

Для повышения производительности вычислений используется библиотека Open MPI [4], реализующая стандарт MPI, что позволяет значительно ускорить выполнение численных операций за счет использования параллельных вычислений на множестве узлов. При работе на кластерах с графическими ускорителями дополнительно планируется задействовать библиотеки cuBLAS или RAPIDS cuML из NVIDIA CUDA для ускорения выполнения операций благодаря использованию графических ускорителей [5].

Предлагаемая система базируется на архитектуре клиент-сервер с распределенными вычислениями. Серверная часть реализует расширенный язык SQL, поддерживающий векторные представления документов, использует распределенные вычисления по модели MapReduce [6] и аппаратное ускорение посредством CUDA. Клиентская часть представляет

собой интерфейс для взаимодействия с пользователем и выполнения запросов. Хранилище данных реализует для хранения векторов единое логическое пространство на физически разделенных узлах.

Реализация данной работы в среде ССУБД Катарсис обусловлена её высокой производительностью и поддержкой параллельных вычислений, что делает её идеальным выбором для работы с большими объемами данных. Использование суперкомпьютерных технологий позволяет значительно ускорить выполнение операций по сравнению с традиционными реляционными СУБД, а также получить значительное снижение трудоемкости реализации прикладных расчетов.

Таким образом, *целью* данной работы является создание расширения для ССУБД Катарсис, поддерживающее основные операции, необходимые для задач анализа документов, их классификации и кластеризации.

Основные этапы данной работы включают введение новых типов данных на уровне файлового хранения, реализацию поддержки типов в буферах, журналировании и на уровне синтаксиса. Планируется рассмотреть возможность реализации индексирования по новому типу данных. Вычислительные задачи будут реализованы в составе штатного конвейера реляционных операций ССУБД Катарсис посредством введения новой материализуемой операции.

Результаты исследования позволят расширить функциональность ССУБД Катарсис, делая её удобным и эффективным инструментом для решения задач документной лингвистики и информационного поиска.

ЛИТЕРАТУРА

1. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2005. – 1328 с.: ил. – Парал. тит. англ
2. Параллельные вычислительные системы: учебное пособие / Н. Ю. Сиротинина, О. В. Непомнящий, К. В. Коршун, В. С. Васильев. – Красноярск: Сиб. федер. ун-т, 2019. – 178 с.
3. Группировка, корреляция, распознавание образов: (статистические методы классификации и измерения связей) / И. И. Елисеева, В. О. Рукавишников. – М.: Статистика, 1977. – 143 с.
4. Введение в параллельное программирование на основе технологий MPI и OpenMP: Учебное пособие / Е. В. Земляная, М. В. Башахин. – Дубна: Государственный университет «Дубна», 2023. – 101 с.
5. Программирование графических устройств. Технология CUDA: Учебное пособие / Д.Н. Тумаков, Д. Х. Гиниятова, В.С. Гуськов – Казань: Казанский государственный университет, 2022. – 87 с.
6. MapReduce: внутри, снаружи или сбоку от параллельных СУБД? / С. Д. Кузнецов. – Труды Института системного программирования РАН, 2010. – 18с.

УДК: 004.8:028.5

А.С. Борисов (4 курс бакалавриата),
К.А. Туралчук, к.т.н, доцент

РАЗРАБОТКА СИСТЕМЫ НА ОСНОВЕ LLM ДЛЯ ГЛУБОКОГО ЧТЕНИЯ ЛИТЕРАТУРНЫХ ТЕКСТОВ С КОНТЕКСТУАЛЬНЫМИ АННОТАЦИЯМИ

В последние 10 лет социологи и отраслевые эксперты отмечают рост интерес к чтению книг в России, в первую очередь — электронных книг. Классическую литературу россияне называют среди топ-3 любимых жанров (22%) [1]. Однако восприятие художественной литературы затрудняется такими факторами как когнитивная сложность языка, культурно-историческая дистанция, особенности восприятия [2,3].

Тем временем Large Language Models (LLM) уже активно используют в образовательном процессе. Основные практики применения сводятся к следующим подходам: адаптированные подсказки, контекстуальные пояснения, уточняющие вопросы, упрощение текста, диалоговый режим [4]. Отдельного внимания заслуживает развитие образовательных инструментов на

основе подхода Retrieval-Augmented Generation (RAG), который позволяет заставить модель отвечать на вопросы на базе предоставленных материалов и таким образом снизить количество галлюцинаций [5]. Примерами таких решений являются Google NotebookLM, Google Learn About, Яндекс Нейро.

В гуманитарном образовании LLM тоже начинают использовать. Американский стартап Rebind.ai решает проблемы восприятия классической литературы с помощью LLM. Это приложение для чтения, которое позволяет задавать книге вопросы; ответы на них генерирует LLM на основе интервью с современными исследователями. Журнал Time признал приложение одним из главных изобретений 2024 года [6], однако это решение обладает рядом недостатков: а) отсутствие поддержки русской литературы; б) низкая скорость пополнения библиотеки.

Разрабатываемая система призвана предоставить аналогичный Rebind.ai опыт для русскоязычной аудитории и позволяет сократить время и ресурсы на создание цифровых изданий с поддержкой AI благодаря использованию существующих материалов. Архитектура решения представлена на рисунке 1.

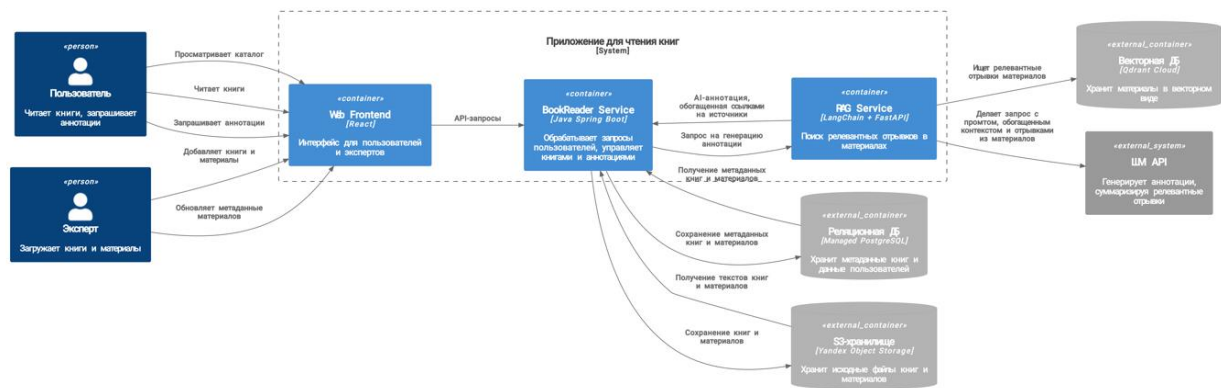


Рисунок 1 – Архитектура системы для глубокого чтения с контекстуальными аннотациями

«Глубокое чтение» — это медитативное погружение в книгу, позволяющее лучше понять и оценить сложный текст [7]. Использовать такой подход в предлагаемой системе помогают аннотации, генерируемые с помощью LLM на базе материалов, отобранных экспертами. Пользователь в интерфейсе приложения может в любом месте задать вопрос и получить сумаризированный ответ со ссылками на источники. Процесс формирования подобных аннотаций представлен на рисунке 2.

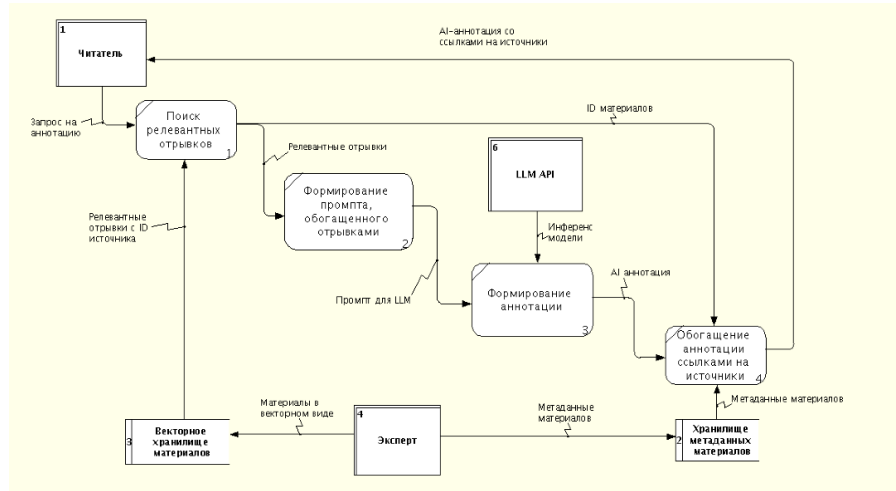


Рисунок 2 – Процесс формирования аннотаций с помощью RAG

Таким образом использование LLM и RAG позволяет снизить когнитивные и культурные барьеры при чтении сложных текстов, а также увеличивает точность ответов LLM и уровень доверия к системе благодаря опоре на экспертные материалы. Разрабатываемая

система предлагает новый формат взаимодействия с текстом — интеллектуальный партнер, который помогает читателю мыслить глубже и осмысленнее.

Помимо коммерческого использования разрабатываемого решения по модели B2C, его можно интегрировать в гуманитарные образовательные платформы (Арзамас, Страдариум, Правое полушарие интроверта), а также применять на уроках литературы в школах.

ЛИТЕРАТУРА

1. Всероссийский центр изучения общественного мнения (ВЦИОМ). Чтение в эпоху цифры // [Электронный ресурс]. – 18.11.2024. – Режим доступа: <https://wciom.ru/analytical-reviews/analiticheskii-obzor/chtenie-v-ehpokhu-cifry> (дата обращения: 12.01.2025).
2. Обласова Т. В., Логинова Е. А. Почему подростки не любят читать классику? // Филологический класс. – 2020. – № 2 (25). – Режим доступа: <https://filclass.ru/images/JOURNAL/2020-2-25/22.pdf> (дата обращения: 12.01.2025).
3. Борисов С. Б. Историко-культурный комментарий как необходимое условие эффективного освоения школьной программы по литературе // Педагогика. Вопросы теории и практики. – 2024. – Том 9, Выпуск 7. – Режим доступа: <https://pedagogy-journal.ru/en/article/ped20240080/fulltext> (дата обращения: 12.01.2025).
4. Wang S., Xu T., Li H., Zhang C., Liang J., Tang J., Yu P. S., Wen Q. Large Language Models for Education: A Survey and Outlook // arXiv preprint arXiv:2403.18105. – 2024. – Режим доступа: <https://arxiv.org/abs/2403.18105> (дата обращения: 12.01.2025).
5. Gupta S., Ranjan R., Singh S. N. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions // arXiv preprint arXiv:2410.12837. – 2024. – Режим доступа: <https://arxiv.org/abs/2410.12837> (дата обращения: 12.01.2025).
6. Stokel-Walker C. An Expert Book Club. Rebind // Time. – 30.10.2024. – Режим доступа: <https://time.com/7094572/rebind/> (дата обращения: 08.02.2025).
7. Birkerts S. The Gutenberg Elegies: The Fate of Reading in an Electronic Age. – Boston: Faber and Faber, 1994. — Selected passages, Open Book Systems — [Электронный ресурс]. – Режим доступа: <http://archives.obs-us.com/obs/english/books/nn/bdbirk.htm> (дата обращения: 08.02.2025)

УДК 004.453

Д.К. Борисов (4 курс бакалавриата),
А.Н. Яковлев, ст. преподаватель,
Е.В. Скуднева, к.т.н., доцент

РЕАЛИЗАЦИЯ ОПЕРАЦИЙ ЧИСЛЕННОГО РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ССУБД КАТАРСИС

Применение суперкомпьютерных технологий широко распространено в мире. На практике часто возникают задачи, которые требуют обработки большого объема данных, и в которых суперкомпьютер был бы эффективен, однако прикладное применение суперкомпьютеров ограничено [1], в том числе из-за отсутствия инфраструктурного ПО, такого как системы управления базами данных.

Популярные реляционные СУБД не могут быть эффективно использованы на суперкомпьютере, так как спроектированы для других сценариев использования. Для суперкомпьютера характерно наличие большого количества данных и выполнение вычислительно сложных операций над ними. Для повышения эффективности обработки таких запросов возможно использовать несколько вычислительных узлов. Поэтому СУБД для суперкомпьютера (СуперСУБД, ССУБД) должна осуществлять деление задач на подзадачи и распределение нагрузки по вычислителям.

В рамках реляционной системы управления базами данных для суперкомпьютера – ССУБД Катарсис, решается задача упрощения работы с данными и предоставление дополнительных возможностей по обработке данных для научных и прикладных исследований совместно с решением реляционных задач.

При этом используются особенности суперкомпьютера в СКЦ Политехнический – так реализуемая ССУБД сохраняет данные в общей системе хранения данных, а вычислительные операции производятся на узлах, которые соединены с хранилищем и между собой сетью InfiniBand, имеющую топологию утолщенное дерево.

Дифференциальные уравнения являются одним из основных инструментов для моделирования процессов, изменяющихся во времени. Однако аналитическое решение часто невозможно или слишком сложно. Поэтому на практике широко применяются численные методы [2].

Общий подход к численному решению дифференциального уравнения состоит в его преобразовании в разностное уравнение, которое затем решается пошаговым методом [2], например, одним из явных методов Рунге-Кутты. Такая операция при определенных ограничениях, например, высокой точности и большом объеме точек, в которых выполняются расчеты, может быть чрезвычайно затратной по времени и вычислительным ресурсам. В таких случаях становится оправданным выполнение этой операции с использованием возможностей суперкомпьютерного оборудования.

Таким образом, целью данной работы является реализация операций для численного решения дифференциальных уравнений в ССУБД Катарсис на аппаратных мощностях суперкомпьютера.

Общий вид разностного уравнения:

$$x_{n+1} = x_n + \int_{t_n}^{t_{n+1}} f(\tau, x(\tau)) d\tau \quad (1)$$

Широко применяются явные методы Рунге-Кутты, которые представляют собой последовательный расчет коэффициентов [3], количество которых зависит от порядка метода, каждый из которых требует вычисления функции f . Однако во многих случаях вычисление функции f является одной из самых вычислительно затратных операций.

Если задача представляет собой систему дифференциальных уравнений, а x_n и $f(t, x)$ являются векторами довольно большой размерности, то можно получить ускорение за счет параллельного вычисления компонентов вектора x_{n+1} . В этом случае ССУБД может обеспечить планирование и параллельное выполнение требуемых вычислений.

Кроме того, при решении некоторых прикладных задач функция f будет содержать большое количество обращений к большим данным, которые могут выполняться параллельно. Например, это могут быть сведения о рельефе местности, о воздушной и водной среде в задачах прогнозирования гидрометеорологических явлений. При решении таких задач наряду с реляционными операциями могут требоваться и операции из линейной алгебры, например, сплайн-аппроксимация. Таким образом, использование ССУБД может значительно упростить и ускорить данный процесс.

При этом различные операции могут быть выполнены либо на узлах суперкомпьютера, либо с использованием других вычислительных средств в составе суперкомпьютера, например, на высокопроизводительных графических процессорах с поддержкой CUDA [4]. Также рассматривается выполнение операций на узлах с общей разделяемой памятью (NUMA).

В зависимости от особенностей решаемой задачи, соотношения объема данных и вычислений, задержек и скорости передачи данных между вычислительными средствами эффективность применения рассматриваемых аппаратных средств может быть различной и представляет собой отдельный предмет исследования.

Для эффективной реализации вычислений во внимание берется работа В.О. Гаврикова, А.Н. Яковлева, Е.В. Скудновой, в которой в рамках ССУБД Катарсис реализуется аппарат операций линейной алгебры посредством введения на уровне ядра СУБД нового типа данных, представляющего матрицы.

Таким образом, для выполнения поставленной задачи в рамках ССУБД Катарсис требуется реализовать операторы для численного решения дифференциальных уравнений. Для эффективной реализации потребуется разработать планировщик, который позволит

распределять подзадачи на наиболее подходящее для них на момент планирования оборудование для минимизации общего времени выполнения задач.

Вопрос осуществления статистического анализа исполнения параллельных операций в ССУБД Катарсис с учетом аппаратных особенностей применяемого суперкомпьютерного оборудования также представляет собой важную научную задачу, которая будет актуальна на будущих этапах развития проекта.

ЛИТЕРАТУРА

1. Абрамов Н. С., Абрамов С. М. Ноябрь 2022: состояние и перспективы развития суперкомпьютерной отрасли в мире и в России // Программные системы: теория и приложения. - 2023. - №2. - С. 49–93.
2. С. М. Устинов, В. А. Зимницкий Вычислительная математика. - СПб.: БХВ-Петербург, 2009. - 336 с.
3. Ващенко Г.В. ЯВНЫЕ МЕТОДЫ ТИПА РУНГЕ-КУТТЫ И ИХ ПАРАЛЛЕЛЬНЫЕ АНАЛОГИ // Международный журнал прикладных и фундаментальных исследований. - 2011. - №6. - С. 98-100.
4. Ahnert K., Demidov D., Mulansky M. Solving ordinary differential equations on GPUs // Numerical Computations with GPUs. – Cham : Springer International Publishing, 2014. – С. 125-157.

УДК 004.453

А.А. Борисов (4 курс бакалавриата),
С.В. Хлопин, к.т.н., доцент

АЛГОРИТМ И ПРОГРАММА УПРАВЛЕНИЯ ДАТЧИКОМ НАЛИЧИЯ И КОНЦЕНТРАЦИИ НЕФТЕСОДЕРЖАЩИХ ПРИМЕСЕЙ

В рамках выполнения работ по гранту Российского научного фонда (РНФ) по научному проекту «Оптический датчик наличия и уровня загрязнённости нефтепродуктами и тяжелыми частицами жидкостей» создаётся лабораторный стенд. Лабораторный стенд, на котором установлен датчик, представляет собой трубу, в которой движется исследуемая жидкость. Скорость потока можно регулировать насосом. Непосредственно перед корпусом датчика расположен дозатор, позволяющий вносить в поток загрязнения из нефтепродуктов. В корпусе датчика установлены точечные источники ультрафиолетового излучения. Они позволяют облучать поток жидкости и вызывать вторичное флуоресцентное свечение и в дальнейшем, регистрировать фотокамерой через смотровое окно [1].

Подсчет концентрации производится за счет вычисления объёма нефтесодержащих пузырей в общем объёме жидкости, прошедшей через датчик. Скорость потока определяется датчиком расхода жидкости FS200A, G1/2 (1-30L/min).

Целью представляемой работы является разработка алгоритма и программы для управления датчиком наличия и концентрации нефтесодержащих примесей.

Для достижения поставленной цели решались следующие задачи:

1. Разработка алгоритма измерения скорости потока жидкости.
2. Разработка алгоритма, контроля частоты включения ультрафиолетового источника света и синхронной работы камеры, регистрирующей свечение нефтепродуктов в корпусе датчика.
3. Реализация программы, использующей разработанные алгоритмы.

Общий принцип работы программы:

1. В пользовательском интерфейсе устанавливаются значения для частоты зажигания светодиодов и задержки на получение кадра (последовательности кадров) камерой.
2. В режиме постоянного мониторинга производится считывание с датчика расхода жидкости скорости потока жидкости и передача с контроллера через com-порт на пользовательское приложение.

3. В программном интерфейсе при нажатии на кнопку старт, выполняется функционал одиночного или серийного выполнения рабочего цикла: включение светодиодов и получения кадров с камеры. Включение и выключение светодиодов производится микроконтроллером, связь с которым осуществляется через com-порт. В результате выполнения рабочего цикла, в выбранной пользователем папке, создается одиночный файл или серия файлов с указанием в названии временной метки создания кадра.

Программа написана на языке программирования C++, который обеспечивает высокую производительность и возможность работы с данными на низком уровне [2].

Пользовательская программа написана под операционную систему Windows. Для создания графического интерфейса используется интерфейс прикладного программирования (API). API предоставляет программисту набор готовых классов, функций, структур и констант.

Программирование контроллера произведено в среде Arduino IDEА [5]. В качестве языка программирования использовался "Arduino programming language".

Передача сигнала между пользовательским приложением и управляющей платой микроконтроллера осуществляется через com-порт. Порт работает на стандартной скорости 115200 бит/с.

Для получения изображений с камеры используется библиотека OpenCV [3]. OpenCV является ведущей библиотекой компьютерного зрения с открытым исходным кодом, которая предоставляет обширный набор функций и алгоритмов для работы с изображениями и видео. OpenCV широко используется в индустрии и научных исследованиях благодаря своей высокой производительности, гибкости и простоте использования.

Блок-схема разработанного алгоритма представлена на рисунке 1.



Рисунок 1 – Блок-схема алгоритма программы управления датчиком наличия и концентрации нефтесодержащих примесей

В результате работы программы получают снимки, на основе которых можно вычислить объём примесей в жидкости [4].

Исследование выполнено за счет гранта Российского научного фонда № 24–21–20116 (<https://rscf.ru/project/24-21-20116/>) и гранта Санкт-Петербургского научного фонда № 24-21-20116.

ЛИТЕРАТУРА

1. Пипия Г.Т., Салех Л.О.А., Сушников В.А., Хлопин С.В., Черненькая Л.В. Оптический датчик наличия и концентрации нефтесодержащих примесей // 51 школа-конференция «Актуальные проблемы механики», 2024. – с. 263–264

2. C++ [Электронный ресурс] Режим доступа: <https://cplusplus.com/>
3. OpenCV [Электронный ресурс] Режим доступа: <https://opencv.org/>
4. Салех Л.О.А., Хлопин С.В., Черненькая Л.В. Алгоритм вычисления объёма примесей в жидкости по оптическим данным // Известия Тульского государственного университета. Технические науки. – 2023. – No 7. – с.562–569.
5. Arduino IDE [Электронный ресурс] Режим доступа: <https://www.arduino.cc/en/software/>

УДК 004.65 + 004.438

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

РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ РАБОТЫ С ХРОНОЛОГИЧЕСКИМИ ТИПАМИ ДАННЫХ В РЕЛЯЦИОННЫХ СУБД

В настоящее время обработка хронологических данных приобретает всё большую актуальность [1,2]. Это обусловлено как стремительным развитием технологий хранения информации, которые делают сбор огромных объёмов данных, так и появлением новых областей, требующих анализа временных данных. Также важной задачей остаётся прогнозирование на основе временных рядов, например, анализ данных о погодных условиях позволяет предсказать вероятные периоды засухи или сильных осадков, что важно для управления сельскохозяйственными ресурсами и предотвращения убытков. В данной статье описывается разработка библиотеки-расширения для работы с хронологическими типами данных в СУБД PostgreSQL [3,4] на языке Rust [5] с использованием библиотеки PGRX [6].

Разрабатываемая библиотека предназначена для работы с интервальными типами данных PostgreSQL. Ключевые особенности:

- Создание универсального уровня абстракции, которое позволяет уйти от решения частных задач к охвату их широкого спектра.
- Разработка математического аппарата для формализации временных запросов.
- Реализация набора хронологических операций.

Библиотека-расширение разработана для PostgreSQL на языке Rust с использованием библиотеки PGRX. Выбор инструментов для реализации обусловлен следующими их возможностями:

– PostgreSQL является одной из самых популярных СУБД с открытым исходным кодом. Поддерживает широкий спектр типов, включая интервальные типы данных. Предоставляет документацию [7] и API, что позволяет интегрировать расширения, которые могут содержать пользовательские функции, типы данных и даже низкоуровневые оптимизации.

– Rust обеспечивает безопасное управление памятью благодаря системе владения (ownership) и заимствования (borrowing), что исключает такие ошибки, как утечка памяти, висячие указатели и доступ к неинициализированным данным. Также язык имеет богатую экосистему инструментов, которые упрощают процесс управления зависимостями, сборки и тестирования.

– PGRX предоставляет удобные механизмы для реализации различных аспектов работы с PostgreSQL, включая сериализацию, десериализацию, создания API с использованием макросов и вспомогательных функций, возможность добавлять новые типы.

Таким образом, выбранные инструменты позволяют эффективно разрабатывать расширения для СУБД, обеспечивая безопасность и удобство разработки. Также реализация встраиваемой библиотеки-расширения избавляет от необходимости писать хранимые процедуры и функции на языке SQL, как следствие, позволяя добиться высокой производительности.

Прототип библиотеки реализован и протестирован. Оценка характеристик включает:

– Удобство интегрирования расширения в локальную СУБД: для интегрирования достаточно клонировать репозиторий проекта и перенести файл с расширением .so в директорию pg_config.

– Удобство использования: расширение предоставляет ключевые хронологические операции, описанные в документации с примерами использования.

– Безопасность: вся логика работы инкапсулирована.

– Расширяемость: библиотека распространяется, как проект с открытым исходным кодом, а значит любой заинтересованный может клонировать репозиторий и добавить недостающие операторы.

– Производительность: для оценки производительности скорость выполнения функций из библиотеки-расширения сравнивалась со скоростью выполнения хранимых процедур, написанных на языке SQL [8].

Разработанная библиотека-расширение демонстрирует эффективный подход к решению задач, связанных с обработкой данных с хронологическими типами. Выбранные инструменты доказали свою эффективность. Это делает библиотеку перспективным решением для использования в проектах, требующих работы с временными данными, и открывает возможности для её дальнейшего совершенствования и адаптации под новые задачи.

ЛИТЕРАТУРА

1. Steiner, A. N. M. C. Implementing Temporal Databases in Object-Oriented Systems: Database Systems for Advanced Applications (DASFAA), 1997. — 381-390 с.
2. Костенко Б.Б. История и актуальные проблемы темпоральных баз данных — МГУ, 2007. [Электронный ресурс] Режим доступа: <https://citforum.ru/database/articles/temporal/>
3. Postgres. Первое знакомство / Лузанов П. В. — 8-е изд. — М.: Постгрес Профессиональный, 2022. — 183 с.;
4. PostgreSQL изнутри. / Рогов Е. В.— М.: ДМК Пресс, 2022. — 660 с.
5. Программирование на Rust. / Клабник С., Николс К.— М.: Питер, 2021. — 592 с.
6. Документация библиотеки PGRX. [Электронный ресурс] Режим доступа: <https://crates.io/crates/pgrx/>
7. PostgreSQL: Documentation. [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/>
8. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.

УДК 004.457

Д.Н. Виниченко (4 курс бакалавриата),
Е.Е. Андрианова, ст. преподаватель

ПРИМЕНЕНИЕ LSM-ДЕРЕВЬЕВ ДЛЯ ЛОКАЛЬНЫХ NOSQL-ХРАНИЛИЩ В АВТОНОМНЫХ СИСТЕМАХ

В современных автономных системах, таких как встроенные устройства, IoT, автономные программные агенты, мобильные приложения и другие решения, не использующие отдельные серверы баз данных, особую важность приобретают локальные высокопроизводительные хранилища данных. Такие хранилища должны обеспечивать надежное хранение, иметь высокую скорость обработки данных и потреблять как можно меньше ресурсов. Одним из примеров автономной системы, требующей локального хранилища, являются устройства на базе Android, такие как умные часы на базе Wear OS или умные термостаты. В качестве используемого хранилища здесь может выступить такой инструмент Jetpack Datastore. В случае с носимыми устройствами DataStore может использоваться для хранения пользовательских настроек, данных о физической активности и параметров работы устройства без необходимости постоянного соединения с сервером. В умных термостатах DataStore применяется для сохранения расписания температурных

режимов, предпочтений пользователей и истории изменений, обеспечивая автономную работу даже при отсутствии подключения к облаку. Упомянутое решение позволяет сохранять данные с помощью key-value хранилищ на основе SharedPreferences или сериализуемых объектов в формате Protocol Buffers. Однако в сценариях, требующих высокой производительности при частых операциях записи, могут использоваться NoSQL-базы данных, построенные на LSM-деревьях (Log-Structured Merge-Tree). Этот подход позволяет минимизировать нагрузку на дисковую подсистему, оптимизируя работу с энергоэффективными хранилищами и SSD-накопителями.

Цель данной работы – разработка встроенного локального NoSQL-хранилища на основе LSM-деревьев, способного обеспечить эффективную работу в автономных системах с ограниченными вычислительными ресурсами. Для её достижения необходимо решить следующие задачи:

- Анализ существующих подходов к локальному хранению данных в автономных системах.
- Сравнительный анализ структур хранения с точки зрения производительности и надежности.
- Разработка прототипа локального NoSQL-хранилища с использованием LSM-деревьев.
- Экспериментальная оценка производительности разработанного хранилища.
- Определение преимуществ и недостатков предложенного подхода.

Подход будет реализован в среде программирования Java, так как данный язык обеспечивает кроссплатформенность и высокую производительность. Одним из ключевых преимуществ является возможность запуска JVM практически на любом устройстве.

Одним из ключевых элементов хранилища является структура данных LSM-деревьев, обеспечивающая эффективное управление записями, их компактификацию и быстрый доступ. В ходе сравнительного анализа были рассмотрены альтернативные структуры, такие как B-деревья, которые реализуются в средстве MapDB, а также готовые решения, реализующие LSM-деревья, например, RocksDB, Apache Cassandra, BadgerDB. В сравнении с B-деревьями именно LSM-деревья демонстрируют большую эффективность на запись, в то время как B-деревья – на чтение.

Однако существующие решения не всегда подходят для встроенных Java-хранилищ. Так, *Apache Cassandra* представляет собой мощную реализацию LSM-дерева, но она ориентирована на распределённые системы и не подходит для встроенных решений. *RocksDB*, будучи высокопроизводительным хранилищем, не является нативным для Java, так как написан на C++, что требует использования обёрток, и, хотя это не является критическим недостатком, такое решение не является нативным. Также стоит отметить, что функционал *RocksDB* избыточен для задач встраиваемого хранилища. *BadgerDB* – одно из наиболее успешных решений для автономных систем, но оно написано на языке Golang, что делает его неиспользуемым в Java-проектах. *MapDB*, работающее с Java, реализует схожий с нашим подход, но использует B-деревья вместо LSM.

Описание ключевых компонентов архитектуры:

1. LSM-дерево - основная структура хранения:
 - WAL (Write-Ahead Log) - журнал предзаписи для отказоустойчивости.
 - MemTable - структура в памяти, сбрасываемая на диск.
 - SSTable (Sorted String Table) - неизменяемые файлы с отсортированными данными.
 - Compaction Engine - объединяет SSTables, удаляет устаревшие данные.
2. Кэширование – ускорение операций чтения:
 - Block Cache - кэширует блоки данных SSTable.
 - Bloom Filter - быстро определяет, есть ли ключ в SSTable.

Разработка локального NoSQL-хранилища на основе LSM-деревьев находится в активной фазе, и к моменту конференции будет готов рабочий прототип. Ожидается, что:

- Ожидается, что запись данных будет быстрее, чем в решениях на В-деревьях, за счёт особенностей структуры LSM.
- Оптимизированная работа с SSD-дисками позволит минимизировать износ носителей.
- Java-реализация обеспечит лёгкость интеграции в автономные проекты, при этом учитывая требования к ресурсам.

Финальные метрики производительности и оценки эффективности будут представлены по результатам тестирования к моменту конференции.

Разрабатываемое локальное NoSQL-хранилище на основе LSM-деревьев ориентировано на автономные системы, где критически важны высокая скорость записи, эффективная работа с SSD и минимальное потребление ресурсов. Использование Java способствует совместимости и упрощает интеграцию. В дальнейших исследованиях планируется оптимизация механизмов компактификации и кеширования, а также проведение тестирования для подтверждения ожидаемых преимуществ перед существующими решениями.

ЛИТЕРАТУРА

1. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. – Москва: ДМК Пресс, 2022. – 576 с. URL: <https://dataintensive.net/>
2. O’Neil P. et al. The log-structured merge-tree (LSM-tree) // Acta Informatica. – 1996. – Т. 33. – С. 351-385. URL: <https://www.inf.ufpr.br/eduardo/ensino/ci763/papers/lsmtree.pdf>
3. Luo C., Carey M. J. LSM-based storage techniques: a survey // The VLDB Journal. – 2020. – Т. 29. – №. 1. – С. 393-418. URL: <https://arxiv.org/pdf/1812.07527>
4. Shalina, O. V. The study of combining SQL and nosql databases in a heterogeneous system for the development of a project management database / O. V. Shalina, E. E. Andrianova // Theoretical & Applied Science. – 2024. – No. 5(133). – P. 14-17. – DOI 10.15863/TAS.2024.05.133.5. – EDN TJJGEK. URL: <https://www.elibrary.ru/item.asp?id=67955798>
5. Data Layer – Data Store – Android Developers | App architecture URL: <https://developer.android.com/topic/libraries/architecture/datastore?hl=ru>
6. Apache Cassandra Documentation. URL: <https://cassandra.apache.org/doc/latest/>
7. Dong S. et al. Rocksdb: Evolution of development priorities in a key-value store serving large-scale applications // ACM Transactions on Storage (TOS). – 2021. – Т. 17. – №. 4. – С. 1-32. URL: <https://dl.acm.org/doi/full/10.1145/3483840>.

УДК 004.453

В.А. Ворожко (3 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО АДАПТЕРА ИНФОРМАЦИОННО-ТЕХНИЧЕСКОГО ВЗАИМОДЕЙСТВИЯ ВЕДОМСТВЕННОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ С КОМПЛЕКСОМ СРЕДСТВ АВТОМАТИЗАЦИИ СИСТЕМЫ УПРАВЛЕНИЯ ДВИЖЕНИЕМ СУДОВ МИНТРАНС РОССИИ (КСА СУДС) В ИНТЕРЕСАХ МОНИТОРИНГА МОРСКОЙ ОБСТАНОВКИ

Развитие экономики страны зависит от её транспортной инфраструктуры, и ключевую роль в этом играет судоходная отрасль, обеспечивающая до 90% мировых перевозок. Для эффективного управления морской деятельностью требуется оперативный мониторинг местоположения судов, что возможно благодаря автоматизированным системам сбора и обработки информации. Такие системы позволяют отслеживать суда, анализировать их маршруты и предоставлять данные заинтересованным ведомственным структурам.

Существует множество международных и российских систем мониторинга, которые разрабатывались под различные задачи:

– Глобальные системы (Marine Traffic, Vessel Finder, КИИС «МоРе») охватывают большую территорию, но зависят от данных Автоматизированной идентификационной

системы (АИС) [6], что снижает точность и надёжность информации. Например, операторы судов могут вводить некорректные данные, а суда с выключенным транспондером не фиксируются.

- Ведомственные системы (ОСМ Росрыболовства, системы Минобороны) ориентированы на специфические задачи, но не всегда обеспечивают комплексное наблюдение за судоходной обстановкой.

- Региональные системы управления движением судов (СУДС) [2] предоставляют наиболее оперативные и достоверные данные в прибрежной зоне, так как используют не только АИС [6], но и радиолокационные станции, видеонаблюдение, радиосвязь и метеоданные.

Анализ показал, что для мониторинга судоходства в прибрежных акваториях наиболее точным источником информации являются СУДС [2]. Они позволяют:

- Обнаруживать суда даже при выключенном АИС [6] благодаря радиолокационному контролю.
- Верифицировать данные: оператор может сопоставить визуальное наблюдение с радиолокацией и связаться с судном через радиоканал.
- Оперативно передавать информацию без значительных задержек.
- Учитывать гидрометеорологические условия, влияющие на безопасность мореплавания.

Однако СУДС [2] имеют региональные ограничения и не передают данные напрямую в ведомственные информационные системы. Для эффективного межведомственного взаимодействия необходимо разработать адаптер, который обеспечит интеграцию данных СУДС [2] в другие системы, повышая точность и оперативность мониторинга судоходной обстановки.

Таким образом, целью данной работы является создание решения для интеграции данных региональных систем управления движением судов в ведомственные структуры, что позволит оптимизировать контроль судоходной деятельности, повысить достоверность данных и обеспечить их доступность для ведомственных пользователей, отвечающих за безопасность мореплавания в прибрежных акваториях.

Архитектура системы мониторинга судоходства [4] представлена на рисунке 1. Данные о передвижении судов, получаемые из системы управления движением судов, обрабатываются сервером приложений, после чего передаются клиентскому уровню для отображения в пользовательском интерфейсе.

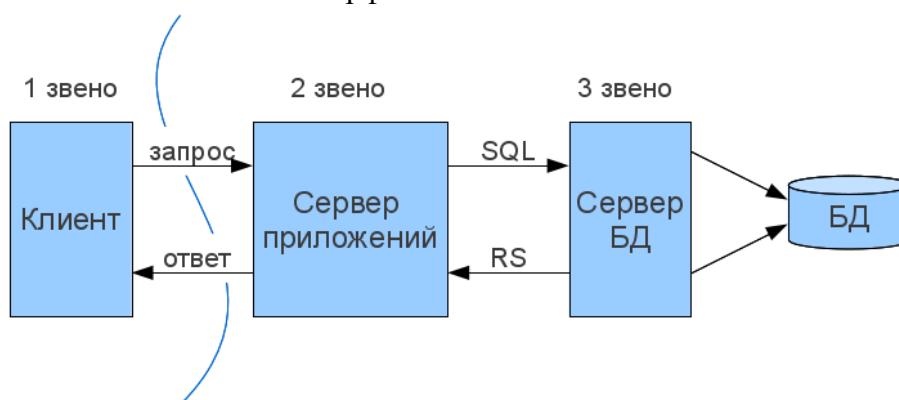


Рисунок 1 – Архитектура системы

Клиентский уровень реализован на языке программирования JavaScript [1] и отвечает за визуализацию информации и взаимодействие с пользователем. Веб-интерфейс обеспечивает отображение актуальных данных о местоположении судов, их маршрутах и параметрах передвижения. Благодаря использованию асинхронных запросов и WebSocket-соединений, интерфейс обновляется в реальном времени без необходимости перезагрузки страницы. Это

позволяет оперативно реагировать на изменения в судоходной обстановке, минимизируя задержки в передаче данных.

Сервер приложений, разработанный на Python [3], обрабатывает входящие данные и взаимодействует с базой данных PostgreSQL [5], обеспечивая хранение и управление данными о судоходстве. Он отвечает за приём информации от системы управления движением судов (СУДС) [2], её предварительную обработку, валидацию и передачу клиентскому уровню.

Передача данных между уровнями осуществляется по протоколу HTTP с использованием REST API, которое поддерживает методы для получения информации о судах, обновления данных и управления пользователями.

Выбранная трёхуровневая архитектурная модель [4] обеспечивает чёткое разделение системы на три уровня: клиентский, сервер приложений и база данных, улучшает масштабируемость и упрощает управление и поддержку системы.

Таким образом, разрабатываемый адаптер СУДС [2] обеспечивает интеграцию данных региональных систем управления движением судов в ведомственные структуры, повышая точность, достоверность и скорость обработки информации.

ЛИТЕРАТУРА

1. Флэнаган Дэвид. JavaScript. Полное руководство. 7-е издание: Диалектика-Вильямс, 2021. – 720 с.
2. Система управления движением судов [Электронный ресурс] Режим доступа : <https://geo-dinamika.ru/products-and-services/vtms/> свободный. – Загл. с экрана
3. Документация Python [Электронный ресурс] Режим доступа : <https://pylessons.readthedocs.io/ru/latest/> свободный. – Загл. с экрана
4. Трёхуровневая архитектура [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Трёхуровневая_архитектура свободный. – Загл. с экрана
5. Е. П. Моргунов. PostgreSQL. Основы языка SQL. : Питер, 2019. – 336 с.
6. Документация АИС [Электронный ресурс] Режим доступа: <https://seacomm.ru/dokumentaciya/stati/avtomaticheskaya-identifikatsionnaya-informatsionnaya-sistema-ais/> свободный. – Загл. с экрана

УДК 004.453

В.О. Гавриков (4 курс бакалавриата),
А.Н. Яковлев, ст. преподаватель,
Е.В. Скуднева, к.т.н., доцент

РАЗРАБОТКА ЗАМКНУТОГО АППАРАТА ОПЕРАЦИЙ ЛИНЕЙНОЙ АЛГЕБРЫ В РАМКАХ ССУБД КАТАРСИС

ССУБД Катарсис — это высокопроизводительная система управления базами данных, разработанная для обеспечения оптимального использования оборудования суперкомпьютеров при решении задач, связанных с обработкой и хранением больших объемов данных. Её архитектура ориентирована на поддержку параллельных вычислений и эффективное распределение ресурсов, что делает её идеальным инструментом для выполнения сложных вычислительных операций, включая задачи линейной алгебры.

Операции сложения, вычитания и умножения входят в стандарт SQL-92 и реализуются в ССУБД Катарсис в рамках конвейера реляционных операций. Однако операндами таких операций могут выступать только элементарные типы данных, что не позволяет эффективно использовать оборудование суперкомпьютера для реализации прикладных научных задач [1].

В рамках настоящей работы предлагается реализация на уровне ядра ССУБД Катарсис нового типа данных *mtr*, предназначенного для хранения и осуществления операций линейной алгебры над матрицами в рамках расширенного синтаксиса SQL.

Решение данной задачи сопряжено с разработкой эффективной структуры хранения на уровне файлов ССУБД, реализация поддержки типа *mtr* в буферах, журналировании, в модуле

синтаксического разбора, материализованной реализации в рамках конвейера операций реляционной алгебры таким образом, чтобы эффективно использовать возможности распределенной обработки данных и параллельных вычислений [2].

Дополнительные операции включают: транспонирование, нахождение обратной матрицы и вычисление собственных значений [3].

Для повышения производительности непосредственных вычислений используется библиотека cuBLAS из состава NVIDIA CUDA [4], что позволяет значительно ускорить выполнение матричных операций за счет использования графических процессоров [5].

Такая реализация основана на архитектуре клиент-сервер с распределенными вычислениями (рисунок 1). Серверная часть в своем ядре реализует расширенный язык SQL, поддерживающий матричные операции и использующий аппаратное ускорение посредством CUDA. Клиентская часть представляет собой интерфейс для взаимодействия с пользователем и выполнения текстовых запросов.

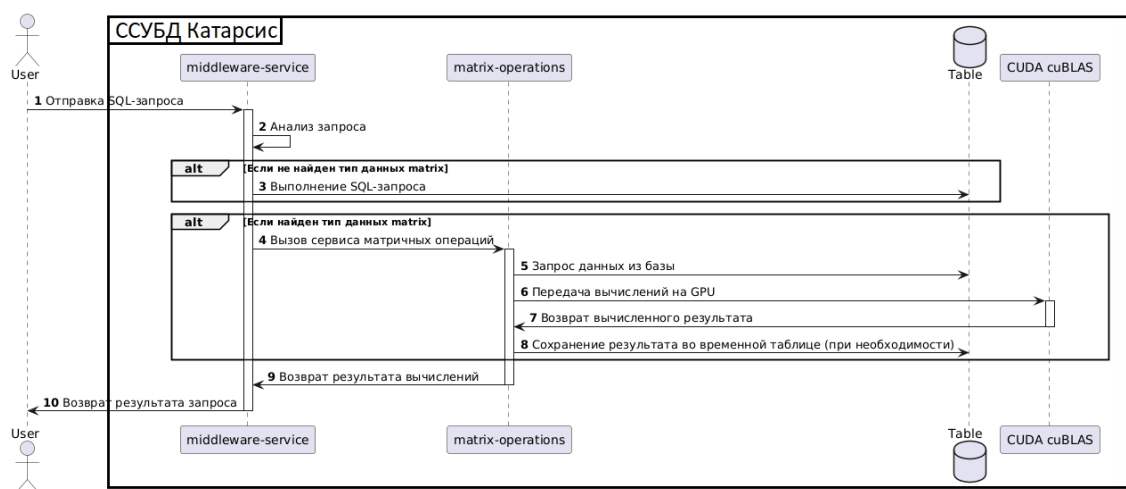


Рисунок 1 – ССУБД Катарис

Также со стороны ССУБД поддерживаются стандартные интерфейсы взаимодействия для разработки прикладного программного обеспечения. Однако реализация поддержки нового типа данных *mtg* в клиентских библиотеках ADO, JDBC не предусмотрена стандартом. Схема реализации приведена на рисунке 1.

Область применения разработки охватывает широкий спектр универсальных задач, востребованных современной наукой и практикой, в том числе: решение систем линейных алгебраических уравнений (СЛАУ), поиск собственных значений и собственных векторов матриц, матричные разложения (LU-разложение, SVD-разложение), численное решение дифференциальных уравнений, решение уравнений высокой степени с высокой точностью и скоростью.

Гипотеза об эффективности легко доказывается в связи с применением высокопроизводительных алгоритмов коммерческих библиотек cuSOLVER и cuBLAS в пределах общепринятых реляционных операций [6].

ЛИТЕРАТУРА

1. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.: ил. — Парал. тит. англ
2. Параллельные вычислительные системы : учебное пособие / Н. Ю. Сиротина, О. В. Непомнящий, К. В. Коршун, В. С. Васильев. - Красноярск : Сиб. федер. ун-т, 2019. - 178 с.
3. Линейная алгебра : учеб. пособие / Н.В. Гредасова, М.А. Корешникова, Н.И. Желонкина [и др.] ; Мин-во науки и высш. образования РФ. Екатеринбург: Изд-во Урал. ун-та, 2019.— 88 с.
4. Запуск кода CUDA на видеокартах AMD [Электронный ресурс] // habr.com. – 2022. – URL: <https://habr.com/ru/articles/701712/>.

5. Тумаков Д.Н. Технология программирования CUDA: учебное пособие / Д.Н. Тумаков, Д.Е. Чикрин, А.А. Егорчев, С.В. Голоусов. – Казань: Казанский государственный университет, 2017. – 112 с.
6. freeCodeCamp выпустила бесплатный курс по программированию CUDA [Электронный ресурс] // habr.com. – 2024. – URL: <https://habr.com/ru/news/846984/>.

УДК 004.453

К.М. Геворков (4 курс бакалавриата),
Е.В. Комарова, ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ЦЕНОВЫМИ ПАРАМЕТРАМИ СДЕЛОК ДЛЯ ФИНАНСОВЫХ ОПЕРАЦИЙ КОРПОРАТИВНЫХ КЛИЕНТОВ В БАНКИНГЕ

Современные финтех-приложения требуют высокой отказоустойчивости, гибкости и масштабируемости, что делает микросервисную архитектуру оптимальным решением. В отличие от монолитных систем, микросервисы позволяют разделять функциональность на независимые сервисы, которые могут разрабатываться, тестироваться и развертываться отдельно, обеспечивая независимость команд разработки и повышая управляемость системы.[1]

В банковском секторе корпоративные клиенты заключают сделки по различным финансовым продуктам, включая валютный обмен, операции РЕПО, сделки с драгоценными металлами и структурные депозиты. Для каждой операции требуется точный расчет ценовых условий, который зависит от валютной пары, объема сделки, рыночных котировок и внутренних тарифов банка. Система управления ценовыми параметрами сделок предназначена для автоматизации и централизованного контроля над этими условиями, обеспечивая прозрачность и снижение операционных рисков.

В основе системы лежит модуль тиринг, который позволяет формировать и управлять наборами ценовых параметров, заранее настроенными для различных типов финансовых операций. Этот подход позволяет банку гибко изменять условия сделок, оперативно адаптируя их к изменениям рыночной ситуации. Использование микросервисной архитектуры в сочетании с микрофронтендами делает систему легко масштабируемой, а применение Module Federation позволяет динамически загружать интерфейсы и обновлять пользовательский опыт без необходимости перезапуска всей системы. [2]

В процессе разработки системы был проведен анализ требований банковского сектора, что позволило определить ключевые аспекты автоматизации управления ценовыми параметрами сделок. Для обеспечения гибкости, отказоустойчивости и безопасности была выбрана микросервисная архитектура, в которой каждый сервис выполняет свою четко определенную задачу. Центральным компонентом системы является модуль тиринг, который отвечает за хранение, редактирование и применение ценовых параметров к финансовым операциям. Этот модуль позволяет создавать настраиваемые пакеты условий, включающие такие параметры, как валютная пара, объем сделки, ценовой уровень, направление сделки (покупка или продажа) и тенор (режим торговли). Благодаря тирингу банк может централизованно управлять тарифами и оперативно адаптировать их под изменяющиеся рыночные условия.

Для удобства взаимодействия пользователей с системой был создан фронтенд-модуль, реализованный с использованием микрофронтенд-архитектуры и Module Federation. Это решение позволило добиться независимого обновления отдельных компонентов пользовательского интерфейса без необходимости перезагрузки всего приложения. Взаимодействие между фронтендом и бэкендом организовано через API Gateway [3], который маршрутизирует запросы к соответствующим сервисам. Бэкенд-сервисы, разработанные на Java (Spring Boot), выполняют обработку заявок, рассчитывают условия сделок и передают обновленные данные в модуль тиринга. Для хранения информации о ценовых параметрах

была выбрана база данных PostgreSQL, обеспечивающая надежность и быстроедействие. [4] Дополнительно была настроена CI/CD-платформа [6], включающая Jenkins, Bitbucket и Nexus, что позволило автоматизировать процесс развертывания и обновления системы. [5] Module Federation позволяет динамически загружать независимые фронтенд-модули, что упрощает развертывание и обновление пользовательского интерфейса без перезапуска всей системы. [1]

Разработанная система управления ценовыми параметрами сделок автоматизирует процесс расчета ценовых условий, снижает операционные риски и повышает прозрачность финансовых операций. Использование микросервисной архитектуры и микрофронтендов позволило создать гибкую, масштабируемую и отказоустойчивую архитектуру, которая легко адаптируется под изменения рынка. В дальнейшем планируется внедрение алгоритмов машинного обучения для прогнозирования оптимальных условий сделок, интеграция с аналитическими системами для детального аудита и оптимизации тарифов, а также расширение списка поддерживаемых финансовых операций.

ЛИТЕРАТУРА

1. Neha Kaushik, Harish Kumar, Vinay Raj, Micro Frontend Based Performance Improvement and Prediction for Web Applications, 16 April, 2024 [Электронный ресурс] Режим доступа: <https://link.springer.com/article/10.1007/s10723-024-09760-8>
2. Micro-Frontends: Application of Microservices to Web Front-Ends, Andrey Pavlenko, Nursultan Askarbekuly, Swati Megha, and Manuel Mazzara, Innopolis University, Innopolis, Russia a.pavlenko@innopolis.ru, {n.askarbekuly,s.megha}@innopolis.university, m.mazzara@innopolis.ru Sitronics Telecom Solutions (MTS Group), Innopolis, Russia. [Электронный ресурс] Режим доступа: <https://jisis.org/wp-content/uploads/2022/11/jisis-2020-vol10-no2-04.pdf>
3. Comparing Serverless and Microservices Architecture Patterns in FinTech, E-ISSN: 2582-2160. [Электронный ресурс] Режим доступа: <https://www.ijfmr.com/papers/2021/6/2750.pdf>
4. Программная реализация микросервиса банковской системы для управления данными пользователей на основе REST архитектуры, Казанский национальный исследовательский технический университет имени А.Н. Туполева-КАИ, 2021. [Электронный ресурс] Режим доступа: <https://elibrary.ru/item.asp?id=44776642>
5. Проектирование и внедрение микросервисной архитектуры в серверное приложение, Толматов Д.Н., Алтайский государственный технический университет им. И. И. Ползунова, 2024. [Электронный ресурс] Режим доступа: <https://elibrary.ru/item.asp?id=68611330>
6. Использование архитектуры микрофронтенда: инструменты и рекомендации для использования, 2023, Группа компаний «Иннотех» [Электронный ресурс] Режим доступа: <https://elibrary.ru/item.asp?id=54608041>

УДК 004.453

М.А. Гайле, Ю.М. Асадулаев (4 курс бакалавриата),
И.В. Зайцев, ст. преподаватель

РАСПРЕДЕЛЁННАЯ СИСТЕМА ПЕРЕБОРА ХЕШЕЙ С ИСПОЛЬЗОВАНИЕМ GPU КЛАСТЕРОВ

Задача перебора хешей имеет большое значение для областей, связанных с криптографией и обеспечением безопасности данных.

Распределённые системы для перебора хешей основываются на параллельной обработке данных с использованием нескольких узлов, что значительно ускоряет вычисления. Основной принцип таких систем — это разбиение задач на чанки, которые обрабатываются одновременно различными вычислительными узлами. Важно учитывать различия в вычислительных мощностях каждого узла, таких как тип и производительность GPU, поскольку это напрямую влияет на скорость выполнения.

Оптимизация распределения задач требует гибкости и настройки системы с учётом доступных мощностей. Когда количество клиентов увеличивается, а вычислительные характеристики узлов могут сильно варьироваться, система должна эффективно

адаптироваться к этим изменениям. Для этого применяются асинхронные инструменты, которые позволяют управлять задачами и результатами, обеспечивая масштабируемость и надёжность системы.

Кроме того, важным аспектом является использование специализированных программ, таких как Hashcat, которые оптимизируют процесс перебора хешей с учётом возможностей различных типов GPU. В целом, создание эффективной распределённой системы для перебора хешей требует правильного выбора технологий и методов для обработки и распределения вычислительных задач.

В рамках работы была разработана архитектура, состоящая из трёх основных компонентов: клиента, сервера и агента. Клиент (с помощью утилиты Hashcat) выполняет задачи по брутфорс-атаке на хеши, сервер отвечает за распределение задач между подключёнными клиентами в зависимости от их вычислительных мощностей, а агент управляет процессом получения данных через телеграм-бота, выполняет расшифровку GPG-файлов и передачу хешей на сервер для дальнейшей обработки.

Разработанная система в отличие от популярного проекта Kraken не использует Minio для хранения хешей и словарей, что обеспечивает большую производительность при работе с большими наборами данных. Кроме того, разработанное решение имеет встроенную систему шедулинга задач и обеспечивает более эффективное распределение нагрузки. При разработке использовался язык программирования Python, инструмент управления контейнерами Docker совместно с Docker-compose, для эффективного управления задачами была использована система Celery, а для хранения информации о хешах и клиентах была выбрана СУБД PostgreSQL.

Важной частью работы стала настройка взаимодействия между компонентами системы. Особое внимание уделялось корректному распределению чанков хешей и учёту вычислительных мощностей клиентов, что позволило значительно повысить эффективность работы системы.

Реализованная система показала хорошие результаты в распределении нагрузки и эффективном использовании GPU для ускорения перебора хешей.

В дальнейшем планируется продолжить работу над улучшением масштабируемости и надёжности системы, а также применение новых методов для повышения её эффективности.

ЛИТЕРАТУРА

1. Klein M. Cryptography: A Very Short Introduction. Oxford University Press, 2019. 144 pp.
2. Menezes A. J., Vanstone S. A., Oorschot P. C. Handbook of Applied Cryptography. CRC Press, 1996. 784 pp.
3. Miller G. L., Haller M. Hashcat: The Definitive Guide to Cracking Passwords and Security. Apress, 2021. 400 pp.
4. Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. 2nd Edition. Prentice Hall, 2007. 784 pp.

УДК 004.42

М.В. Головин (2 курс магистратуры),
С.А. Молодяков, д.т.н., профессор

РАЗРАБОТКА СИМУЛЯТОРА СИСТЕМЫ СБОРКИ, ПОСТАНОВКИ И ТРАЛЕНИЯ ВОЕННЫХ ТРАЛОВ

Современные военные технологии требуют разработки специализированных программ для обучения людей, так как военные технологии дорогостоящие, а также постоянно обновляются[1]. Это приводит к тому, что обучать использованию оборудования студентов лучше сначала с помощью программного обеспечения, которое симулирует реальное

оборудование, его поведение и при совершении ошибок не приводит к негативному исходу и не содержит риска для реальных объектов.

Данная работа посвящена разработке тренажера для моделирования сложных процессов, таких как сборка, постановка и траление военных тралов с контролированием длины буксира, настройки подачи тока, радиуса акустики и получения данных о ходе траления и координатам судна. Данное ПО предназначено для студентов, изучающих эксплуатацию тралового оборудования. Оно позволяет отрабатывать навыки сборки, постановки тралов, а также управления параметрами траления в безопасной виртуальной среде, без риска повредить оборудование. Так же программный продукт позволяет воспроизводить разные погодные условия и окружающую обстановку без необходимости искать нужное для обучения студентов место и ждать настоящего изменения погоды, что позволяет экономить время и не иметь нарушений в расписании обучения из-за неожиданных изменений окружающей среды. Кроме того, тренажер позволяет проверять знания студентов, что делает его хорошим инструментом для образовательных учреждений.

Целью работы является разработка программного обеспечения, которое позволит моделировать процессы сборки, постановки и траления военных тралов разных типов (ПЭМТ, КТ, АТ и тд)[2] и будет интегрировано в систему и будет взаимодействовать с другими частями реализуемого тренажера.

Для достижения цели необходимо решить следующие задачи:

1. Изучить предметную область и составить требования к системе.
2. Изучить и выбрать технологии, методы, существующие решения.
3. Спроектировать архитектуру симулятора, учитывая его взаимодействие со всей системой.
4. Выбрать средства для реализации программного продукта.
5. Реализовать симулятор.
6. Провести тестирование разработанного продукта.
7. Написать инструкцию по использованию реализованного симулятора.

Основные функции симулятора:

1. Возможность выбрать тип собираемого трала для преподавателя, в зависимости от типа судна.
2. Просмотр всех элементов трала в 360 градусов, для того чтобы не ошибиться при выборе.
3. Возможность делать заметки на экране.
4. Сборка и сброс трала в воду в правильной последовательности.
5. Автоматическая сборка для пропуска данного этапа и перехода к этапу траления.
6. Контроль параметров траления, управления длиной трала и возможностью стравить и выбрать трал.
7. Управление подачей тока на тралы.
8. Возможность сброса вехи и контроля натяжения.

Для разработки программного продукта были выбраны следующие средства и технологии:

Для реализации симулятора был выбран язык C++ благодаря высокой производительности, богатому выбору фреймворков и библиотек.

Фреймворк Qt был выбран так это кроссплатформенный фреймворк, который предоставляет обширные библиотеки для работы с графическим интерфейсом и позволяет создавать динамические пользовательские интерфейсы с минимальными усилиями. Данный фреймворк предоставляет удобные инструменты для работы с многопоточность. Позволяя разрабатывать высокопроизводительные приложения, имеет хорошую документацию, активное сообщество разработчиков, а также интегрируется с другими технологиями.

В качестве среды разработки была выбрана IDE Qt Creator, поскольку эта IDE разработана специально для разработки приложений с использованием Qt. Qt Creator имеет удобный интерфейс, удобный текстовый редактор кода, предоставляющий навигацию по коду

и его рефакторинг, а также удобный визуальный редактор интерфейсов, позволяющий сократить написание кода.

Для хранения данных была выбрана СУБД PostgreSQL. PostgreSQL – это ORDBMS, наиболее развитая из открытых СУБД в мире. Она имеет открытый исходный код, позволяет создавать собственные функции, операторы, обеспечивает целостность данных при высокой нагрузке и имеет большое сообщество разработчиков и обширную документацию

На рисунке 1 представлено схема взаимодействия компонентов программного продукта со всей системой, с базой данных, а также схема пользовательского интерфейса.

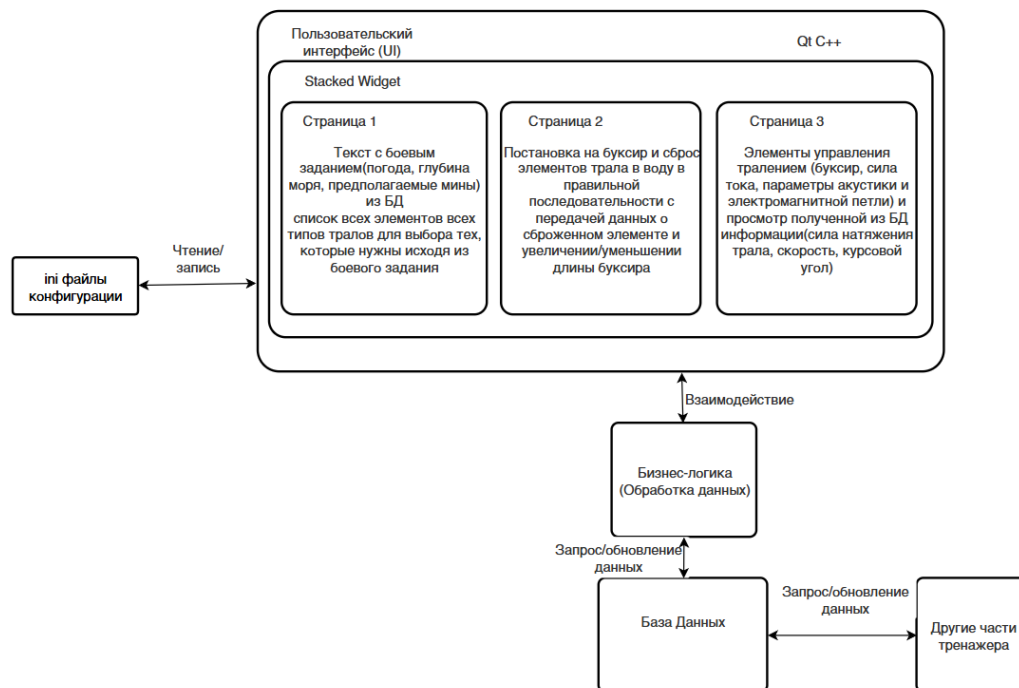


Рисунок 1 – Общая схема взаимодействия компонентов

Таким образом, рассмотрена задача разработки симулятора системы сборки, постановки и траления военных тралов, определены основные функции, разработана структура проекта и выбран стек технологий.

ЛИТЕРАТУРА

1. Коршунов Э. Л. Документы о послевоенном тралении в операционной зоне Северного флота в 1945 г. – 2015 – С.234
2. Йолтуховский В. М. Тральщики ВМФ СССР в предвоенные и военные годы //Судостроение. – 2015. – №. 5. – С. 73-77.
3. Lee Zhi Eng, Qt 6 C++ GUI Programming Cookbook – 2022. ISBN: 978-1801813040 – С 10.

УДК 004.056.5

А.А. Гладышев (4 курс бакалавриата),
Н.В. Щербаков, к.т.н., доцент

РАЗРАБОТКА API-СЕРВИСА ДЛЯ БЕЗОПАСНОГО ХРАНЕНИЯ И УПРАВЛЕНИЯ ДОСТУПОМ К ДАННЫМ

Современные системы хранения и совместного использования данных требуют обеспечения одновременной защищенности, доступности и отказоустойчивости. Традиционные централизованные решения (например, Amazon S3, Google Drive) упрощают администрирование данных, но формируют единую точку отказа и ограничивают контроль пользователей над собственными данными [1]. Применение децентрализованных подходов на базе IPFS (InterPlanetary File System) позволяет распределять файлы по множеству узлов и

сохранять их целостность за счет хранения метаданных, включая хэширование [2]. Дополнительно использование блокчейна обеспечивает фиксацию метаданных файлов и историю их изменений, повышая прозрачность и исключая несанкционированные изменения [3]. Важной компонентой распределенного хранилища данных является удобный механизм разграничения доступа, часто реализуемый на базе ролевой модели (RBAC) [4].

Для обеспечения надежности и гибкости распределенного хранилища используются специальные архитектурные решения. Наиболее целесообразное архитектурное решение базируется на принципах мультисервисной среды, где функциональные компоненты распределяются и выполняются в так-называемых контейнерах. Контейнеры легко адаптируются к особенностям применения, включая как вертикальное, так и горизонтальное масштабирование.

Данная работа представляет оригинальный распределенный сервис для хранения данных. Мы используем IPFS [5] для локального хранения пользовательских файлов в узлах системы. Коммуникация между узлами осуществляется посредством смарт-контрактов в сети Ethereum [7], которые обеспечивают обмен и хранение информации о версиях, правах доступа и фактах изменений. Такая двухуровневая архитектура облегчает масштабирование и позволяет конечным пользователям сохранять контроль над собственными данными без необходимости доверять посторонним точкам централизованного управления и хранения данных.

Таким образом, целью работы является создание решения для безопасного хранения файлов с версионированием и разграничением доступа, основанного на децентрализованных технологиях. В работе приводятся следующие результаты:

1. Обзор существующих подходов к централизованному и децентрализованному хранению данных, включая IPFS и блокчейн-решения.
2. Сравнительный анализ вариантов управления правами доступа (DAC, MAC, RBAC, ABAC) с учетом масштабируемости и удобства администрирования.
3. Архитектура API-сервиса, обеспечивающего интеграцию с IPFS и смарт-контрактом для хранения метаданных и версий файлов.
4. Механизм версионирования и блокчейн-фиксации изменений для обеспечения неизменности ключевых атрибутов.

В качестве основного инструмента для реализации интерфейса с IPFS и блокчейном выбран язык Go [6] ввиду его производительности и минималистичного синтаксиса, что упрощает разработку сетевых сервисов. Разработанный сервис позволяет пользователю загружать файлы, формируя первую версию в децентрализованном хранилище, а также создавать последующие версии с автоматической записью в блокчейн. Права на чтение и редактирование определяются загрузчиком и сохраняются в структуре доступа, которая учитывается при выполнении операций над файлом.

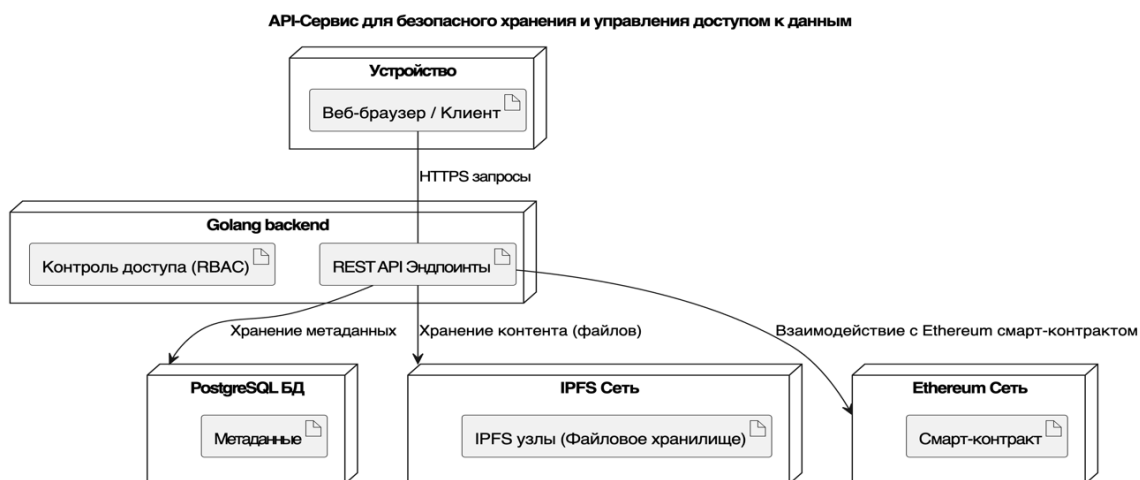


Рисунок 1 – Архитектура системы.

Таким образом, сочетание IPFS и Ethereum обеспечивает прозрачное хранение данных и сохранность истории версий. Применение ролевой модели (RBAC) обеспечивает разграничение прав доступа пользователей к чувствительной информации. Система отличается повышенной устойчивостью к нештатным ситуациям, поскольку отсутствует единая точка отказа, а все изменения метаданных хранятся в блокчейне, что исключает несанкционированные модификации.

ЛИТЕРАТУРА

1. Gabriel T., Cornel-Cristian A., Arhip-Calin M., Zamfirescu A. Cloud Storage. A comparison between centralized solutions versus decentralized cloud storage solutions using Blockchain technology // Материалы 54-й Международной конференции университетов по энергетике (UPEC). 2019. С. 1–5.
2. Alizadeh M., Andersson K., Schelén O. Efficient Decentralized Data Storage Based on Public Blockchain and IPFS // Материалы Азиатско-Тихоокеанской конференции IEEE по компьютерным наукам и обработке данных (CSDE). Голд-Кост, Австралия, 2020. С. 1–8.
3. Bhalibar K., Singh A., Sharma H., Uphadyay A., Gupta H. Centralize Storage System with Encryption vs Decentralize Storage System Using Blockchain // Материалы Международной конференции по инновационным вычислениям и коммуникациям (ICICC), 2022.
4. Лапин С. А. Сравнительный анализ использования существующих моделей разграничения доступа в системах, обладающих равнозначными объектами // Известия АлтГУ. 2016. №1 (89).
5. IPFS документация. [Электронный ресурс] Режим доступа: <https://docs.ipfs.tech/>
6. Golang документация. [Электронный ресурс] Режим доступа: <https://go.dev/doc/>
7. Ethereum документация. [Электронный ресурс] Режим доступа: <https://ethereum.org/en/developers/docs/>

УДК 004.891.2

Н.С. Дергунов (2 курс бакалавриата),
А.Д. Ковалев, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНОЙ ДИАГНОСТИКИ ЗАБОЛЕВАНИЙ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННЫХ СЕТЕЙ ТИПА «ТРАНСФОРМЕР»

Современная медицина сталкивается с необходимостью оперативного и точного дифференциального диагноза, требующего анализа множества симптомов, лабораторных данных и анамнеза пациента. Однако, даже опытные врачи могут испытывать трудности при интерпретации схожих клинических проявлений различных заболеваний. В этом контексте системы искусственного интеллекта способны автоматизировать процесс дифференциальной диагностики становятся ключевым инструментом для снижения числа диагностических ошибок и оптимизации временных затрат. При этом, критически важным аспектом остается доступность таких систем для практикующих врачей, что требует разработки удобных интерфейсов, интегрируемых в повседневную клиническую практику.

Таким образом, главной целью работы является автоматизация процесса дифференциальной диагностики заболеваний за счет реализации и использования программного комплекса для постановки диагнозов на основе имеющихся симптомов пациентов.

Традиционные подходы на основе методов машинного обучения, такие как деревья решений или классические нейронные сети, часто ограничены в обработке сложных контекстуальных зависимостей между симптомами и диагнозами. В последние годы также активно исследуются подходы с использованием больших языковых моделей (LLM) и обучения с подкреплением (RL), которые демонстрируют потенциал в выявлении диагнозов на основе неструктурированных текстовых данных. Таблица 1 содержит результаты сравнительного анализа современных подходов к решению поставленной задачи. В работах использовался датасет DDxPlus [1] в качестве структурированного источника данных. Он

содержит клинические случаи с аннотированными симптомами, демографическими признаками и подтвержденными диагнозами, что обеспечивает репрезентативность обучения и валидность результатов. Датасет DDxPlus используется и в этой работе. Существующие исследования содержат вывод о том, что такие методы, несмотря на их гибкость, могут уступать в точности при работе с ограниченными размеченными медицинскими данными, а также требовать значительных вычислительных ресурсов.

Таблица 1 – Сравнение известных решений

Подгруппа	Модель обучения	Фреймворки	Точность предсказания заболевания (accuracy)	F1-score при определении наиболее вероятных диагнозов	Реализация	Информация о внедрении
Обучение с подкреплением	AARLC	rlpyt	0.99	0.78		Нет
	BASD		0.97	0.83		
	CASANDE		0.98	0.94		
Большие языковые модели [2]	GPT-4	Неизвестно	0.84	-	Закрыта	Есть API (могут быть проблемы с доступом)
	GPT-3.5		0.69	-		
	Gemini Pro		0.65	-		
Совместная работа агентов больших языковых моделей	GPT-4	MdAgents [3]	0.80	-		
	GPT-3.5		0.74	-		
	Gemini Pro		0.79	-		
Трансформеры	DDxT [4]	PyTorch	0.98	0.94		Нет

В отличие от них, трансформерные архитектуры демонстрируют высокую эффективность в задачах обработки последовательностей и анализа семантических связей, что делает их перспективными для медицинских приложений.

В работе предлагается разработать систему дифференциальной диагностики на основе трансформерной модели ELECTRA [5, 6], предобученной на датасете DDxPlus и оснащенной веб-приложением для взаимодействия с пользователями. Выбор ELECTRA обусловлен её высокой эффективностью при меньших вычислительных затратах благодаря механизму замены токенов в процессе предобучения, что особенно актуально для медицинских задач с ограниченными объемами данных. Научная новизна исследования заключается не только в адаптации архитектуры трансформера для анализа многомерных клинических признаков и генерации ранжированного списка дифференциальных диагнозов, но и в создании интуитивно понятного веб-интерфейса, обеспечивающего врачам возможность быстро получать AI-поддержку в режиме реального времени.

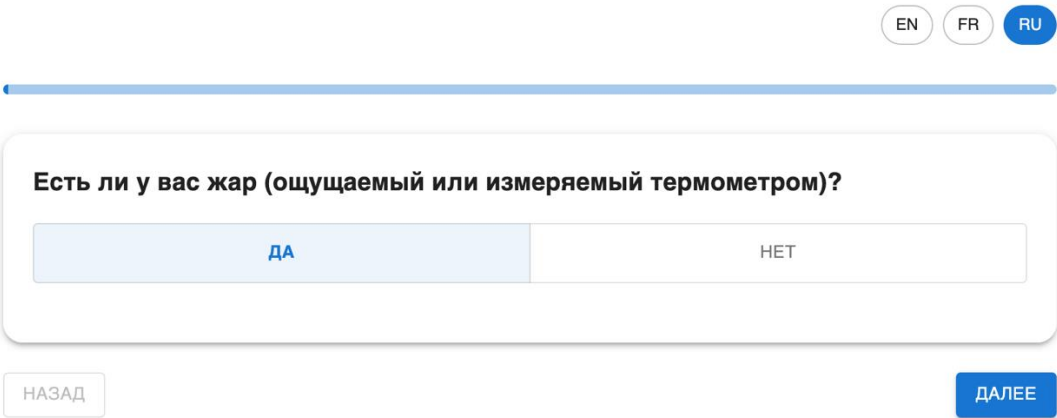


Рисунок 1 – Скриншот веб-интерфейса предлагаемого решения

Отличительной особенностью предлагаемой системы является её практико-ориентированный дизайн: веб-приложение позволяет вводить симптомы и анамнез пациента через удобные формы, а затем получать ранжированный перечень возможных диагнозов с оценкой вероятности и визуализацией значимых признаков через механизм внимания трансформера. Это не только повышает доверие к модели, но и упрощает интеграцию системы в рабочий процесс медицинских специалистов. Веб-решение разработано на базе современных фреймворков: FastAPI для бэкенда и React для фронтенда. Их применение обеспечивает масштабируемость, безопасность данных и кроссплатформенную доступность. Рисунок 1 демонстрирует интерфейс пользователя при ответе на вопросы для выявления заболевания.

ЛИТЕРАТУРА

1. Arsene Fansi Tchango, Rishab Goel, Zhi Wen, Julien Martel, Joumana Ghosn. DDxPlus: A New Dataset For Automatic Medical Diagnosis // arXiv, 2022, arXiv:2205.09148.
2. Yue Huang, Lichao Sun. TrustLLM: Trustworthiness in Large Language Models // arXiv, 2024, arXiv:2401.05561.
3. Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, Hae Won Park. MDAgents: An Adaptive Collaboration of LLMs for Medical Decision-Making // arXiv, 2024, arXiv:2404.15155.
4. Mohammad Mahmudul Alam, Edward Raff, Tim Oates, Cynthia Matuszek. DDxT: Deep Generative Transformer Models for Differential Diagnosis // arXiv, 2023, arXiv:2312.01242.
5. Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators // arXiv, 2020, arXiv:2003.10555.
6. J. S. Pedersen, M. S. Laursen, C. Soguero-Ruiz, T. R. Savarimuthu, R. S. Hansen and P. J. Vinholt. Domain over size: Clinical ELECTRA surpasses general BERT for bleeding site classification in the free text of electronic health records // 2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), Ioannina, Greece, – 2022, pp. 1-4, doi: 10.1109/BHI56158.2022.9926955.

УДК 004.415.2

А.В. Дмитриев (4 курс бакалавриата),
А.М. Хахина, д.т.н., профессор

РАЗРАБОТКА МОДЕЛИ И СИСТЕМЫ АНАЛИЗА И ПРОГНОЗИРОВАНИЯ ПРОДУКТИВНОСТИ СОТРУДНИКОВ НА ОСНОВЕ МЕНЕДЖЕРА ЗАДАЧ

В современных рабочих условиях компании разного масштаба стремятся анализировать, поддерживать и повышать продуктивность своих сотрудников. Однако один из факторов, который влияет на качество и количество выполняемых задач сотрудников – профессиональное выгорание [1]. Данная проблема становится всё более и более значимой, и актуальной, поэтому пишутся статьи, где её пытаются тщательно изучить [2][3]. Более того, проводятся изучения взаимосвязи профессионального выгорания от различных факторов [4].

Существующие решения по управлению задачами, не предоставляют инструментов для глубокого анализа и прогнозирования эффективности выполнения задач и рисков выгорания сотрудников. Большинство доступных решений ориентированы исключительно на управление задачами без учёта внешних факторов и исторических данных, влияющих на продуктивность сотрудников [5]. Например, как иностранные (Jira, ClickUp), так и российские приложения менеджеры задач (Яндекс Трекер, Visary Tracker [6] и другие) не обладают функционалом, описанным ранее.

Таким образом, возникает задача разработки системы, которая будет сочетать функции менеджера задач с возможностями анализа производительности и прогнозирования выгорания с учетом внешних факторов, таких как разные события или праздничные дни. Целью данной системы является использование исторических данных о выполнении задач, внешних

факторов (например, графики праздников или учебных сессий у студентов) и методов математического моделирования для создания прогноза вероятности выполнения задач.

Основная задача, стоящая перед разработкой в рамках данной работы, заключается в создании модели анализа и прогнозирования производительности сотрудников, которая сможет учитывать следующие факторы: текущую нагрузку, внешние обстоятельства и исторические данные. Решение будет реализовано в виде системы с возможностью интеграции в существующие процессы управления задачами или менеджера задач с использованием модели прогнозирования производительности сотрудников.

Для достижения поставленной цели необходимо решить следующие задачи:

- Анализ существующих решений управления задачами с выявлением их недостатков в области анализа продуктивности и прогнозирования рисков.
- Разработка архитектуры системы.
- Выбор и реализация математических моделей для прогнозирования вероятности выполнения задач и рисков выгорания сотрудников.
- Интеграция данных о внешних факторах, такие как праздничные дни или учебные и экзаменационные сессии в модель анализа.
- Реализация системы с применением современного технологического стека.
- Демонстрация работы модели или продукта с внедрением в рабочую команду.

Разработанная система объединит функционал классического таск-трекера и инструменты анализа данных, позволяя учитывать нагрузку сотрудников, исторические показатели выполнения задач и внешние обстоятельства. Математическое моделирование включает использование методов временных рядов, регрессионного анализа или машинного обучения для составления прогнозов выполнения задач и выявления рисков профессионального выгорания. В рамках работы будут проведены исследования на то, какие методы лучше всего себя проявят в работе и будут наиболее точно предсказывать продуктивность сотрудников.

Система предполагает формирование гибкой отчётности с детализацией, адаптированной под потребности разных заинтересованных сторон и команд), включая графические визуализации продуктивности и прогнозов выполнения задач.

Техническая реализация системы предполагает использование следующих технологий:

- Python для математического моделирования (NumPy, SciPy, pandas) [7];
- Веб-фреймворк Django для серверной части приложения;
- JavaScript-библиотека React и фреймворк Tailwind CSS для реализации интерфейса;
- PostgreSQL для хранения и обработки данных.

Использование предложенного технологического стека обеспечивает высокую производительность, масштабируемость и удобство интеграции с уже существующими процессами управления задачами.

Таким образом, предложенная система позволит существенно повысить качество управленческих решений, предотвратить снижение продуктивности сотрудников и минимизировать риски профессионального выгорания, что делает разработку актуальной как в научном, так и в практическом аспектах.

ЛИТЕРАТУРА

1. МКБ-11 для ведения статистики смертности и заболеваемости [Электронный ресурс] / Всемирная организация здравоохранения. – URL: <https://icd.who.int/browse/2024-01/mms/ru#129180281> (дата обращения: 20.01.2025).
2. Нетребко Елена Эдуардовна ПРОФЕССИОНАЛЬНОЕ ВЫГОРАНИЕ // Глобус: психология и педагогика. 2019. №2 (30). URL: <https://cyberleninka.ru/article/n/professionalnoe-vygoranie> (дата обращения: 20.01.2025).
3. Варварова Л. С., Серая Н. Н. СИНДРОМ ПРОФЕССИОНАЛЬНОГО ВЫГОРАНИЯ // Форум молодых ученых. 2018. №12-1 (28). URL: <https://cyberleninka.ru/article/n/sindrom-professionalnoe-vygoraniya-3> (дата обращения: 20.01.2025).

4. Бочкарева Е.Н. Взаимосвязь профессионального выгорания и эмоционального интеллекта сотрудников контактного центра // Психология. Журнал Высшей школы экономики. 2016. Т. 13. № 3. С. 576–587. URL: <https://psy-journal.hse.ru/2016-13-3/196063638.html> (дата обращения: 20.01.2025).
5. Таск-трекеры: помощники в управлении проектами // vc.ru. 2024. [Электронный ресурс] – URL: <https://vc.ru/education/1330109-task-trekery-pomoshniki-v-upravlenii-proektami> (дата обращения: 20.01.2025).
6. Visary Tracker — российский аналог Jira для эффективного управления задачами и проектами // itWeek. 2023. [Электронный ресурс] – URL: <https://www.itweek.ru/management/news-company/detail.php?ID=228030> (дата обращения: 20.01.2025).
7. Кирдяев М. М. Обзор языка программирования Python для решения задач математического моделирования // НиКа. 2016. №. URL: <https://cyberleninka.ru/article/n/obzor-yazyka-programmirovaniya-python-dlya-resheniya-zadach-matematicheskogo-modelirovaniya> (дата обращения: 20.01.2025).

УДК 004.453

О.В. Доценко (4 курс бакалавриата),
Г.В. Смородников, Р.А. Золотарев (2 курс магистратуры),
А.В. Самочадин, к.т.н., доцент

РЕАЛИЗАЦИЯ СРЕДСТВ ПРОГНОЗИРОВАНИЯ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

Облачные вычисления — это инфраструктура, решения, софт и ресурсы, которые провайдер предоставляет через сеть, точнее — через облако [1].

Ключевыми аспектами, связанными с облачными вычислениями, являются:

1. Оптимизация затрат: Избежание дефицита ресурсов; Сокращение избыточного резервирования; Автоматическое масштабирование [2].

Поскольку дефицит/недостаток ресурсов могут привести к большим денежным затратам: Облачные ресурсы стоят дорого, поэтому необходимо стараться избегать лишних затрат, но если ресурсов окажется мало для предоставления оплаченных клиентами услуг, то это приведет к штрафам, что и тоже ведет к большим затратам.

2. Повышение производительности и надежности, а также предотвращение кризисов: Предотвращение перегрузки системы; Оптимизация распределения ресурсов; Улучшение планирования; Прогнозирование сбоев

Некоторые методы прогнозирования позволяют прогнозировать возможные сбои и потенциальные точки перегрузки в работе оборудования или программного обеспечения, эти прогнозы позволяют оптимизировать распределение ресурсов между различными приложениями и сервисами и/или принять меры для их предотвращения

3. Поддержка инноваций и развитие бизнеса: Оптимизация затрат; Привлечение и удержание клиентов. Качественная бесперебойная работа сервиса будет приводить к положительному опыту использования для потребителей, что может привести новых клиентов и предотвратить отток старых.

Облачные среды характеризуются высокой динамичностью, с постоянным изменением нагрузки во времени. Прогнозирование должно учитывать эту динамику и адаптироваться к изменяющимся условиям. Для большинства задач, требуется предсказание на ближайшее будущее требуемых ресурсов облачных вычислений, то есть решение задачи прогнозирования временных рядов.

Прогнозирование временных рядов — это процесс анализа данных временных рядов с использованием статистики и моделирования для дальнейшего прогнозирования и принятия стратегических решений [4].

Основная цель прогнозирования временных рядов — предсказать будущие значения на основе прошлых и текущих данных. Это позволяет определить тренды и паттерны, которые помогают принимать обоснованные решения в бизнесе и науке.

Целью данной работы является предсказание параметров использования ресурсов облачных вычислений на основе прошлых и текущих данных.

Есть разные подходы прогнозирования временных рядов:

1. Авторегрессионные модели: ARIMA, SARIMA, AR, ARMA
2. Методы экстраполяции: Линейная экстраполяция, экспоненциальное сглаживание
3. Методы машинного обучения (регрессия, нейронные сети): xLSTM, LSTM, GRU, RNN, Bidirectional LSTM, LightGBM, Stacked LSTM, TCN, DNN и др

В данной работе использовались методы машинного обучения: xLSTM, LSTM, GRU, RNN, LightGBM, TCN, DNN. Использование других подходов оказалось менее эффективным в силу специфики данных, используемых для обучения.

Для решения задачи прогнозирования предлагается следующая схема (рисунок 1), которая показывает, что входные данные, содержащие информацию о параметрах использования, обрабатываются различными алгоритмами предсказания (модель 1, 2 ... n). Полученные результаты поступают на вход полносвязной нейронной сети, которая формирует общий результат предсказания.

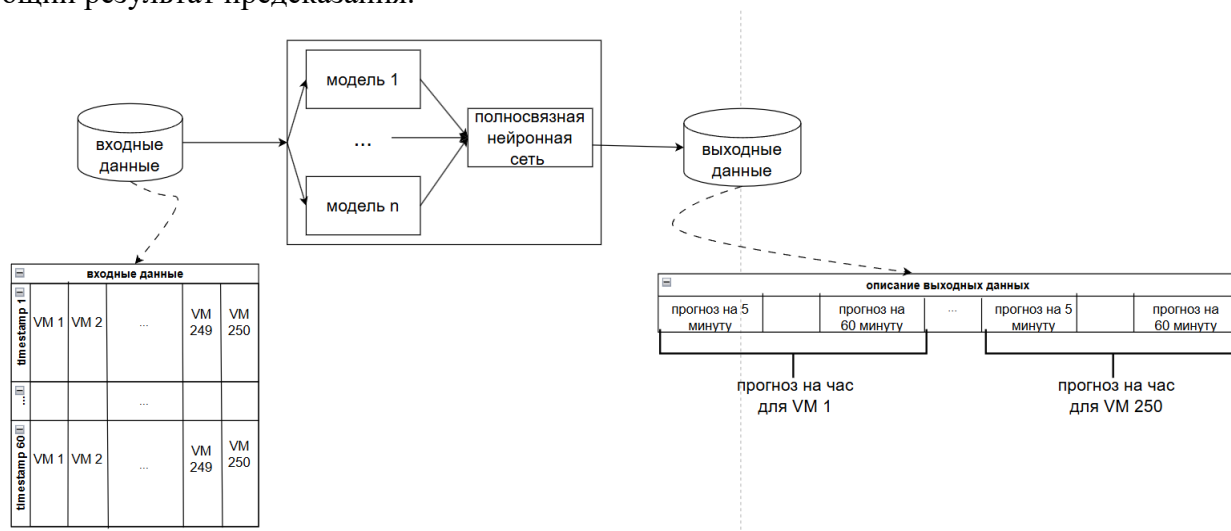


Рисунок 1 – Схема проекта с описанием входных/выходных данных

Технологический стек:

1. Сбор данных: Данные взяты из набора данных GWA-bitbrains [5].
2. Обработка и подготовка данных:
 - Язык программирования: Python.
 - Библиотеки обработки данных: Pandas; NumPy; Scikit-learn; Datetime.
3. Моделирование и машинное обучение:
 - ML/DL Frameworks: TensorFlow: фреймворк для глубокого обучения; Scikit-learn: для классических алгоритмов ML (линейная регрессия, деревья решений, случайный лес).
 - Алгоритмы: TCN; LSTM [3]; ARIMA; XGBoost, LightGBM; Dense: полносвязная нейронная сеть для принятия финального решения на основе предсказаний остальных алгоритмов.
 - Инструмент для машинного обучения: Google Colab: для интерактивной разработки и экспериментов.

В ходе работы над решением данной задачи были реализованы несколько алгоритмов: TCN, LSTM и т.д., после чего собраны в ансамбль с использованием полносвязной нейронной сети.

ЛИТЕРАТУРА

1. Что такое облачные вычисления? Виды технологий и сервисов в облаке. [Электронный ресурс] //Ссылка <https://servercore.com/ru/blog/articles/what-is-cloud-computing-types-of-cloud-based-technologies-and-services/> (дата обращения 11.03.2025)

2. Тенденции развития облачных вычислений 2024 [Электронный ресурс] //Ссылка: <https://blog.rubrain.com/cloud-computing-trends-for-2024.html> (дата обращения 03.03.2025)
3. Exploring Machine Learning Approaches for Time Series [Электронный ресурс] //Ссылка <https://www.geeksforgeeks.org/exploring-machine-learning-approaches-for-time-series/> (дата обращения 03.03.2025)
4. Time Series Analysis and Forecasting [Электронный ресурс] //Ссылка https://translated.turbopages.org/proxy_u/en-ru.ru.9f258f17-67d06565-3d98e560-74722d776562/https/www.geeksforgeeks.org/time-series-analysis-and-forecasting/#what-is-time-series-forecasting (дата обращения 11.03.2025)
5. gwa-bitbrains [Электронный ресурс] //Ссылка: <https://www.kaggle.com/datasets/gauravdhamane/gwa-bitbrains> (дата обращения 11.03.2025)

УДК 004.453

И.Г. Дусеев (4 курс бакалавриата),
Е.Е. Андрианова, ст. преподаватель

ИССЛЕДОВАНИЕ МЕТОДОВ ПРОЕКТИРОВАНИЯ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ДЛЯ ВЕБ-СЕРВИСА И МОБИЛЬНОГО ANDROID-ПРИЛОЖЕНИЯ ПО РЕЦЕНЗИРОВАНИЮ И ПОДБОРУ КНИГ

В условиях цифровизации выбор книг и их обсуждение переходят в онлайн-среду, что сталкивает пользователей с проблемой информационной перегрузки. Для решения этой задачи важны платформы, оптимизирующие поиск и предоставляющие персонализированные рекомендации. Разработка рекомендательной системы, основанной на коллаборативной и контентной фильтрации, а также гибридных алгоритмах, становится ключевым элементом для учета предпочтений и поведения пользователей.

Общая схема работы рекомендательной системы, после использования сервиса формируется профиль пользователя с учетом его предпочтений, предпочтения пользователя сравниваются с общим набором книг в базе данных, в последствии после сравнения выдаются рекомендации (рисунок 1):

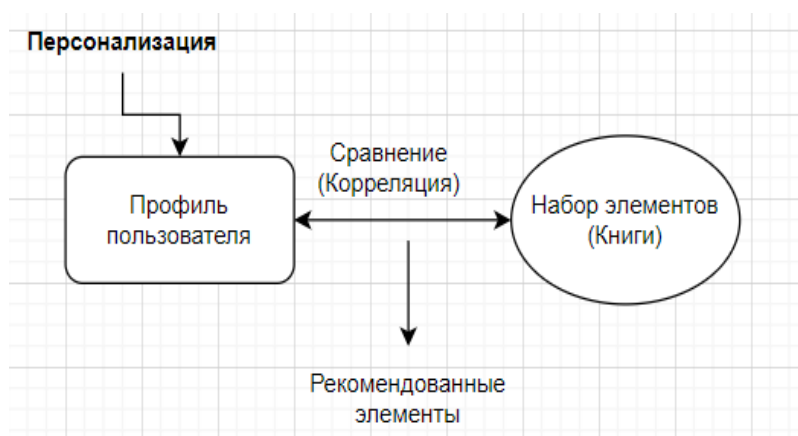


Рисунок 1 – Общая схема работы рекомендательной системы

Рекомендательные системы можно условно разделить на три основные категории, каждая из которых имеет свои подходы и особенности:

Контентно-ориентированный подход (Content-based Recommendation):

Рекомендации основаны на содержательных характеристиках книг и предпочтениях конкретного пользователя. Например, если пользователь положительно оценил книгу из жанра "фантастика", ему будут предложены другие книги того же жанра, похожего автора или с аналогичным описанием[1].

Алгоритм работы: Создание профиля книги жанр, автор, ключевые слова, далее анализ активности пользователя (прочитанные книги, рейтинги), затем сравниваются профили книг и предпочтений пользователя с использованием метрик (косинусное сходство, TF-IDF).

Преимущества: Простота интерпретации, Адаптация под индивидуальные предпочтения, не требует большого количества данных о других пользователях.

Недостатки: Система не предложит что-то вне зоны интереса пользователя, требуется детальная информация о контенте каждой книги

Коллаборативный фильtring (Collaborative Filtering):

Рекомендации формируются на основе предпочтений других пользователей. Если два пользователя оценили похожие книги схожим образом, то книги, которые нравятся одному, с большой вероятностью понравятся и другому[2].

Алгоритм работы: Построение матрицы взаимодействий пользователей с книгами, строки соответствуют пользователям, а столбцы – книгам (например, выставленные оценки, написанные рецензии, добавление в избранное), Применение алгоритмов User-based Collaborative Filtering, Item-based Collaborative Filtering для рекомендаций пользователям:

Технологии: библиотека Surprise (Python), Spark, MLlib.

Преимущества: учитывает коллективные предпочтения, что даёт разнообразные рекомендации, не требует подробных характеристик книг (они могут быть неизвестны), легко расширяется для большого количества пользователей и книг.

Недостатки: Проблема холодного старта: нет данных о новых книгах или новых пользователях, система может работать плохо с недостаточными данными.

Гибридные методы (Hybrid Recommendations):

Гибридные системы объединяют сильные стороны нескольких подходов (контентно-ориентированных рекомендаций и коллаборативного фильтра) для более точных и разнообразных рекомендаций [3].

Алгоритм работы: скомбинировать результаты двух или более алгоритмов (например, с помощью взвешенного среднего), создание метамодели, которая использует рекомендации нескольких моделей и обучается на их результатах (Stacking).

Преимущества: устранение слабостей отдельных подходов. Например, гибридная система решает проблему холодного старта, возникающую в коллаборативном фильтре, за счёт анализа контента, более точные и релевантные рекомендации за счёт объединения сил разных методов

Недостатки: более сложная реализация и вычислительная затратность

Вывод: реализация рекомендательной системы должна сочетать несколько подходов. Контентно-ориентированные методы учитывают индивидуальные предпочтения, коллаборативная фильтрация — коллективный опыт, а гибридный подход объединяет их преимущества. Это обеспечит персонализацию, удобство и конкурентное преимущество системы.

ЛИТЕРАТУРА

1. Rendhiva, Lidia & Hartanti, Dwi & Yuliana, Margaretha. (2024). Indibiz Product Recommendation System Using Knowledge Base Method. Jurnal Inotera. 9. 391-400. 10.31572/inotera.Vol9.Iss2.2024.ID381.
2. Husain, Hisham & Monteil, Julien. (2024). Geometric Collaborative Filtering with Convergence. 10.48550/arXiv.2410.03064.
3. Noordi, Daffa & Hasanah, Herliyani & Sumarlinda, Sri. (2024). Marvel Movie Recommendation System Using Hybrid Item-Based and Content-Based Filtering Methods. TIERS Information Technology Journal. 5. 13-19. 10.38043/tiers.v5i1.5209.

ОБЗОР МЕТОДОВ РАЗРАБОТКИ ANDROID ПРИЛОЖЕНИЯ ДЛЯ ПОДГОТОВКИ К
ОЛИМПИАДЕ ПО СУБД POSTGRESQL

В современном мире цифровых технологий мобильные приложения стали неотъемлемой частью повседневной жизни. Среди множества операционных систем для мобильных устройств распространённость Android составляет более 70% [1]. Это делает разработку приложений для этой системы важным и востребованным направлением для изучения.

Для подготовки студентов Политеха к олимпиаде по СУБД PostgreSQL [2] необходимо создать удобное мобильное приложение, которое позволит изучать теоретическую информацию, проходить тестирование и решать задачи уровня олимпиады.

Целью данной статьи является проведение обзора основных методов разработки Android приложений с целью выявить лучший для написания требуемого приложения.

Для достижения этой цели необходимо:

1. Рассмотреть конструкторы, движки, платформы для создания мобильных приложений и их возможности, ограничения.

2. Провести сравнительный анализ каждого из методов.

Данный обзор так же, поможет начинающим разработчикам определить для себя интересные инструменты, которые можно будет изучить и использовать в дальнейшем для реализации своих проектов.

Конструкторы для создания мобильных приложений представляют собой специализированные платформы, которые позволяют собирать приложения из готовых шаблонов и модулей, что предоставляет возможность получить желаемое приложение людям без знания программирования. Наиболее популярными являются конструкторы: AppsGeyser, Draftbit, Kodika. Преимущества этого способа создания приложения заключаются в экономии бюджета, скорости, простоте. Недостатками же являются зависимость от платформы конструктора, ограничения производительности, ограниченность функционала и дизайна [3]. Разработанное приложение для Политеха должно быть производительным, так как необходима высокая скорость работы с базой данных для быстрой обработки пользовательских запросов, поэтому данный подход не желателен.

Движком называется программная платформа, которая предоставляет набор инструментов, библиотек и функций для упрощения разработки приложений. Они могут выполнять рендеринг графики, управление ресурсами, взаимодействие с аппаратными компонентами устройства. Такая возможность позволяет разработчикам заострить внимание на реализации бизнес-логики. Движки часто используются для создания игр, но их функционал также применяют в неигровых проектах в сферах архитектуры, машиностроения и анимации. Одним из самых популярных является движок Unity, к нему предоставлена обширная документация, база знаний и активное сообщество разработчиков. Unity поддерживает множество плагинов и инструментов, поэтому проект можно легко адаптировать под нужды проекта. Однако приложения, созданные на этом инструменте, весьма тяжеловесные и требуют оптимизации [4]. Известен своими графическими возможностями движок Unreal Engine, плюсом для разработчиков является хорошая оптимизация благодаря языку программирования C++, а для непрограммистов возможность создать приложение с помощью визуального языка программирования Blueprints. Недостатки этого движка заключаются в дороговизне дополнительных модулей и высокие системные требования. Так как приложение для подготовки к олимпиаде не подразумевает сложной анимации и графики, то данный метод непригоден.

Наиболее сложным методом является написание полностью самостоятельно кода для мобильного приложения ввиду необходимости обладания качественной технической подготовки. Этот метод дорогостоящий как по времени, так и по материальным ресурсам, но

он позволяет достичь выполнения всех требований, предъявленных к проекту. Android Studio является официальной IDE для разработки приложений для операционной системы Android. Android Studio предоставляет специализированный функционал для упрощения разработки такие, как система сборки на основе Gradle, интеграция с Git, инструмент для работы с пользовательским интерфейсом Layout Inspector и многие другие. Приложения состоят из компонентов, которые можно вызывать по отдельности. Компоненты представляют собой пользовательский интерфейс, например, основной компонент Activity запускается в момент нажатия на иконку приложения или перехода в него через всплывающие уведомления. Рекомендующим языком программирования является Kotlin, есть возможность использовать Java и C++. Таким образом, разработка на описанной платформе обладает следующими характеристиками: высокая энергоэффективность, масштабируемость, производительность, безопасность [5], [6]. Еще одной известной платформой является Xamarin, предоставляющая среду выполнения автоматической обработки процессов выделения памяти, сборки мусора и т. д. Xamarin содержит строго типизированные привязки к большинству базовых пакетов SDK Android, интеграцию с Visual Studio, доступ к нативным API, что позволяет писать приложения с высокой производительностью. Недостатком приложений, написанных на этой платформе, является большой размер занимаемой памяти в сравнении с приложениями, написанными в Android Studio. Для создания качественных и продуманных приложений этот способ является наиболее подходящим, хоть и достаточно дорогостоящим.

После проведенного обзора методов можно сделать вывод, что разрабатывать приложение для СПбПУ им. Петра Великого стоит на платформе Android Studio вследствие обширного функционала, большого сообщества, подробной документации и возможности продумать архитектуру приложения самостоятельно. Поддержка языка программирования Kotlin позволит писать менее громоздкий код в сравнении с C#, который поддерживается в Xamarin. Наличие инструмента для просмотра и отладки пользовательского интерфейса в Android Studio позволит быстро проанализировать иерархию UI, что поможет разумно распределять время при разработке приложения.

В данной работе были рассмотрены три метода разработки Android приложений. В зависимости от целей и потребностей стоит выбирать один из методов. В случае потребности в быстром и дешевом решении можно воспользоваться конструкторами. Для сложных высокопроизводительных приложений лучше писать свой код, так как готовые шаблонные решения не приведут к успешному, уникальному продукту, который будет приносить высокий доход.

ЛИТЕРАТУРА

1. Mobile Operating System Market Share Worldwide: [Электронный ресурс] Режим доступа: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения: 10.03.2025).
2. СУБД Postgres Pro – IT-Планета 2025 – BRAIM: [Электронный ресурс] Режим доступа: https://challenge.braim.org/landing/postgres_contest.
3. Создание программ на Андроид: как и на чем писать OTUS: [Электронный ресурс] Режим доступа: <https://otus.ru/journal/sozдание-programm-na-android-kak-i-na-chem-pisat/>
4. Стёпушкина П. К. ИГРОВЫЕ ДВИЖКИ ДЛЯ ОБРАЗОВАТЕЛЬНЫХ ЦЕЛЕЙ // Вестник науки. 2024. №5 (74). Режим доступа: <https://cyberleninka.ru/article/n/igrovyie-dvizhki-dlya-obrazovatelnyh-tseley>.
5. Нуждин Д. Г. ОЦЕНКА И ВЫБОР НАИБОЛЕЕ ПОДХОДЯЩИХ ИНСТРУМЕНТОВ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ // Universum: технические науки. 2023. №11-1 (116). Режим доступа: <https://cyberleninka.ru/article/n/otsenka-i-vybor-naibolee-podhodyaschih-instrumentov-razrabotki-mobilnyh-prilozheniy>
6. Фешина Е. В., Куштано С. А., Крамаренко Т. А., Скорбатьюк Р. Я. АНАЛИЗ ТЕХНОЛОГИЙ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ И ИНФОРМАЦИОННЫХ СИСТЕМ НА БАЗЕ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2022. №1 (296). Режим доступа: <https://cyberleninka.ru/article/n/analiz-tehnologiy-razrabotki-mobilnyh-prilozheniy-i-informatsionnyh-sistem-na-baze-operatsionnoy-sistemy-android>

РАЗРАБОТКА ИНТЕРАКТИВНОГО ТРЕНАЖЁРА ПО SYSTEM DESIGN

В современном мире многие разработчики сталкиваются с трудностью проектирования масштабируемых и отказоустойчивых систем, особенно на ранних этапах своей карьеры. Однако традиционные методы обучения системному проектированию часто ограничены теорией и недостатком практических заданий. Цель данного проекта — разработка интерактивного тренажёра по System Design, позволяющего пользователям практиковаться в проектировании распределённых систем.

В рамках нашего проекта мы выделили следующие ключевые задачи проекта:

1. Разработка интуитивно понятного веб-интерфейса для проектирования высоконагруженных систем.
2. Формирование библиотеки задач, охватывающих ключевые аспекты проектирования: масштабируемость, отказоустойчивость, распределённые базы данных, кеширование и балансировку нагрузки.
3. Реализация механизма автоматической проверки решений.
4. Обучение пользователей через встроенные образовательные модули.

Архитектура нашего приложения построена по принципу «Цитадель». Такой подход обеспечивает надежность, безопасность и гибкость, позволяя масштабировать систему и выделять дополнительные микросервисы при необходимости. При разработке системы мы используем современные технологии:

1. Backend:

- Java 21 — современный язык программирования.
- Spring Boot 3.4.0 — фреймворк для создания производительных и масштабируемых приложений на Java.
- Liquibase — управление миграциями базы данных.
- JOOQ — безопасное взаимодействие с базами данных.
- PostgreSQL 15 — надёжная реляционная база данных.
- KeyCloak — система управления доступом и аутентификацией.
- Swagger — автоматизированная документация API.
- Prometheus и Grafana — инструменты для мониторинга и логирования.

2. Frontend:

- React 18.3.1 — современная библиотека для клиентской части приложения.
- Typescript 5.6.2 — язык программирования
- Zustand — управление состоянием приложения.
- Tanstack Router — библиотека для маршрутизации в React-приложениях,
- Jest — тестирование пользовательского интерфейса.
- Tanstack React Query — удобное взаимодействие с API.
- Zod — валидация данных для взаимодействия с API.
- I18n — библиотека, предназначенная для реализации поддержки нескольких языков в приложении.
- React Flow — визуализация компонентов и диаграмм.
- Chakra UI — библиотека готовых UI компонентов.

3. Инфраструктура:

- Docker-Compose, Kubernetes — инструменты для контейнеризации и управления сервисами.
- GitLab CI — автоматизация CI/CD.

Рассмотрим ключевые характеристики разрабатываемой системы:

1. Производительность — тренажер обрабатывает запросы пользователей с высокой скоростью благодаря использованию современных технологий.
2. Масштабируемость — архитектура построена по принципу «Цитадель», который обеспечивает гибкость и возможность выделения дополнительных сервисов.
3. Безопасность — использование Keycloak для управления аутентификацией и авторизацией повышает защиту пользовательских данных.
4. Удобство использования — веб-интерфейс на React 18.3.1 обеспечивает интуитивно понятное взаимодействие с системой.

Разрабатываемый тренажер по System Design является инновационным продуктом для подготовки IT-специалистов. Он позволит развивать навыки проектирования сложных высоконагруженных систем, дополняя традиционные методы обучения практическими заданиями. В дальнейшем планируется расширение функционала приложения и внедрение машинного обучения для моделирования системы в режиме реального времени.

ЛИТЕРАТУРА

1. Григорьев М. В., Григорьева И. И. Проектирование информационных систем. - Москва: Юрайт, 2019. - 278 с.
2. Spring Boot // Spring URL: <https://docs.spring.io/spring-boot/index.html> (дата обращения: 01.12.2024).
3. Documentation // React URL: <https://react.dev/learn> (дата обращения: 01.12.2024).
4. Getting started with jOOQ // Jooq URL: <https://www.jooq.org/doc/3.21/manual-single-page/#getting-started> (дата обращения: 11.12.2024).
5. Documentation // PostgreSQL URL: <https://www.postgresql.org/docs/> (дата обращения: 05.12.2024).
6. Spring Modulith // Хабр URL: <https://habr.com/ru/articles/701984/> (дата обращения: 15.12.2024).
7. Getting Started with jOOQ // Baeldung URL: <https://www.baeldung.com/jooq-intro> (дата обращения: 12.12.2024).

УДК 004.021

М.А. Житнухина (4 курс бакалавриата),
Ю.В. Литвинов, к.т.н., ст. преподаватель

SENSOR FUSION ДЛЯ УЛУЧШЕНИЯ ВЫКЛАДЫВАНИЯ МОДЕЛИ ДОРОГИ В 3D В ADAS

В настоящее время рынок автомобилей с системами помощи водителю (англ. ADAS, Advanced driver-assistance systems) растет [4]. В таких системах важно уметь точно проецировать объекты из плоскости изображения в систему координат дороги, чтобы отдавать команды при управлении, основываясь на как можно более точной информации об окружающем мире. Для того, чтобы делать такие проекции, необходимо знать ориентацию камеры относительно дороги. При движении автомобиля камера изменяет свое положение относительно дороги из-за встречающихся на дороге неровностей, «лежащих полицейских», сильно меняется угол тангажа. Точность проецирования повышается, если учитывать эти изменения.

Для оценки изменения тангажа чаще всего используют камеру или инерциальный измерительный модуль (англ. IMU, Inertial measurement unit), состоящий из гироскопа, акселерометра и магнетометра.

Компенсация тряски при помощи камеры как правило реализуется при помощи оптического потока [1], но таким способом не всегда возможно точно оценить изменение угла: при отсутствии далеких объектов или в темноте данных с камеры недостаточно.

При использовании данных с IMU возникает проблема зашумленности данных и погрешностей при проставлении временных меток [7].

Можно добиться более точной оценки ориентации камеры, если использовать несколько датчиков одновременно. Процесс объединения данных с разных датчиков называется sensor fusion. Существуют разные методы sensor fusion, например, фильтр Калмана [2], комплементарный фильтр [3].

Целью данной работы является реализация алгоритма sensor fusion для оценки изменения угла наклона автомобиля в режиме реального времени.

Для достижения этой цели необходимо решение следующих задач.

1. Обзор существующих решений sensor fusion.
2. Реализовать алгоритм определения ориентации по IMU.
3. Реализовать алгоритм определения ориентации по данным с камеры.
4. Реализовать алгоритм sensor fusion.
5. Протестировать алгоритм.

Алгоритм определения ориентации камеры был реализован при помощи оптического потока. Для последовательных кадров вычисляется сдвиг в пикселях, который переводится в угол, внутренние параметры камеры (фокальное расстояние и оптический центр) известны.

В данном решении оценивается ориентация по IMU при помощи фильтра Мэджвика [5]. Существуют две вариации этого фильтра: с использованием магнетометра и без. Было решено не использовать магнетометр, так как его данные зашумлены на автомобиле. В качестве аргументов он принимает данные гироскопа и акселерометра.

Для оценки сдвига используются производная угла, получаемого при помощи камеры, и данные гироскопа. Вычисляется сдвиг по времени, при котором значение кросс-корреляции двух временных рядов наиболее близко к единице.

Для реализации sensor fusion планируется в вариацию фильтра Мэджвика, использующую магнетометр, вместо данных магнетометра подавать другие данные. Как один из параметров будет подаваться угол камеры, в качестве двух других будут взяты экстраполированные значения углов, полученных от IMU по предыдущим измерениям.

Для реализации используется язык C++, так как на нем можно писать достаточно производительный код для использования в системах реального времени. Ссылка на репозиторий не приведена из-за соглашения о неразглашении.

Тестирование алгоритма планируется внутри симулятора Webots [6], а также при помощи оценки накопленной ошибки на круговом маршруте на автомобиле. Будет произведена визуальная оценка, стабилизируется ли изображение данным алгоритмом.

ЛИТЕРАТУРА

1. Оптический поток. [Электронный ресурс] Режим доступа: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html (дата обращения: 17 марта 2025 г.).
2. Ribeiro Maria, Ribeiro Isabel. Kalman and Extended Kalman Filters: Concept, Derivation and Properties. — 2004. — 04. — URL: <https://doi.org/10.1115/1.3662552> (дата обращения: 17 марта 2025 г.).
3. Kundra László, Ekler Péter, Charaf Hassan. Improving orientation estimation in mobiles with built-in camera // 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom). — 2013. — P. 765–770. — URL: <https://ieeexplore.ieee.org/document/6719202> (дата обращения: 17 марта 2025 г.).
4. Рынок ADAS. [Электронный ресурс] Режим доступа: <https://www.polarismarketresearch.com/industry-analysis/advanced-driver-assistance-systems-adamarket> (дата обращения: 17 марта 2025 г.).
5. Sebastian O. H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical Report, University of Bristol, 2010. — 04. — URL: https://x-io.co.uk/downloads/madgwick_internal_report.pdf (дата обращения: 17 марта 2025 г.).
6. Webots. [Электронный ресурс] Режим доступа: <https://cyberbotics.com/> (дата обращения: 17 марта 2025 г.).
7. Aerts Peter, Demeester Eric. Time Synchronisation of Low-cost Camera Images with IMU Data based on Similar Motion // International Conference on Informatics in Control, Automation and Robotics. — 2019. — 01. — P. 292–299. — URL: <https://www.scitepress.org/Papers/2019/78376/78376.pdf> (дата обращения 7 марта 2025 г.).

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ДИСКРЕТНО-СОБЫТИЙНОГО
МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ОБЩЕСТВЕННОГО ТРАНСПОРТА.

Современные системы моделирования транспортных систем позволяют анализировать и оптимизировать движение общественного транспорта в условиях реальной городской инфраструктуры [6]. Такие системы обеспечивают возможность оценки загруженности маршрутов, прогнозирования пассажиропотока и выявления узких мест транспортной сети. Однако существующие решения зачастую являются зарубежными продуктами, а также сложны в конфигурировании и не обладают возможностью использования собственных оптимизационных алгоритмов, таких как, например, вычисление данных по разным маршрутам параллельно для ускорения вычислений.

Одним из способов повышения эффективности моделирования является разработка веб-сервиса, предоставляющего API для взаимодействия с системой, позволяющей проводить имитационное моделирование, управлять входными данными, анализировать результаты и формировать отчеты в удобном формате. Использование современных технологий, таких как Python[1] и FastAPI[2], обеспечивает достаточную производительность, а применение PostgreSQL[3] в качестве базы данных позволяет эффективно обрабатывать большие объемы информации о транспортной сети и пассажиропотоках.

Результатами моделирования являются данные о перевозке пассажиров в транспортной сети, которые могут быть использованы для принятия решений по построению оптимального расписанию движения общественного транспорта включая конфигурацию транспортных средств, их количество и интервалы движения.

Таким образом, целью данной работы является разработка программного средства для моделирования движения общественного транспорта, обеспечивающего гибкость настройки, высокую точность моделирования и удобство анализа результатов.

Для достижения этой цели необходимо решение следующих задач:

1. Обзор существующих подходов к моделированию транспортных потоков.
2. Проведение сравнительного анализа найденных решений.
3. Разработка модели для транспортных потоков с учетом загруженности маршрутов и пассажиропотоков.
4. Реализация данного подхода в виде веб-сервиса с API.
5. Проведение экспериментов и анализ эффективности предложенного решения.

Основным компонентом разработанной системы является дискретно-событийная модель, реализованная с использованием языка программирования Python. Данная модель позволяет учитывать различные характеристики транспортных средств, такие как вместимость и скорость, а также параметры движения пассажиров. Веб-сервис на базе FastAPI обеспечивает доступ к функциональности системы, а PostgreSQL используется для хранения данных о конфигурации транспортной сети - маршрутах, пассажиропотоках.

Архитектура разработанной системы представлена на рисунке 1. Входные данные, включая количество транспортных средств, их характеристики и расписание движения отправляются пользователем, конфигурация транспортной сети предварительно загружается в базу данных, после чего запускается процесс моделирования. В ходе моделирования автоматически собирается статистика по следующим параметрам:

- Загрузка транспортных средств.
- Пассажиропотоки на маршрутах.
- Среднее и максимальное время ожидания пассажиров.

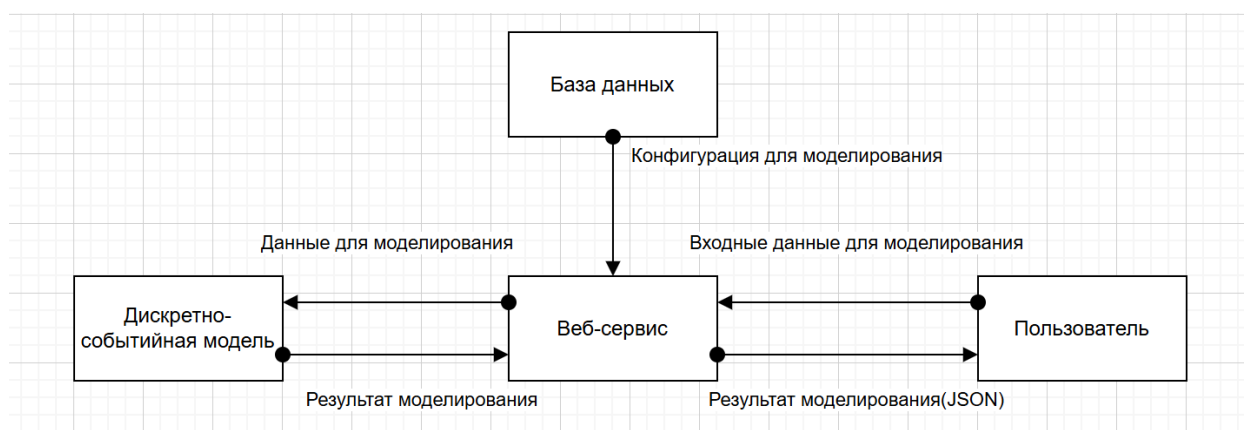


Рисунок 1 – Архитектура системы моделирования

Результаты моделирования доступны через API, позволяя пользователям анализировать данные в формате JSON и подключать внешние инструменты визуализации, которые умеют работать с данным форматом.

В рамках данной работы был проведен сравнительный анализ существующих решений, а также разработан и реализован новый подход к моделированию транспортных потоков с учетом реальных данных и пользовательских настроек. Программное средство обеспечивает удобный доступ к информации о движении общественного транспорта, позволяя эффективно управлять транспортной сетью и принимать обоснованные решения по ее оптимизации.

ЛИТЕРАТУРА

1. Python programming language. [Электронный ресурс] Режим доступа: <https://www.python.org/>
2. PostgreSQL [Электронный ресурс] Режим доступа: <https://www.postgresql.org/>
3. FastAPI framework [Электронный ресурс] Режим доступа: <https://fastapi.tiangolo.com/>
4. Информационные системы и технологии в логистике и на транспорте.
Константин Павлович Манжула.
5. Русак С.Н. Зверко Е.К. Методы имитационного моделирования.
6. Евин И.А., Соловьев А.А., Хабибулин Т.Ф. Модели общественного транспорта Москвы на основе теории сложных сетей.

УДК 004.42

А.Р. Загородняя (3 курс бакалавриата),
В.Э. Шамаков, к.т.н., доцент

АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ СКЛАДОМ НА ОСНОВЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

В условиях современного рынка эффективное управление складом является ключевым фактором конкурентоспособности компании. Предлагается решение для автоматизации складского учета на основе мобильного приложения, интегрированного с системой управления складом. Разработанное приложение позволяет оптимизировать процессы приемки, хранения, отгрузки товаров, а также проведения инвентаризации с использованием мобильных устройств.

Традиционные системы управления складом часто связаны с высокими затратами на внедрение и обслуживание, а также ограничены в возможностях мобильного доступа. Разработка собственного мобильного приложения позволяет учесть специфику бизнес-процессов компании, сократить издержки и повысить оперативность работы склада [1].

Проведен сравнительный анализ популярных систем управления складом: SAP EWM [2], 1С:Предприятие 8 WMS Логистика [3], МойСклад [4]. Выявлены их преимущества и

недостатки, обоснована необходимость разработки собственного решения с учетом специфики компании.

Результаты сравнения различных вариантов (описанных выше) при необходимости автоматизации работы склада сведены в таблице 1.

Таблица 1 – Сравнительная характеристика систем управления складом

Характеристики	SAP EWM	1С:Предприятие 8. WMS Логистика	МойСклад (SaaS)	Собственная разработка
Складские операции	Большой спектр	Большой спектр	Базовые	Большой спектр
Дополнительные операции	Требуется дополнительных решений	Возможна разработка на базе встроенного языка	Отсутствуют	Любые по необходимости
Поддержка отечественного учета	При необходимости	Ориентирована на отечественный учет	При необходимости	Ориентация на любой учет при разработке
Работа с оборудованием (весы, сканеры и т.п.)	При необходимости	Легко интегрируется с любым оборудованием	Только оговоренным	Ориентация на любое оборудование при разработке
Технологии радиочастотной идентификации	Возможно использование	Возможно использование	Только оговоренные	Ориентация на любые технологии при разработке
Удаленное использование	Необходимы дополнительные решения	Необходимы дополнительные решения	Ориентирована на использование	Ориентация на любые технологии при разработке
Возможность использования бесплатных продуктов	Для малых предприятий	Для малых предприятий	Только тестовый период	Да

Разработанная система представляет собой клиент-серверное приложение. Серверная часть реализована с использованием СУБД SQL Server 2019. Мобильное приложение предоставляет следующий функционал:

1. авторизация пользователей;
2. оформление приходных и расходных документов;
3. проведение инвентаризации с помощью сканирования штрих-кодов;
4. отслеживание движения товаров на складе;
5. формирование отчетов о наличии и движении товаров.

Для хранения данных используется реляционная база данных, спроектированная с учетом принципов нормализации и целостности. Разработаны сущности для хранения информации о товарах, контрагентах, сотрудниках, документах и др.

В результате разработки получено мобильное приложение, позволяющее автоматизировать учет товаров на складе, сократить количество ошибок и повысить эффективность работы. Дальнейшее развитие системы связано с внедрением технологий радиочастотной идентификации (RFID), интеграцией с WMS [5] системами верхнего уровня и расширением функциональности мобильного приложения.

ЛИТЕРАТУРА

1. Как мобильное приложение помогает оптимизировать бизнес-процессы производства. [Электронный ресурс] Режим доступа: <https://delaweb.ru/blog/kak-mobilnoe-prilozhenie-pomogaet-optimizirovat-biznes-procetsy-proizvodstva/>

2. Расширенное управление складами (EWM) [Электронный ресурс] Режим доступа: <https://www.itctg.ru/solutions/sap-ewm>
3. 1С. Отраслевые и специализированные решения. 1С: Предприятия. [Электронный ресурс] Режим доступа: <http://www.solutions.1c.ru/catalog/wms4/features>
4. МойСклад. Торговля, склад и CRM онлайн. [Электронный ресурс] Режим доступа: www.moysklad.ru/upravlenie-skladom/
5. Технологии RFID идентификации. [Электронный ресурс] Режим доступа: <https://isbc-rfid.ru/applications/warehouses/>

УДК 004.7

З.Д. Иванов (3 курс бакалавриата),
Б.М. Медведев, к.т.н., доцент

РАЗРАБОТКА ИНСТРУМЕНТА ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ ДЛЯ ОТСЛЕЖИВАНИЯ МАРШРУТА ВЫЗОВА НА СЕТИ ОПЕРАТОРА СВЯЗИ

Современные сети операторов связи представляют собой сложные инфраструктуры, требующие постоянного мониторинга и анализа для обеспечения стабильной работы. Одной из ключевых задач является отслеживание маршрута вызова, что позволяет оперативно выявлять и устранять неполадки, а также оптимизировать работу сети. В процессе развития сети количество точек коммутации и управления увеличилось, что привело к сложности диагностики ошибок, таких как потеря соединения между абонентами. Разработка инструмента для автоматизации этого процесса является актуальной задачей, так как ручное отслеживание маршрута вызова требует значительных временных и человеческих ресурсов.

Целью работы является разработка программного инструмента для автоматизации процесса отслеживания маршрута вызова на сети оператора связи. Для достижения этой цели решаются следующие задачи:

1. Разработка алгоритмов взаимодействия с промежуточным сервером на базе протокола SSH, анализа подробной записи о вызове (ПЗВ) центрального коммутатора и определения маршрута вызова на основе данных из различных точек коммутации.
2. Разработка алгоритмов синхронизации времени и анализа лог-файлов систем телефонной сети.
3. Разработка программных средств отслеживания маршрута вызова.
4. Тестирование разработанных программных средств.

Алгоритм агрегации данных контролирует весь процесс построения вызова, формируя запросы в системы и составляя на их основе последовательность точек маршрутизации.

Алгоритм взаимодействия с промежуточным сервером позволяет по имени сервера и наборам команд выполнять запросы выборки на внутренних серверах. Полученные ответы он конвертирует в формат входных данных алгоритма анализа системных данных вызова.

Алгоритм анализа системных данных вызова, читая лог-файлы системы «АнтиФрод», серверов переадресации, SIP-телефонии и записи из БД PostgreSQL, возвращает данные о результате прохождения вызова на каждой точке маршрутизации, изменения номеров и следующую точку маршрутизации, при наличии.

Алгоритм синхронизации времени лог-файлов включен в алгоритм анализа лог-файлов, выявляет расхождения в временных значениях и корректирует их для исключения расхождений.

Структура разработанных программных средств приведена на рисунке 1.

Модуль агрегации данных на основе пользовательского ввода составляет SSH-запросы на сервера телефонии. SSH-клиент обеспечивает приём и передачу SSH-запросов с поддержкой аутентификации и шифрования данных. Модуль анализа системных данных вызова выполняет чтение данных из различных точек коммутации, включая синхронизацию

времени событий. Модуль визуализации составляет схематическое представление маршрута вызова на основе данных модуля агрегации.

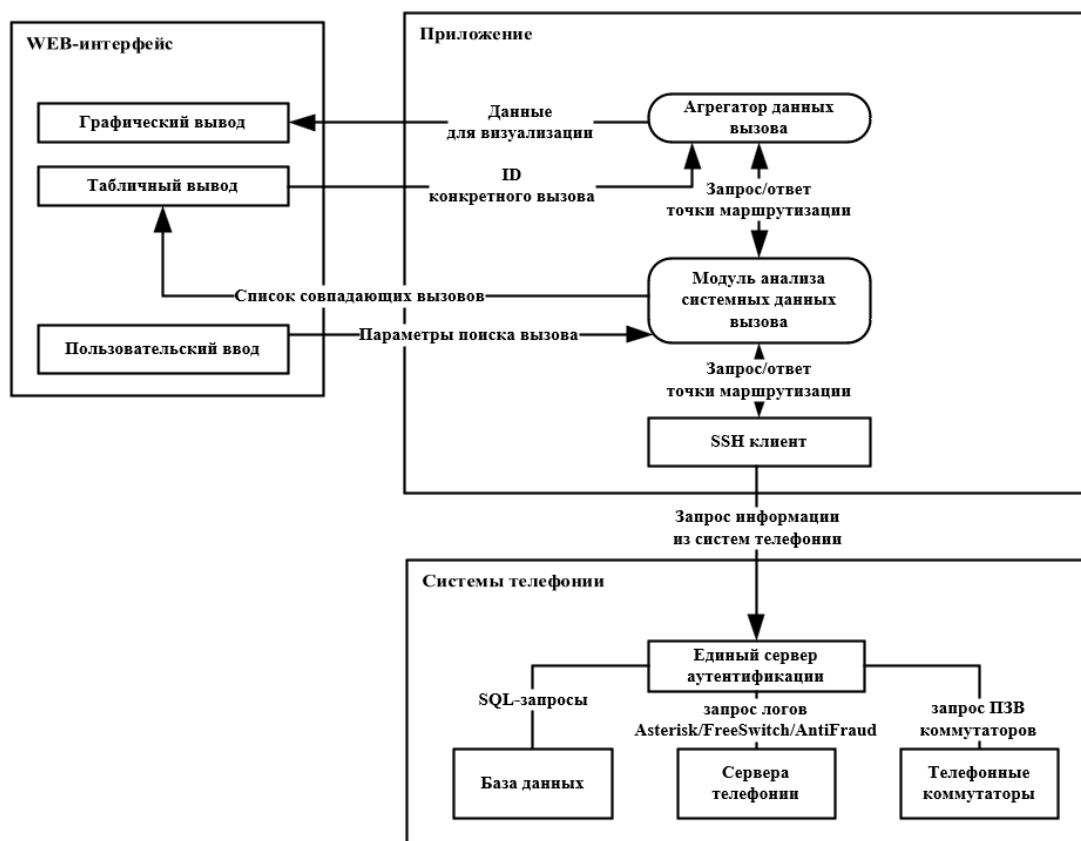


Рисунок 1 – Структура разработанных программных средств

Для тестирования разработанных программных средств было создано порядка 25 тестов проверки корректности подключения к промежуточному серверу, аутентификации SSH-сессии, передачи запросов и анализа лог-файлов. Выполненная оценка производительности показала, что обеспечивается обработка порядка 12-16 вызовов в час. Результаты тестирования с использованием модульного тестирования фреймворка JUnit показали высокую точность определения маршрута вызова более 95%. Оставшиеся 5% связаны в основном с устаревшими АТС, не поддерживающими современные протоколы сигнализации.

Разработанные программные средства могут быть использованы в составе систем технической поддержки операторов связи для повышения эффективности диагностики и устранения неполадок. Автоматизация процесса отслеживания маршрута вызова значительно сокращает время диагностики.

Задачами дальнейшего исследования являются расширение функциональности инструмента для поддержки протоколов сигнализации H.323 для видеоконференцсвязи и CAS для поддержки аналоговой сети, а также оптимизация программных средств для работы в сетях с высокой нагрузкой.

ЛИТЕРАТУРА

1. Иванов И. И., Петров А. А. Современные методы мониторинга сетей связи // Журнал телекоммуникаций. – 2023. – № 5. – С. 45-50.
2. Smith J., Brown R. Network Monitoring and Analysis. – 2022. – 300 p.
3. Протокол SIP: основы и применение / Под ред. В. В. Сидорова. – М.: Издательство "Связь", 2021. – 200 с.
4. Оптимизация сетей связи: методы и инструменты / Под ред. П. П. Иванова. – М.: Издательство "Техносфера", 2022. – 250 с.
5. Java JSch [Электронный ресурс] Режим доступа: <http://www.jcraft.com/jsch/>

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО РЕШЕНИЯ ДЛЯ УДАЛЁННОГО ПОДКЛЮЧЕНИЯ
К ГРАФИЧЕСКОЙ СРЕДЕ ОПЕРАЦИОННЫХ СИСТЕМ НА БАЗЕ ЯДРА LINUX

Для удалённого подключения к компьютерам операционной системы Windows наиболее распространённым и производительным решением является протокол RDP (Remote Desktop Protocol), разработанный корпорацией Microsoft. RDP позволяет передавать изображение, события клавиатуры и мыши, звук, прокидывать USB-устройства, а также поддерживает продолжение удалённой работы в сессии, ранее запущенной локально, и наоборот. Отдельно стоит отметить удобство окна подключения – виртуальный экран будет растягиваться автоматически при изменении размеров окна, что бывает очень полезно при работе на нескольких мониторах с разным разрешением.

При переходе на операционную систему на базе ядра Linux, продвинутый пользователь, часто использовавший ранее удалённое подключение RDP на Windows, захочет завладеть схожими возможностями, но столкнётся с проблемой – аналогичного решения, включающего весь вышеупомянутый функционал, не существует.

RDP-сервер, разработанный Microsoft, является закрытым решением – корпорация не стала разрабатывать его для Linux. Однако, энтузиасты решили воспроизвести работу этого сервера, что привело к появлению «xrdp» – RDP-сервера с открытым исходным кодом. Протестировав данное решение, стоит отметить, что при должной настройке (пусть и местами сложной – в частности, для настройки звука необходимо устанавливать «xrdp» с помощью специальных скриптов установки [1]), данное решение позволит получить весь функционал, доступный в Windows. Но при работе по глобальной сети Интернет производительность подключения значительно падает. Попытки оптимизировать соединение через настройку параметров «xrdp» и «xrdp-sesman» [2], а также прокидывание через SSH [3] с применением алгоритмов сжатия не помогли решить проблему. Только значительное уменьшение передаваемых цветов [4] может сделать соединение более стабильным, но значительно ухудшает сам процесс работы.

Существует ещё один графический протокол – VNC (Virtual Network Computing) [5]. Он является прямым аналогом RDP, а первый VNC-сервер имеет открытый исходный код, что привело к появлению множества аналогичных VNC-решений – как расширяющих «x11vnc», так и написанных с нуля. Наиболее продвинутым и стабильным из них является TigerVNC.

Клиент TigerVNC позволяет передавать изображения и события клавиатуры и мыши, а также может синхронизировать буфер обмена. Ещё он позволяет напрямую настраивать параметры подключения Xvnc, в отличие, например, от решения ThinLinc, которое разрабатывается теми же разработчиками TigerVNC как решение для корпоративных организаций. Клиент TigerVNC есть во всех популярных дистрибутивах в официальных репозиториях, в том числе и для Windows, что делает данное решение кроссплатформенным. Также, в отличие от некоторых других VNC-клиентов (noVNC, RealVNC, TurboVNC, UltraVNC), клиент может автоматически менять размер виртуального экрана при изменении окна подключения – клиент посылает серверу соответствующие запросы при изменении размеров окна, а на стороне сервера скрипты меняют размер дисплея X11.

VNC разрабатывался специально для передачи трафика по сети, и его производительность при подключении по Интернету, по сравнению с «xrdp», намного выше. Но VNC поддерживает передачу только изображения графической сессии и событий клавиатуры и мыши, а соединение не является безопасным, так как ограничивается небольшим паролем до восьми символов и не поддерживает шифрование [6]. Данная проблема решается использованием SSH-туннелирования. Через SSH будем также подключать звук, используя «pactl» (PulseAudio) [7], а для USB-устройств – используя «usbip» (USB IP) [8].

Ещё одна проблема, которая не была упомянута – это возможность входа в одну и ту же сессию, локально или удалённо. На Windows по RDP мы можем зайти в локальную сессию и продолжить работу, а в Linux мы так сделать не сможем – каждое отдельное подключение пользователя рассматривается как новая сессия. Это вызвано тем, что Linux изначально проектировался для мультипользовательской и мультисессионной работы. А современные окружения рабочих столов, таких как KDE, не допускают в системе запуск нескольких своих процессов под одним пользователем одновременно. Это приводит к чёрным экранам при попытке удалённого входа при наличии уже запущенной локальной сессии. Решить данную проблему можно путём использования виртуальной сессии VNC. При авторизации локально мы сразу будем создавать виртуальную сессию и подключаться к ней через простой оконный менеджер. При удалённом подключении, будем эту сессию перехватывать, тем самым отключая пользователя локально – имитируя то, как работает сессия RDP в операционных системах Windows. Условный алгоритм работы приведён на рисунке 1.

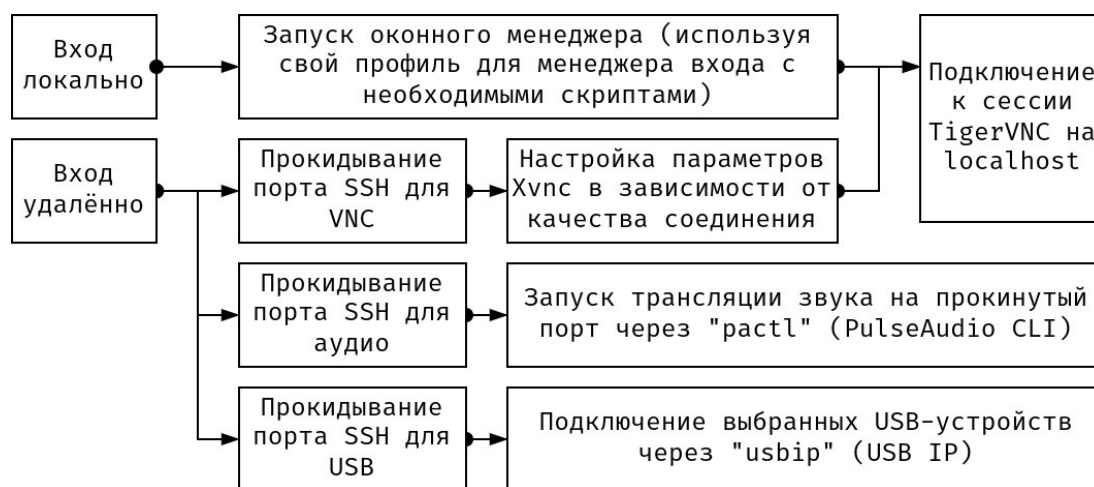


Рисунок 1 – Условный алгоритм при подключении локально и удалённо

Таким образом, объединив кроссплатформенное решение TigerVNC со скриптами по прокидыванию звука и USB-устройств, туннелированием по SSH для шифрования трафика, и отдельными скриптами по управлению сессиями пользователей при их входе локально и удалённо, мы сможем создать монолитное решение для удалённого подключения к графической среде операционной системы на базе ядра Linux.

ЛИТЕРАТУРА

1. xRDP – Easy install xRDP on Ubuntu [Электронный ресурс]. Режим доступа: <https://c-nergy.be/blog/?p=19228>
2. Slow Ubuntu Remote Desktop using xrdp [Электронный ресурс]. Режим доступа: <https://superuser.com/questions/1539900/slow-ubuntu-remote-desktop-using-xrdp>
3. Speed-optimization of Xrdp server connection when connected via SSH tunnel [Электронный ресурс]. Режим доступа: <https://serverfault.com/questions/1079559/speed-optimization-of-xrdp-server-connection-when-connected-via-ssh-tunnel>
4. Xrdp Slow [Электронный ресурс]. Режим доступа: <https://github.com/neutrino-labs/xrdp/issues/1600>
5. All you need to know about VNC remote access technology [Электронный ресурс]. Режим доступа: <https://discover.realvnc.com/what-is-vnc-remote-access-technology>
6. VNC – Security [Электронный ресурс]. Режим доступа: <https://en.wikipedia.org/wiki/VNC#Security>
7. How to carry audio over SSH [Электронный ресурс]. Режим доступа: <https://askubuntu.com/questions/371687/how-to-carry-audio-over-ssh>
8. USB/IP [Электронный ресурс]. Режим доступа: <https://wiki.archlinux.org/title/USB/IP>

РАЗРАБОТКА СРЕДСТВ ОЦЕНКИ ИНФОРМАТИВНОСТИ И ДИЗАЙНА СТРАНИЦ МАРКЕТПЛЕЙСОВ НА ОСНОВЕ АЙТРЕКИНГА

Актуальность работы заключается в увеличивающейся доли маркетплейсов в качестве каналов продаж, а, следовательно, важности повышения уровня конкурентоспособности создаваемых разработчиками веб-ресурсов. Её можно повысить при уделении должного внимания разработке дизайна пользовательского интерфейса. Технологий трекинга взгляда все больше пользуются спросом на рынке, однако полностью готовые к использованию решения они имеют слишком большую стоимость, что делает актуальным разработку проектов с открытым исходным кодом.

Целью работы является разработка средств оценки для анализа содержания и удобства страниц маркетплейсов на основе айттрекинга. Для достижения поставленной цели необходимо решить следующие *задачи*:

1. Обзор и сравнительный анализ существующих инструментальных средств айттрекинга при оценке содержания и удобства GUI.
2. Проектирование и реализация средств айттрекинга при оценке содержания и удобства GUI.
3. Разработка метода ранжирования маркетплейсов с целью выявления наиболее оптимального решения из представленных на рынке конкурентоспособных программ.
4. Проведение предварительных экспериментов.

Общая *архитектура* приложения представлена на рисунке 1.

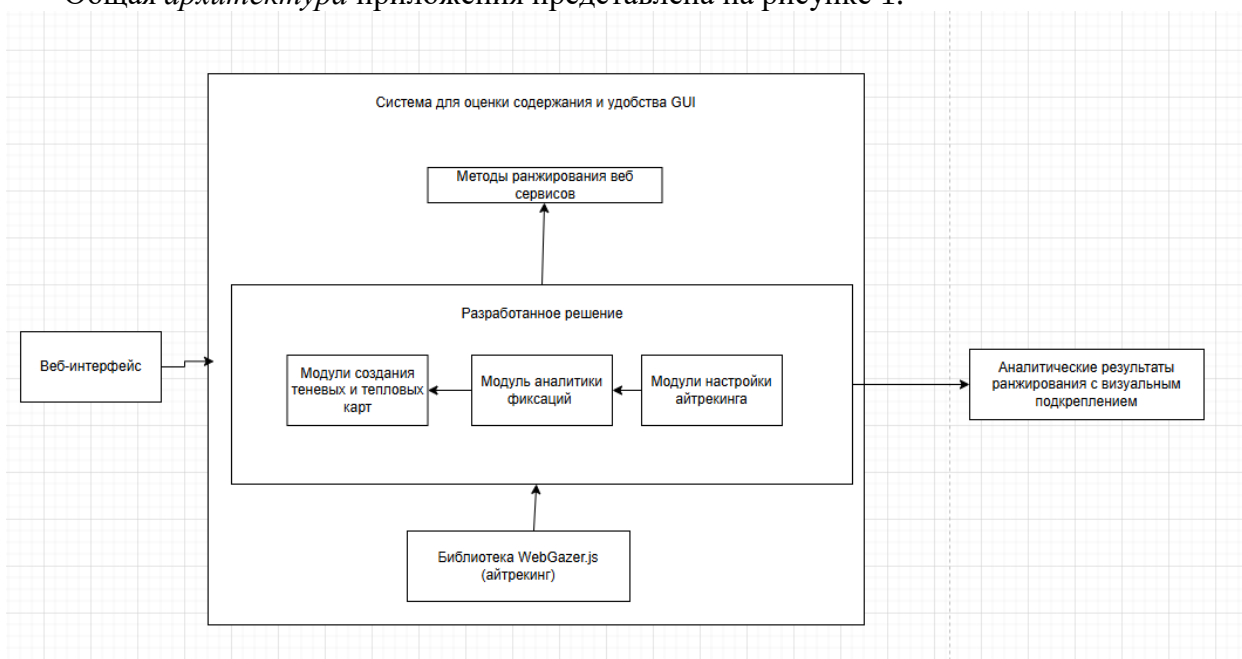


Рисунок 1 – Архитектура разработанного средства трекинга

Приложение для аналитики интерфейса реализовано с использованием web-технологий и алгоритмов компьютерного зрения [5], в частности, библиотеки WebGazer.js[1] с применением языков программирования JavaScript [2] и Python [3]. Данная библиотека позволяет отслеживать взгляд пользователя в реальном времени. В ходе работы получены количественные метрики взаимодействия пользователей с интерфейсами, а также теневые и тепловые карты взглядов для визуального отображения результатов, которые использованы для ранжирования веб-страниц маркетплейсов.

Предварительные результаты, демонстрирующие тепловые карты, приведено на рисунке 2.



Рисунок 2 – Демонстрация анализа тепловых карт

Разработанный инструмент показывает высокие *результаты* при аналитики пользовательских интерфейсов и предоставляет визуализированные данные, для упрощения дальнейшей работы с аналитикой интерфейсов. С помощью разработанных тепловых карт удалось провести анализ сайтов Ozon, Wildberries и Яндекс.Маркет. В частности, удалось подтвердить работу F-паттерна [4], а также выявить преимущества и недостатки в расположении блоков информации на странице.

ЛИТЕРАТУРА

1. Basurah M., Swastika W., Kelana O. H. Implementation of Face Recognition and Liveness Detection System Using TensorFlow.js // Jurnal Informatika Polinema. – 2023. – P. 509-516.
2. Python Documentation [Электронный ресурс] Режим доступа: <https://www.python.org/doc/>
3. JavaScript reference [Электронный ресурс] Режим доступа: <https://devdocs.io/javascript/>
4. UI паттерны и их польза в дизайне интерфейсов // TenChat [Электронный ресурс] Режим доступа: <https://tenchat.ru/media/1421234-ui-patterny-i-ikh-polza-v-dizayne-interfeysov>
5. Мурадов Ю.,Атаев Р., Беглиева А. Компьютерное зрение и искусственный интеллект // Научный электронный журнал "Матрица нсаучного познания". - 2025. - №12-2. - С. 60-63.

УДК 004.4

С.С. Казанцева, М.С. Силин (4 курс бакалавриата),
А.В. Самочадин, к. т. н., доцент

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ АРБИТРАЖА КРИПТОВАЛЮТ

Криптовалютный рынок характеризуется высокой волатильностью и разницей цен на одни и те же активы на различных торговых площадках. Арбитраж криптовалют представляет собой стратегию, основанную на использовании этих ценовых расхождений для получения прибыли [1]. Однако ручной поиск арбитражных возможностей требует значительных временных затрат и постоянного мониторинга множества бирж, что делает процесс неэффективным.

Существующие автоматизированные решения для арбитража криптовалют обладают рядом недостатков [2]. Многие из них ориентированы исключительно на централизованные биржи (CEX) и не учитывают особенности децентрализованных бирж (DEX). Другие сложны в настройке и использовании, что ограничивает их доступность для широкого круга пользователей. В связи с этим возникает необходимость разработки удобного и эффективного инструмента, учитывающего особенности как CEX, так и DEX, с целью автоматизации мониторинга данных и выявления арбитражных возможностей.

Разрабатываемое веб-приложение для арбитража криптовалют предназначено для автоматического сбора данных о котировках с различных бирж, анализа ценовых различий и оперативного уведомления пользователей о выявленных арбитражных возможностях.

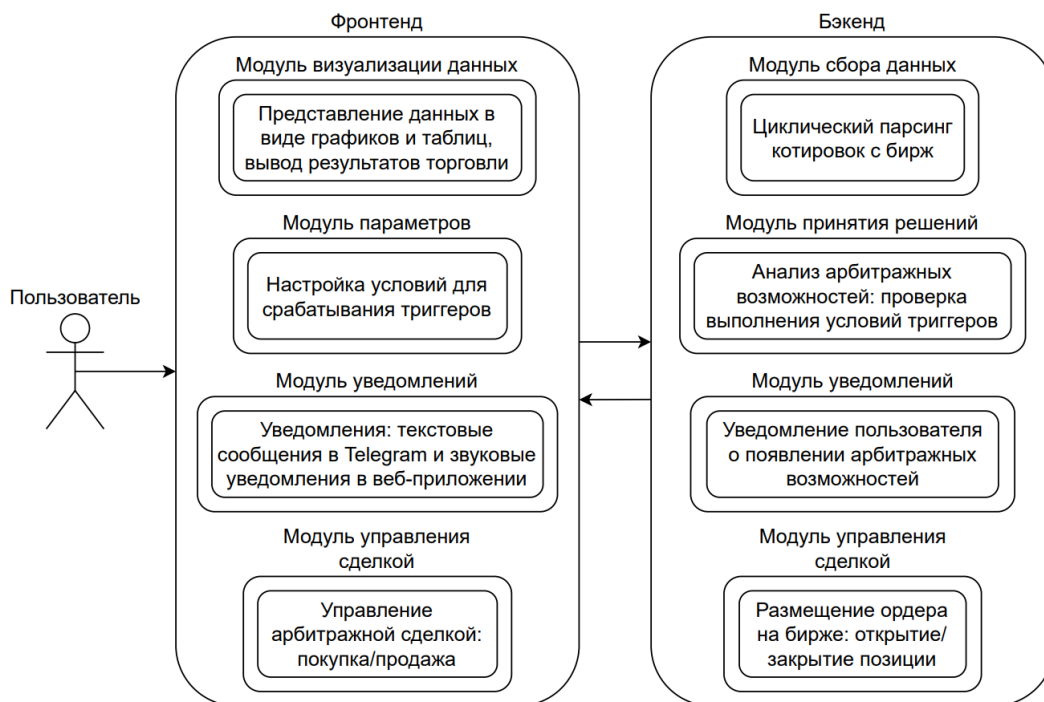


Рисунок 1 – Архитектура веб-приложения

На рисунке 1 представлена архитектура веб-приложения. Система выполняет циклический сбор данных о ценах криптовалют с бирж CEX и DEX с интервалом в одну секунду. Полученные данные передаются в модуль принятия решений, который проверяет выполнение условий триггеров для арбитражных сделок. Настроить условия для срабатывания триггеров, например, при возникновении разницы в 3% среди рассматриваемых цен, пользователь может в модуле параметров. При выполнении условий система отправляет уведомления о появлении арбитражной возможности: пользователь получает текстовое сообщение в Telegram и звуковое уведомление в веб-приложении. Далее пользователю предоставляется возможность совершения торговой сделки через соответствующий модуль управления сделкой. После исполнения ордеров система отображает прибыль/убыток, реализованные в рамках данной арбитражной сделки.

Бэкенд приложения реализован на ASP.NET Core [3], что обеспечивает высокую производительность по сравнению с большинством других фреймворков. Он выполняет функции сбора, обработки и анализа данных, а также генерации уведомлений. Данные поступают в модуль обработки, а после сохраняются в базе данных PostgreSQL. PostgreSQL гарантирует надежность хранения данных благодаря поддержке ACID-транзакций. Анализ арбитражных возможностей выполняет модуль принятия решений, а внешние уведомления отправляются через Telegram API [4]. Для асинхронного взаимодействия компонентов используется RabbitMQ, который эффективно управляет обменом сообщениями между сервисами. Для обработки фоновых задач используется Hangfire, что позволяет масштабировать нагрузку. Логирование реализовано через Serilog и SEQ, что упрощает мониторинг и отладку.

Фронтенд приложения реализован на Blazor WebAssembly [5], который позволяет создавать интерактивные пользовательские интерфейсы с помощью C# и обеспечивает повышение производительности разработки, обслуживания приложения, а также высокую скорость работы интерфейса. Он отвечает за визуализацию данных, отображение результатов анализа и торговли, а также за реализацию уведомлений. Взаимодействие с бэкендом

осуществляется через HTTP-запросы. UI-компоненты из Radzen повышают скорость разработки, также используются JavaScript-библиотеки: Chart.js предоставляет удобные инструменты для визуализации данных, jQuery используется для управления разметкой страниц. Взаимодействие пользователя с системой осуществляется через доступный интерфейс, обеспечивающий наглядное представление данных и возможность настройки предусмотренных параметров.

В результате работы было успешно разработано веб-приложение для арбитража криптовалют. Архитектура реализована, алгоритмы сбора и обработки данных работают стабильно. Система уведомлений оперативно информирует пользователей о возникновении новых арбитражных возможностей. В данный момент приложение функционирует круглосуточно. С целью обеспечения максимального удобства и эффективности в работе с платформой на основе отзывов пользователей проведена оптимизация аналитических алгоритмов и усовершенствован пользовательский интерфейс.

ЛИТЕРАТУРА

1. Binance. «Арбитраж криптовалют: что это, как работает, и как зарабатывать на разнице курсов криптовалют» [Электронный ресурс] Режим доступа: <https://www.binance.com/ru/blog/all/арбитраж-криптовалют-что-это-как-работает-и-как-зарабатывать-на-разнице-курсов-криптовалют-1969102862550780670#Стратегии-заработка>
2. Jon Nielsen. «8 Best Crypto Arbitrage Scanners in 2024». [Электронный ресурс] Режим доступа: <https://coincodex.com/article/27241/crypto-arbitrage-scanner/>
3. ASP.NET Core. [Электронный ресурс] Режим доступа: <https://dotnet.microsoft.com/ru-ru/apps/aspnet>
4. Telegram APIs. [Электронный ресурс] Режим доступа: <https://core.telegram.org/>
5. Blazor. [Электронный ресурс] Режим доступа: <https://dotnet.microsoft.com/ru-ru/apps/aspnet/web-apps/blazor>

УДК 004.4

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

УПРАВЛЕНИЕ РИСКАМИ НА ПРЕДПРИЯТИИ С ПОМОЩЬЮ МОБИЛЬНОЙ СИСТЕМЫ

В производственной сфере всегда присутствует риск возникновения нештатных ситуаций, и для обеспечения безопасности сотрудников необходимо иметь эффективную систему управления профессиональными рисками. Идея создания данной системы возникла после прочтения различных новостей о несчастных случаях и ЧП на предприятиях. Согласно подсчетам Международной Организации Труда ежегодно на различного рода предприятиях погибают около 2,3 миллионов мужчин и женщин в результате несчастных случаев или воздействия вредных факторов производственной среды [1-2]. Наиболее распространёнными причинами травмирования сотрудников являются [3]:

- технические причины (конструктивные недостатки и плохое техническое состояние оборудования);
- организационные причины (нарушение правил эксплуатации оборудования, недостатки содержания территории и рабочих мест);
- санитарно-гигиенические причины.

Целью работы является разработка мобильной системы для управления профессиональными рисками. На данный момент у сотрудников большинства предприятий уходит большое количество времени и сил на идентификацию и анализ рисков. А весь документооборот происходит на бумажных носителях, что ведет к тому, что на классифицирование выявленных рисков и разработку мероприятий для их устранения

требуются значительные трудозатраты от всех участников процесса управления профессиональными рисками. Для достижения поставленной цели необходимо решить следующие задачи:

- изучить предметную область;
- провести анализ существующих решений;
- составить требования к системе;
- спроектировать и разработать архитектуру системы;
- спроектировать и разработать мобильное приложение;
- протестировать работу всей системы.

В создаваемой системе пользователю будут доступны вкладки “Реестр рисков”, “Мероприятия”, “Аналитика” и личный кабинет пользователя. Все риски, выявленные и устраненные за последний месяц, а также их текущее состояние отображаются на вкладке “Реестр рисков”. На этой же вкладке предусмотрена следующая основная функциональность:

- фильтрация данных в реестре рисков;
- внесение в реестр выявленных рисков;
- просмотр и редактирование подробной карточки риска;
- добавление корректирующих мероприятий в подробной карточке риска;
- добавление фото и видеоматериалов в карточку.

Вкладка “Мероприятия” предназначена для хранения проведенных мероприятий по воздействию на профессиональный риск. Вкладка “Аналитика” предназначена для эффективного анализа предпринимаемых мероприятий направленных на устранение рисков. На этой же странице можно посмотреть графическую информацию о рисках, получить статистические сведения о выявленных и устраненных рисках, а также узнать, какова степень влияния и среднее время необходимое на устранение угроз.

Разрабатываемая система позволит оптимизировать документооборот, а также приведет к снижению рисков возникновения нештатных ситуаций, повышению оперативности реагирования и принятию корректирующих мер по устранению профессиональных рисков.

В настоящее время доступная информация о подобных приложениях и системах ограничена. Большинство программных продуктов сфокусированы на управлении финансовыми, проектными и операционными рисками, но лишь часть из них оценивает профессиональные риски на предприятиях различных отраслей. Например, существует решение от компании «1С», под названием «1С:Управление холдингом 8. Управление рисками и мероприятиями», которое позволяет работать с профессиональными рисками, но данная подсистема может не подойти многим малым и средним предприятиям ввиду ее сложности и дороговизны.

Архитектура мобильного приложения построена по многослойной модели (presentation, domain, data), что обеспечивает гибкость и удобство масштабирования. Клиентская часть разрабатывается на Kotlin (Android) [4-5]. Взаимодействие с серверной частью построено на REST API с использованием PostgreSQL [6] в качестве СУБД.

Разрабатываемая система управления профессиональными рисками будет представлять собой решение, которое отлично подойдет для небольших компаний. Основными преимуществами будут простота и удобство использования, а также наличие всего необходимого функционала для оценки и контроля профессиональных рисков.

ЛИТЕРАТУРА

1. Мировая статистика. [Электронный ресурс] Режим доступа: <https://www.ilo.org/moscow/areas-of-work/occupational-safety-and-health/>
2. Оценка и управления профессиональными рисками [Электронный ресурс]. Режим доступа: <https://eisot.rosmintrud.ru/otsenka-i-upravlenie-professionalnymi-riskami>
3. Виды и причины несчастных случаев на производстве [Электронный ресурс]. Режим доступа: <https://www.protrud.com/>
4. Мартин, Р. Чистая архитектура: Искусство разработки программного обеспечения / Р. Мартин. – Москва: Питер, 2018. – 352 с.

5. Kotlin Language Documentation. [Электронный ресурс]. Режим доступа: <https://kotlinlang.org/docs/home.html>
6. PostgreSQL Documentation [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/docs/>

УДК 004.4

Е. О. Карасева (4 курс бакалавриата),
Ю.В. Литвинов, к.т.н., ст. преподаватель

РЕАЛИЗАЦИЯ МЕССЕНДЖЕРА В ВЕБ-СЕРВИСЕ ДЛЯ ПОМОЩИ БЕЗДОМНЫМ ЖИВОТНЫМ

Проблема бездомных животных как никогда актуальна. Оперативная помощь животным требует координации действий волонтеров, ветеринаров и других специалистов. Веб-сервис, разрабатываемый компанией YADRO, призван объединить информацию по вопросам пропажи животных, их отлова, лечения и последующего пристройства. Одним из ключевых элементов взаимодействия между пользователями должен стать мессенджер, позволяющий оперативно передавать информацию в личных и групповых чатах. Браузерных чатов с открытым исходным кодом и таким же стеком технологий, который можно было бы внедрить, не существует.

Целью работы является разработка и внедрение функциональности чатов с мгновенной синхронизацией сообщений. Для достижения цели ставились задачи:

1. Провести обзор технологий для моментальной отправки сообщений и выбрать подходящие для проекта библиотеки.
2. Добавить поддержку мессенджера на стороне базы данных.
3. Спроектировать и реализовать программный интерфейс для мессенджера, а также поддержать чаты на фронтенде.
4. Протестировать функциональность, провести опробацию.

Перед началом разработки были выявлены требования к мессенджеру – моментальная отправка сообщений, возможность общения как в личных, так и в групповых чатах.

Были проанализированы методы синхронизации сообщений в реальном времени [1]. Для создания чата было решено использовать WebSockets, так как они подходят для сценариев, где предполагаются частые обновления, низкая задержка и поддержка одновременной отправки данных, в отличие от SSE, более эффективны по сравнению с Long Polling и не требуют настройки отдельного сервиса, как WebRTC.

Для реализации клиентской части веб-сервиса был использован язык программирования TypeScript и фреймворк Angular, для серверной части – язык программирования Golang и фреймворк Fiber. В качестве системы управления базами данных использовалась PostgreSQL. Соответственно, библиотеки для работы с WebSockets выбирались с учетом данных ограничений.

Для серверной части была выбрана библиотека fiber/websocket, так как совместимый пакет снижает сложность проекта. На клиентской части использовался нативный WebSocket API, чтобы не переусложнять код, так как вся логика работы с веб-сокетами происходит в серверной части.

Для взаимодействий с чатами были в базу данных были добавлены таблицы chats (информация о чате), chats_members (информация об участниках чата), messages (информация о сообщениях), message_read (во сколько и кем прочитано определенное сообщение). Также в рамках работы велось взаимодействие с таблицами users и posts, которые уже находились в базе данных. Актуальная документация базы данных представлена по ссылке [2]. Для актуализации базы были созданы миграции [3, 4].

В проект были добавлены сервисы чатов, сообщений и участников чата [5]. В основу методов API легли операции CRUD, но вместо удаления данных из базы было решено помечать их deleted. Так как чаты доступны только авторизованным пользователям, в каждом методе проверяется наличие токена авторизации в запросе и ведется работа с id текущего пользователя, получаемого из токена. Методы возвращают ответ успеха (200 OK или 201 Created), если запланированное действие выполнено, либо одну из следующих ошибок в случае сбоев: 400 Bad Request, 401 Unauthorized, 500 Internal Server Error.

Реализация мгновенных сообщения была сделана в виде типа WebSocketManager, который хранит идентификаторы чатов и управляет активными соединениями для каждого чата. При создании сервера на отдельные маршруты вида /ws/{chat_id} добавляется middleware для инициализации и регистрации веб-сокета соединения в конкретном чате, а также эндпоинт, предназначенный для обработки веб-сокета сообщений.

Когда пользователь заходит в чат, подгружаются сообщения из базы и открывается веб-сокета соединение. Мы отправляем, получаем и сразу же отрисовываем на экране сообщения до тех пор, пока соединение не закроется. Это произойдет, если пользователь выйдет из чата или закроет страницу. При следующем заходе в чат старые сообщения отобразятся уже из базы, а веб-сокета соединение откроется заново. Если пользователь выйдет из чата, о новых сообщениях он узнает, когда они подгрузятся в базу и отобразятся в виде неп прочитанных на панели чатов слева.

Полностью статические страницы на фронтенде, отвечающие за клиентскую часть, были модифицированы в соответствии с логикой серверной части приложения. Были добавлены websocket-service для подключения к чату и инициации на клиенте веб-сокета соединения, и chat-service для отправки запросов к серверу на разные url в зависимости от задачи – открыть чат, отправить сообщение и так далее.

Для тестирования бекенда приложения были использованы пакеты для тестов в Go: testify и mockery. Методика System Usability Scale (SUS) показала высокий уровень удобства использования (91,3 балла при среднем значении 68).

Пулл-реквест [6] с реализацией мессенджера предложен в репозиторий проекта, на момент 07.10.2024 тесты и линтер в CI проходят.

В результате работы была разработана основная функциональность мессенджера в веб-сервисе для помощи бездомным животным. В рамках работы были решены следующие задачи:

1. Проведен обзор технологий мгновенного обмена сообщениями, выбраны подходящие библиотеки.
2. Добавлена поддержка мессенджера на стороне базы данных.
3. Спроектирован и реализован программный интерфейс для мессенджера, добавлена поддержка чатов на фронтенде, обновлен дизайн страниц.
4. Добавленная функциональность протестирована, проведена опробация.

ЛИТЕРАТУРА

1. WebSockets, SSE, Long-Polling, WebRTC и WebTransport: сравнение технологий. [Электронный ресурс] Режим доступа: <https://rxdb.info/articles/websockets-sse-polling-webrtc-webtransport.html>
2. Документация базы данных проекта SOS-Kotopes. [Электронный ресурс] Режим доступа: <https://dbdocs.io/kleo-53/SOS-kotopes>
3. Изменения в структуре базы данных. [Электронный ресурс] Режим доступа: <https://github.com/kotopesp/sos-kotopes/pull/30>
4. Правки в структуре базы данных. [Электронный ресурс] Режим доступа: <https://github.com/kotopesp/sos-kotopes/pull/47>
5. API мессенджера проекта SOS-Kotopes. [Электронный ресурс] Режим доступа: <https://mocoibelcr.apidog.io/>
6. Основная реализация мессенджера. [Электронный ресурс] Режим доступа: <https://github.com/kotopesp/sos-kotopes/pull/8>

РАЗРАБОТКА ВЕБ-ИНТЕРФЕЙСА И TELEGRAM-БОТА ДЛЯ УПРАВЛЕНИЯ
УМНОЙ ТЕПЛИЦЕЙ НА ОСНОВЕ IOT

Автоматизация управления теплицами с применением технологий Интернета вещей (IoT) позволяет повысить эффективность выращивания сельскохозяйственных культур, минимизировать ручной труд и сократить затраты ресурсов.

Целью работы является разработка веб-интерфейса с использованием SvelteKit[1] и Telegram-бота с использованием библиотеки Telegram Bots API[2] разработанные для удаленного мониторинга и управления параметрами теплицы.

Для достижения цели поставлены выполнены *задачи*: Веб-интерфейс был реализован с использованием фреймворка SvelteKit[1] и языка TypeScript[3]. Архитектура одностраничного приложения (SPA)[4] позволила обеспечить динамическое обновление данных без перезагрузки страницы, что критически важно для мониторинга состояния теплицы в реальном времени. Бэкенд системы разработан на Java, обеспечивая взаимодействие с датчиками температуры, влажности и исполнительными устройствами. Взаимодействие между клиентской и серверной частью реализовано через REST API[5]. Дополнительно разработан Telegram-бот на Java с использованием библиотеки Telegram Bots API, который предоставляет пользователям возможность удаленного управления системой и получения уведомлений о текущем состоянии теплицы.

Архитектура системы включает:

- Веб-интерфейс: SPA приложение на TypeScript и фреймворке SvelteKit .
- Telegram-бот: Java приложение с использованием библиотеки Telegram Bots API.

В ходе работы был создан и протестирован веб-интерфейс, обеспечивающий:

- Подключение и настройку новых датчиков и исполнительных устройств.
- Мониторинг данных с датчиков в реальном времени.
- Настройку профилей автоматического полива и управление ими.
- Отображение статистики работы системы.
- Интеграцию с Telegram-ботом для оперативного управления.

Разработанный интерфейс был успешно развернут на Raspberry Pi 4, а тестирование показало корректность работы всех функций и стабильность работы системы при длительном использовании.

Выводы: Веб-интерфейс и Telegram-бот обеспечивают удобный доступ к управлению параметрами умной теплицы, позволяют гибко настраивать режимы работы и получать актуальную информацию о состоянии системы в режиме реального времени. Разработанное решение продемонстрировало эффективность применения IoT-технологий в автоматизации теплиц. В дальнейшем планируется интеграция алгоритмов машинного обучения для предиктивного анализа параметров микроклимата и оптимизации управления системой.

ЛИТЕРАТУРА

1. SvelteKit Documentation. 2023. URL: <https://kit.svelte.dev/docs>
2. Telegram Bots API Documentation. 2023. URL: <https://core.telegram.org/bots/api>
3. Svelte with TypeScript Documentation. 2023. URL: <https://svelte.dev/docs/typescript>
4. Single Page Application (SPA) Overview. 2023. URL: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>
5. REST API Overview. 2023. URL: <https://developer.mozilla.org/en-US/docs/Glossary/REST>

РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ ЭФФЕКТА КОГЕРЕНТНОГО ПЛЕНЕНИЯ НАСЕЛЕННОСТЕЙ В ЯЧЕЙКАХ КОНЕЧНОГО РАЗМЕРА В ОПТИЧЕСКИ ПЛОТНОЙ СРЕДЕ

Эффект когерентного пленения населенностей (КПН) наблюдается в многоуровневых системах и представляет собой результат формирования суперпозиционных состояний, которые не взаимодействуют с оптическими полями. Этот эффект становится особенно важным при определенных условиях частотных расстройств и интенсивности световых волн, что приводит к состоянию КПН, при котором система практически не взаимодействует с внешним полем. Такое поведение характерно для систем, допускающих интерференцию нескольких каналов возбуждения [1,2]. КПН открывает новые возможности для поддержания квантовых состояний, что способствует увеличению времени их жизни и улучшению стабильности различных энергетических переходов. Благодаря узкой полосе резонанса этот эффект активно используется в разработке атомных стандартов частоты [3,4] и др.

Целью данной работы является разработка программного модуля для численного моделирования эффекта КПН в ячейках конечного размера в оптически плотной среде и создания на его основе клиент-серверного приложения. Математическая модель для расчёта комплексных частот Раби $V_1(z)$ и $V_2(z)$ описывается следующей системой дифференциальных уравнений:

$$\begin{cases} \frac{dV_1}{dz} = -\frac{2\pi i n_a k_1 |d_{31}|^2}{\hbar} \langle \rho_{13} \rangle, & V_1|_{z=0} = V_{10} \\ \frac{dV_2}{dz} = -\frac{2\pi i n_a k_2 |d_{32}|^2}{\hbar} \langle \rho_{23} \rangle, & V_2|_{z=0} = V_{20} \end{cases}, \quad z \in [0, a] \quad (1)$$

где $\langle \rho_{13} \rangle$ и $\langle \rho_{23} \rangle$ — это оптические когерентности ρ_{13} и ρ_{23} , усредненные по всем скоростным группам атомов, имеющие вид:

$$\langle \rho_{13} \rangle = \frac{2}{V_{T_r} \sqrt{\pi}} \int_0^\infty e^{-\left(\frac{V_z}{V_{T_r}}\right)^2} f(V_z, V_1, V_2, \rho_1^{+*}, \rho_3^{+*}, \rho_1^{-*}, \rho_3^{-*}) dV_z, \quad (2)$$

$$\langle \rho_{23} \rangle = \frac{2}{V_{T_r} \sqrt{\pi}} \int_0^\infty e^{-\left(\frac{V_z}{V_{T_r}}\right)^2} g(V_z, V_1, V_2, \rho_1^{+*}, \rho_2^{+*}, \rho_1^{-*}, \rho_2^{-*}) dV_z, \quad (3)$$

в которых $\rho_k^+, \rho_k^-, k=1,2,3$ находятся из решения систем линейных алгебраических уравнений:

$$A^+ \rho^+ = -b^+, \quad A^- \rho^- = -b^-, \quad (4)$$

в которых матрицы и правые части зависят от V_z, V_1, V_2 , а f и g заданные функции. В модель входят параметры: a - размер ячейки; n_a - концентрация атомов газа; k_1 и k_2 - волновые числа соответствующих частотных составляющих электромагнитных полей; d_{31} и d_{32} - матричные элементы оператора дипольного элемента; $\hbar$ - постоянная Планка; V_z - скорость атомов вдоль оси z ; V_{T_r} - относительная скорость атомов; $\rho_k^+, \rho_k^-, k=1,2,3$ связаны с населённостями

основного и возбуждённого состояния, соответствующими двум скоростным группам $+kV_z$ и $-kV_z$; z - координата, направленная по оси, вдоль которой распространяется лазерное излучение; $i = \sqrt{-1}$; $*$ - символ комплексного сопряжения.

Программа разработана на языке Fortran в среде Code::Blocks под ОС Linux. Система дифференциальных уравнений, дополненная двумя сопряжёнными уравнениями, интегрируется неявным методом с автоматическим выбором шага и контролем точности. Матрица Якоби вычисляется численно. На каждом шаге интегрирования вычисление интегралов осуществляется по формулам Ньютона-Котеса 8-го порядка точности с автоматическим выбором шага с заданным допуском на точность. Так как подинтегральная функция зависит от ρ_k^+ , ρ_k^- , то каждое её вычисление приводит к решению алгебраических систем (4), которые решаются методом Гаусса.

Программа работает в двух режимах. Первый режим обеспечивает расчёт частот Раби $V_1(z)$ и $V_2(z)$ и интенсивности излучения $\frac{|V_1(z)|^2}{|V_{10}|^2}$ и $\frac{|V_2(z)|^2}{|V_{20}|^2}$, а также ρ_k^+ , ρ_k^- , $k=1,2,3$ для заданного набора скоростей V_z . Во втором режиме рассчитываются формы спектра КПН от координаты z , что требует примерно сотни расчётов в первом режиме. Вычислительный модуль на языке Fortran для численного моделирования эффекта КПН запускается сервисом на Go, общающимся с пользовательским веб-интерфейсом на JavaScript через HTTP. Такая реализация позволяет удобно задавать параметры задачи и визуализировать результаты. Go-сервис отделяет вычисления от интерфейса, повышая гибкость и масштабируемость. Он может быть развернут отдельно, освобождая ресурсы клиента и ускоряя обработку данных. Веб-интерфейс обеспечивает доступ к результатам с любых устройств. В будущем система может поддерживать параллельные вычисления с несколькими Go-сервисами, что ускорит моделирование и позволит исследовать более сложные сценарии.

ЛИТЕРАТУРА

1. G. Alzetta et al. // Nuovo Cim. В **36** (1), 5 (1976).
2. Б.Д. Агапьев, М.Б. Горный, Б.Г. Матисов и др. // УФН, **163**, 1 (1993).
3. J. Vanier // Appl. Phys. В **81**, 421 (2005).
4. С.А. Зибров, В.Л. Величанский, А.С. Зибров и др. // Письма в ЖЭТФ, **82**, 534 (2005).

УДК 004.93'12

С.А. Клюкин (4 курс бакалавриата),
О.В. Александрова, ст. преподаватель

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ ТРАНСФОРМАТОРОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНИЧЕСКОГО ЗРЕНИЯ

В работе отдела технического контроля на производстве была выявлена операция, которая может принести наибольшую выгоду при автоматизации [1] – это проверка электрической прочности.

– При проверке электрической прочности изделий в большинстве случаев сотруднику нужно ждать 60 секунд, пока проходит проверка одного изделия.

– Проверка электрической прочности изделий и пробоя требует допуск (не каждый сотрудник может проводить такие испытания).

– Высокие требования к технике безопасности.

– Работа повторяющаяся. Работником совершаются одни и те же действия.

Целью данной работы является автоматизация операции отдела технического контроля за счёт создания подхода автоматического перемещения изделий [2] и запуска электрических испытаний.

Для достижения цели необходимо решить следующие задачи:

1. Распознавание изделий для захвата
2. Интеграция с промышленным манипулятором
3. Интеграция с коммутатором контактов изделия
4. Интеграция с пробойной установкой
5. Интеграция с системой учета испытаний

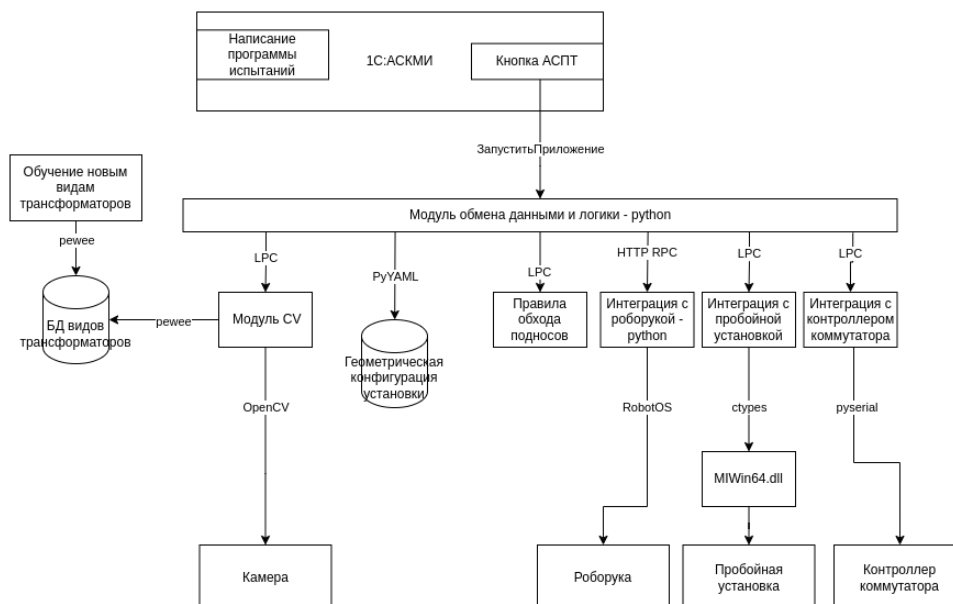


Рисунок 1 – Архитектура программной системы

Архитектура системы тестирования трансформаторов представлена на рисунке 1. На вход подается набор параметров испытания. Система обращается к телевизионной камере и получает позицию следующего изделия [3]. Система обращается к промышленному манипулятору для перемещения между позициями, к коммутатору для замыкания контактов испытания, к пробойной установке для проведения испытания посредством существующей библиотеки для работы с измерительными приборами. Модуль обучения новым видам трансформаторов - это программа с пользовательским интерфейсом для составления обучающей выборки для алгоритмов компьютерного зрения.

Решение основывается на языке python и различных его библиотеках: tkinter, pewee, opencv, pyyaml, ctypes, pyserial [4].

Основные преимущества решения заключаются в том, что в отличие от существующих систем автоматизации в результате испытания не содержится относительно большая погрешность, и в процессе испытания не отнимается время на ручное соединение контактов [5].

ЛИТЕРАТУРА

1. Ильин Д. В. Алгоритмы машинного зрения для автоматической классификации кристаллов / Ильин Д. В. [Электронный ресурс] // ЭБ СПбПУ : [сайт]. — URL: <https://elib.spbstu.ru/dl/3/2020/vr/vr20-4000.pdf/view> (дата обращения: 20.02.2025).
2. Молодяков С.А. Низкоуровневая обработка изображений в компьютерном зрении. СПб.:Изд-во Политехн. ун-та, 2021.-197с.
3. Низамутдинова Д. И. Компьютерное зрение в промышленной дефектоскопии / Низамутдинова Д. И. [Электронный ресурс] // ЭБ СПбПУ : [сайт]. — URL: <https://elib.spbstu.ru/dl/3/2024/vr/vr24-5283.pdf/view> (дата обращения: 20.02.2025).

4. Передовые производственные технологии: компьютерные (супер-компьютерные) технологии и организация наукоемких производств : сборник тезисов докладов Национальной научно-практической конференции с международным участием, 26 мая 2023 г. / под ред. д-ра техн. наук, проф-ссора Ю. Я. Болдырева, д-ра экон. наук, профессора П. А. Аркина. – СПб. : ПОЛИТЕХ-ПРЕСС, 2023. – 218 с.
5. Бессонов А. Н. Программа для распознавания дорожных знаков на основе технологии компьютерного зрения / Бессонов А. Н. [Электронный ресурс] // ЭБ СПбПУ : [сайт]. — URL: <https://elibr.spbstu.ru/dl/3/2024/vr/vr24-766.pdf/view> (дата обращения: 20.02.2025).

УДК 004.4

В.В. Кислов (2 курс магистратуры),
Н.М. Лыченко, д.т.н., профессор

СИСТЕМА АВТОМАТИЧЕСКОГО МОНИТОРИНГА И АНАЛИЗА ТОРГОВЫХ ПРЕДЛОЖЕНИЙ НА МАРКЕТПЛЕЙСАХ

Современные маркетплейсы (Ozon, Aliexpress, Wildberries и др.) предлагают миллионы товаров с постоянно меняющимися ценами. Пользователю зачастую нелегко находить и отслеживать наиболее выгодные предложения, своевременно реагировать на колебания стоимости и оценивать качество товаров на основании большого количества отзывов. Для решения этих задач необходим инструмент, способный автоматически мониторить цены, уведомлять о снижениях, а также учитывать мнения других покупателей и сопоставлять товары по ключевым критериям (цена, рейтинг, количество покупок).

Разработанная система выполняет следующие функции:

- сбор данных с маркетплейсов в режиме реального времени или по заданному расписанию;
- мониторинг и рассылка уведомлений: при снижении цены пользователю отправляется уведомление (например, в мессенджер Telegram), что существенно упрощает поиск выгодных предложений;
- анализ отзывов: система суммаризирует отзывы с помощью OpenAI API [1], выделяя ключевые преимущества и недостатки, что даёт возможность быстро ознакомиться с опытом других покупателей и исключить из рассмотрения неинформативные комментарии;
- присвоение рейтингов: каждому товару автоматически начисляются баллы на основе показателей (количество покупок, оценка в отзывах, цена), итоговый рейтинг помогает пользователю выбрать оптимальный вариант среди нескольких похожих товаров.

Техническая реализация основана на платформе .NET и языке C# [2]. Для уведомлений в Telegram используется специальный бот, взаимодействующий с сервером по протоколу HTTP (периодически или по событию) [3]. Система может быть запущена как в контейнере (Docker) [4], так и на виртуальном сервере. Архитектура системы представлена на рисунке 1. Далее приведена оценка характеристик демонстрационной версии:

- удобство развертывания: система не требует значительных аппаратных ресурсов, легко интегрируется и масштабируется;
- скорость реакции: при оптимальном интервале опроса API задержка между фактическим изменением цены и уведомлением в Telegram не превышает нескольких секунд;
- качество анализа отзывов: использование OpenAI API даёт возможность быстро выявлять сильные и слабые стороны товаров, а также предоставлять их в сжатом виде, что значительно упрощает принятие решения;
- точность итоговой оценки: расчёт взвешенного балла с учётом рейтинга, количества покупок и динамики цен демонстрирует высокую степень совпадения с реальными предпочтениями пользователей.

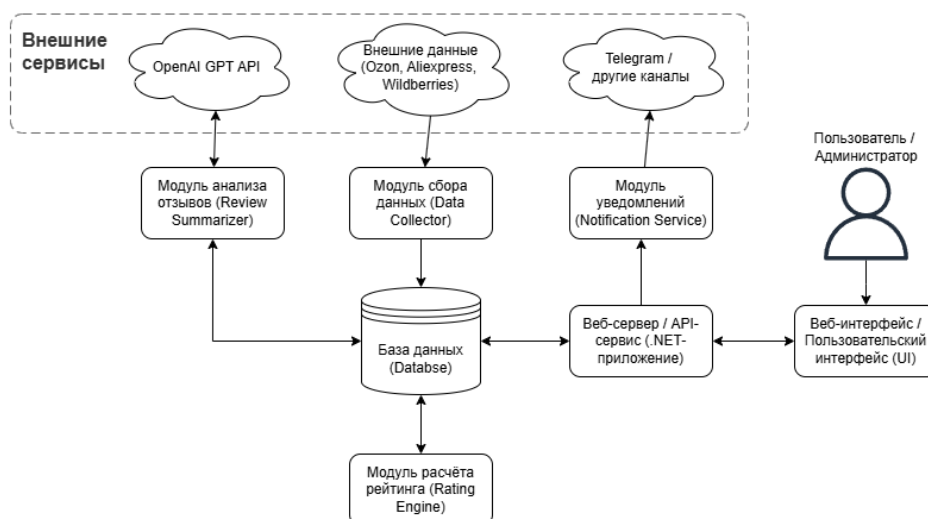


Рисунок 1 – Архитектура системы

Для удобства использования в системе предусмотрен веб-интерфейс, позволяющий пользователям настраивать целевые товары, задавать критерии сравнения и периодичность проверок. При необходимости можно подключать дополнительные источники информации, например, исторические данные о ценах, чтобы определять тренды или прогнозировать дальнейшие изменения стоимости. Реализованы базовые механизмы ограничения доступа и разграничения прав, позволяющие использовать продукт как в личных целях (мониторинг нескольких товаров), так и в корпоративной среде (отслеживание конкурентных позиций и формирование аналитических отчётов).

В ходе тестирования подтвердились корректность отправки уведомлений и точность формирования рейтингов. В дальнейшем планируется расширение списка маркетплейсов, интеграция дополнительных платформ для уведомления и реализация прогноза изменения цен (например, с помощью статистических или машинных методов), а также внедрение инструментов визуализации динамики цен и рейтингов в реальном времени.

ЛИТЕРАТУРА

1. OpenAI GPT API документация. [Электронный ресурс]. Режим доступа: <https://platform.openai.com/docs/overview>
2. ASP.NET Web APIs | Rest APIs with .NET and C#. [Электронный ресурс]. Режим доступа: <https://dotnet.microsoft.com/en-us/apps/aspnet/apis>
3. Telegram Bot API. [Электронный ресурс]. Режим доступа: <https://core.telegram.org/bots/api>
4. Docker Documentation. [Электронный ресурс]. Режим доступа: <https://docs.docker.com>

УДК 004.891.2

Ч.Э. Козырев (2 курс магистратуры),
А.Д. Ковалев, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ ОЦЕНКИ ПЛАТЕЖЕСПОСОБНОСТИ БАНКОВСКИХ КЛИЕНТОВ

Современная банковская деятельность неразрывно связана с управлением финансовыми рисками, среди которых ключевое место занимает кредитный риск, то есть вероятность невозврата заемных средств клиентом. В условиях высокой конкуренции и ужесточения регуляторных требований банки стремятся оптимизировать процессы принятия решений о выдаче кредитов, минимизируя потери и обеспечивая собственную финансовую

устойчивость. Одним из инструментов решения этой задачи является скоринг, который является технологией, преобразующей персональные и финансовые данные клиента в числовую оценку, отражающую вероятность своевременного погашения долга.

Разработка программного обеспечения для определения платежеспособности клиента с использованием скоринг системы позволяет компаниям автоматизировать и ускорить процесс принятия решений, улучшить качество кредитного анализа, повысить точность прогнозирования платежеспособности клиентов, снизить риски для бизнеса.

Основной целью данной работы является разработка комплексной скоринг системы, основанной на применении современных методов машинного обучения, для точного и объективного определения платежеспособности банковских клиентов.

На данный момент для поддержки бизнес-процессов кредитования применяются различные типы программ. Согласно данным аналитического агентства TAdviser, наиболее популярными информационными системами являются QlicView, Deductor, SAP BW, Форсайт и др. Перечисленные аналитические продукты реализуют функции оценки кредитоспособности заемщиков, в том числе осуществляют расчет рейтинга клиентов по различным критериям. Но эти инструменты обладают рядом недостатков. Например, перед их использованием необходимо пройти дополнительное обучение, чтобы иметь возможность пользоваться всеми функциями. Также, не во всех из них есть удобный интерфейс и возможность дообучения моделей.

Многие ученые и специалисты затронули в своих трудах проблемы оценивания кредитоспособности заемщика, среди них работы Н.П. Белотеловой [1], О.И. Лаврушина [2] и др. Целый ряд исследований был посвящен разработке методики применения нейронных сетей для оценки кредитоспособности заемщика, например, работы Е.А. Ангелини [3], С. Гоуль [4], М.А. Власенко [5] и др.

Активная популяризация цифровых технологий привела к необходимости замены классического скоринга более продвинутыми методами машинного обучения, основанными на искусственном интеллекте. Методы К-ближайших соседей (KNN), случайный лес (Random Forest, RF), XGBoost (Extreme Gradient Boosting), логистическая регрессия (Logistic Regression, LR) и метод опорных векторов (Support Vector Machine, SVM) были выбраны в качестве базовых для изучения, поскольку чаще всего они используются в качестве эталонных моделей в области кредитного скоринга. Их сравнение представлено в таблице 1.

Таблица 1 – Сравнение методов

Dataset	Model	Accuracy %	Precision (0:1)	Recall (0:1)	F1_Score	G-Mean	ROU-AUC	Kappa
German	KNN	72.8	0.85:0.19	0.82:0.23	0.84:0.21	0.431	0.524	0.05
	RF	79.6	0.85:0.25	0.92:0.14	0.88:0.18	0.364	0.532	0.08
	XGBoost	82.5	0.85:0.30	0.97:0.08	0.74:0.35	0.270	0.521	0.06
	LR	62.8	0.90:0.24	0.63:0.63	0.74:0.35	0.629	0.629	0.16
	SVM	61.5	0.89:0.23	0.63:0.60	0.74:0.34	0.615	0.615	0.14

Основу предлагаемой скоринговой системы составляет набор данных о заемщиках, на котором обучается алгоритм оценивания новых клиентов. В рамках работы был произведен анализ двух крупнейших датасетов на сегодняшний день. Результаты их сравнения представлены в таблице 2.

Таблица 2 – Сравнение датасетов

Название дата сета	GermanCreditData	CreditScore
Количество уникальных записей	1000	1001
Количество атрибутов	20	87
Основная направленность данных	Финансовое положение клиента	Доходы и расходы клиента

Предлагаемый в работе подход к разработке скоринг системы представлен на схеме на рисунке 1.



Рисунок 1 – Схема предлагаемой скоринг системы

В качестве датасета используется исходный набор данных German Credit Data. Эти данные проходят предварительную подготовку и поступают в модуль обучения алгоритма. В результате обучения получается модель, которая в дальнейшем используется в обработчике заявок.

Пользовательский интерфейс предлагаемой скоринг системы содержит анкету с полями для ввода данных. Введенные пользовательские данные преобразуются в числовые значения, и затем эти значения передаются в обработчик заявок с обученной моделью. После обработки данных результат работы системы выводится в пользовательском интерфейсе. Результат представляет собой численное значение вероятности того, что клиент вовремя вернет заемные средства. Кроме того, отличительной особенностью разработанного программного инструмента является предоставление рекомендации о том, на какую сумму можно одобрить кредит конкретному клиенту.

ЛИТЕРАТУРА

1. Белотелова Н. П. Проблемы банковской системы РФ в современных условиях // Актуал. проблемы соц.-экон. развития России. 2020. № 1. С. 103–106.
2. Роль кредита и модернизация деятельности банков в сфере кредитования: учеб. пособие / под ред. О. И. Лаврушина. М.: Кнорус, 2013. 267 с.
3. Angelini E., di Tollo G., Roli A. A Neural Network Approach for Credit Risk Evaluation // Kluwer Academic Publishers. 2006.
4. Goyal S. Credit Risk Prediction Using Artificial Neural Network Algorithm. Data Science Central, 2018.
5. Власенко М. Оценка кредитоспособности физических лиц при помощи инструментария классификационных нейронных сетей // Банк. вестн. 2021. № 3 (692). С. 19–30

УДК 004.453

Е.А. Козлова (3 курс бакалавриата),
И.В. Зайцев, ст. преподаватель

АВТОМАТИЗАЦИЯ СОСТАВЛЕНИЯ РАСПИСАНИЯ УЧЕБНОЙ СЕССИИ

Составление расписания экзаменационной сессии — это трудоёмкая задача, требующая учета множества факторов: нормативных требований по времени подготовки к различным видам аттестации, доступности аудиторий, занятости преподавателей. Поиск удобных вариантов в процессе первоначального формирования расписания и его последующих вынужденных корректировок в случае форс-мажорных обстоятельств нередко требуют длительного подбора приемлемого варианта, поэтому использование автоматизации для решения таких задач может существенно сэкономить время сотрудников.

Дополнительная сложность задачи автоматизации составления расписания заключается в необходимости учета множества жестких и мягких ограничений, которые можно классифицировать следующим образом.

- Временные ограничения: сложность заключается в многочисленности требований к расписанию, включающих в себя как установленные правила, ограничивающие допустимую нагрузку на студентов и преподавателей, так и разнообразные по своему виду пожелания, касающиеся дат, времени и совмещения аттестаций для некоторых групп.

- Пространственные ограничения: доступность аудиторий с нужными характеристиками и их расположение в разных корпусах.

Для решения задачи автоматизации составления расписания предлагается использовать комбинацию алгоритмических методов и структур данных, позволяющих на каждом шаге контролировать соответствие генерируемого расписания всем правилам и пожеланиям. Анализ проблемы показал, что требования к расписанию значительно меняются и крайне трудно все их предвидеть, формализовать и учесть в программном продукте. Поэтому разрабатываемый проект является лишь инструментом для повышения возможностей составителей расписания, цель которого — сэкономить время и повысить качество результата их работы.

Исходные данные для составления расписания задаются в файле формата Excel, который включает следующие листы:

- Дни сессии: даты проведения сессии и выходные дни.
- Потоки: информация о группах студентов.
- Аудитории: список аудиторий с их характеристиками (вместимость, тип и т. д.).
- Информация о зачётах, курсовых работах, зачётах с оценками, курсовых проектах и экзаменах, которые запланированы на ближайшую сессию.
- Итог: лист для записи итогового расписания.

Программа взаимодействует с файлом Excel с использованием специализированных библиотек (например, `libxl` или `xlnt`), что позволяет:

- Считывать данные из соответствующих листов.
- Обрабатывать данные с учетом всех ограничений.
- Записывать итоговое расписание на лист «Итог».

Для обработки данных и распределения аттестаций предлагается использовать следующие алгоритмы и методы:

- Жадный алгоритм: на каждом шаге выбирается наиболее подходящая дата и аудитория для текущей аттестации, исходя из доступности ресурсов и временных ограничений.
- Приоритетная очередь: для управления очередью аттестаций, где приоритет определяется типом аттестации (например, экзамены имеют более высокий приоритет, чем зачеты) и временем на подготовку.
- Поиск в глубину (DFS) или поиск в ширину (BFS): для проверки доступности аудиторий и преподавателей в определенные даты и время.
- Динамическое программирование: для оптимизации распределения аудиторий и временных интервалов.

Программа учитывает нагрузку на аудитории и преподавателей, обеспечивая непротиворечивое их распределение. Аудитории: выбор наиболее подходящей аудитории для каждой аттестации с учетом ее типа и вместимости. Преподаватели: учитываются их предпочтения и ограничения на количество аттестаций в день.

После распределения всех аттестаций программа генерирует итоговое расписание, которое включает:

- Таблицу с расписанием для каждой группы студентов, включая даты, время, аудитории и преподавателей.
- О выполнении правил и пожеланий.

– Возможны случаи, когда заданные в исходной информации ограничения не позволяют построить полностью непротиворечивое расписание, информация об этом также включается в результирующий файл.

Программа предоставляет информацию о результатах составления расписания:

– Процент успешно распределённых аттестаций.

– Процент выполненных пожеланий, выполненные и невыполненные отмечаются в соответствующих строках исходных данных.

– Подсказки по устранению проблем в исходных данных с целью успешного составления непротиворечивого расписания.

В случае, если составление расписания программе не удалось завершить полностью, для составителя расписания предполагается возможность скорректировать исходные данные (например, в сторону ослабления ограничений по аудиториям или датам) и снова запустить программу составления расписания.

Предложенный проект автоматизации формирования расписания сессии является экспериментом в рамках одного из институтов СПбПУ для оценки целесообразности разработки расширенной версии инструмента автоматизации такого типа, имеющего целью помочь сотрудникам генерировать удобные расписания сессий во всём вузе.

ЛИТЕРАТУРА

1. Иванов А. А., Петров Б. Б. Автоматизация составления расписания в вузах: проблемы и решения // Информационные технологии в образовании. – 2020. – Т. 15. – № 3. – С. 45-52.
2. Сидоров В. В., Кузнецов Д. Д. Алгоритмы оптимизации расписаний учебных занятий // Программные системы и вычислительные методы. – 2019. – Т. 12. – № 4. – С. 67-74.
3. Smith J., Brown R. Automated Scheduling in Higher Education: A Review of Current Approaches // Journal of Educational Technology. – 2021. – Vol. 18. – No 2. – P. 89-102.
4. Смирнов А. А., Фёдоров В. В. (2021). Опыт использования алгоритмов динамического программирования для составления расписаний в вузах. Вестник СПбПУ, Т. 18, № 2, С. 89-102.

УДК 004.738.5

А.А. Колосовская (4 курс бакалавриата),
Т.В. Леонтьева, к.т.н., доцент

ПЛАТФОРМА ДЛЯ ПОИСКА И ОРГАНИЗАЦИИ МЕРОПРИЯТИЙ ПО НАСТОЛЬНЫМ ИГРАМ: РАЗРАБОТКА И РЕАЛИЗАЦИЯ

Современные информационные технологии радикально меняют способы организации досуга и коммуникации между людьми. Рост интереса к специализированным онлайн-платформам обусловлен потребностью в создании сообществ, объединённых общими увлечениями. Цель данной работы заключается в разработке веб-приложения, позволяющего пользователям находить единомышленников для совместного проведения настольных игр, участвовать в рейтинговых мероприятиях, обмениваться отзывами и мотивировать активное участие посредством системы уровней и достижений.

Для реализации данного проекта выбрана современная архитектура, основанная на принципах микросервисного подхода, что позволяет разнести функциональные блоки приложения на независимые компоненты и обеспечить гибкость, масштабируемость и отказоустойчивость системы. Серверная часть разрабатывается с использованием Spring Boot – мощного фреймворка для создания микросервисов, упрощающего разработку, тестирование и интеграцию модулей [6]. Взаимодействие между сервисами организовано посредством Apache Kafka – системы обмена сообщениями, позволяющей реализовать асинхронное уведомление пользователей о новых событиях и изменениях в реальном времени [3].

Надёжное хранение информации о пользователях, настольных играх, отзывах и мероприятиях обеспечивается за счёт использования реляционной базы данных PostgreSQL, которая благодаря продвинутым механизмам индексирования и транзакционной обработке

позволяет оперативно выполнять выборки и обновления данных даже при высоких нагрузках [5].

Клиентская часть приложения реализуется с применением современных JavaScript-фреймворков (React, Angular), что обеспечивает создание динамичных, адаптивных и интуитивно понятных интерфейсов. Предварительное прототипирование в Figma позволяет на ранних этапах оптимизировать пользовательский опыт и визуальное оформление, а плавные анимации и переходы, реализованные с помощью CSS и JavaScript, способствуют повышению вовлечённости пользователей.

Особое внимание уделено процессам автоматизированного развёртывания и управления инфраструктурой. Использование контейнеризации с Docker и систем оркестрации Kubernetes позволяет ускорить цикл разработки, обеспечить лёгкое масштабирование сервисов, повысить отказоустойчивость и снизить операционные затраты [2]. Эти инструменты автоматизируют деплоймент, мониторинг и управление ресурсами приложения в облачной среде, что критически важно при росте числа пользователей.

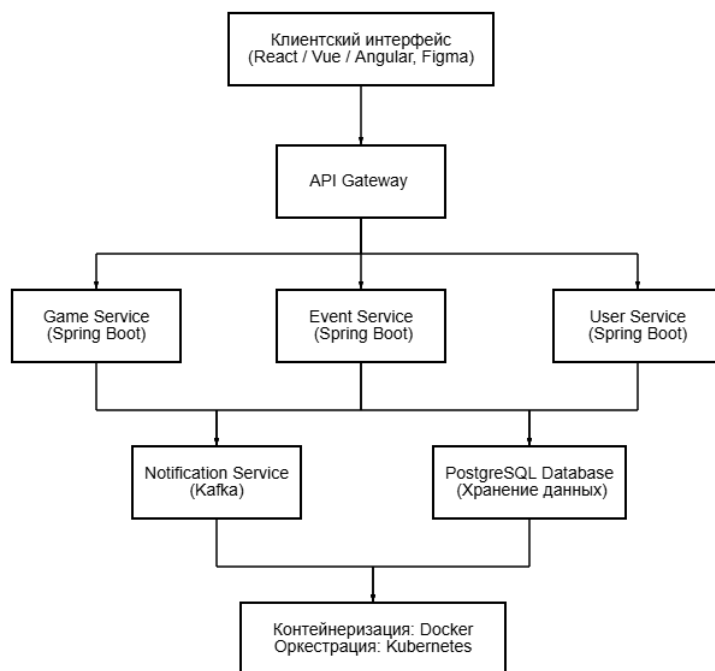


Рисунок 1 – Схема организации системы веб-приложения

Данная схема демонстрирует общую архитектуру системы: клиентский интерфейс взаимодействует с серверной частью через API, после чего запросы распределяются между микросервисами, каждый из которых выполняет свою специализированную функцию. Взаимодействие между сервисами осуществляется с помощью Apache Kafka [3], а централизованное хранение данных организовано в базе PostgreSQL [5]. Контейнеризация и оркестрация обеспечивают гибкость и масштабируемость всего решения [2].

Особое внимание уделено и интеграции элементов геймификации, таких как система уровней, рейтингов и виртуальных наград. Внедрение данных механизмов, подтверждённое современными исследованиями в области мотивационных информационных систем, способствует повышению вовлечённости пользователей и улучшению качества взаимодействия с платформой [4].

Таким образом, представленный подход базируется на использовании передовых технологий, позволяющих создавать высокопроизводительные и масштабируемые веб-приложения. Решение обеспечивает эффективную интеграцию функциональных модулей, динамичный и удобный интерфейс, а также гибкие механизмы управления данными и инфраструктурой. В результате пользователи получают возможность не только находить партнёров для настольных игр, но и активно участвовать в жизни сообщества посредством рейтинговых систем, автоматического начисления опыта (XP) и модерации контента.

Для реализации поставленных целей в работе решаются следующие задачи:

1. Проведение анализа современных инструментов разработки веб-приложений и выявление существующих ограничений платформ для поиска единомышленников.
2. Проектирование архитектуры системы с разделением на микросервисы и реализация асинхронного обмена сообщениями с помощью Apache Kafka [3].
3. Разработка серверной части на базе Spring Boot [6], обеспечивающей гибкость и масштабируемость приложения.
4. Создание адаптивного клиентского интерфейса с применением современных JavaScript-фреймворков и предварительное прототипирование в Figma.
5. Интеграция систем контейнеризации и оркестрации (Docker, Kubernetes) для автоматизации развёртывания и управления инфраструктурой [2].
6. Проведение комплексного тестирования и демонстрация эффективности предлагаемого решения.

Полученные результаты и выбранный подход демонстрируют практическую применимость современных технологий в решении актуальных задач, а также обоснование выбора инструментов на основе их проверенной эффективности и функциональных возможностей. Представленная схема организации системы служит теоретической базой для дальнейших исследований и оптимизации платформы.

ЛИТЕРАТУРА

1. Сэм Ньюмен. Создание микросервисов. – 2021. – 624pp.
2. Александр Бранд, Рич Ландер, Джош Росс, Джон Харрис, Kubernetes на практике. – 2022. – 354pp.
3. Документация Kafka. [Электронный ресурс] Режим доступа: <https://kafka.apache.org/documentation.html>
4. Koivisto, Jaakko, and Juho Hamari. “The Rise of Motivational Information Systems: A Review of Gamification Research.” International Journal of Information Management. – 2019.
5. PostgreSQL Global Development Group. PostgreSQL Documentation. [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/index.html>
6. Spring Boot Reference Guide. [Электронный ресурс] Режим доступа: <https://docs.spring.io/spring-boot-data-geode-build/1.1.x/reference/html5/>

УДК 004.453

Е.Д. Кошкарёва (3 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

ИНСТРУМЕНТ ДЛЯ УДАЛЕННОГО УПРАВЛЕНИЯ КОНФИГУРАЦИЕЙ ПРИЛОЖЕНИЙ

Приложения часто развиваются в условиях динамически изменяющихся требований и необходимости быстрого внедрения новых функций. Для обеспечения гибкости и удобства управления функциональностью всё чаще используется подход Remote Configs (удалённые настройки) [1].

Удалённые настройки позволяют управлять поведением и параметрами приложений без необходимости выпуска новых версий. Это особенно важно для мобильных приложений, где каждое обновление требует прохождения проверки в магазинах приложений. С помощью удалённых конфигураций можно вносить изменения в режиме реального времени, быстро адаптироваться к новым условиям и обеспечивать стабильность работы приложения.

Использование удалённых конфигураций стало стандартом для многих крупных компаний, таких как Google, Facebook и Netflix. Разработка решения в этой области повышает эффективность процессов управления приложением, что делает данную тему особенно актуальной для исследования и применения.

Одно из самых популярных решений - Firebase Remote Config [2]. Однако он зависит от инфраструктуры Google, что в текущих условиях может привести к сбоям или недоступности сервиса. Также существуют ограничения на количество ключей и объём данных, что может быть критичным для масштабных проектов.

Также существует RuStore Remote Config [3], но он требует времени на изучение, особенно для разработчиков, уже работающих с другими инструментами. Кроме того, его функциональность может быть менее развитой по сравнению с Firebase, особенно в плане интеграции с аналитическими сервисами.

Таким образом, целью данной работы является создание инструмента для удаленного управления конфигурацией приложений. Инструмент должен предоставлять возможность отдельные функции приложения удаленно без необходимости запуска масштабного обновления или развертывания его новой версии.

Главной идеей приложения является веб-сервис, который позволяет динамически управлять данными в приложении. Веб-сервис принимает данные в формате json, которые загружаются в приложение пользователя во время его запуска. Есть возможность изменять существующие данные и добавлять новые, а также разделять данные по разным версиям приложения.

Выбран стек .NET C#, MongoDB и React, обеспечивающий производительность, безопасность и масштабируемость. .NET отличается высокой скоростью обработки API-запросов и встроенной защитой данных [4]. MongoDB позволяет хранить динамические JSON-данные без строгих схем, обеспечивая гибкость и быструю синхронизацию [5]. React используется для создания удобного веб-интерфейса с модульной структурой [6]. Совместное использование этих технологий создаёт эффективную систему для удалённого управления конфигурациями, адаптивную к изменениям и масштабируемую.

Приложение использует JSON-файлы для хранения и обновления параметров, позволяя изменять конфигурацию без правок в коде. На этапе первоначальной загрузки JSON-файл с параметрами становится первой версией (v1.0) и получает уникальный ID. Этот идентификатор применяется в приложении для обращения к актуальным данным. Во время запуска система загружает последнюю версию JSON, преобразует его в объект и делает доступным для работы.

Обновление конфигурации происходит путём загрузки нового JSON-файла, при этом версия увеличивается автоматически. После следующего запуска устаревшие данные заменяются актуальными, что позволяет оперативно вносить изменения в настройки приложения. Чтобы избежать ошибок, предусмотрены механизмы валидации, проверяющие структуру JSON, корректность типов данных и наличие обязательных полей. Кеширование данных обеспечивает стабильную работу даже при отсутствии соединения с сервером.

Схему работы сервиса демонстрирует рисунок 1.

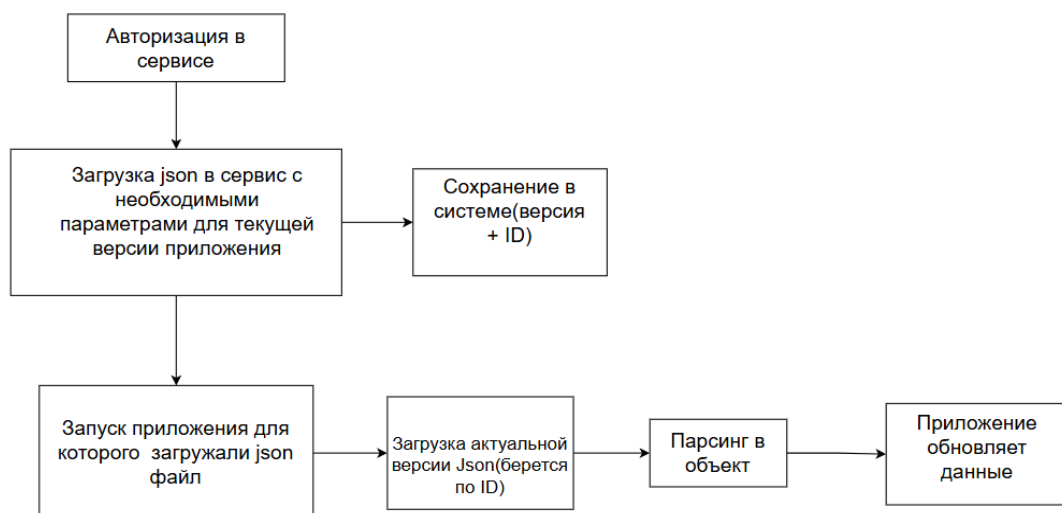


Рисунок 1 – схема работы сервиса удаленных конфигураций

Такой подход обеспечивает гибкость конфигурации, позволяет оперативно обновлять параметры без перекомпиляции приложения и гарантирует стабильность работы за счёт проверки входных данных.

ЛИТЕРАТУРА

1. Microsoft. Azure App Configuration Documentation [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/en-us/azure/azure-app-configuration/>, свободный. – Загл. с экрана.
2. Google. Firebase Remote Config Documentation [Электронный ресурс]. – Режим доступа: <https://firebase.google.com/docs/remote-config>, свободный. – Загл. с экрана.
3. RuStore. Документация RuStore Remote Config [Электронный ресурс]. – Режим доступа: <https://www.rustore.ru/help/developers/tools/remote-config/general>, свободный. – Загл. с экрана.
4. Bass L., Clements P., Kazman R. Software Architecture in Practice [Электронный ресурс]. – Режим доступа: <https://www.oreilly.com/library/view/software-architecture-in/9780132942785/>, свободный. – Загл. с экрана.
5. Turnbull J. The Art of Monitoring [Электронный ресурс]. – Режим доступа: <https://artofmonitoring.com/>, свободный. – Загл. с экрана.

УДК 004.42

М.А. Коновалов (2 курс магистратуры),
В.В. Амосов, к.т.н., доцент

РАЗРАБОТКА ДИЗАЙН СИСТЕМЫ ДЛЯ ПРИЛОЖЕНИЙ ПОД ПЛАТФОРМУ iOS

Современная разработка мобильных приложений требует создания эффективных решений для проектирования интерфейсов, которые отвечают требованиям удобства, гибкости и доступности. Дизайн-системы [1] стали ключевым инструментом для решения этих задач, предоставляя наборы унифицированных компонентов, стилей и правил, которые способствуют единообразию, ускоряют разработку и упрощают масштабируемость приложений.

Для приложений под платформу iOS дизайн-система играет важную роль, обеспечивая соответствие интерфейсов рекомендациям **Human Interface Guidelines** (HIG) [2] и упрощая процесс адаптации интерфейсов под различные устройства и разрешения экранов.

Таким образом *целью* работы является проектирование и реализация дизайн системы на операционной системе iOS.

Для достижения данной цели необходимо решение следующих *задач*:

1. Провести исследование и провести сравнительный анализ существующих решений и подходов к реализации дизайн систем на мобильных платформах. Выявить недостатки и сформулировать преимущества.
2. Разработать механизм по работе с палитрой.
3. Реализовать компоненты для разработки сценариев.
4. Разработать механизм по поддержке смены темы в рамках компонентов.
5. Реализовать механизм по поддержке динамических шрифтов и доступности (для людей с ограниченными возможностями) в рамках компонентов
6. Поддерживать работу дизайн системы на фреймворке uikit и swiftui[3].

На рынке существует большое количество различных дизайн систем. Были рассмотрены наиболее популярные из них: Xela, Lightning Design System, Atlassian Design System.

№	Название	Возможность менять компоненты под свой дизайн	Поддержка темной темы	Поддержка доступности для людей с ограниченными возможностями	Наличие необходимых компонентов для создания полноценных приложений под платформу iOS	Поддержка возможности использовать дизайн систему на платформе iOS
1	Xela UI kit for Swift iOS components	+	+	-	Отсутствует TabBar. Нет палитры.	-
2	Lightning Design System	+	+	+	Нет TabBar. Нет банеров. Нет атрибутированного текста.	-
3	Atlasian Design System	+	-	-	Нет TabBar. Нет банеров. Нет атрибутированного текста. Нет палитры	-

Рисунок 1 – сравнительная таблица существующих решений

В ходе анализа было установлено, что все существующие решения соответствуют критериям частично. Из проведенного анализа следует что необходимо разработать дизайн систему, которая будет удовлетворять всем требованиям.

Дизайн система была реализована на языке программирования Swift [4] – родном языке платформы iOS. Для создания палитры была использована система токенов. Токены разбиваются на отдельные списки перечисления, которые относятся к определенной категории принадлежности. Например, список токенов для задания цвета текста, фона, иконок. Токен представляет из себя название цвета из палитры например primary или secondary и является абстракцией над настоящим значением цвета. Это нужно чтобы поддерживать различные значения цветов в разных темах, кроме того, использование абстракции значительно упрощает рефакторинг.

Верстка компонентов реализована на основе системных компонентов нативного фреймворка UIKit. Кроме того, для поддержки нового фреймворка для верстки SwiftUI, компоненты дизайн системы были переведены на него с помощью подхода UIRepresentableView[5]. Это позволило не переписывать все компоненты на новый фреймворк.

Архитектура реализации компонентов представлена на рисунке 2. Она заключается в разбиении компонентов на более маленькие компоненты(базовые). Данная архитектура имеет следующие преимущества:

- Компоненты разделены на несколько уровней(модулей), это позволяет переиспользовать их и подменять реализацию при необходимости
- При изменении реализации на определенном уровне, изменения сразу попадают во все места использования компонента.

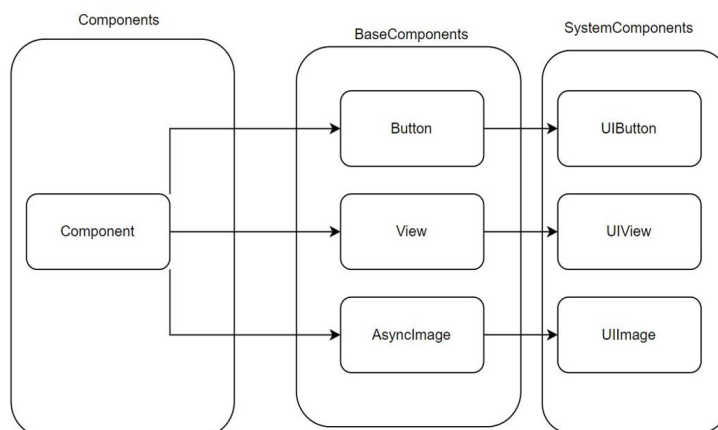


Рисунок 2 – архитектура создания компонента

Так же можно отметить при наличии бизнес-логики или модели в компоненте использовалась архитектура MVC [6].

Реализация темизации была основана на паттерне **Notifications** (рассылки) [7]. Каждый компонент подписывается на события и при обновлении на основе значения токена из объекта палитры (или из токена шрифта) берется необходимое значение.

Таким образом были рассмотрены существующие дизайн системы. В результате анализа были выявлены основные требования, которым должна соответствовать программная реализация дизайн системы на платформе iOS. Предложена архитектура и технологический стек для реализации.

ЛИТЕРАТУРА

1. Mobile Design Systems – Tips and Examples [Электронный ресурс] Режим доступа: <https://www.uxpin.com/studio/blog/mobile-design-systems/>
2. Human Interface Guidelines. [Электронный ресурс] Режим доступа: <https://developer.apple.com/design/human-interface-guidelines/>
3. Introducing SwiftUI. [Электронный ресурс] Режим доступа <https://developer.apple.com/swiftui/>
4. The Swift Programming Language Book [Электронный ресурс] Режим доступа: <https://www.swift.org/documentation/>
5. UIViewRepresentable explained to host UIView instances in SwiftUI. [Электронный ресурс] Режим доступа: <https://www.avanderlee.com/swiftui/integrating-swiftui-with-uikit/>
6. Булыга А.И., АНАЛИЗ И СРАВНЕНИЕ АРХИТЕКТУРНЫХ ШАБЛОНОВ ПРОЕКТИРОВАНИЯ, ИСПОЛЬЗУЕМЫХ ПРИ РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ ПЛАТФОРМЫ IOS Научный журнал № 1(63), 2022.
7. Karoly Nyisztor, Monika Nyisztor Design Patterns in Swift 5. - Razeware LLC , 2019. - 277 с.

УДК 004.932.72'1

Р.А. Кумище, Я.В. Беккер (2 курс бакалавриата),
Е.Г. Хитров, д.т.н., доцент

МЕТОДЫ АДАПТАЦИИ ДОМЕНА И ПЕРЕНОСА ОБУЧЕНИЯ ДЛЯ РАСПОЗНАВАНИЯ ДЕФЕКТОВ МАТЕРИАЛОВ ПО СНИМКАМ ПОВЕРХНОСТИ

В эпоху активного развития технологий машинного обучения и компьютерного зрения одной из ключевых проблем остается недостаток данных, необходимых для разработки и последующей программной реализации моделей в определенных доменах. Это особенно актуально для задач, где снимки объектов исследования редки [1], либо их получение связано с рисками — например, в промышленности. Однако часто существуют схожие домены, в которых данные более доступны, но вопрос их прямого использования для решения целевой задачи требует отдельной проработки.

Целью работы является изучение вопроса эффективности методов адаптации домена [1] и переноса обучения [1] для решения проблемы ограниченности данных при обучении моделей компьютерного зрения в задаче детекции. Исследуется практическая задача из металлургической отрасли — распознавание трещин металла на производстве, где объем данных (например, сет [2]) ограничен из-за редкости дефектов и сложности получения снимков. При этом доступен большой по объему набор данных трещин в бетоне [3], который, хотя и относится к другому домену, имеет схожие визуальные характеристики. В нашей работе последовательно решены следующие задачи:

1. обучение модели на целевом домене (трещины в металле, сет [2]),
2. обучение модели на схожем наборе данных (трещины в бетоне [3], адаптация домена),
3. последующее дообучение на целевых данных [2] (перенос обучения).

На настоящем этапе разработки в качестве базовой модели для обучения использована искусственная нейронная сеть (ИНС) YOLOv8s.pt [4], предназначенная для решения задач детекции объектов в реальном времени. Базовая модель отличается высокой скоростью работы, гибкостью настроек, открытым исходным кодом и лицензией для некоммерческого использования [4], что делает ее применение популярным в исследованиях в области компьютерного зрения.

В ходе работы были проведены три эксперимента и получены три обученные модели для последующего тестирования:

1. Обучение модели на целевом домене — наборе трещин металла [2]. Набор состоит из 155 изображений трещин в металле в формате JPG, из которых 142 составили обучающую выборку, остальные 13 — валидационную [2]. Представлена модель 1 «Целевой домен».

2. Обучение модели на схожем домене — наборе снимков трещин бетона [3]. Набор состоит из 3829 изображений трещин бетона в формате JPG, из которых 3717 были использованы для обучения, 112 для валидации модели [3]. Представлена модель 2 «Схожий домен», при тестировании проверяется адаптация домена.

3. Дообучение модели 2 на наборе трещин металла [2]. Представлена модель 3 «Схожий + целевой домены», при тестировании проверяется модель с переносом обучения.

В экспериментах обучение выполнялось в течении 100 эпох (гиперпараметр $epoch = 100$), размер изображений составлял 640x640 пикселей ($imgsz = 640\text{ pcs}$), размер батча 8 снимков ($batch = 8$).

Программная часть стенда для эксперимента реализована в среде Jupyter Notebook на языке Python версии 3.9.21. Обучение и тестирование моделей проводилось с использованием графического процессора NVIDIA GeForce RTX 3060 12 ГБ.

Сравнение точности представленных моделей на тестовом наборе целевых данных, представляющем собой подвыборку сета [2], не использовавшуюся при обучении, выполнено по метрикам усредненной точности (mAP50) [5], точности (Precision) и полноты (Recall). Подвыборка для тестирования моделей включала в себя 56 изображений трещин металла в формате JPG. Гистограмма сравнения трех моделей на тестовом наборе данных по указанным метрикам представлена на рисунке 1.

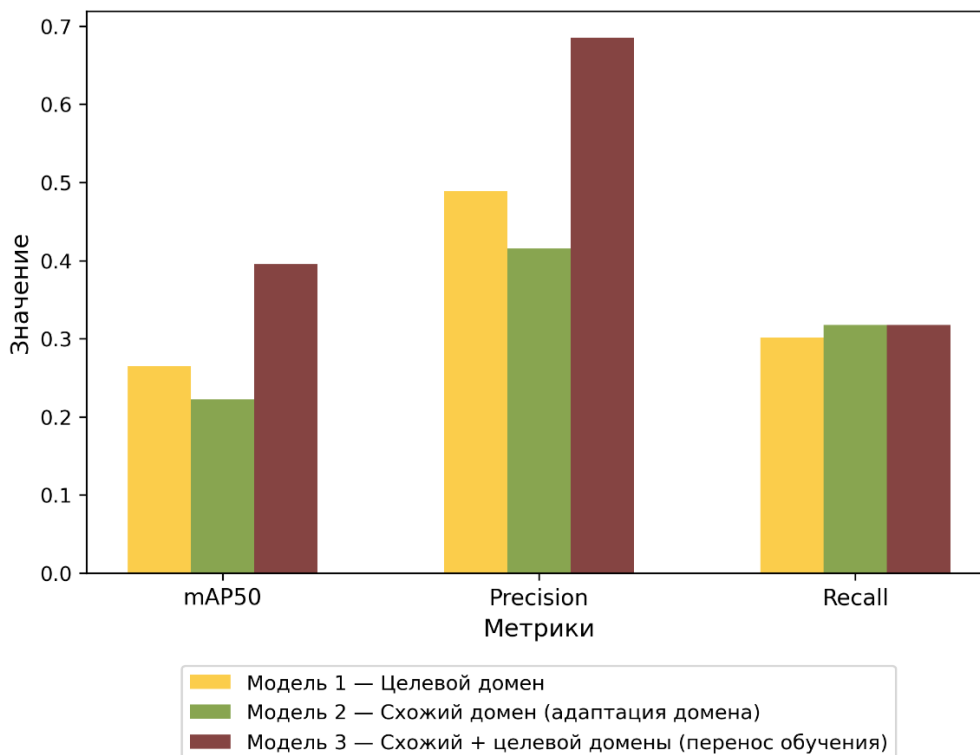


Рисунок 1 – Сравнение метрик mAP50, Precision и Recall для экспериментальных моделей

Результаты исследования показали, что модель 2, применяющая метод адаптации домена, при тестировании улучшила показатель Recall на 5%, при этом уменьшила значения mAP50 и Precision на 16% и 15% соответственно относительно модели 1, обученной на малом наборе целевых данных. Модель 3, использующая метод переноса обучения, демонстрирует улучшение всех основных метрик: значение mAP50 выросло на 49% относительно модели 1; значение Precision выросло на 40%; значение Recall осталось равным значению второй модели — прирост на 5% относительно модели 1, обученной на целевых данных.

Исходя из результатов экспериментов можно предварительно сделать вывод, что для задач детекции трещин металла при ограниченном объеме целевых данных и при наличии схожих доменов, метод переноса обучения является перспективным подходом, в то время как метод адаптации домена не демонстрирует существенного улучшения метрик качества распознавания.

Для дальнейшей проработки и получения более полных сведений о влиянии различных факторов на качество работы моделей при детекции трещин металла рекомендуется провести дополнительные исследования с увеличенными объемами данных, вариацией гиперпараметров и расширенным набором экспериментальных условий, включая различные модели ИНС, что позволит более точно оценить потенциал каждого из методов.

ЛИТЕРАТУРА

1. Hutter F., Kotthoff L., Vanschoren J. (2019). Automated Machine Learning - Methods, Systems, Challenges. DOI: 10.1007/978-3-030-05318-5.
2. YOLO v7 metal defect instance Dataset > Overview [Электронный ресурс] Режим доступа: <https://universe.roboflow.com/yolo-v7-segmentation/yolo-v7-metal-defect-instance>
3. Crack Dataset Overview [Электронный ресурс] Режим доступа: <https://universe.roboflow.com/university-bswxt/crack-bphdr>
4. Ultralytics YOLO Docs. [Электронный ресурс] Режим доступа: <https://docs.ultralytics.com/ru>
5. YOLO Показатели производительности – Ultralytics YOLO Docs [Электронный ресурс] Режим доступа: <https://docs.ultralytics.com/ru/guides/yolo-performance-metrics/#class-wise-metrics>

УДК 004.82

Л.А. Лаврова (1 курс магистратуры),
К.А. Туральчук, к.т.н., доцент

ОБЗОР ПОДХОДОВ К ВАЛИДАЦИИ LLM-МОДЕЛЕЙ

Современные большие языковые модели (Large Language models, далее - LLM) демонстрируют высокую производительность в генерации связных текстов, однако одной из ключевых проблем, возникающих при разработке программных решений, использующих LLM, остается валидация таких моделей [1], то есть оценка их способности генерировать релевантные, информативные и достоверные ответы. Многие аспекты их использования до настоящего времени изучены слабо, поскольку LLM получили распространение сравнительно недавно. В частности, валидация диалоговых LLM-моделей остается актуальной исследовательской и практической задачей, для решения которой применяются различные классы метрик [1]. Задача включает в себя анализ различных аспектов, таких как лексическая точность, семантическая эквивалентность и фактологическая достоверность. Существующие методы валидации, основанные на лексических, семантических и фактологических метриках, применяются изолированно, что ограничивает их общую эффективность [1].

Для более подробного анализа эффективности существующих подходов к валидации диалоговых LLM-моделей, результаты которого позволят уточнить перспективные направления дальнейших исследований, рассмотрим существующие подходы, а также выявим ограничения каждого из методов в контексте их применимости для оценки качества генерации.

Лексические метрики, такие как BLEU и ROUGE, основаны на сопоставлении текста модели с эталонным ответом на уровне слов или фраз. BLEU, разработанный для оценки машинного перевода, использует модифицированную точность n-грамм для сравнения переведенных текстов с эталонными [2]. ROUGE анализирует совпадения n-грамм между сгенерированным текстом и эталонным, что полезно для задач автоматического резюмирования [3]. Однако эти метрики не учитывают синонимичность и вариативность формулировок, что ограничивает их применимость для оценки семантической эквивалентности.

Семантические метрики, такие как BERTScore и BARTScore, используют контекстуальные эмбединги для оценки смысловой близости между ответом модели и эталоном. BERTScore вычисляет семантическое сходство между токенами на уровне эмбедингов, что позволяет учитывать смысловую эквивалентность текстов [4]. BARTScore, основанный на модели BART, оценивает вероятность генерации текста на основе эталонного, что позволяет анализировать семантическую полноту и достоверность сгенерированных текстов [5]. Однако эти метрики также могут быть нечувствительны к фактической точности, что ограничивает их способность обнаруживать галлюцинации в ответах LLM.

Метрики фактической точности, такие как FactCC и FEQA, направлены на проверку достоверности и корректности текстов, которые генерирует модель. FactCC использует синтетически созданные данные для обучения модели, которая проверяет фактологическую согласованность сгенерированных текстов с исходным документом [6]. FEQA, в свою очередь, оценивает фактическую правильность сгенерированных текстов через автоматически сгенерированные вопросы и ответы, что позволяет обнаруживать галлюцинации и противоречия [7]. Эти метрики обладают высокой способностью к обнаружению галлюцинаций в ответах LLM, но требуют наличия исходного документа для проверки.

Проведенный обзор показывает, что существующие методы валидации диалоговых LLM-моделей имеют как сильные, так и слабые стороны. Лексические метрики, такие как BLEU и ROUGE, эффективны для оценки лексической точности, но не учитывают семантическую эквивалентность и фактологическую достоверность. Семантические метрики, такие как BERTScore и BARTScore, лучше оценивают смысловую близость, но могут быть нечувствительны к фактической точности. Метрики фактической точности, такие как FactCC и FEQA, обладают высокой способностью к обнаружению галлюцинаций, но требуют наличия исходного документа для проверки.

Результаты выполненного обзора позволяют предположить, что для совершенствования валидации диалоговых LLM перспективен комплексный подход, объединяющий семантический и фактологический анализ. Такой подход предполагает использование мультикомпонентной модели оценки, которая включает семантический модуль, оценивающий схожесть между входным текстом и сгенерированным, и фактологический модуль, выполняющий проверку достоверности утверждений. Предполагается, что такой подход обеспечит более точную валидацию диалоговых LLM-моделей по сравнению с использованием отдельных метрик.

ЛИТЕРАТУРА

1. Chen X., Gao C., Chen C., Zhang G., Liu Y. An empirical study on challenges for LLM application developers // ACM Trans. Softw. Eng. Methodol. – 2024. – URL: <https://arxiv.org/abs/2408.05002>
2. Papineni K., Roukos S., Ward T., Zhu W. J. BLEU: a method for automatic evaluation of machine translation // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL). — Philadelphia: Association for Computational Linguistics, 2002. – P. 311-318.
3. Chin-Yew Lin. . ROUGE: A Package for Automatic Evaluation of Summaries // Text Summarization Branches Out. – Barcelona, Spain: Association for Computational Linguistics, 2004. – P. 74-81.
4. Zhang T., Kishore V., Wu F., Weinberger K., Artzi Y. BERTScore: Evaluating Text Generation with BERT // Proceedings of the International Conference on Learning Representations (ICLR). – 2020. – URL: <https://arxiv.org/abs/1904.09675>

5. Yuan W., Neubig G., Liu P. BARTSCORE: Evaluating Generated Text as Text Generation // 35th Conference on Neural Information Processing Systems. – Sydney: Neural Information Processing Systems, 2021. – URL: <https://doi.org/10.48550/arXiv.2106.115206>.
6. Kryscinski W., McCann B., Xiong C., Socher R. Evaluating the factual consistency of abstractive text summarization // Conference on Empirical Methods in Natural Language Processing. – 2020. – P. 9332–9346.
7. Durmus E., He H., Diab M. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL). — :Association for Computational Linguistics, 2020. – P. 5055–5070.

УДК 004.4'236

Д.С. Лебедев (3 курс бакалавра),
А.П. Маслаков, ст. преподаватель

ИНТЕРАКТИВНОЕ ОБУЧАЮЩЕЕ ПРИЛОЖЕНИЕ ДЛЯ ИЗУЧЕНИЯ ОСНОВ ПРОГРАММИРОВАНИЯ И РОБОТОТЕХНИКИ

В настоящее время большинство образовательных учреждений в небольших городах сталкиваются со сложностями при обучении детей программируемой робототехнике. Основная проблема заключается в высоких затратах на оборудование, т.к. физические робототехнические конструкторы стоят дорого и обеспечить один учебный класс набором таких конструкторов многим не под силу [1].

Одним из способов решения описанной проблемы могло бы стать использование программных продуктов, которые можно запустить на персональном компьютере и которые позволяли бы обучать детей программируемой робототехнике без необходимости приобретения дорогостоящего оборудования. Персональные компьютеры и ноутбуки широко распространены и имеются в каждой школе, поскольку они являются неотъемлемой частью учебного процесса, особенно на уроках информатики. Использование виртуальных образовательных сред позволяет не только экономить бюджет, но и значительно расширяет возможности интерактивного обучения [2], предоставляя учащимся доступ к разнообразным сценариям программирования.

Одним из таких возможных инструментов является Lego Mindstorms Robot Inventor [3], который предоставляет ПО для программирования физических роботов Lego с помощью визуального блочного программирования. Но этот инструмент для полноценной работы все ещё требует наличие физического робота, стоимость которого высока.

Также существует VEXcode VR [4] – приложение для программирования роботов в виртуальной среде, но оно не поддерживает русский язык.

Таким образом, целью данной работы является создание интерактивного приложения с возможностью программирования робота, который выполнял бы заданные ему операции в трехмерной виртуальной среде.

Для достижения этой цели необходимо решение следующих задач:

1. Обзор существующих решений для обучения детей основам программирования и робототехнике.
2. Проведение сравнительного анализа найденных решений.
3. Формирование требований к собственной разработке.
4. Реализация в виде десктопного приложения.

Главным идеей приложения является виртуальная трехмерная среда с роботом, который программируется при помощи визуального блочного программирования.

Приложение было реализовано в виде десктопного приложения на языке программирования C# [5] в мультиплатформенной среде для разработки 3D-приложений Unity [6]. Пользователю предоставляется возможность конфигурировать робота различными

датчиками и сенсорами, а после предлагается написать для него программу, для выполнения роботом всевозможных задач, начиная с поиска и движения по линии и заканчивая поиском выхода из лабиринта. В ходе работы пользователь сможет отслеживать выполнение программы в режиме реального времени, что способствует более глубокому пониманию принципов работы алгоритмов.

Для робота была реализована реалистичная физика движения, учитывающая особенности динамики и инерции. После полного отключения моторов и включения тормоза он не останавливается мгновенно, а продолжает двигаться по инерции до полной остановки. Также учтены силы трения и сопротивления, поэтому при недостаточной мощности моторов робот не способен преодолевать подъемы.

Кроме того, для робота был разработан набор датчиков, принципы действия которых основаны на их физических аналогах:

- Датчик линии, определяющий количество отраженного света от поверхности.
- Датчик касания, работающий по принципу замыкания электрической цепи.
- Датчик цвета, фиксирующий цвет поверхности в трехбитном формате.
- Датчик освещенности, оснащенный фоторезистором, выходной сигнал которого зависит от интенсивности света.
- Светодиод, который загорается при подаче на него сигнала.
- Дальномер, использующий эхолокацию: он посылает сигнал и принимает его отражение.

Основой блочного программирования является BlocksEngine2 – ассет из Asset Store, предоставляющий ядро и каркас для создания собственных блоков для управления целевым объектом – роботом. Общие блоки в нем уже присутствуют, такие как:

- Блоки управления – это блоки для управления ходом выполнения программы, т.е. циклы, условия, блоки ожидания.
 - Блоки операторов – это блоки для выполнения арифметических операций: сложение, умножение, вычитание, деление, округление и т.п.
 - Блоки переменных – это блоки для создания переменных и управления их состоянием.
- Как уже было сказано, BlockEngine2 предоставляет лишь каркас, а реализация блоков для управления роботом проходила собственными силами. Были реализованы следующие группы блоков:

- Блоки датчиков – это блоки, которые позволяют получать значения с датчиков робота.
- Блоки движения – это блоки для управления моторами робота.
- Блоки дальномера – это блоки для управления дальномером робота.

Применение блочного программирования в сочетании с реалистичной физической моделью робота дает возможность учащимся не только изучать программирование, но и понимать основы работы датчиков и сенсоров. Реализованный набор датчиков и механизмы управления движением обеспечивают взаимодействие с виртуальной средой близкое к реальному.

ЛИТЕРАТУРА

1. Робототехника: перспективы и проблемы обучения детей [Электронный ресурс]. – Режим доступа : <https://portamur.ru/texts/novolt/robototekhnika-perspektivy-i-problemy-obucheniya-detey/>, свободный. – Загл. с экрана.
2. Интерактивные методы как инновационные подходы в обучении [Электронный ресурс]. – Режим доступа : <https://scipress.ru/pedagogy/articles/interaktivnye-metody-kak-innovatsionnye-podkhody-v-obuchenii.html>, свободный. – Загл. с экрана.
3. LEGO® MINDSTORMS®| About | Official LEGO® Shop US [Электронный ресурс]. – Режим доступа : <https://www.lego.com/en-us/themes/mindstorms/about>, свободный. – Загл. с экрана.
4. VEXcode Virtual Robots (VR) – UK – VEX Robotics [Электронный ресурс]. – Режим доступа : <https://www.vexrobotics.com/vexcode/vr>, свободный. – Загл. с экрана.
5. C# | Современный язык программирования с открытым исходным кодом для .NET [Электронный ресурс]. – Режим доступа : <https://dotnet.microsoft.com/ru-ru/languages/csharp>, свободный. – Загл. с экрана.

6. Платформа и редактор для разработки RT3D-контента | Unity [Электронный ресурс]. – Режим доступа : <https://unity.com/ru/products/unity-engine>, свободный. – Загл. с экрана.

УДК 004.652

Р.А. Лисовик (2 курс магистратуры),
П.Д. Дробинцев, к.т.н., доцент

ВЫБОР АРХИТЕКТУРЫ БАЗЫ ДАННЫХ ПРИЛОЖЕНИЯ АВТОСЕРВИСА STREET LEGAL SERVICE

Разработка приложения автосервиса всегда сопровождается необходимостью реализации базы данных. Для обеспечения надлежащего качества приложения нужно заранее выбирать такую архитектуру БД, которая будет устойчива к масштабированию, а также снизит дублирование одной и той же информации [5]. В таких условиях, следует рассмотреть несколько методологий реализации хранилищ данных, выделить их преимущества и недостатки, а после сравнить между собой с целью определения наилучшей. В данной статье будут рассмотрены три методологии: Data Vault, Anchor Modeling, Dimensional Modeling.

Data Vault – это подход проектирования хранилища данных, который стал эволюцией модели звезды и 3 нормальной формы (3НФ). Модель включает в себя три основных компонента: хаб, ссылка, спутник [1]. Такой подход адаптирует нашу БД под возможность расширения, снижает дублирование информации [2]. В то же время большое число таблиц приводит к росту JOIN-ов в запросах, а также к растущей сложности при проектировании БД.

Anchor Modeling – это один из гибких подходов к проектированию хранилищ данных. Модель включает в себя Якоря, Атрибуты, Связи, а также Узел [3]. Данный подход хорошо адаптирован под растущие проекты, экономит место, поскольку не содержит pull-данных и дублирующих данных [4]. В то же время, данный подход крайне сложен в реализации, и часто его внедрение может быть избыточным. Как и Data Vault, данная модель требует большого числа JOIN-ов для выполнения запросов.

Dimensional Modeling – это традиционный стиль проектирования хранилища данных. Она обладает простой структурой, которая включает таблицу факта для того или иного процесса, а также зависимые от неё таблицы измерений. Среди плюсов модели можно выделить её простую реализацию, простую структуру данных, которая будет понятна и сторонним пользователям. В то же время модель плохо подходит для растущих или крупных проектов.

Для сравнения методологий между собой будет вычисляться интегральный показатель качества. Он вычисляется как сумма значений всех показателей качества, умноженных на их весовой коэффициент. В качестве показателей качества были выделены данные из таблицы 1.

Таблица 1 – Показатели качества

Показатель	Весовой коэффициент
Масштабируемость	0.4
Дублирование данных	0.3
Лёгкость реализации	0.3

В результате была получена таблица 2, в которой оценены те или иные показатели качества для каждой из методологий.

Таблица 2 – Значения для каждой из методологий

Методология	Масштабируемость	Дублирование	Сложность
Data Vault	0.7	0.5	0.5
Anchor Modeling	0.8	0.6	0.1
Dimensional Modeling	0.5	0.4	0.6

Были получены интегральные показатели качества для каждой из методологий. Результаты были приведены в таблице 3.

Таблица 3 – Значения показателя для каждой из методологий.

Методология	Показатель
Data Vault	0,58
Anchor Modeling	0,53
Dimensional Modeling	0,5

Результаты исследования подтверждают, что для реализации базы данных приложения подходит модель Data Vault. Выбор данной методологии обусловлен её способностью к доработке хранилища, снижения дублирующей информации, а также приведением таблиц к ЗНФ. В то же время, было выделено, что в данной модели порождается большое число таблиц, что приводит к большой затратности по времени при написании запросов, а также проектировании хранилища.

ЛИТЕРАТУРА

1. Конаков П.О. РАЗРАБОТКА МЕТОДОЛГИИ УНИФИКАЦИИ ЗАГРУЗКИ ДАННЫХ В ХРАНИЛИЩЕ ДАННЫХ МОДЕЛИ DATA VAULT // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2023. - 2023. - С. 78-84.
2. Павлович Н.В. DATA VAULT: ПРЕИМУЩЕСТВА И НЕДОСТАТКИ // Новые информационные технологии в научных исследованиях НИТ-2021. материалы XXVI Всероссийской научно-технической конференции студентов, молодых ученых и специалистов.. - Рязань: 2021. - С. 35-36.
3. Дзюба А.Г., Рочкина Е.А., Ефимова Е.В. СРАВНЕНИЕ ГИБКИХ МЕТОДОЛОГИЙ ПРОЕКТИРОВАНИЯ DWH (DATA VAULT И ANCHOR MODELING) // НОВЫЕ НАПРАВЛЕНИЯ НАУЧНОЙ МЫСЛИ. Сборник научных статей Национальной (Всероссийской) научно-практической конференции. - Ростов-на-Дону: 2022. - С. 533-535.
4. Жернаков Р.С., Кислицын Е.В. ХАРАКТЕРИСТИКА И ОСОБЕННОСТИ МЕТОДОЛОГИИ ANCHOR MODELING // Современные тенденции развития науки и технологий. - 2016. - С. 33-36.
5. Ерошенко Я.Б., Самхарадзе К.К. ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ ПУТЕМ НОРМАЛИЗАЦИИ БАЗ ДАННЫХ // Научный аспект. - 2017: 2017. - С. 104-117.

УДК 004.453

А.А. Лихачева (4 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ АНАЛИЗА КОМПЬЮТЕРНЫХ ИГР ДЛЯ ИХ РАНЖИРОВАНИЯ И СОСТАВЛЕНИЯ РЕКОМЕНДАЦИЙ ПО УЛУЧШЕНИЮ

На современном рынке компьютерных игр у разработчиков возникает потребность в инструментах улучшения пользовательских качеств реализуемых продуктов с целью увеличения их коммерческой успешности [1, 2]. В связи с этим актуальной становится задача разработки специализированных аналитических систем, способных на основе описательных характеристик игры прогнозировать её потенциальный успех на рынке.

Таким образом, *цель исследования* — изучение существующих методов оценки и рейтингования компьютерных игр с акцентом на использование технологий машинного обучения, анализа текстов и построения прогнозов. Для достижения цели решены следующие задачи:

1. Анализ научная литература, публикации и инструменты, посвященные методам оценки в индустрии компьютерных игр, а также разработке рекомендательных систем.
2. Выбор и подготовка набора данных о компьютерных играх.
3. Реализация прогностической модели для рейтингования компьютерных игр.
4. Выбор технологий и реализация веб-приложения для визуализации результатов анализа коммерческой успешности компьютерных игр.

5. Тестирование и апробация разработанной системы.

Обзор существующих систем ранжирования выявил тенденцию к интеграции в них гибридных методов, использующих комбинации методов глубокого обучения и свёртки критериев. К достоинствам таких подходов относятся высокая точность и адаптивность рекомендаций. Среди недостатков можно выделить высокое потребление ресурсов и сложности, связанные с обработкой и анализом текстовых данных. На рисунке 1 представлена диаграмма, иллюстрирующая предложенный алгоритм оценки и рейтингования видеоигр.

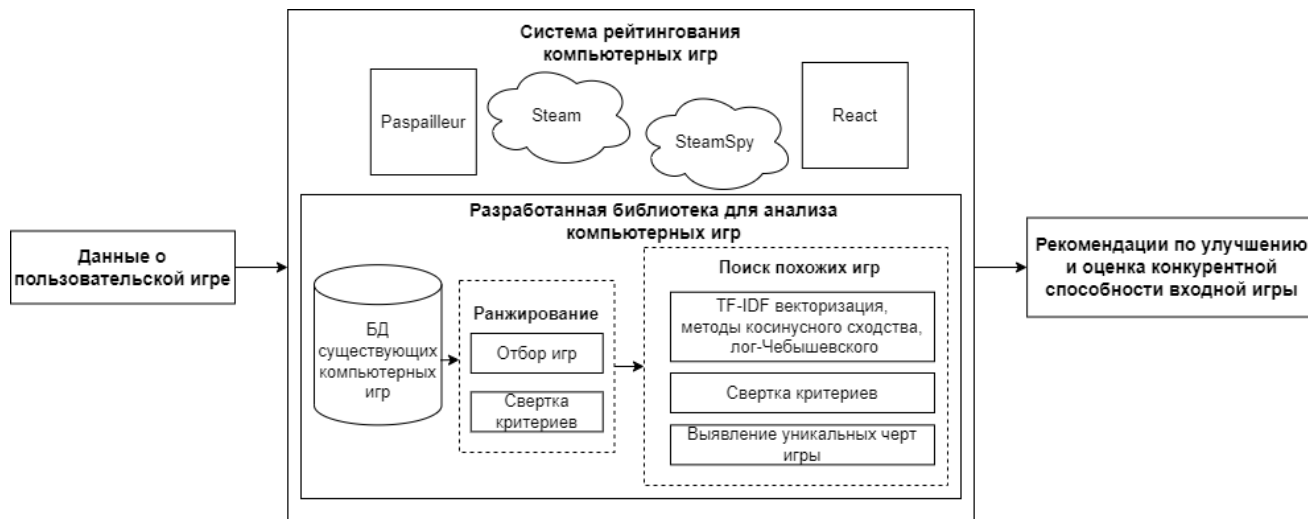


Рисунок 1 – Архитектура системы

Разработанная система классификации видеоигр предоставляет гибридный подход, включающий анализ отзывов и использование глубокого обучения. В рамках данного подхода предложен метод, учитывающий следующие аспекты: для обработки текстовых данных используется косинусное сходство на основе TF-IDF векторов, для числовых – лог-чебышевская аппроксимация [3], которая вычисляет максимальное отклонение между логарифмами нормализованных значений. Для интеграции различных критериев используется теория важности критериев и метод взвешенной суммы [4].

Как показывают результаты исследований, жанровая специфика, описание и цена компьютерной игры определяют текстовые характеристики отзывов, которые связаны с востребованностью продукта – положительные отзывы коррелируют с ростом продаж независимо от сегмента игры [5]. Именно поэтому архитектура системы предполагает анализ данных о введённой игре (жанры, механики, описание и т.д.) и её сравнение с существующими играми на основе контентных характеристик (контентная фильтрация). Затем система анализирует отзывы и метрики наиболее похожих игр (рейтинги, стоимость, аудитория и другие параметры) и вычисляет оценку их возможного успеха на рынке. Для хранения данных выбрана СУБД MongoDB, что обусловлено её способностью эффективно работать с неструктурированными данными, такими как описания игр и отзывы пользователей, которые хранятся в формате JSON-объектов.

Итоговая система способна предоставить список рекомендаций по схожим играм, а также числовую оценку коммерческого успеха, рассчитанную на основе контентного сходства и рыночных факторов. Важной особенностью системы является её способность к диверсификации рекомендаций, что достигается за счёт комбинации методов сходства и различия.

Реализованный комплексный подход к рейтингу компьютерных игр сочетает в себе использование алгоритмов рекомендательных систем, анализа текстовых данных и машинного обучения, не только оценивая коммерческий потенциал продукта, но и формируя стратегические рекомендации для его улучшения. Система учитывает сходства с другими играми для привлечения аудитории, заинтересованной в аналогичных продуктах, и выявляет уникальные черты разрабатываемой игры. Результат рекомендаций направлен на

минимизацию рисков для разработчиков, маркетологов и инвесторов за счёт предоставления обоснованных прогнозов и предложений по повышению конкурентоспособности продукта на рынке.

ЛИТЕРАТУРА

1. Condensing Steam: Distilling the Diversity of Gamer Behavior / M. O'Neill [и др.] // Proceedings of the 2016 Internet Measurement Conference. — 2016. — URL: <https://api.semanticscholar.org/CorpusID:15134651>.
2. The influencing factors of game brand loyalty / X. Zhao [и др.] // Heliyon. — 2024. — Т. 10. — e31324. — DOI 10.1016/j.heliyon.2024.e31324.
3. Solving multicriteria problems of rating alternatives based on pairwise comparisons / Krivulin [и др.]. — 2024. — DOI 10.32603/2071-2340-2024-2-5-29. — URL: <http://cte.eltech.ru/ojs/index.php/kio/article/view/1833>.
4. Подиновский В. В. Идеи и методы теории важности критериев в многокритериальных задачах принятия решений. — 1-е изд. — Москва, 2019. — 102 с. — (Дата обращения: 15.01.2025).
5. Wang D., Moh M., Moh T.-S. Using Deep Learning and Steam User Data for Better Video Game Recommendations // . — Tampa, FL, USA: Association for Computing Machinery, 2020. — С. 154—159. — (Сер.: ACMSE '20). — DOI 10.1145/3374135.3385283. — URL: <https://doi.org/10.1145/3374135.3385283>.

УДК 004.031.43:004.032.26

Д. Майстренко (4 курс бакалавриата),
О.Г. Малеев, к. т. н., доцент

ВИРТУАЛЬНЫЙ ПОМОЩНИК ДЛЯ ЗАКАЗА ПРОДУКТОВ ПИТАНИЯ

Сегодня проблема рационального использования продуктов питания и сокращения пищевых отходов становится всё более значимой. Умение правильно организовать запас продуктов в холодильнике становится важной задачей для повышения уровня осознанного потребления. Традиционные методы контроля запасов, основанные на контроле «вручную», не обеспечивают достаточного уровня контроля, испытывают негативное влияние человеческих факторов, таких как невнимательность, а также требуют выделения отдельных временных ресурсов, которых в современном мире у человека становится всё меньше.

Одним из способов решения данной проблемы является создание частично автоматизированной системы по контролю за продовольственными запасами. В этом контексте особое внимание необходимо уделить разработке мощной и надёжной backend-системы, которая обеспечит все необходимые функциональности для пользователей, а также будет служить базой для создания различных frontend-решений на её основе.

Предлагаемое решение использует программное обеспечение, написанное с целью частично автоматизировать процесс контроля за продовольственными товарами. Обработка информации осуществляется посредством серверного решения для отслеживания имеющихся запасов, а также нейросетью, прогнозирующей необходимые будущие закупки продуктов на основе анализа текущего потребления и учёта предстоящих значимых событий, таких как праздники.

Архитектура предлагаемой системы изображена на рисунке 1 и включает в себя следующие компоненты:

1. Программное обеспечение, написанное при помощи фреймворка FastAPI [2], обрабатывающее и возвращающее обработанные данные в Restful формате (включая, но не ограничиваясь) о наличии продовольственных запасов и их количестве, данные о конкретных типах запасов, запросы на пополнение запасов, прогнозы будущих закупок и др.
2. Реляционную базу данных PostgreSQL [1], хранящую информацию о имеющихся продовольственных запасах и запросов на их пополнение.

3. Нейросетевой модуль, прогнозирующий будущие закупки на основании имеющейся в базе данных информации о потреблении и количестве значимых дат на прогнозируемый период.

Передача данных между клиентом и сервером осуществляется посредством архитектурного стиля взаимодействия REST и спецификации OpenAPI [3], что позволяет на основании данного backend-решения независимое создание различного рода frontend-решений, не привязываясь к особенностям внутренней реализации приложения.

Архитектура системы, разработанная с учетом будущих потребностей и расширяемости, предоставляет не только возможности для надежного контроля за продовольственными запасами, но и открывает новые перспективы в сфере интеграции с онлайн-сервисами по заказу продуктов. В последующих веках появляется возможность подключения к магазинам, позволяющим оформлять онлайн-доставку. Как итог, пользователи смогут не только отслеживать имеющиеся продукты, но также и автоматически или вручную заказывать необходимое через интегрированные системы доставки без необходимости

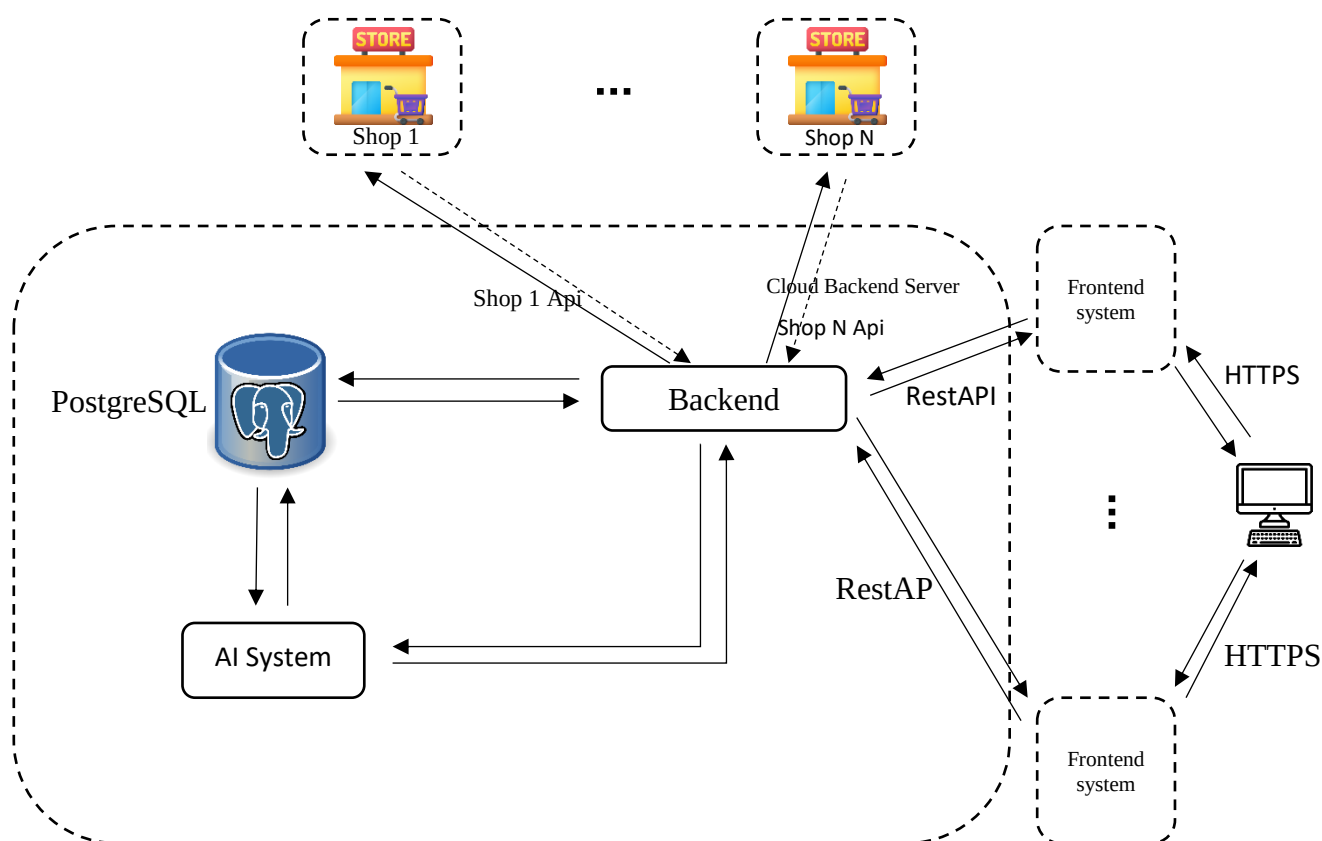


Рисунок 1 – Архитектура системы

Таким образом, основной целью данной работы является оптимизация расходов на продовольственные запасы, путём частичной автоматизации контроля и, в перспективе, приобретения продуктов, а также снижения влияния человеческого фактора, что позволит значительно снизить временные затраты на пополнение провизии.

ЛИТЕРАТУРА

1. Рогов Е. В. PostgreSQL 16 изнутри – М.: ДМК Пресс, 2024. – 664 с.
2. Любанович Б. FastAPI: веб-разработка на Python – Алматы: Sprint Book, 2024. – 282 с.
3. Гоф Дж. Проектирование архитектуры API – Астана: АЛИСТ, 2024. – 288 с.

ИНТЕЛЛЕКТУАЛЬНАЯ ПЛАТФОРМА МОНИТОРИНГА ПОЧВ И ВОДНЫХ РЕСУРСОВ
В СЕЛЬСКОМ ХОЗЯЙСТВЕ

Постановка задачи. Эффективное земледелие требует постоянного контроля влажности почвы и качества воды на сельскохозяйственных угодьях [1]. Традиционные методы мониторинга (полевые замеры, лабораторный анализ) недостаточно оперативны и точны, что затрудняет своевременное принятие агротехнических решений. Возрастающая роль технологий точного земледелия и цифровизации сельского хозяйства стимулирует разработку автоматизированных систем мониторинга окружающей среды [2].

Поэтому возникает необходимость разработки интеллектуальной системы, способной в реальном времени контролировать ключевые параметры почвы и воды, прогнозировать их изменения и предупреждать негативные последствия климатических и антропогенных факторов. Постановка задачи сводится к разработке адаптивной системы мониторинга, обеспечивающей аграриев достоверной информацией для оптимизации полива и улучшения урожайности при минимизации затрат ресурсов.

Краткое описание проекта. Разрабатываемая интеллектуальная система мониторинга, включает сеть сенсорных узлов, канал передачи данных и аналитический модуль. Цель разработки – обеспечить точный и оперативный контроль состояния почвы (в частности, влажности, концентрации солей и температуры) с применением методов искусственного интеллекта для прогнозирования состояния почвы. На поле будет развернута сенсорная инфраструктура: датчики влажности, температуры и концентрации солей, установленные в различных зонах участка. Собранные данные проходят предварительную обработку (фильтрацию шумов, калибровку) и по беспроводному каналу поступают на серверный аналитический модуль. Архитектура системы, показанная на рисунке 1, модульная и масштабируемая, что позволяет гибко наращивать число датчиков и интегрировать систему с существующими информационно-управляющими платформами агропредприятия.

Прогноз состояния почвы основан на интеллектуальных методах анализа данных. В серверном модуле применяется обученная нейронная сеть для прогнозирования динамики показателей почвы на основе поступающих временных рядов данных. Использование методов глубокого обучения [3] позволяет выявлять скрытые зависимости в данных и с высокой точностью предсказывать, например, снижение влажности почвы до критического уровня. Это даёт возможность заблаговременно принимать меры – регулировать режим полива или вносить корректировки в агротехнологии. Кроме того, будут реализованы интеллектуальные алгоритмы обнаружения аномалий, сигнализирующие о резких отклонениях (например, о резком засолении или переувлажнении участка). Интеллектуальная система адаптируется к изменяющимся условиям, уточняя прогнозные модели на основе данных датчиков. Предложенная архитектура отличается высокой надежностью, гибкостью конфигурации и совместимостью с технологиями «умного» земледелия. Это позволяет интегрировать мониторинг в единую цифровую экосистему управления фермерским хозяйством [4].

Оценка характеристик демонстрационной версии. Разработан и протестирован опытный образец системы. Демонстрационная версия включала набор из нескольких беспроводных сенсорных модулей, объединенных в сеть, и облачный сервис аналитики. Испытания проводились на учебно-опытном участке: датчики непрерывно измеряли влажность и температуру почвы. Результаты тестирования подтвердили работоспособность предложенной архитектуры в реальном времени. Система обеспечила оперативную передачу данных (задержка не превышала нескольких секунд) и высокую точность измерений влажности (погрешность не более 2–3%). В настоящее время разрабатывается нейросетевой модуль, который будет предсказывать тенденции изменения влажности почвы на 6–8 часов

вперед, что позволит своевременно среагировать на надвигающееся пересыхание участка. Таким образом, будет продемонстрирована эффективность применения методов машинного обучения для агромониторинга: по сравнению с традиционными подходами точность прогноза состояния почвы возросла, а риск упущенных критических ситуаций снизится. Архитектура системы показала высокую масштабируемость – при подключении дополнительных датчиков и узлов производительность системы существенно не снизилась.

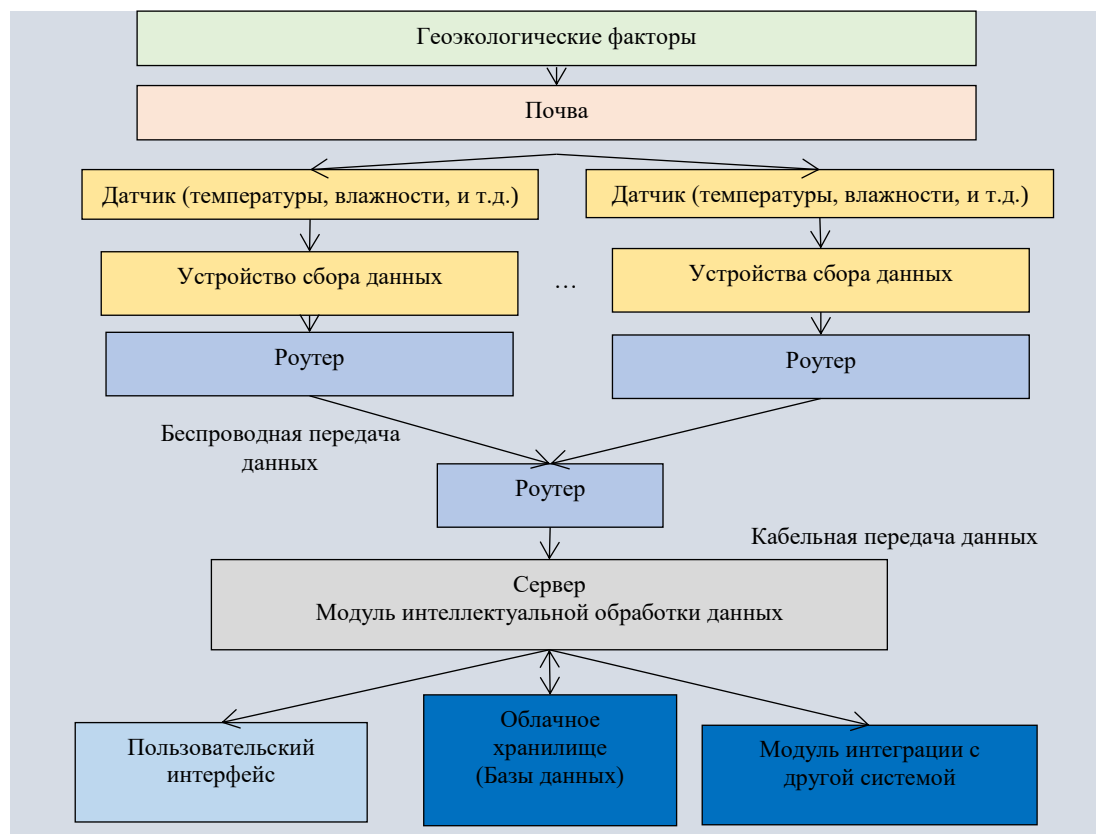


Рисунок 1 – Архитектура интеллектуальной системы мониторинга состояния почвы

В целом, предложенная интеллектуальная система мониторинга состояния воды и почв продемонстрировала потенциал для повышения эффективности сельского хозяйства. Ожидается, что её внедрение позволит оптимизировать график полива и внесения удобрений, сократить непроизводительные потери воды до 20–30%, а также повысить урожайность за счёт более точного соблюдения агротехнологических параметров. Реализация таких систем способствует повышению эффективности использования ресурсов и устойчивости агроэкосистем в условиях изменяющегося климата [5]. Работа над проектом продолжается, планируется расширение функциональности системы и проведение длительных полевых испытаний для окончательной количественной оценки её экономической эффективности.

ЛИТЕРАТУРА

1. Балабанов В.И., Железова С.В., Березовский Е.В., Беленков А.И., Егоров В.В. «Навигационные системы в сельском хозяйстве. Координатное земледелие». Под общ. ред. проф. В.И. Балабанова. – М.: Изд-во РГАУ-МСХА им. К.А. Тимирязева, 2013. – 143 с.
2. Бородина О.Б., Гвоздева О.В., Синица Ю.С., Колбнева Е.Ю. Цифровое сельское хозяйство: настоящее и будущее (обзор международной практики) // Московский экономический журнал. – 2021. – № 1. – С. 45–60.
3. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016. – 775 p.
4. Добровлянин В.Д., Антинескул Е.А. Цифровизация сельского хозяйства: текущий уровень цифровизации в Российской Федерации и перспективы дальнейшего развития // Экономика сельского хозяйства России. – 2021. – № 3. – С. 29–34.
5. Якушев В.В. «Точное земледелие: теория и практика». – СПб.: ФГБНУ АФИ, 2016. – 364 с.

РАЗРАБОТКА ВЕБ-СЕРВИСА ДЛЯ АВТОМАТИЗИРОВАННОГО СОЗДАНИЯ
DOCKER-ОБРАЗОВ

Технологии контейнеризации, в частности Docker, активно применяются в современной разработке программного обеспечения для упрощения развертывания, масштабируемости и обеспечения воспроизводимости сред выполнения. Однако, Docker требует от пользователей определенных знаний синтаксиса команд и умения управлять средой выполнения на уровне операционной системы. Для того чтобы создать Docker-образ, необходимо вручную писать Dockerfile, настраивать зависимости, управлять контейнерами и выполнять команды через терминал [1], что в различных ситуациях может занимать много времени.

На сегодняшний день для работы с Docker существует ряд инструментов, таких как Docker Desktop, Docker Hub, а также CI/CD-платформы, включая GitHub Actions и GitLab CI/CD. Эти решения позволяют создавать, хранить и управлять контейнерами, но требуют установки Docker и дополнительных инструментов на локальном компьютере. Это не всегда удобно, особенно в случаях, когда использование контейнеров требуется лишь эпизодически или с разных устройств.

Таким образом, целью работы является создание веб-сервиса, предоставляющего удобный графический интерфейс, который автоматизирует процесс создания Docker-образов, позволяя пользователям загружать файлы, выбирать параметры и зависимости и автоматически генерировать Dockerfile и разворачивать контейнер, тем самым упрощая процесс и делая его доступным для более широкой аудитории. Для достижения поставленных целей необходимо сформулировать требования.

Функциональные требования:

- Загрузка исходных файлов проекта через веб-интерфейс.
- Автоматическая генерация Dockerfile для выбранного проекта на основе загруженных данных [2].
- Возможность выбора стека технологий (например, версию интерпретатора или компилятора и необходимые библиотеки и зависимости).
- Инициализация процесса сборки Docker-образа через Docker API с последующим отслеживанием его статуса.
- Возможность скачать Docker-образ в формате .tar.
- Развертывание контейнера на сервере после создания образа.
- Хранение информации о пользователях и их проектах в базе данных.

Нефункциональные требования:

- Доступный и интуитивно понятный интерфейс.
- Готовность системы к масштабированию для работы с большим количеством пользователей.
- Высокая скорость обработки запросов и генерирования образов.

Система построена по клиент-серверной архитектуре и включает в себя несколько основных компонентов. Frontend реализован с использованием React.js — JavaScript-библиотеки для создания пользовательских интерфейсов. Она оптимизирована для создания динамичных и высокоотзывчивых интерфейсов [3]. Для серверной части был выбран FastAPI. Это высокопроизводительный фреймворк для разработки API на языке программирования Python [4]. Он позволит эффективно взаимодействовать с Docker API и поддерживать асинхронное выполнение операций. Для работы с базой данных используется PostgreSQL [5], так как она является высокопроизводительной СУБД и предоставляет возможности для

масштабирования при последующей работе с данными пользователей и их проектами. Архитектура системы представлена на рисунке 1

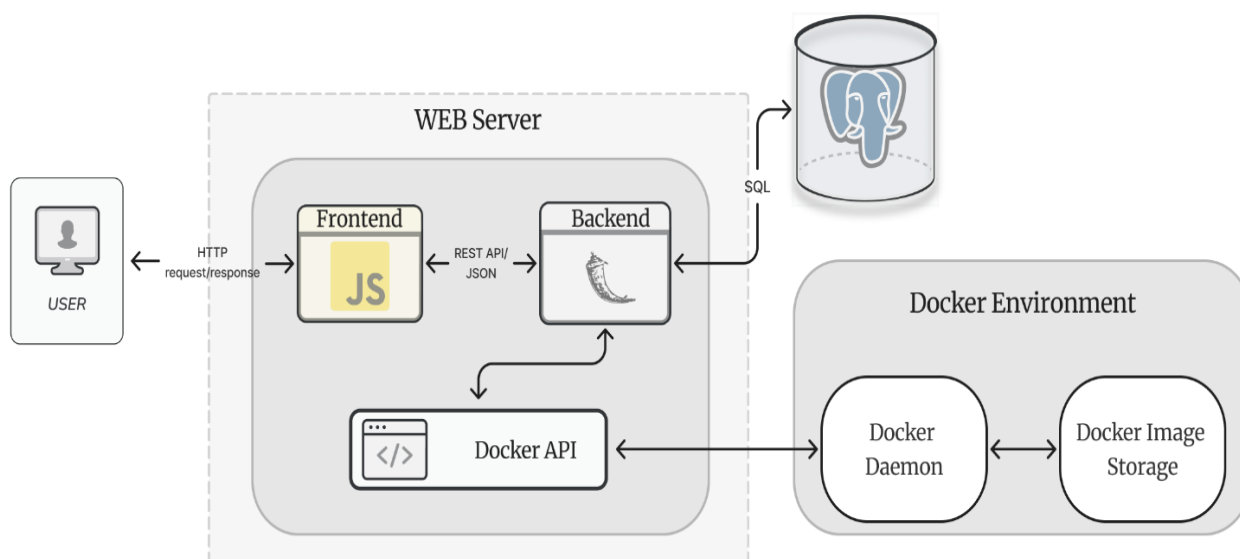


Рисунок 1 – Архитектурная модель системы

Frontend взаимодействует с Backend через REST API, передача данных осуществляется по протоколу HTTPS в формате JSON. Сервер обрабатывает пользовательские запросы и взаимодействует с PostgreSQL через ORM SQLAlchemy. Коммуникация с Docker API осуществляется через локальный Unix-сокет (`unix:///var/run/docker.sock`) [6]. Так сервер напрямую взаимодействует с Docker Daemon для управления контейнерами и отслеживания процесса сборки в реальном времени и получает результат обратно через тот же сокет.

Процесс выполнения запросов организован следующим образом. Пользователь загружает исходные файлы через веб-интерфейс, отправляя POST-запрос на сервер. После этого сервер предлагает пользователю выбрать параметры контейнера. На основе этих параметров сервер генерирует Dockerfile. Далее Backend отправляет команду `docker build` через Docker API, взаимодействуя с Docker Daemon [7]. В процессе сборки используется Docker Image Storage, куда будет сохраняться готовый образ, который может быть использован для дальнейших действий. После завершения сборки сервер сохраняет информацию о созданном образе в базе данных и предоставляет пользователю возможность либо скачать готовый Docker-образ в формате `.tar`, либо развернуть контейнер на сервере. Развертывание контейнера на сервере осуществляется с помощью команды `docker run`, которая создает контейнер из образа, запускает его и возвращает ссылку для доступа к контейнеру через веб-интерфейс.

ЛИТЕРАТУРА

1. Документация Docker API. [Электронный ресурс]. Режим доступа: <https://docs.docker.com/engine/api/v1.41/>
2. Kim G., Debois P., Willis J., Humble J. The DevOps Handbook. IT Revolution Press, 2016.
3. Документация React. [Электронный ресурс]. Режим доступа: <https://react.dev/learn>
4. Документация FastAPI. [Электронный ресурс]. Режим доступа: <https://fastapi.tiangolo.com/>
5. Документация PostgreSQL. [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/docs/>
6. Документация Docker. [Электронный ресурс]. Режим доступа: <https://docs.docker.com/>
7. Гош С. Docker без секретов: Пер. с англ. — СПб.: БХВ-Петербург, 2023. — 224 с.

ПРОГРАММНАЯ СИСТЕМА АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ВЗАИМОДЕЙСТВИЯ С ПОСТАВЩИКАМИ ИНФОРМАЦИОННО- ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

В современном бизнесе поставщики информационных технологий критически важны для повышения эффективности и конкурентоспособности. Эффективное взаимодействие с ИТ-поставщиками необходимо для успешного функционирования организаций. Существующие методы учета, такие как Excel, имеют недостатки: низкая автоматизация, высокая вероятность ошибок и трудоемкость.

В настоящее время обработка предложений о сотрудничестве осуществляется следующим образом: ответственный сотрудник получает входящее коммерческое предложение по электронной почте, проводит его анализ и вручную вносит данные о компании-партнере в Excel-файл, что создает ряд серьезных проблем - отсутствие синхронизации при пересылке информации приводит к быстрому устареванию данных, незащищенный формат хранения создает риски утечки конфиденциальной информации, а полная зависимость от одного специалиста не только создает организационные риски, но и затрудняет масштабирование процесса при увеличении объема поступающих предложений.

В связи с этим, создание автоматической системы обеспечит централизованный доступ к информации, повысит прозрачность и упростит принятие решений.

Целью работы является повышение эффективности учета и управления взаимодействием поставщиками ИТ решений за счет подхода автоматизации процессов обработки предложений, хранения данных о поставщиках и анализа их деятельности, реализованного в веб-приложении.

Задачи, необходимые для достижения цели:

- Провести анализ текущих процессов и требований к системе
- Представить сравнительный анализ подходов автоматизации процессов обработки предложений (Excel, 1С и другие)
- Предложить программную архитектуру системы автоматизации управления
- Реализовать предложенный подход в веб приложении с использованием стека технологий (JS, React [1], Electron [2], GoLang, MongoDB [3], Docker [4])
- Провести экспериментальные исследования, доказывающие эффективность системы.

Первоначальная реализация системы предусматривает использование десктопного приложения, которое обеспечит удобный доступ к основным функциям и возможностям системы. Но благодаря выбранному стеку технологий, при запросе от заказчика, приложение можно будет представить в виде сайта. Разработка приложения осуществляется с использованием современного JavaScript в сочетании с популярным фреймворком React, что гарантирует высокую производительность и отзывчивость интерфейса. Для создания десктопной версии применяется фреймворк Electron, который позволяет использовать веб-технологии для разработки нативных приложений под различные операционные системы. Выбор данных технологий обусловлен их способностью обеспечивать эффективное взаимодействие с пользователем и быструю обработку данных.

Серверная часть системы построена на микросервисной архитектуре [5] с использованием языка программирования Go, что обеспечивает высокую производительность и эффективное управление ресурсами. Для контейнеризации и оркестрации сервисов применяется Docker, позволяющий создавать изолированные окружения и упрощающий процесс развертывания. В качестве системы управления базами данных выбрана MongoDB, которая предоставляет гибкие возможности для хранения и обработки данных. Для обеспечения безопасности реализована JWT-аутентификация, а интеграция с электронной

почтой осуществляется через SMTP-протокол, что позволяет организовать надежную систему обмена сообщениями. Такой технологический стек и архитектурное решение обеспечивают масштабируемость, надежность и высокую производительность всей системы.

Архитектура системы управления поставщиками представлена на рисунке 1[6]

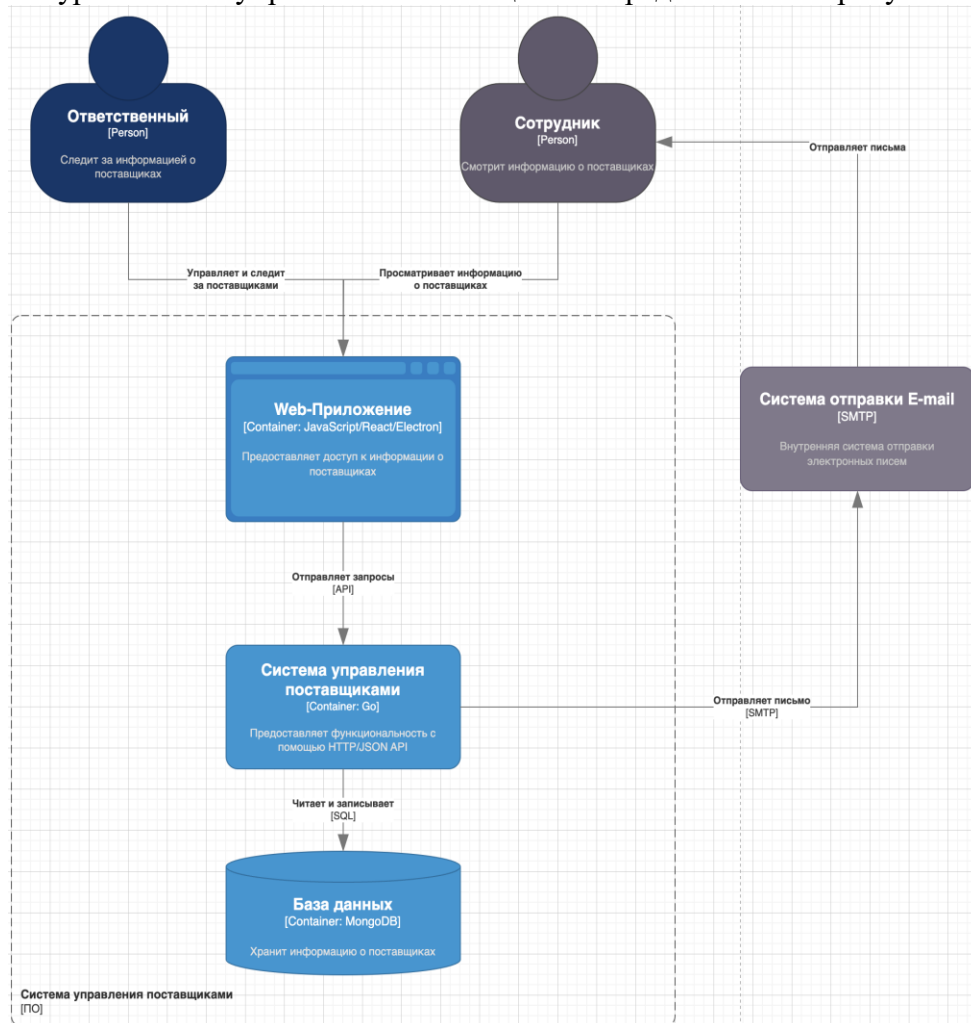


Рисунок 1 – Архитектура системы

Таким образом, разработанная система автоматизации управления поставщиками ИТ-решений представляет собой эффективное решение, отвечающее современным требованиям бизнеса. Внедрение данной системы позволило организации оптимизировать процессы взаимодействия с поставщиками, повысить качество принимаемых решений и обеспечить надежное хранение информации о партнерских отношениях.

ЛИТЕРАТУРА

1. React. The library for web and native user interfaces [Электронный ресурс] <https://react.dev/> (Дата обращения 14.02.2025)
2. ElectronJS [Электронный ресурс] <https://www.electronjs.org/> (Дата обращения 14.02.2025)
3. MongoDB: The Developer Data Platform [Электронный ресурс] <https://www.mongodb.com/> (Дата обращения 20.02.2025)
4. James Turnbull. "The Docker Book" Version v17.03.0 - 2017
5. Что такое микросервисная архитектура. [Электронный ресурс] URL: <https://selectel.ru/blog/what-is-microservice-architecture/> (Дата обращения 20.02.2025)
6. C4Models. Diagrams. [Электронный ресурс] URL: <https://c4model.com/diagrams> (Дата обращения 26.02.2025)

РАЗРАБОТКА СЕРВЕРНОЙ СОСТАВЛЯЮЩЕЙ ПРОГРАММНОГО
ПРОДУКТА POLYHABR

В процессе обучения каждый сталкивался с проблемой того, что не знал, как сделать или с чего начать выполнение курсовой, лабораторной или научной работы. Данную проблему возможно решить путем использования поисковой системы Интернета, однако часто искомый ответ там отсутствует. Обращение за помощью к сокурсникам тоже не всегда эффективно: одни испытывают неловкость при просьбах, другие могут отказать. Личный опыт подобных ситуаций побудил меня к разработке решения этой проблемы.

Основная цель проекта - разработка веб-платформы «Polyhabr», предназначенной для поддержки студентов в выполнении академических работ. Сервис предоставит доступ к архивным материалам прошлых лет, которые послужат ориентиром для решения текущих задач. Пользователи смогут делиться своими наработками, участвовать в профессиональном диалоге через комментарии к публикациям, а также оценивать контент и сохранять его в персональные закладки. Функционал включает систему регистрации, поиск по заголовкам с ранжированием результатов, фильтрацию материалов по дате добавления и популярности.

Внедрение этого инструмента позволит оптимизировать учебный процесс: сократить временные затраты на поиск информации, оперативно решать возникающие вопросы и повышать уровень академической подготовки. Уникальность платформы заключается в её узкой направленности — все материалы связаны с СПбПУ Петра Великого, что обеспечивает релевантность контента исключительно для учащихся этого вуза. Дополнительно будет интегрирован персонализированный алгоритм подбора материалов на основе предпочтений пользователя.

Техническая реализация включает серверную часть и базу данных [1]. Для backend-разработки выбран стек Kotlin + Spring Boot + JPA [2-5]. Kotlin, как статически типизированный язык для JVM, сочетает лаконичный синтаксис с повышенной безопасностью (включая обработку null-значений) и поддержкой функционального программирования, что упрощает поддержку кода. JPA автоматизирует взаимодействие с базой данных, минимизируя ручное написание SQL-запросов, а Spring Boot ускоряет настройку и деплой приложения.

В качестве СУБД используется PostgreSQL [6] - расширяемая объектно-реляционная система с открытым кодом, известная надёжностью и соответствием стандартам SQL. Её функционал позволяет реализовать сложные запросы и обеспечить целостность данных, что критично для образовательной платформы.

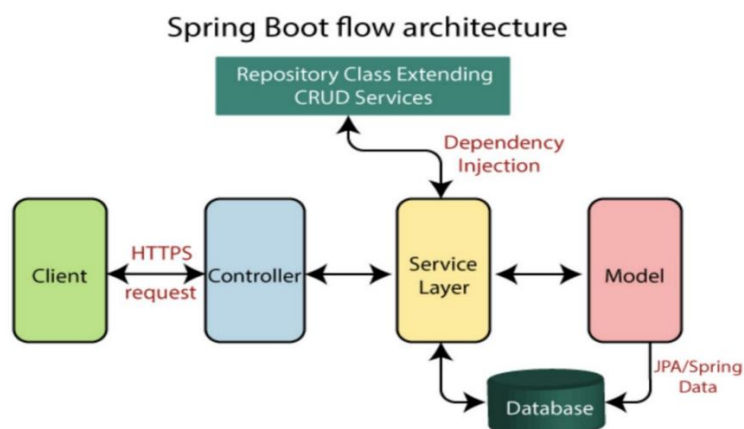


Рисунок 1 – Архитектура разрабатываемой системы

ЛИТЕРАТУРА

1. Технология программирования. Основные понятия и подходы [Электронный ресурс] Режим доступа: <https://studfile.net/preview/2953279/>
2. Java Persistence API (JPA) [Электронный ресурс] Режим доступа: <https://www.ibm.com/docs/en/was-liberty/base?topic=overview-java-persistence-api-jpa>
3. Kotlin. Basic syntax [Электронный ресурс] Режим доступа: <https://kotlinlang.org/docs/basic-syntax.html>
4. Spring Boot Reference Documentation [Электронный ресурс] Режим доступа: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
5. JPA - Краткое руководство [Электронный ресурс] Режим доступа: https://translated.turbopages.org/proxy_u/en-ru.ru.5bd8d95e-6419aea7-21845570-74722d776562/https/www.tutorialspoint.com/jpa/jpa_quick_guide.htm
6. PostgreSQL Documentation [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/>

УДК 004.453

П.А. Мезенцева (4 курс бакалавриата),
О.В. Прокофьев, ст. преподаватель

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ ПОДДЕРЖКИ МИГРАЦИЙ

Управление изменениями схемы базы данных представляет собой одну из ключевых задач в разработке информационных систем, особенно в контексте поддержки и обновления крупных и сложных баз данных. Этот процесс требует систематического подхода для отслеживания, контроля и применения изменений, что позволяет обеспечить целостность данных, согласованность и стабильную работу приложений.

Для автоматизации этого процесса используются системы управления миграциями [1]. Эти специализированные инструменты позволяют документировать историю изменений, управлять версиями схемы базы данных и обеспечивать воспроизводимость изменений в различных средах — от разработки до эксплуатации. Благодаря таким системам команды разработчиков могут согласованно вносить изменения, минимизируя риск конфликтов и ошибок при модификации структуры базы данных [2].

Одним из наиболее популярных инструментов для автоматизации управления миграциями является Liquibase [3]. Однако процесс написания миграций в Liquibase может сопровождаться рядом трудностей, поскольку поддержка нескольких форматов с уникальным синтаксисом типов изменений (change types) и набором обязательных атрибутов требует повышенного внимания от разработчика при отсутствии вспомогательных инструментов. Это увеличивает риск ошибок и время, затрачиваемое на создание и поддержку миграций [4].

Таким образом, целью работы является разработка способа для снижения трудоемкости и минимизации ошибок при написании миграций Liquibase путем разработки плагина для текстового редактора Visual Studio Code (VS Code).

Для достижения этой цели необходимо решение следующих задач:

1. Анализ существующих решений по автоматизации управления миграциями в Liquibase.
2. Предложение решения, учитывающего недостатки рассмотренных подходов.
3. Проектирование архитектуры и выбор инструментов для реализации.
4. Разработка плагина.
5. Демонстрация снижения трудоемкости при использовании разработанного решения.

В ходе сравнительного анализа существующих решений было выявлено несколько ключевых ограничений, которые необходимо устранить в разрабатываемом инструменте. Прежде всего, плагин должен быть интегрирован в среду разработки, а не функционировать как отдельное решение. Также важным требованием является поддержка intellisense-подсказок для всех форматов файлов описания миграций Liquibase (xml, json, yml), в отличие

от существующих инструментов, ориентированных на единственный формат. Кроме того, предлагаемые подсказки должны быть контекстно-зависимыми, адаптируясь к конкретной ситуации, а не ограничиваться статическими шаблонами.

В качестве основного инструмента разработки был выбран Visual Studio Code с использованием VS Code Extension API [6]. Для реализации плагина применяется среда выполнения Node.js, а в качестве языка программирования – JavaScript.

Основные компоненты приложения, использующего систему управления миграциями для отслеживания и применения изменений схемы базы данных, представлены на рисунке 1.

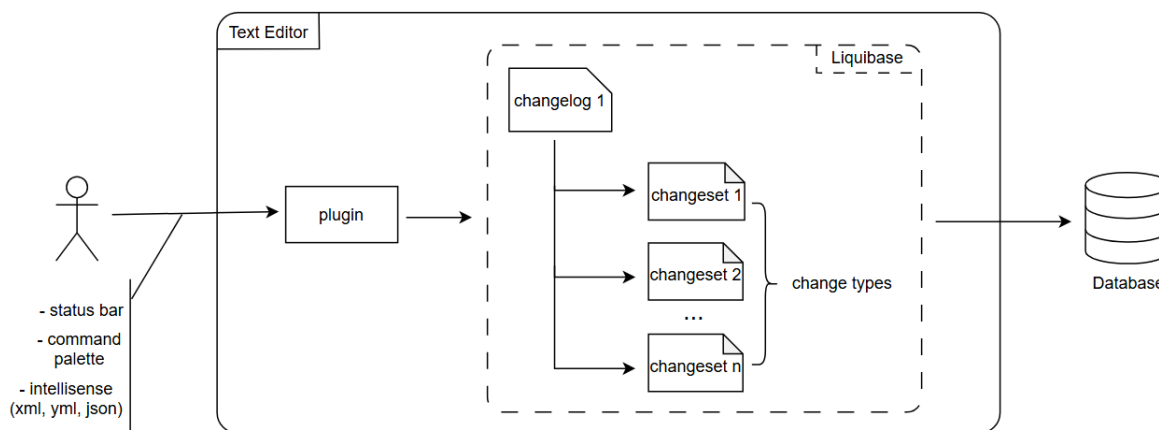


Рисунок 1 – Схема работы с Liquibase

Разрабатываемое решение взаимодействует с текстовым редактором через строку состояния, палитру команд и автодополнение кода, предоставляя пользователю вспомогательные механизмы для написания основных компонентов рассматриваемого инструмента – changelog и changeset с использованием доступных change types. [5]

Организация структуры миграций реализуется через иерархическую систему трех ключевых компонентов. Changelog представляет собой основной файл, содержащий полную историю изменений базы данных и определяющий порядок их выполнения. Changeset – это атомарная единица изменений с уникальным идентификатором, позволяющая отслеживать статус выполнения каждой миграции. Change types представляют predetermined операции для описания содержимого миграции, из которых состоят changesets и которые преобразуются в SQL-команды при применении к базе данных.

На текущий момент заложены основы для реализации автодополнений, обеспечивающих контекстно-зависимые подсказки при написании скриптов миграций, а также разработан кастомный статус-бар для отображения и выбора вспомогательных функций через палитру команд.

Таким образом, предложенный функционал позволит разделить ответственность: разработчик сможет заниматься логической структурой и содержанием миграций, в то время как технические аспекты будет брать на себя плагин.

ЛИТЕРАТУРА

1. Schema migration [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Schema_migration (дата обращения: 07.03.2025).
2. Bringing Databases Into the DevOps Fold [Электронный ресурс]. – Режим доступа: https://resources.liquibase.com/hubfs/Resources/eBook_DevOps_Bringing-Databases-DevOps-Fold.pdf (дата обращения: 07.03.2025).
3. Introduction to Liquibase [Электронный ресурс]. – Режим доступа: <https://docs.liquibase.com/concepts/introduction-to-liquibase.html> (дата обращения: 09.03.2025).
4. Jain M. Automated Liquibase Generator And Validator (ALGV) / M. Jain // ResearchGate. – 2020. – С. 250. – URL:

https://www.researchgate.net/publication/281814836_Automated_Liquibase_Generator_And_ValidatorALGV (дата обращения: 07.03.2025).

5. Concepts: Changelogs, Changeset and Change Types [Электронный ресурс]. – Режим доступа: <https://docs.liquibase.com/concepts/home.html> (дата обращения: 09.03.2025).
6. Visual Studio Code Extension API [Электронный ресурс]. – Режим доступа: <https://code.visualstudio.com/api> (дата обращения: 09.03.2025).

УДК 004.416

Л.А. Мясоедов (2 курс магистратуры),
А.И. Тышкевич, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ ДЛЯ ОФОРМЛЕНИЯ КОМИССИОННЫХ КРЕДИТНЫХ СТРАХОВЫХ И НЕСТРАХОВЫХ ПРОДУКТОВ ДЛЯ НУЖД БАНКА

Целью разработки является создание гибкой и масштабируемой системы, которая обеспечивает удобное оформление кредитных, страховых и нестраховых продуктов с автоматическим расчетом комиссий и дополнительной стоимости. Основные задачи включают:

- Разработка микросервисной архитектуры для гибкости и масштабируемости[2].
- Обеспечение удобного пользовательского интерфейса для оформления продуктов.
- Автоматизация расчета комиссий и дополнительных услуг.
- Повышение скорости обработки заявок и минимизация ошибок в расчетах.

Современный банковский рынок испытывает повышенный спрос на кредитные продукты, дополненные страховыми и нестраховыми услугами. Автоматизация процессов оформления и расчета комиссий позволяет:

- Снизить операционные затраты и исключить человеческие ошибки.
- Ускорить процесс обработки заявок, обеспечивая конкурентное преимущество.
- Повысить удовлетворенность клиентов за счет быстрого и удобного оформления продуктов.

Большинство существующих решений представляют собой монолитные системы с ограниченной масштабируемостью и фиксированным функционалом. Они требуют значительных затрат на адаптацию под нужды банка и не обладают гибкостью в настройке комиссий и дополнительных услуг. Это создает необходимость разработки более гибкой микросервисной архитектуры.

Архитектура. Проект построен на основе гексагональной архитектуры[3] (Hexagonal Architecture), которая обеспечивает четкое разделение слоев: Домен, Порты и Адаптеры. Это позволяет изолировать бизнес-логику от внешних фреймворков и технологий, что облегчает тестирование и повышает модульность компонентов.

Микросервисная архитектура проекта включает:

- Оформление продуктов.
- Расчет комиссий и дополнительной стоимости.
- Управление настройками комиссий.
- Авторизацию и аутентификацию.

Используется API Gateway для маршрутизации запросов и обеспечения масштабируемости и отказоустойчивости системы.

Используемые технологии:

- Язык программирования: Java 17+ [4,5].

Java 17+ был выбран благодаря его стабильности, производительности и поддержке современных функций, таких как записи (records) и улучшенная работа с памятью. Это позволяет писать более чистый и эффективный код, что особенно важно для сложных систем,

где требуется высокая надежность и производительность. Использование современных версий Java также обеспечивает долгосрочную поддержку и безопасность.

– Контейнеризация: Docker.

Docker используется для создания изолированных сред выполнения приложений, что упрощает развертывание и обеспечивает согласованность работы приложения на разных этапах разработки и в различных окружениях. Это позволяет избежать проблем, связанных с различиями в окружениях, и ускоряет процесс разработки и тестирования.

– Оркестрация контейнеров: Kubernetes [6].

Kubernetes обеспечивает автоматическое масштабирование, отказоустойчивость и управление контейнерами, что критически важно для высоконагруженных систем. Это позволяет системе адаптироваться к изменяющимся нагрузкам и минимизировать простои, обеспечивая высокую доступность и устойчивость к сбоям.

– Сборка и управление зависимостями: Gradle.

Gradle был выбран благодаря его гибкости и производительности. Он позволяет эффективно управлять зависимостями, автоматизировать сборку и тестирование, что сокращает время на подготовку релизов и уменьшает вероятность ошибок, связанных с ручным управлением зависимостями.

– Фреймворк: Spring (Spring Boot, Spring Data, Spring Security) [8].

Spring Boot используется для быстрого создания микросервисов с минимальной конфигурацией, что ускоряет разработку. Spring Data упрощает работу с базами данных, а Spring Security обеспечивает надежную защиту приложения. Это позволяет сосредоточиться на бизнес-логике, не тратя время на рутинные задачи.

CI/CD. Для непрерывной интеграции и развертывания используется:

- TeamCity — для автоматического тестирования и сборки артефактов.
- Git (Bitbucket) — для управления версиями кода и проведения Code Review.
- Развертывание в Kubernetes с использованием Docker-контейнеров. [9]

Система CI/CD обеспечивает высокую производительность и качество кода за счет автоматизации всех этапов разработки, тестирования и развертывания. [10] Система CI/CD на основе TeamCity и Git (Bitbucket) автоматизирует процессы сборки, тестирования и развертывания, что значительно сокращает время выпуска новых версий и уменьшает количество ошибок, вызванных человеческим фактором. Развертывание в Kubernetes с использованием Docker-контейнеров обеспечивает высокую отказоустойчивость и масштабируемость системы. В результате, объем ручного труда сокращается, а система становится более устойчивой к сбоям, что позволяет быстрее реагировать на изменения и обеспечивать стабильную работу приложения.

ЛИТЕРАТУРА

1. Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional.
2. Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
3. Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.
4. Bloch, J. (2018). Effective Java (3rd Edition). Addison-Wesley Professional.
5. Kelsey, C., Hightower, K., Burns, B. (2019). Kubernetes: Up and Running. O'Reilly Media.
6. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.
7. Pivotal Team. (2020). Cloud Native Java: Designing Resilient Systems with Spring Boot, Spring Cloud, and Cloud Foundry. O'Reilly Media.
8. Ford, N., Parsons, R., & Kua, P. (2017). Building Evolutionary Architectures: Support Constant Change. O'Reilly Media.
9. Turnbull, J. (2018). The Docker Book: Containerization is the New Virtualization. Independently published.
10. Hohpe, G., & Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional.

ТРЕКИНГ ПОСЕТИТЕЛЕЙ ОБЩЕСТВЕННЫХ ПРОСТРАНСТВ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

В условиях роста конкуренции в сфере торговли и необходимости повышения качества обслуживания актуальной задачей становится анализ поведения посетителей общественных пространств. Традиционные методы сбора данных, основанные на анкетировании и опросах, не обеспечивают достаточной точности и оперативности, а также требуют значительных временных и человеческих ресурсов. Такие подходы ограничены в возможностях проведения статистического анализа, что затрудняет выявление закономерностей в поведении клиентов и принятии обоснованных управленческих и логистических решений.

Одним из способов решения этой проблемы является создание автоматизированной системы мониторинга, позволяющей регистрировать перемещение клиентов в торговом пространстве и анализировать их маршруты и поведение на основе видеопотоков с камер наблюдения [1]. Важной составляющей данной системы должен быть модуль сбора статистики, который позволит анализировать потоки посетителей в разное время суток, выявлять наиболее популярные маршруты и оценивать эффективность рекламных зон внутри магазина.

Предлагаемая система трекинга общественных пространств использует видеопотоки с нескольких камер, установленных в торговых зонах. Обработка данных осуществляется с помощью нейронных сетей, которые позволяют детектировать посетителей на видеопотоке, отслеживать их перемещение и строить траектории движения на карте помещения, а также анализировать статистику поведения покупателей [3].

Архитектура системы представлена на рисунке 1 и состоит из следующих компонентов:

1. Локальный модуль сбора данных включает камеры, установленные в магазине, и локальный компьютер, передающий видеопотоки на сервер.
2. Серверный модуль обработки выполняет обработку видео с помощью нейросетевых алгоритмов, таких как YOLO для детекции людей, DeepSORT для трекинга и MMPose для анализа поз и выявления характерных движений посетителей [5]. Данные о перемещении посетителей и их траекториях сохраняются в базе данных PostgreSQL [2] для дальнейшего анализа и построения статистики.
3. Веб-интерфейс системы, разработанный на основе библиотеки Django (серверная часть) и React (клиентская часть), предоставляет возможность просмотра видеопотоков, отображения траекторий движения на интерактивной карте магазина и анализа статистики.

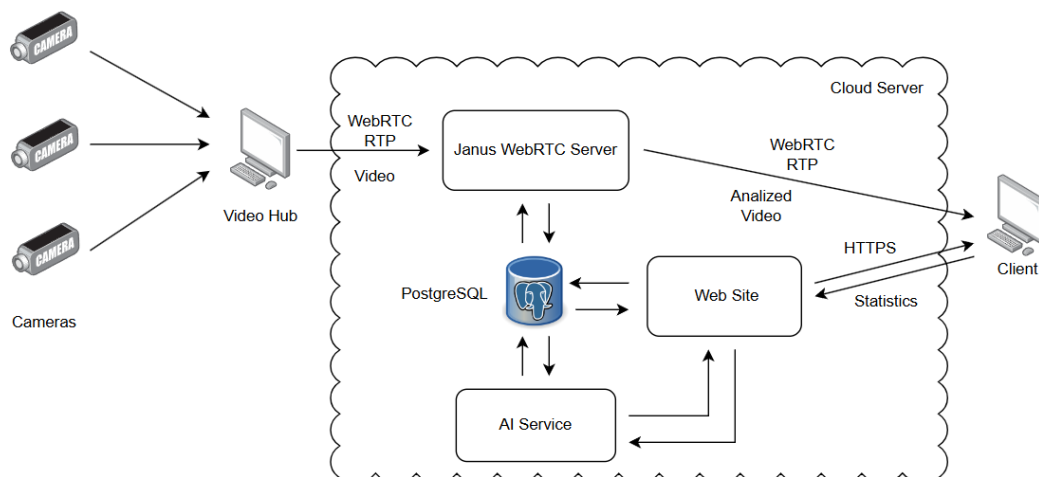


Рисунок 1 – Архитектура системы

Передача данных между компонентами осуществляется с использованием протокола WebRTC на основе RTP, что обеспечивает минимальную задержку и высокое качество видео [6]. Для управления потоками и маршрутизации видеоданных применяется сервер Janus WebRTC, обеспечивающий надежную передачу и поддержку многопользовательского взаимодействия [4].

Таким образом, основная цель данной работы состоит в повышении эффективности работы торгового бизнеса и оптимизации общественных пространств путем разработки автоматизированной системы мониторинга, сбора и анализа статистических данных. При этом особое внимание уделяется созданию качественного программного продукта с удобным и настраиваемым пользовательским интерфейсом, который может быть адаптирован к различным условиям планировки и успешно интегрирован в множество общественных пространств.

В ходе разработки была протестирована демонстрационная версия системы, включающая веб-интерфейс и локальный сервер обработки видео. Помимо основных функций детекции и мониторинга, система предоставляет статистику о посещаемости и передвижении клиентов.

Ключевыми характеристиками системы являются поддержка одновременной обработки более 24 видеопотоков с камер, что позволяет охватить значительное пространство и качественно проводить мониторинг. Процесс обработки видеопотока отличается низкой задержкой, составляющей всего 150–200 миллисекунд, что гарантирует анализ данных в реальном времени. Дополнительно предусмотрен интуитивный пользовательский интерфейс, который позволяет настраивать параметры камер, просматривать маршруты передвижения посетителей и анализировать статистические данные, обеспечивая владельцам общественных пространств удобный контроль над системой мониторинга.

Результаты тестирования показывают, что система эффективно справляется с задачей отслеживания и трекинга посетителей. В дальнейшем планируется развитие системы для внедрения в другие торговые объекты. Важными направлениями улучшения станут репликация базы данных для повышения отказоустойчивости, а также создание расширенного статистического отчета, позволяющего анализировать тенденции и оптимизировать размещение товаров и маркетинговых зон.

ЛИТЕРАТУРА

1. Богуш Р. П., Захарова И. Ю. Клиент-серверная система для идентификации и мониторинга перемещения людей в помещениях на основе анализа видеоданных / Вестник Полоцкого государственного университета. Серия С. Фундаментальные науки. 2020. №12. С. 13–18.
2. Рогов Е. В. PostgreSQL 16 изнутри – М.: ДМК Пресс, 2024. – 664 с.
3. Чен Х., Игнатъева С. А. Повторная идентификация людей в системах видеонаблюдения с использованием глубокого обучения: анализ существующих методов / Автоматика и телемеханика. 2023. №5. С. 61–112.
4. Janus: the general purpose WebRTC server. 2025. [Электронный ресурс] Режим доступа: <https://janus.conf.meetecho.com> (дата обращения: 05.03.2025)
5. Terven J., Cordova-Esparza D. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8. 2023. [Электронный ресурс] Режим доступа: <https://arxiv.org/abs/2304.00501> (дата обращения: 25.02.2025)
6. WebRTC RTP Transport. 2025. [Электронный ресурс] Режим доступа: <https://w3c.github.io/webrtc-rtptransport> (дата обращения: 03.03.2025)

ПРОДВИНУТЫЙ РЕДАКТОР ДЛЯ ХУДОЖЕСТВЕННОЙ РАБОТЫ С ИЗОБРАЖЕНИЕМ

Стремительное развитие технологии CMOS-сенсоров способствовало росту популярности творческой фотографии, однако профессиональные фотографы сталкиваются с рядом проблем при обработке изображений: во-первых, ограниченный динамический диапазон формата JPG/PNG не отвечает современным требованиям к высококачественным изображениям и обработке в родном формате RAW [1]. Во-вторых, традиционное программное обеспечение для редактирования изображений ограничено однослойной архитектурой и не может поддерживать сложное маскирование и обработку формата RAW. И хотя формат RAW может предоставить более богатую информацию об изображении, его большой размер приводит к резкому увеличению стоимости локального хранения.

С этой целью в данной работе облачное хранилище интегрируется поверх основной функции обработки изображений, что позволяет не только удовлетворить потребности в разборе данных RAW, многослойном маскировании, управлении цветом и защите авторских прав в рабочем процессе профессиональной фотографии. В то же время, благодаря распределенной архитектуре и возможностям частного развертывания, она преодолевает ограничения существующего коммерческого программного обеспечения с точки зрения суверенитета данных и эффективности передачи, а также предоставляет набор решений для полного цикла обработки, учитывающих производительность и безопасность хранения.

Для достижения поставленных целей использована клиент-серверная архитектура на языке C++, разделённая на два основных модуля: облачное хранение и обработка изображений.

Модуль облачного хранения использует распределённую схему хранения для высоких нагрузок [2]. Для серверной части: используется FastDFS для создания масштабируемых кластеров хранения данных [3]. Для достижения высокого параллелизма и обработки различных запросов используется Nginx, Spawn-CGI и CGI. Для ускорения чтения часто используемых данных Redis и HiRedis используются для хранения и обработки токенов аутентификации пользователей. Базы данных MySQL используются для постоянного хранения пользовательских данных и метаданных файлов. Для удобного развёртывания приватного облачного хранилища на любом сервере окончательная упаковка выполнена с помощью технологии контейнеризации Docker. С помощью этого решения реализованы функции многопользовательской регистрации и входа, загрузки, скачивания и удаления файлов, совместного использования файлов и т. д., что отвечает требованиям безопасности, расширяемости и переносимости. Архитектура модуля показана на рисунке 1.

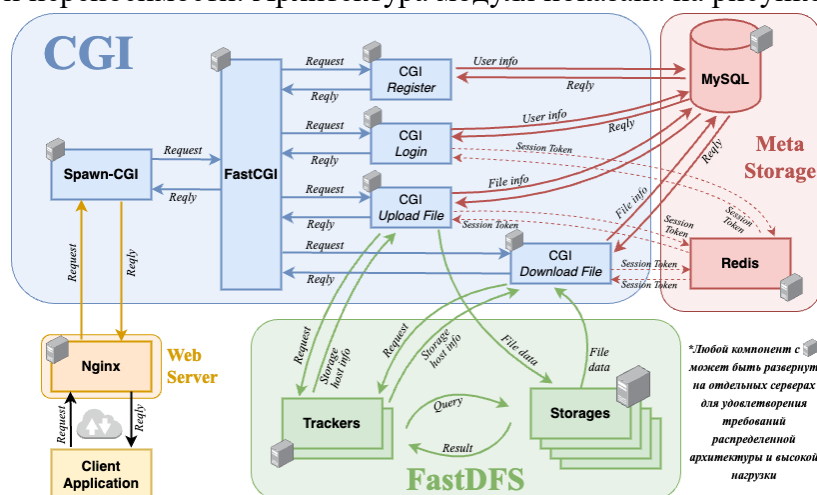


Рисунок 1 – Архитектура распределенной высоконагруженной облачной системы хранения

Для клиентской части используется фреймворк Qt и его элементы: QThread для многопоточной обработки, QNetwork для сетевого взаимодействия с серверной частью и т. д. [4]. Для разделения и упрощения логики транзакций мы используем механизм, основанный на шаблоне наблюдателя, для связи различных объектов внутри программного обеспечения [5]. Управление памятью объектов осуществляется с помощью механизма дерева объектов [6]. Для чтения RAW-данных используется LibRaw. Для обработки изображений, таких как многослойное наложение, локальное маскирование и т. д., используется OpenCV. Для управления цветом используется OpenColorIO. Для эффективного вывода файлов в различных форматах используется OpenImageIO. Для защиты авторской информации мы используем OpenStego для стеганографии метаданных EXIF и темного водяного знака. Учитывая, что не у всех пользователей есть серверы, полностью отделен клиент от сервера, и пользователи могут использовать все остальные функции без использования облачных сервисов. С помощью данного решения получена поддержка формата RAW различных производителей камер, расширена обработка изображений и цвета, получена защита авторских прав, многопоточность обработки, многоязычность и т. д. Архитектура клиентской части показана на рисунке 2.

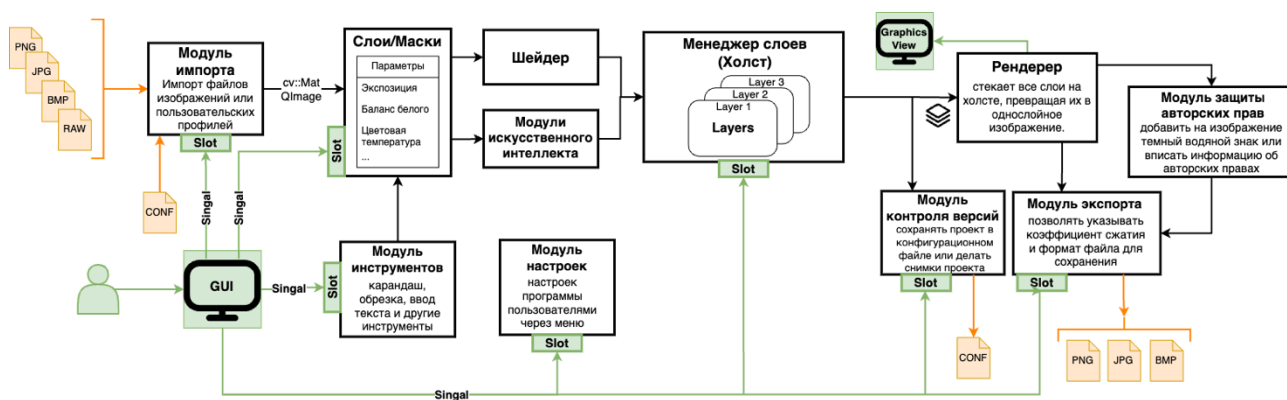


Рисунок 2 – Архитектура модуля обработки изображений

Кроме того, использованы и другие программные средства. Qt Design Studio для проектирования графического интерфейса. Qt Linguist для поддержки нескольких языков. Cppcheck и Clazy для тестирования программного обеспечения. Git для версионирования. GitHub используется в качестве хостинга кода, а также для реализации автоматической верификации, сборки и выпуска программного обеспечения с помощью CI/CD, предоставляемого GitHub Actions.

На стадии тестирования передачи файлов по сети, клиент находился в Санкт-Петербурге, Россия, в среде с пропускной способностью загрузки и скачивания 800 Мбит/с. Сервер находился в провинции Шаньдун, Китай, в среде с пропускной способностью для загрузки 100 Мбит/с и для скачивания 1000 Мбит/с. Скорость загрузки файлов пользователем может достигать 18 МБ/с, стабилизируясь на уровне 10 МБ/с, а скорость скачивания файлов стабилизируется на 7 МБ/с.

Для обработки изображений использовались RAW файлы в формате CR3, снятые на камеру Canon EOS R5, размер файла 70 МБ. RAW файлы могут быть корректно прочитаны и подвергнуты многослойной обработке в реальном времени, с последующим экспортом в необходимый пользователю формат.

ЛИТЕРАТУРА

1. Yao, P. (2012). The Advantages of RAW Format Image Post-processing. In: Zhao, M., Sha, J. (eds) Communications and Information Processing. Communications in Computer and Information Science, vol 289. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-31968-6_74
2. Мэн Цзянин. Дизайн и реализация высоконагруженной облачной системы хранения на основе распределенной архитектуры. [Электронный ресурс]. URL: <https://github.com/nekosilverfox/foxcloud>

3. LIU, XIAOYAN, QING YU, and JIANWEI LIAO. FastDFS: a high performance distributed file system. Applications: an international journal of research and surveys 5.6 (2014): 1741-1746.
4. Dey, Nibedit. Cross-Platform Development with Qt 6 and Modern C++: Design and build applications with modern graphical user interfaces without worrying about platform dependency. 2022.
5. Signals & Slots. [Электронный ресурс]. URL: <https://doc.qt.io/qt-6/signalsandslots.html>
6. Object Trees & Ownership. [Электронный ресурс]. URL: <https://doc.qt.io/qt-6/objecttrees.html>

УДК 004.7

И.В. Новицкий (3 курс бакалавриата),
Б.М. Медведев, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ СИСТЕМЫ КОНТРОЛЯ ПЕРСОНАЛА НА ОПАСНЫХ ПРОИЗВОДСТВАХ

Десятки тысяч зарегистрированных в течение года травм персонала на производстве делают исключительно актуальным применение систем контроля за персоналом. На опасных производствах такие системы позволяют избежать части аварийных ситуаций, а в случае их наступления вовремя оказать необходимую помощь.

Технические средства комплекса контроля персонала содержат датчики состояния человека в виде бейджа или браслета [1], которые передают данные по радиоканалу, сети считывателей данных из датчиков, сервера обработки данных и рабочих мест операторов. Места расположения считывателей выбираются с учетом особенностей контролируемой территории и распространения радиосигнала [2]. Считыватели объединяются в ячеистую сеть для доставки сообщений в сервер [3]. Сервер декодирует сообщения, полученные от считывателей, выполняет обработку информации и обеспечивает хранение данных. Результаты вычислений такие, как координаты датчиков, обнаруженные события, изменения состояния персонала, используются для формирования отчетов на рабочем месте администратора.

Для управления комплексом необходимо разработать программные средства, включая встраиваемое ПО датчиков, средства считывания и маршрутизации сообщений, обработку данных и интерфейс пользователя [4].

Целью работы является разработка программных средств системы контроля персонала на опасных производствах, обеспечивающих обнаружение и отображение опасных состояний персонала (ухудшение здоровья работника, нахождение в опасных зонах, столкновения) путем анализа данных, полученных от датчиков.

Для достижения поставленной цели решены следующие задачи:

1. Разработка сервисов администрирования и управления системой.
2. Реализация фреймворка для интеграции модулей в графическое приложение.
3. Разработка программных средств буферизации и воспроизведения RTLS данных.
4. Модульное и системное тестирование разработанных программных средств с использованием датчиков (акселерометры, гироскопы), стационарных приемников и базовой станции.

Разработанная архитектура программных средств приведена на рисунке 1. Сервис RabbitMQ работает в качестве брокера сообщений MQTT [5]. Формируемая очередь преобразуется в Json формат и рассылается по очередям (топикам) Kafka менеджера. Сообщения считываются вспомогательными периферийными сервисами, такими как сервис идентификации носителя метки, сегмента сети. Делается расчет RTLS параметров с применением алгоритма TDoA [6], технологической операции и конфигурации метки. Результаты вычислений фиксируются в noSQL базе данных для аналитики, и в SQL базе данных для формирования отчетов.

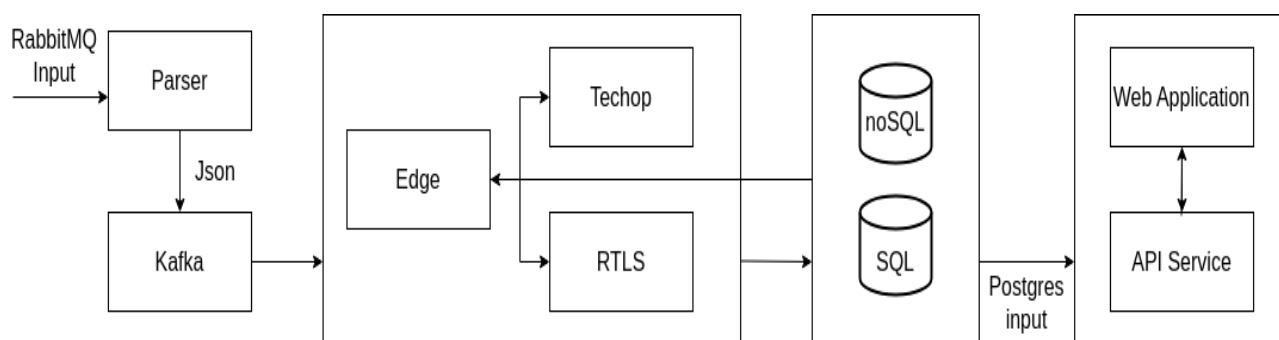


Рисунок 1 – Архитектура программных средств.

Графический интерфейс позволяет формировать отчеты о деятельности предприятия в виде таблиц, графиков и диаграмм, получать информацию о состоянии объекта в 3d, обозревать деятельность персонала в режиме реального времени. Графический интерфейс также позволяет разграничивать и администрировать доступ к системе, настраивать элементы управления и встраивать модули сторонних веб-приложений.

Для разработки ПО использовались языки программирования Java, Python, C++, JavaScript, фреймворки Spring, JUnit, Vue, библиотеки TensorFlow, Three.js. В качестве средств поддержки тестирования применялись Grafana, Postman. В качестве средства разворачивания использовались Docker контейнеры. Для тестирования программных средств было создано 1211 теста, обеспечивающих 92% покрытия кода. Системное тестирование проводилось в подготовленном помещении с размеченными зонами. Тестирование работы системы в реальных условиях проводилось в шахте ГЭС промышленного предприятия, а также в области проведения работ на открытой местности.

Тестирование системы показало определение нахождения персонала в радиусе 0,5 метра. В зависимости от конфигурации системы, задержка получения отчета по всем измерениям составляет от 1 до 5 минут. Вероятность корректного определения состояния персонала не менее 72%.

Разработанные программные средства могут быть использованы в составе системы контроля персонала с количеством сотрудников более 30 человек. Увеличение охвата достигается добавлением базовых узлов и зонированием обработки сенсорных данных.

ЛИТЕРАТУРА

1. Michael J. McGrath, Cliodhna Ní Scanail, Dawn Nafus "Sensor Technologies". - Apress Open, 2013. - 293 с.
2. Jerrin Simla. A, Rekha Chakravarthy, Megalan Leo. L "An Experimental study of IoT-Based Topologies on MQTT protocol for Agriculture Intrusion Detection" // Measurement: Sensors. - 2022. - №24. - С. 4.
3. Тину Оза (Tinu Oza) "Применение датчиков и беспроводных технологий" // Беспроводные Технологии. - 2021. - №3. - С. 42.
4. Н.С. Мальцева, А. Д. Зубова, И. Н. Марышева "Анализ способов построения беспроводных сенсорных сетей" // Ташкентский университет информационных технологий. - 2018. - №2. - С. 32-33.
5. Аль-Обайди А.М.Ж., Заливин А.Н., Бабаринов С.Л. "Локализация сенсорных узлов в беспроводных сенсорных сетях" // Экономика. Информатика.. - 2024. - №1. - С. 239.
6. "Why RabbitMQ?" // RabbitMQ: One broker to queue them all | RabbitMQ URL: <https://www.rabbitmq.com/> (дата обращения: 15.03.2025).
7. "Time of arrival" // Wikipedia URL: https://en.wikipedia.org/wiki/Time_of_arrival (дата обращения: 15.03.2025).

РАЗРАБОТКА МЕНЕДЖЕРА ЗАДАЧ. ПРОГНОЗ ДЕЙСТВИЙ НА ОСНОВЕ ML

Современные системы управления проектами позволяют отслеживать выполнение задач, координировать работу команд и оптимизировать рабочие процессы. Однако, несмотря на широкий выбор инструментов, таких как Trello, Jira и Yandex Tracker, большинство решений ограничиваются базовой аналитикой и не предоставляют автоматического прогнозирования сроков завершения проекта. Да, менеджеры и программисты должны уметь сами просчитывать сроки выполнения проектов. Однако в современном мире, где проекты становятся всё более сложными, а условия их выполнения — всё более динамичными, одной только профессиональной интуиции и опыта может быть недостаточно. А неверное планирование сроков может привести к срыву дедлайнов и неэффективному распределению ресурсов.

В такой ситуации появляется запрос на инструменты, которые могли бы помочь в расчёте сроков проекта. Предложенная система, использующая методы машинного обучения, способная анализировать динамику выполнения задач, учитывать сложность работы, особенности команды, а также данные из предыдущих проектов, может стать надёжным помощником для менеджеров и программистов, предлагая им поддержку в раннем выявлении рисков и увеличении объективности прогнозов.

Таким образом, целью данной работы является исследование возможности разработки интеллектуальной системы управления проектами с функцией предсказания сроков завершения проекта. Для её реализации требуется решить следующие задачи:

1. Создание менеджера задач с возможностью управления проектами, назначением исполнителей, указанием сроков и сложности задач.
2. Сбор статистических данных о ходе выполнения проекта, включая количество задач по сложности и статусу, размер команды и временные параметры.
3. Интеграция машинного обучения для автоматического прогнозирования сроков и вероятности завершения проекта в установленный срок.
4. Разработка пользовательского интерфейса для мониторинга прогресса, просмотра аналитики и получения рекомендаций.

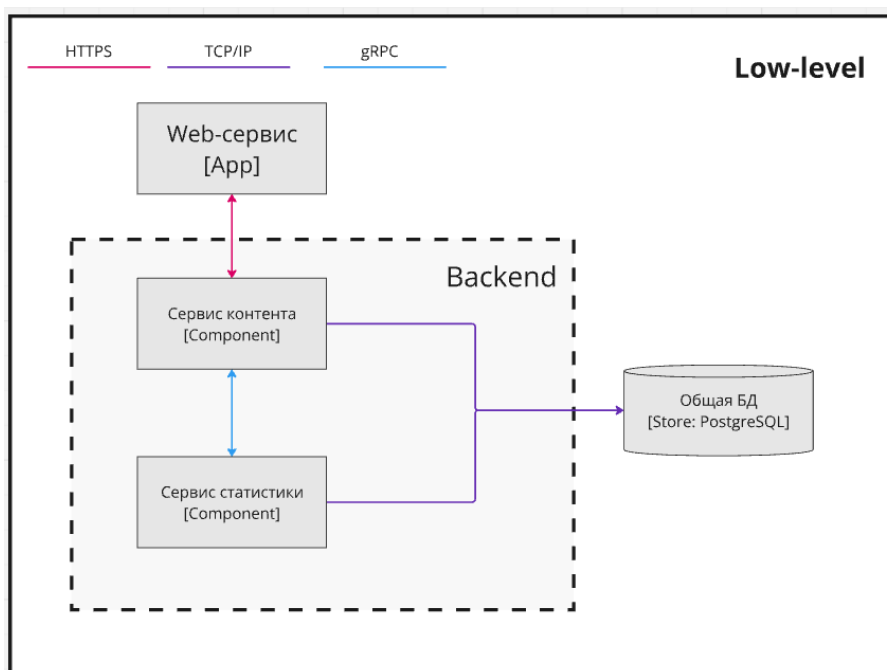


Рисунок 1 – Архитектура предлагаемого решения

Архитектура системы представлена на рисунке 1. Менеджер задач реализован в виде микросервисной структуры:

- Web-сервис – интерфейс управления задачами.
- Сервис контента – обработка логики проектов и задач.
- Сервис статистики – анализ статистики и прогнозирование сроков.
- Общая БД – база данных для хранения информации о задачах и проектах.

Желаемое и текущее состояние проекта обновляются в базе данных, а предсказания ML-модели обновляются при изменении ключевых параметров. (так как предполагается, что проекты в системе будут относительно долговременные, то есть изменения ключевых параметров не будут сильно частыми).

Разработка системы ведётся с использованием следующих технологий:

- Python + TensorFlow [1] – для машинного обучения. (Сервис статистики)
- React [2] – для клиентской части. (Web-сервис)
- Spring Boot (Java) [3] – для серверной логики. (Сервис контента)
- PostgreSQL [4] – для хранения данных. (Общая БД)
- Docker [5] – приведённые микросервисы будут развёртываться в контейнерах Docker.

Существующие решения в области менеджмента проектов не предоставляют интегрированного предиктивного анализа, что создаёт необходимость в разработке системы, объединяющей управление задачами и прогнозирование сроков. Реализованный подход позволит автоматически оценивать риски, повышать точность планирования и снижать трудозатраты на анализ сроков выполнения проектов.

ЛИТЕРАТУРА

1. Tensorflow API Documentation. [Электронный ресурс] Режим доступа: https://www.tensorflow.org/api_docs
2. React Documentation. [Электронный ресурс] Режим доступа: <https://react.dev/learn>
3. Spring Framework Documentation. [Электронный ресурс] Режим доступа: <https://docs.spring.io/spring-framework/reference/index.html>
4. PostgreSQL Documentation. [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/>
5. Docker Docs. [Электронный ресурс] Режим доступа: <https://docs.docker.com/>

УДК 004.4'2

Е.А. Никифорова (4 курс бакалавриата),
Т.В. Коликова, ст. преподаватель

СОЗДАНИЕ ПРОГРАММНОЙ СРЕДЫ ДЛЯ МЕСТООПРЕДЕЛЕНИЯ ИСТОЧНИКА РАДИОИЗЛУЧЕНИЯ

Системы обнаружения и локализации источников радиоизлучения становятся всё более востребованными. Они применяются для мониторинга воздушного пространства, обеспечения безопасности и анализа радиосигналов. Такие системы способны обрабатывать огромные объёмы данных, поступающих от различных источников, будь то самолёты, беспилотники или наземные объекты. Однако обработка этих данных требует значительных ресурсов и сложной инфраструктуры, что делает процесс трудоёмким и дорогостоящим.

Для упрощения этой задачи и повышения эффективности работы систем используется модульный подход. Модульная архитектура позволяет разбить систему на независимые компоненты, каждый из которых выполняет свою функцию [2]. Это не только упрощает разработку и тестирование, но и делает систему более гибкой и адаптируемой под различные задачи. Таким образом, цель данной работы — создание системы обнаружения и локализации источников радиоизлучения, которая сможет эффективно обрабатывать данные и визуализировать результаты обнаружения месторасположения источников радиосигналов [1].

Данный проект представляет собой масштабную систему, состоящую из нескольких модулей, каждый из которых отвечает за определённый этап обработки данных [3]. В рамках этого проекта моя задача — разработка модуля визуализации, который является важной частью системы. Этот модуль отвечает за отображение данных на карте, что позволяет пользователю легко интерпретировать результаты работы системы.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Реализация модуля визуализации. Разработать модуль картографии, который будет отображать местоположение источников на карте. Это ключевая задача, так как визуализация данных является важной частью системы, позволяющей пользователю легко интерпретировать результаты.

2. Интеграция модулей. Обеспечить взаимодействие между модулями приёма, обработки, расчёта и визуализации для создания единой системы.

3. Провести тесты системы, чтобы показать её работоспособность и эффективность [4].

Система состоит из нескольких ключевых модулей, каждый из которых выполняет свою роль в процессе обнаружения и локализации источников радиоизлучения. На рисунке 1 представлена схема системы, иллюстрирующая взаимодействие модулей системы.

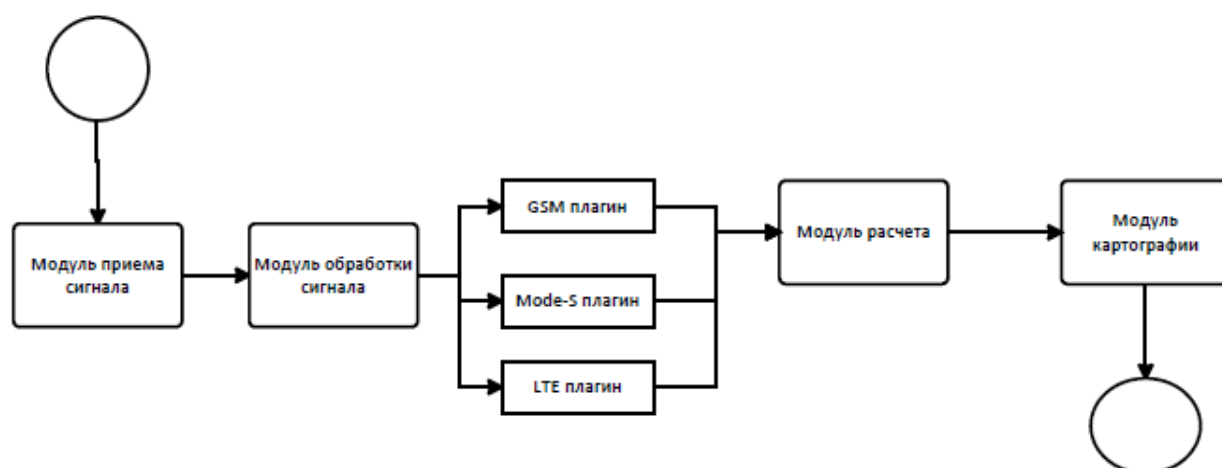


Рисунок 1 – Архитектура системы местоопределения источника радиоизлучения

1. Модуль приёмника сигнала. Этот модуль отвечает за захват радиосигналов из окружающего пространства.

2. Модуль обработки сигнала. Центральный компонент системы, который принимает сырые данные от приёмника и декодирует их. В зависимости от типа сигнала, модуль выбирает подходящий плагин для обработки [5].

3. Модуль расчёта. После обработки сигнала данные передаются в модуль расчёта, где выполняются необходимые вычисления.

4. Модуль картографии. Финальный этап — визуализация данных. Этот модуль отображает местоположение источников на карте, предоставляя пользователю удобный интерфейс для мониторинга.

ЛИТЕРАТУРА

1. Рушечников, Я. И. Информационная технология локализации источника радиоизлучения на основе программно-определяемой радиосистемы / Я. И. Рушечников, В. В. Данилов // Вестник Донецкого национального университета. Серия Г: Технические науки. – 2020. – № 2. – С. 48-54. – EDN XFSESS.
2. Терехов, А. И. Особенности модульного подхода к разработке программного обеспечения / А. И. Терехов, К. С. Бузунова // Технологии информационного общества : X Международная отраслевая научно-техническая конференция: сборник трудов, Москва, 16–17 марта 2016 года. – Москва: ООО "Издательский дом Медиа паблишер", 2016. – С. 242. – EDN VYFVOJ.

3. Громаков, Ю. А. Оптимальная обработка радиосигналов большими системами / Ю. А. Громаков, И. А. Голяницкий, В. А. Шевцов ; Громаков Ю. А., Голяницкий И. А., Шевцов В. А.. – Москва : Эко-Трендз, 2004. – (Библиотека МТС). – ISBN 5-88405-069-0. – EDN QMNBUN.
4. Харичев, Е. А. Методика проведения модульного тестирования программного обеспечения / Е. А. Харичев // Информационные технологии моделирования и управления. – 2010. – № 1(60). – С. 37-41. – EDN KZJDLV.
5. Система управления и сбора данных на основе GSM канала / Р. И. Шагиев, А. В. Карпов, С. А. Калабанов, Р. Р. Фатыхов // Ползуновский вестник. – 2013. – № 2. – С. 214-218. – EDN RBPYEX.

УДК 004.42

А.В. Пахомов (4 курс бакалавриата),
Е.В. Комарова, ст. преподаватель

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ СОЗДАНИЯ ИНТЕРАКТИВНЫХ КАРТ С ПОЛЬЗОВАТЕЛЬСКИМИ КОЛЛЕКЦИЯМИ МЕТОК

В настоящее время наблюдается активное развитие методов визуализации геоданных, что связано с возрастающими требованиями к гибкости и адаптивности инструментов работы с пространственной информацией [1]. Существующие решения, такие как открытые картографические платформы и коммерческие API, зачастую не удовлетворяют потребности пользователей в полной кастомизации: они ограничены в возможностях интеграции пользовательских изображений, добавлении мультимедийных описаний и организации совместного редактирования меток [2]. Эти ограничения становятся особенно ощутимыми при работе как с реальными картами, так и с вымышленными пространствами, используемыми в образовательных, исследовательских и креативных проектах.

Целью данной работы является создание инструмента для разработки интерактивных карт, который позволяет пользователям формировать, редактировать и делиться коллекциями геометок как для карт реального мира, так и для пользовательских изображений.

Такие возможности особенно востребованы у различных категорий пользователей:

- команд, работающих с локальными данными, требующими гибкой визуализации;
- писателей, создающих карты вымышленных миров для литературных произведений;
- разработчиков настольных игр, нуждающихся в интерактивных картах для игровых сценариев.

Для достижения поставленной цели было необходимо решить ряд *задач*. В рамках проведённого теоретического анализа изучены современные методы работы с интерактивными картами и особенности реализации пользовательских коллекций меток [3]. Были определены основные направления для формирования концепции нового программного инструмента, который позволит:

- интегрировать пользовательские изображения (например, сканы бумажных карт и карты вымышленных миров);
- обеспечивать добавление мультимедийных описаний и комментариев к меткам;
- группировать метки в коллекции для удобного структурирования информации;
- встраивать интерактивные карты в сторонние веб-ресурсы.

На основе полученных теоретических выводов сформирована концептуальная архитектура будущего программного продукта, учитывающего масштабируемость, удобство интеграции и пользовательский опыт. Также разработан прототип системы описаний и комментариев к меткам, включая поддержку мультимедийного контента с возможностью интеграции виджета для размещения карт на сторонних веб-сайтах.

На основе полученных теоретических выводов сформирована концептуальная архитектура будущего программного продукта. Клиентская часть будет создана с использованием Vue.js и библиотеки Leaflet, чтобы обеспечить адаптивный и удобный интерфейс при работе с картами.

Серверная же будет реализована на базе Python с фреймворком FastAPI, что позволит добиться асинхронности, высокой производительности и масштабируемости системы [3]. Кроме того, FastAPI позволит генерировать документацию API в формате OpenAPI и Swagger, что значительно упростит дальнейшую интеграцию инструмента. В качестве СУБД была выбрана PostgreSQL с использованием расширения PostGIS, что позволит эффективно хранить и обрабатывать геопространственные данные.

Прототип системы позволит использовать в качестве карт как схемы реальных городов на основе OpenStreetMaps, так и пользовательские изображения, добавлять подробные описания и комментарии к меткам, а также организовывать их в удобные коллекции [2]. Будет использоваться Nominatim API для поиска местоположений по названию или адресу, а также для определения адреса по заданным координатам. Кроме того, решение будет поддерживать интеграцию интерактивных карт в сторонние веб-ресурсы, что обеспечит расширенный охват аудитории. Такой подход гарантирует гибкость, надёжность и соответствие современным требованиям работы с геоинформационными данными.

ЛИТЕРАТУРА

1. Аникеева О. С. Публикация карт в сети интернет: эволюция картографии // Наука. Инновации. Технологии., 2015., № 2., С. 78—85., URL: <https://cyberleninka.ru/article/n/publikatsiya-kart-v-seti-internet-evolyutsiya-kartografii>.
2. Marcante A., Provenza L. P. Social Interaction through Map-based Wikis // PsychNology Journal., 2008., Vol. 6, no. 3., URL: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=71b052267a5bad3fffa9003df22f3689b82ac4ea>.
3. Niu H., Silva E. A. Crowdsourced Data Mining for Urban Activity: Review of Data Sources, Applications, and Methods // Journal of Urban Planning and Development., 2020., Vol. 146, no. 2., P. 04020007., URL: [https://ascelibrary.org/doi/abs/10.1061/\(ASCE\)UP.1943-5444.0000566](https://ascelibrary.org/doi/abs/10.1061/(ASCE)UP.1943-5444.0000566).
4. Bubalo M., Van Zanten B. T., Verburg P. H. Crowdsourcing geo-information on landscape perceptions and preferences: A review // Landscape and Urban Planning., 2019., Vol. 184., P. 101–111., URL: <https://www.sciencedirect.com/science/article/pii/S0169204619300143>.
5. Heipke C. Crowdsourcing geospatial data // ISPRS Journal of Photogrammetry and Remote Sensing., 2010., Vol. 65, no. 6., P. 550–557., URL: <https://www.sciencedirect.com/science/article/pii/S0924271610000602>.

УДК 004.453

А.В. Пахомов (4 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА SBFL СРЕДСТВА ЛОКАЛИЗАЦИИ ОШИБОК В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

Локализация ошибок в программном обеспечении является критически важным этапом жизненного цикла продукта, способствующим снижению затрат на его сопровождение и повышению надёжности. Своевременное выявление дефектов позволяет оперативно реагировать на проблемы и значительно ускоряет процесс их устранения [1,2].

Целью исследования – разработка и внедрение программного средства, позволяющего автоматизировано определять подозрительные участки исходного кода на основе анализа тестовых результатов и данных о покрытии кода. Для достижения цели решены следующие задачи:

1. Исследовать SBFL-подход к локализации ошибок в программном обеспечении.
2. Спроектировать и реализовать средство автоматизированного определять строки кода – кандидаты на первопричины ошибок.
3. Протестировать и апробировать разработанное SBFL-средство.

Основа функционирования разработанного средства заключается в комбинированном использовании нескольких метрик: коэффициента Жаккара, Tarantula, Spearman's footrule, Kendall tau, Op2 и Ochiai. Каждая из этих метрик оценивает степень «подозрительности» строк кода, опираясь на соотношение успешных и проваленных тестов, а также на данные о покрытии кода [3].

SBFL-средство разработано на платформе C# с использованием WPF, также имплементирована консольная реализация. Продукт принимает на вход тестовые данные и результаты анализа покрытия, реализуя алгоритм ранжирования строк по уровню вероятности наличия дефекта. В качестве тестового примера использовались программы из набора Siemens Suite. Сравнение тестовых результатов и данных покрытия позволило выявить участки кода, наиболее подверженные ошибкам. Результаты эксперимента подтвердили, что комбинированное применение нескольких метрик существенно повышает точность локализации дефектов.

Потоки данных при работе с SBFL-средством представлены на рисунке 1.
Скриншот работы программы на данных Siemens Suite представлен на рисунке 2.

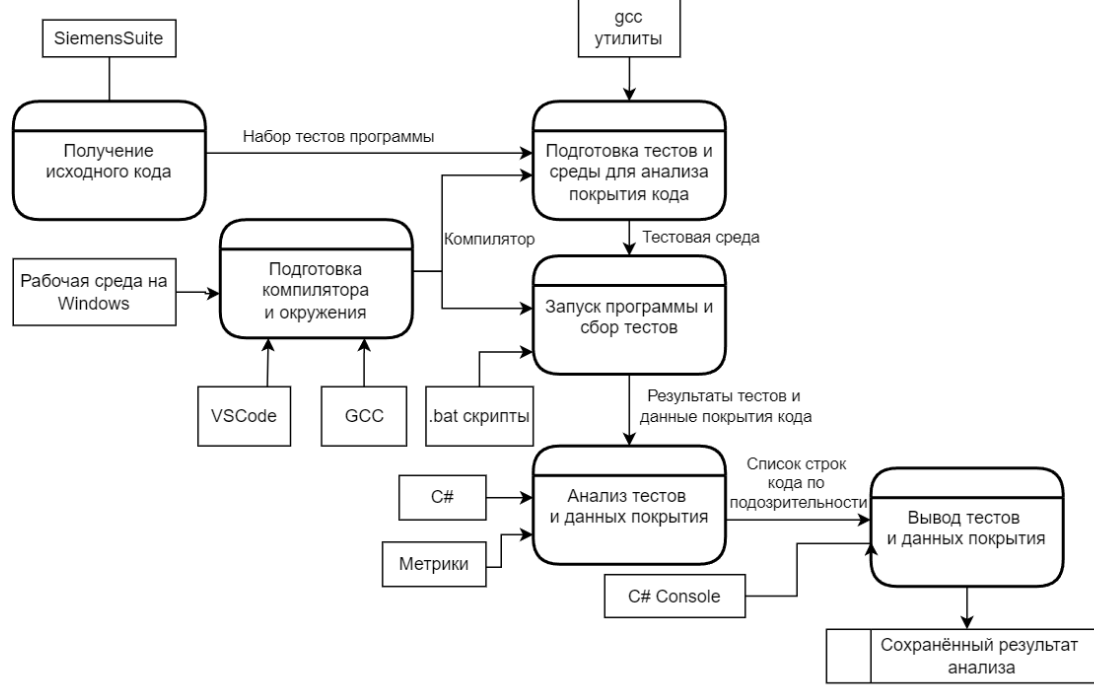


Рисунок 1 – Потоки данных при работе SBFL-средства

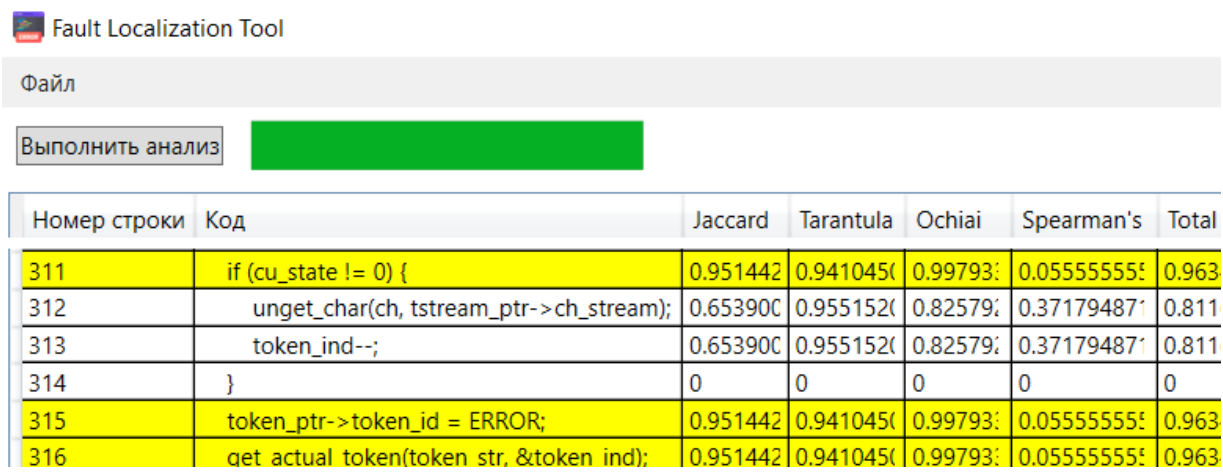


Рисунок 2 – Скриншот с результатом работы GUI версии SBFL-средства

Разработанный инструмент демонстрирует высокую эффективность в автоматизированном определении дефектных участков кода, в частности на Siemens Suite, что

может быть полезно для оптимизации процессов отладки и сопровождения программного обеспечения [4].

ЛИТЕРАТУРА

1. Li X., Zhang L. Transforming Programs and Tests in Tandem for Fault Localization // Proceedings of the ACM on Programming Languages. — 2017. — Т. 1, OOPSLA. — С. 1-30.
2. Effective Fault Localization using Probabilistic and Grouping Approach / K. Kumar [и др.] // International Journal of Software Engineering and Knowledge Engineering. — 2017. — Т. 27, № 4. — С. 631—665. — URL: <https://doi.org/10.1142/S021819401750026X>.
3. Alternative Ranking Distance Metrics for Fault-Focused Clustering in Parallel Fault Localization / Y. Li [и др.] // International Journal of Performability Engineering. — 2023.
4. Effective Fault Localization using Probabilistic and Grouping Approach / K. Kumar [и др.] // International Journal of Software Engineering and Knowledge Engineering. — 2017. — Т. 27, № 4. — С. 631—665. URL: <https://doi.org/10.1142/S021819401750026X>.

УДК 004.4

Ф.А. Пашин (4 курс бакалавриата),
Н.В. Воинов, к.т.н., доцент

РАЗРАБОТКА IOS-ПРИЛОЖЕНИЯ ДЛЯ ПОИСКА И БРОНИРОВАНИЯ АВТОМОЕК

Мы живем в эпоху цифровизации – в такое время, когда многие услуги могут быть улучшены благодаря использованию информационных технологий [1,2]. Одной из таких услуг является мойка автомобиля, а улучшение возможно через цифровизацию процесса записи на автомойку путем разработки специального мобильного приложения.

Такое приложение должно предоставлять возможность искать и бронировать автомойку. Клиенту это позволит сразу узнать всю необходимую информацию о загрузке и доступных услугах, а также избежать звонков и личного посещения автомойки для записи. Для владельца это способ привлечь новых клиентов, автоматизировать процесс взаимодействия с ними и сделать его более прозрачным.

Исследование российского рынка показало, что данная ниша остается незанятой. На текущий момент существует лишь один функционирующий сервис (MoyService [3]), представленный в нескольких малонаселенных городах. Таким образом, выпуск подобного приложения на рынок имеет значительные перспективы.

При реализации таких проектов зачастую ключевую роль играет скорость выхода на рынок, а это означает, что выбранные технологии и решения должны позволить быстро создать работающее приложение. Одним из способов достижения этой цели является использование готовых библиотек компонентов – кирпичиков, из которых можно сконструировать желаемый продукт без значительных временных затрат. На платформе для хостинга проектов GitHub представлена масса вариантов на любой вкус, также существуют обширные курируемые списки [4], помогающие находить решение под конкретную задачу. Лицензии многих библиотек позволяют использовать их в том числе и для коммерческой разработки. По этим причинам готовые библиотеки компонентов активно применялись в процессе разработки приложения.

В качестве языка программирования был выбран современный язык Swift [5], который является стандартом в iOS-разработке, а в основу проекта был положен фреймворк RxSwift [6]. Этот фреймворк реализует концепцию реактивного программирования, которая как нельзя лучше подходит для мобильной разработки, где данные строго связаны с их представлением на экране, и при изменении в одном месте они должны гарантированно измениться и в другом. Реактивное программирование позволяет задавать такие связи, а также обеспечивает декларативный стиль написания кода, что делает разработку проще. Пользовательский интерфейс создавался при помощи различных библиотек и инструмента Interface Builder, а в качестве архитектуры использовалась рекомендованная Apple архитектура MVC [7].

Для обмена данными с сервером был выбран распространенный архитектурный подход REST API, который для выполнения CRUD-операций использует HTTP-методы, такие как GET, POST, PUT, DELETE. Этот подход является достаточно простым, но в то же время несет в себе все необходимое для решения поставленных задач. Сетевой слой приложения был реализован с использованием популярной библиотеки Alamofire.

В процессе разработки активно использовались UML-диаграммы. Это позволяло видеть общую картину и помогало принимать различные архитектурные решения, а также дало возможность упростить реализацию некоторых сценариев, смоделировав их прежде в виде схем.

В результате работы над проектом было разработано полностью рабочее приложение, обладающее следующими возможностями:

- Отображение автомоек на карте.
- Отображение информации о ценах и услугах автомоек.
- Поиск автомоек.
- Запись на автомойку с выбором услуги и времени.
- Построение маршрута до автомойки.
- Настройка профиля пользователя.
- Добавление автомобилей пользователя.
- Добавление банковских карт для оплаты.

ЛИТЕРАТУРА

1. Толстиков, Г. Н. Разработка Android приложения системы поддержки принятия решений / Г. Н. Толстиков, В. В. Амосов // 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. 130-132. – EDN GUXTHF.
2. Абисалов, Г. Р. Разработка Android приложения сервиса доставки и сбора продуктов / Г. Р. Абисалов, В. В. Амосов // 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. 4-5. – EDN DHWJZR.
3. Moy Service. [Электронный ресурс] Режим доступа: <https://moyservice.com/>
4. Awesome iOS. [Электронный ресурс] Режим доступа: <https://github.com/vsouza/awesome-ios>
5. Swift. [Электронный ресурс] Режим доступа: <https://www.swift.org/>
6. RxSwift. [Электронный ресурс] Режим доступа: <https://github.com/ReactiveX/RxSwift>
7. Model-View-Controller (MVC) in iOS – A Modern Approach. [Электронный ресурс] Режим доступа: <https://www.kodeco.com/1000705-model-view-controller-mvc-in-ios-a-modern-approach>

УДК 519.1

В.В. Пехтерев (4 курс бакалавриата),
К.В. Пшеничная, к.т.н., доцент

ВЫБОР ОПТИМАЛЬНОГО МАРШРУТА ПЕРЕДАЧИ ДАННЫХ ПО КОМПЬЮТЕРНОЙ СЕТИ

Рассматривается задача организации передачи данных по компьютерной сети между несколькими филиалами крупной организации. Каждый филиал связан с другими через защищённые каналы связи с использованием виртуальных частных сетей (рисунок 1). Стоимость использования канала определяется объёмом трафика, необходимого для передачи данных с учётом шифрования. Задача относится к одной из самых известных задач комбинаторной оптимизации – задаче коммивояжёра [1, 2].

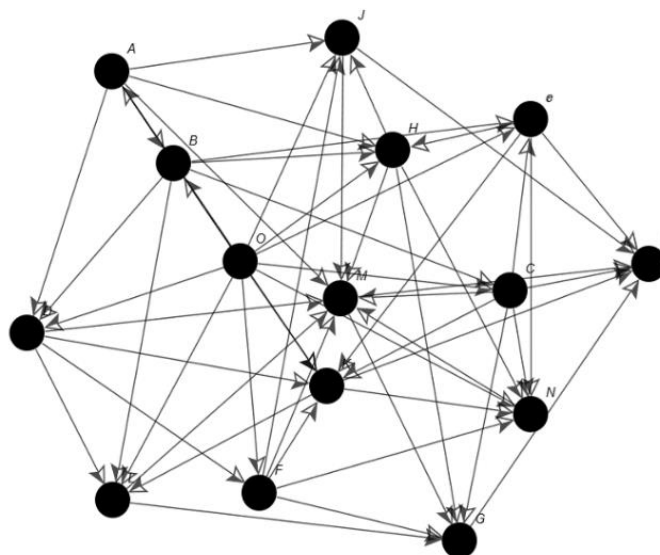


Рисунок 1 – Структурная модульной программы

Исходные данные:

1. Матрица расстояний, где каждая ячейка матрицы определяет объём трафика (в гигабайтах) для передачи данных между филиалами i и j (рисунок 2).
2. Количество филиалов n .
3. Индекс филиала, с которого начинается маршрут.

Место положение	0	A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	0	15	30	25	40	17	32	-	28	-	26	31	42	-	35
A	15	0	8	-	18	-	-	-	20	-	11	19	-	26	-
B	30	8	0	9	16	14	-	-	14	-	-	-	23	-	-
C	25	-	9	0	-	21	-	16	-	30	-	33	-	40	19
D	40	18	16	-	0	-	22	-	-	-	-	29	25	38	-
E	17	-	14	21	-	0	-	-	27	14	-	27	-	-	24
F	32	-	-	-	22	-	0	10	-	-	22	26	-	35	31
G	-	-	-	16	-	-	10	0	24	17	-	-	29	38	-
H	28	20	14	-	-	27	-	24	0	-	19	-	-	29	23
I	-	-	-	30	-	14	-	17	-	0	24	30	-	39	-
J	26	11	-	-	-	-	22	-	19	24	0	-	-	33	-
K	31	19	-	33	29	27	26	-	-	30	-	0	13	-	20
L	42	-	23	-	25	-	-	29	-	-	-	13	0	24	-
M	-	26	-	40	38	-	35	38	29	39	33	-	24	0	17
N	35	-	-	19	-	24	31	-	23	-	-	20	-	17	0

Рисунок 2 – Матрица расстояний

Задача состоит в том, чтобы спланировать маршрут, при котором обеспечивается минимизация общей стоимости передачи данных. Общая стоимость линейно зависит от стоимости шифрования канала между главным офисом и дочерними компаниями, а также от расстояния между самими компаниями. При этом путь передаваемых данных должен начинаться и заканчиваться в центральном офисе. Так как информация является крайне важной, то она должна пройти через абсолютно каждый дочерний узел один раз, при том связно.

Граф рассматриваемой задачи не является полностью связным, что накладывает ограничения на возможность перехода между узлами.

В качестве метода решения задачи был выбран генетический алгоритм [3]. Генетический алгоритм является эвристическим методом, который не пытается просчитать все возможные

маршруты, а использует мягкие вычисления [4]. Он применяет случайные мутации, кроссоверы и селекцию для поиска решений, поэтому может находить хорошие решения за разумное время, обеспечивает баланс между исследованием новых решений и улучшением уже найденных, легко модифицируется и распараллеливается для повышения эффективности.

Разработана программа на языке Python в двух версиях:

1. С использованием динамического программирования и параллельными вычислениями для разбиения задачи на подзадачи с использованием мемоизации и рекурсии;

2. Без использования динамического программирования и параллельных вычислений, с использованием улучшенной эвристики на этапе инициализации (метод ближайших соседей).

Программа была протестирована для варианта матрицы расстояний, представленной на рисунке 2. Результат работы программы с количеством особей в популяции $p = 400$, количеством поколений $g = 10000$ и количеством узлов $V = 15$ следующий: наикратчайший путь: О-А-Н-В-С-Н-М-Л-К-Д-Ф-Г-И-Е-О. Наикратчайшее расстояние: 250. Время работы программы 39,9 сек.

ЛИТЕРАТУРА

1. Корте Б. Комбинаторная оптимизация. Теория и алгоритмы // Московский центр непрерывного математического образования, 2015.
2. Новиков Ф.А. Дискретная математика для программистов // М.: Питер, 2002.
3. Вирсански Э. Генетические алгоритмы на Python / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020.
4. Мягкие вычисления и измерения. Теоретические основы и методы: монография. Том 1./ под ред. проф. С.В. Прокопчиной. – М.: ИД «НАУЧНАЯ БИБЛИОТЕКА», 2017.

УДК 004.042.

В.О. Пономаренко, Е.К. Артеменко, А.О. Багрей (3 курс бакалавриата),
К.В. Кулаков, П.Я. Нейдорф (4 курс бакалавриата),
Р.В. Ахпашев, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ СБОРА И АНАЛИЗА ДАННЫХ С РЕАЛЬНЫХ СЕТЕЙ МОБИЛЬНОЙ СВЯЗИ

Современные мобильные сети испытывают проблемы с качеством связи, особенно при высокой нагрузке. Для их решения разработан программный комплекс, собирающий данные о качестве связи и местоположении пользователей, которые анализируются на сервере и отображаются на frontend. В работе представлена система мониторинга мобильных сетей, включая сбор, обработку и визуализацию данных.

4G LTE (Long-Term Evolution) — четвертое поколение мобильных сетей, обеспечивающее быстрый интернет и улучшенную голосовую связь с использованием технологии OFDM (Orthogonal Frequency Division Multiplexing) для повышения пропускной способности и снижения задержек.

Android-приложение использует встроенные датчики и API для сбора данных о сети. Одним из ключевых источников информации является GPS, определяющий местоположение устройства, что важно для построения карт покрытия сети.

Приложение также получает сведения о соседних базовых станциях через API CellInfo. Данные, такие как идентификатор соты (Cell Identity, CI) и уровень сигнала (dBm), помогают отслеживать состояние сети и выявлять возможные проблемы, например, слабый сигнал или частые переключения между сотами. Для передачи данных на сервер используется WebSocket с минимальной задержкой, а формат JSON упрощает обработку.

На сервере обрабатываются данные с мобильных устройств, где алгоритмы выявляют аномалии, такие как коллизии PCI (Physical Cell Identity), снижение качества сигнала и избыточные хэндоверы, что позволяет диагностировать проблемы и оптимизировать сеть.

Коллизия PCI возникает, когда несколько базовых станций используют одинаковые идентификаторы PCI, мобильное устройство может ошибочно запросить более быстрый канал для передачи данных, что приведет к ошибкам декодирования и снижению скорости передачи. Эффект "ping-pong" возникает из-за частых переключений мобильного устройства между сотами, увеличивая нагрузку на сеть и энергопотребление. Анализ данных выявляет зоны с нестабильным покрытием, что позволяет улучшить устойчивость соединений и снизить нагрузку на оборудование.

Frontend системы мониторинга мобильных сетей визуализирует данные с Android-устройств, которые передаются на сервер для анализа и отображаются на frontend. Web-приложение предоставляет следующий функционал: отображение значений RSRP/RSRQ (треккинг пользователя), выделение проблемных зон с низким уровнем RSRP/RSRQ (представлен на рисунке 1), отображение handovers (куда и откуда был переход) представлено на рисунке 2, PCI коллизий (mod 3/mod 6).

Присутствует возможность фильтрации выводимой информации по времени, значениям RSRP/RSRQ и операторам, что позволяет гибко анализировать данные под необходимые задачи. Есть фильтрация по времени, значениям RSRP/RSRQ и операторам, что помогает гибко анализировать данные. В зависимости от выбранной метрики RSRP или RSRQ отображаются соответствующие столбцы с [цвет-значения] для ясного понимания диапазонов значений.



Рисунок 1 – Отображение зон с низким уровнем RSRP/RSRQ

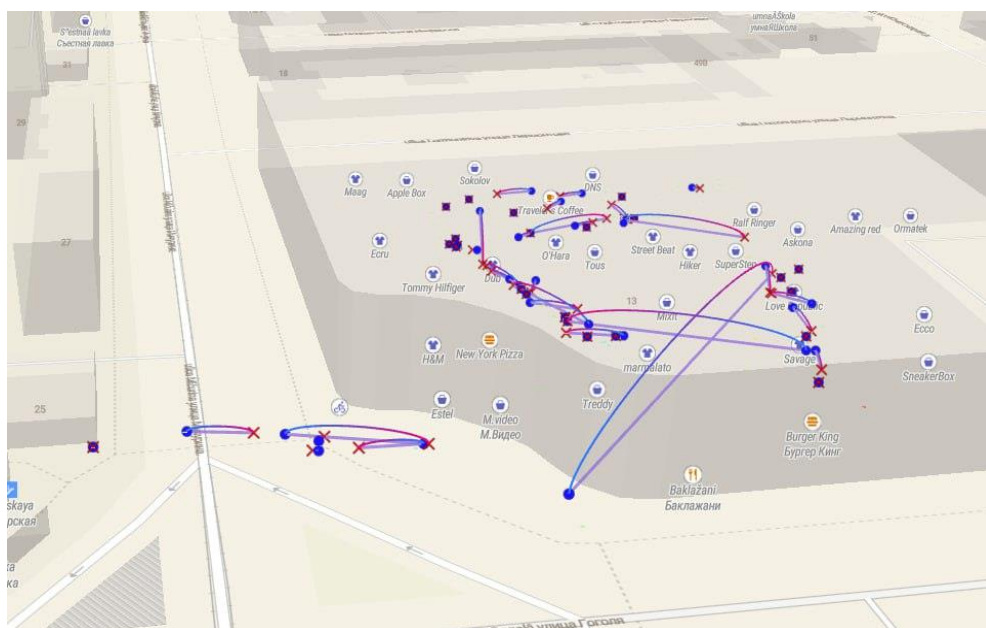


Рисунок 2 – Отображение handover

Представлена система мониторинга мобильных сетей, использующая Android-приложение для сбора данных о сигнале и местоположении пользователей. GPS и API CellInfo отслеживают параметры сети, а WebSocket — быстрый анализ. Система выявляет проблемы, такие как коллизии PCI, снижение сигнала и хэндоверы, а визуализация помогает их устранять. В будущем планируется расширение анализа и интеграция с операторами.

ЛИТЕРАТУРА

1. Domes S. Основы React: всё, что нужно знать для начала работы. [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/ruvds/articles/343022/>
2. Ryabko B., Ryabko D. Provably secure universal steganographic systems. [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/553126/>
3. Хинчман-Домингес А., Блейк Мик Дж. Программирование на Kotlin для Android // ВHV-СПб. 2024. – С. 336.
4. Тихвинский В.О., Юрчук А.Б. Сети мобильной связи LTE. Технологии и архитектура // М: Эко-Трендз, 2010. – С. 284.
5. Moskala M., Wojda I. Android Development with Kotlin // Packt Publishing, 2017. – С. 440.
6. Donovan A.A., Kernighan B.W. The Go Programming Language // Addison-Wesley Professional, 2015. – С. 400.

УДК 004.4

В.О. Пономаренко (4 курс бакалавриата),
Б.М. Медведев, к.т.н, доцент

РЕАЛИЗАЦИЯ КЭШИРОВАНИЯ НОВОСТНОЙ ЛЕНТЫ ДЛЯ IOS-ПРИЛОЖЕНИЯ «ОДНОКЛАССНИКИ»

Разработчики современных мобильных приложений сталкиваются с растущими требованиями к эффективному управлению изменяемыми данными и поддержке автономного режима. Для приложения социальной сети «Одноклассники», насчитывающей миллионы пользователей, постоянный рост количества новостей требует улучшения механизмов кэширования для быстрого доступа и оффлайн режима новостной ленты [1]. Текущий алгоритм кэширования на основе файлов JSON показал ряд недостатков, таких как значительное время на чтение файла при записи страниц, невозможность действий в оффлайн режиме [2].

Цель работы – разработка программных средств кэширования новостной ленты с использованием базы данных YarpDatabase, обеспечивающих уменьшение задержек обработки данных и оффлайн-режим.

Задачи работы:

- Разработка архитектуры кэширования на базе YarpDatabase [3, 4].
- Разработка программных средств модуля кэширования для новостной ленты в приложении “Одноклассники”.
- Интеграция в приложение и тестирование разработанных программных средств.

На рисунке 1 представлена блок-схема работы модуля кэширования.

База данных YarpDatabase содержит четыре коллекции: Meta, Pages, Header и Banners. При запуске приложение проверяет наличие и актуальность кэша по времени создания. Если кэш валиден, данные объединяются из всех коллекций методом readMergedStream, иначе делается запрос к API и результат сохраняется через putFirstPage. При прокрутке ленты до конца формируется запрос к API с сохраненным anchor и новые данные добавляются методом putPage, который проверяет лимит через canAddPage (сравнивает текущее количество с maxCachedFeeds). При обновлении состояние сбрасывается и кэш обновляется. Чтение происходит путем сортировки и объединения всех страниц, добавления заголовка и актуальных баннеров.

Предложенное решение обеспечивает эффективную работу с данными за счет разделения их на отдельные коллекции, что позволяет выполнять точечное обновление без необходимости перезаписи всего кэша, что приводит к значительному сокращению количества операций чтения, поскольку новый механизм не требует объединения данных с полным файлом кэша при каждом обновлении.

Важной особенностью реализованного решения является базовая поддержка оффлайн-режима для просмотра ленты без доступа к интернету. В дальнейшем планируется расширить функциональность для поддержки действий в оффлайн-режиме, таких как лайки и другие взаимодействия с контентом. Для тестирования эффективности разработанного решения будет проведен эксперимент с постепенной раскаткой функциональности на разные группы пользователей с замером ключевых метрик производительности и взаимодействия, что позволит оценить влияние нового механизма кэширования на пользовательский опыт в реальных условиях.

Разработанные программные средства кэширования новостной ленты будут использованы в следующей версии приложения «Одноклассники».

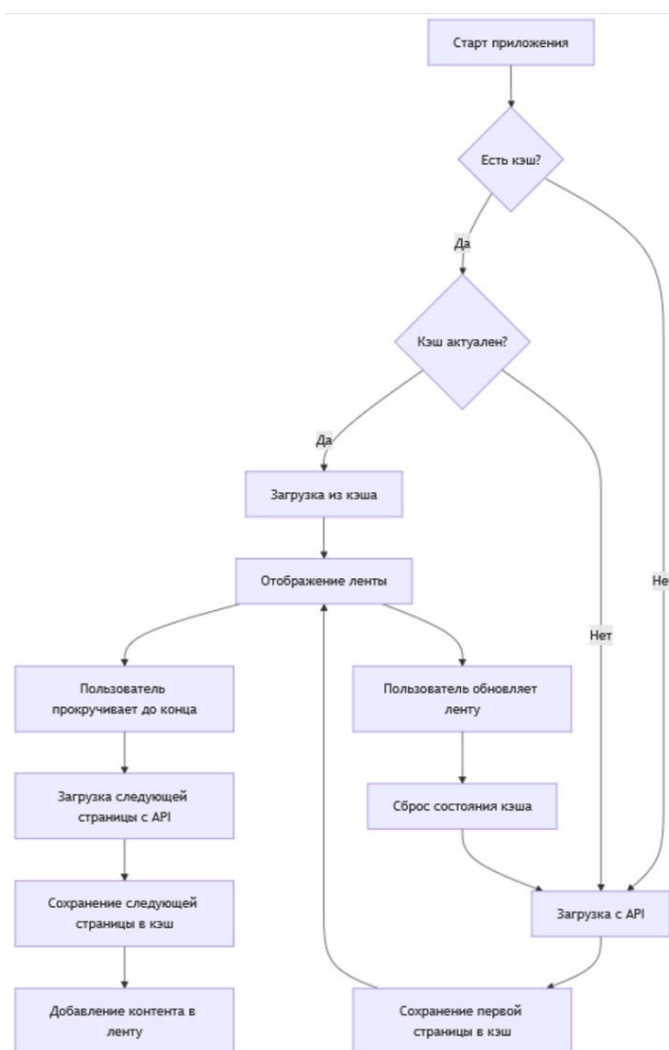


Рисунок 1 – Схема работы модуля кэширования

ЛИТЕРАТУРА

1. Apple Developer. Caching Network Data – URL: <https://developer.apple.com/tutorials/app-dev-training/caching-network-data> (дата обращения: 24.12.2024).
2. Baljit Kaur. Implementing Data Caching in iOS: Strategies for Performance Optimization – Medium. URL: <https://medium.com/swifty/implementing-data-caching-in-ios-strategies-for-performance-optimization-7117182be987> (дата обращения: 24.12.2024).

3. YapDatabase – GitHub Repository. URL: <https://github.com/yapstudios/YapDatabase> (дата обращения: 24.12.2024).
4. Apps using Yap Database SDK – Fork. URL: <https://fork.ai/technologies/database/yap-database> (дата обращения: 24.12.2024).
5. DZone. Offline Mode: Key Strategies and Benefits for iOS – URL: <https://dzone.com/articles/offline-mode-key-strategies-and-benefits> (дата обращения: 24.12.2024).

УДК 004.42

А.П. Парамонова (3 курс бакалавриата),
В.Э. Шмаков, к.т.н, доцент

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ РАСПИСАНИЕМ

В современном мире важное значение имеет эффективное управление временем, тайм-менеджмент. На данный момент на рынке имеются приложения, направленные на управление расписанием, однако эти приложения часто имеют ограничения и могут быть ограничены в функционале. Практическая значимость работы заключается в создании приложения, которое может охватить большую аудиторию и будет востребовано как среди студентов и школьников, так и более старшей аудитории. Помимо этого, для еще большего охвата аудитории систему должна быть кроссплатформенной (исполняться как под android, так и под ios).

Основной целью данного проекта является анализ потребностей пользователей и разработка кроссплатформенного мобильного приложения, позволяющего автоматизировать управление расписанием.

Таким образом, для достижения поставленной цели необходимо решить такие задачи, как:

- Провести сравнительный анализ существующих средств разработки кроссплатформенных мобильных приложений.
- Провести анализ существующих решений.
- Сформировать требуемый функционал системы на основе потребностей пользователей.
- Реализовать систему с помощью выбранных технологий программирования.

Проводился анализ таких приложений Structured [1], Any.do [2] и Todoist [3]. Результаты указаны в таблице 1.

Таблица 1 – Сравнительный анализ функционала приложений

	Structured	Todoist	Any.do
Просмотр по дням	имеется	имеется	имеется
Просмотр по неделям	отсутствует	отсутствует	отсутствует
Просмотр по месяцам	имеется	имеется	отсутствует
Выбор даты и времени для задачи	имеется	имеется	отсутствует
Напоминания	только в платной подписке	только в платной подписке	имеется
Повторение	только в платной подписке	отсутствует	имеется
Приоритет	отсутствует	имеется	отсутствует
Заметки	отсутствует	отсутствует	имеется
Форматирование заметок	отсутствует	отсутствует	отсутствует

В результате изучения существующих аналогов системы и требований пользователей был сформирован необходимый функционал системы:

1. Просмотр задач в виде календаря по месяцам.
2. Просмотр задач по неделям.
3. Просмотр задач по дням.
4. Функция добавления задач на каждый день (с возможностью указания времени начала).
5. Функция, позволяющая настраивать статус задачи и обозначать ее приоритет.
6. Функция для установки напоминаний о предстоящих задачах и событиях. (например, за день, за час).
7. Функция для введения повторяющихся задач (например, каждую неделю или каждый день).
8. Функция для добавления заметок к каждой задаче. Редактор заметок включает в себя текстовый редактор с такими возможностями как:
 - ввод текста и его форматирование (жирный, курсив, изменение шрифта и цвета, подчеркивание и зачеркивание текста);
 - создание нумерованных и маркированных списков;
 - создание списков с чекбоксами;
 - создание таблиц.

В ходе проведения сравнительного анализа средств программирования, был выделен популярный фреймворк React-native [4]. Он предоставляет набор компонентов, позволяющих реализовывать кроссплатформенную разработку, и полностью подходит для реализации поставленной задачи. Для разработки приложения был использован язык программирования JavaScript [5]. Также, использовался фреймворк Expo [6]. Expo – это набор инструментов, сервисов и библиотек, упрощающих разработку на React-Native и тестирование приложения. Так, например, Expo предоставляет доступ к приложению Expo go с помощью которого легко можно протестировать приложение на реальном устройстве, как на системе android, так и на системе ios. Для хранения данных была выбрана легковесная СУБД SQLite [7]. С помощью нее реализуется локальная база данных. Использование локальной базы данных позволяет приложению не зависеть от наличия интернет – соединения и быть полностью функциональным даже при его отсутствии.

Архитектура приложения состоит из трех уровней. Первый уровень содержит весь пользовательский интерфейс. На втором уровне реализована логика приложения. Этот уровень является связующим звеном между пользовательским интерфейсом и третьим уровнем архитектуры – базой данных.

ЛИТЕРАТУРА

1. Structured [Электронный ресурс]. Режим доступа: <https://structured.app/>
2. Any.do [Электронный ресурс]. Режим доступа: <https://www.any.do/>
3. Todoist [Электронный ресурс]. Режим доступа: <https://www.todoist.com/ru>
4. React Native [Электронный ресурс]. Режим доступа: <https://reactnative.dev/>
5. JavaScript [Электронный ресурс]. Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
6. Expo [Электронный ресурс]. Режим доступа: <https://docs.expo.dev/get-started/introduction/>
7. SQLite [Электронный ресурс]. Режим доступа: <https://www.sqlite.org/about.html>

РАЗРАБОТКА СИМУЛЯТОРА ПЭТ-СКАНЕРА ДЛЯ ЛУЧЕВОЙ ТЕРАПИИ

Лучевая терапия представляет собой высокоэффективный метод лечения онкологических заболеваний, базирующийся на точной доставке дозы облучения в определенную область человеческого тела. Однако, несмотря на значительные успехи в этой области, существует фундаментальная проблема – методики лучевой терапии, принятые в настоящее время, основаны на предположении о статичности опухоли. Это предположение играет критическую роль в точности процедуры, и любое движение пациента, будь то дыхание или случайные телесные движения, может привести к существенным погрешностям в доставке облучения.

В основе современных разработок в области лучевой терапии лежит стремление к максимальной точности и эффективности в доставке облучения опухолевой ткани. В этом контексте, ключевую роль играет подход лучевой терапии под биологическим контролем, предполагающий использование аппаратуры с обратной связью [1]. Этот подход предоставляет возможность активного реагирования на динамические изменения внутренней структуры пациента в режиме реального времени, что повышает точность и предсказуемость процедуры.

Подобный аппарат лучевой терапии активно анализирует биологические параметры организма пациента в процессе облучения. Одним из ключевых элементов этого подхода является использование позитронно-эмиссионной томографии (ПЭТ) в качестве "биологического ориентира" [2]. ПЭТ предоставляет детальную информацию о метаболизме тканей, что делает его ценным инструментом для наблюдения за изменениями внутри организма пациента в реальном времени [3].

Однако, несмотря на потенциальные преимущества данного метода, возникает проблема, связанная с продолжительным временем, требующимся для получения высококачественных ПЭТ-изображений [4]. Решение этой проблемы находится на поверхности – необходимо использовать быстро получаемые "частичные изображения" [2]. Это позволяет учесть динамику изменений в организме пациента и, таким образом, активно корректировать направление луча в режиме реального времени.

Разработка и внедрение лучевой терапии под биологическим контролем представляет собой важный шаг в улучшении результатов лечения онкологических заболеваний. Этот подход не только повышает точность доставки облучения, но и открывает новые перспективы для персонализированного и эффективного лечения пациентов.

Процесс тестирования аппарата лучевой терапии требует наличия модели пациента. В настоящее время для этих целей активно применяются фантомы – контейнеры, имитирующие форму конкретной части человеческого тела, оснащенные цилиндрами внутри. Заполненный различными жидкостями, фантом подвергается сканированию.

Однако, несмотря на эффективность использования фантомов, существует проблема – тестирование приборов становится сложным процессом из-за зависимости лучей от зашумленных данных ПЭТ. Точность результатов тестирования напрямую зависит от конкретной реализации вероятностного распределения линий отклика ПЭТ, что создает дополнительные сложности при анализе и интерпретации данных.

Для преодоления сложностей, связанных с натурными моделями, предлагается использовать компьютерные модели пациента. Зная внутреннюю структуру фантома, можно ручным образом составить "идеальное" ПЭТ-изображение. Это изображение становится основой для создания компьютерной модели пациента.

Результатом реализации такого подхода будет являться автоматизация процесса тестирования. Генерация линий отклика происходит на основе идеализированного ПЭТ-

изображения. Сгенерированные данные подаются на вход программного модуля, который работает с ними как с данными, полученными естественным путем.

Такой компьютерный подход к моделированию позволит более эффективно проводить тестирование приборов лучевой терапии, минимизируя влияние шумов и предоставляя более стабильные результаты. Это открывает новые возможности для повышения эффективности методов лучевой терапии под биологическим контролем.

Симулятор ПЭТ-сканера, получающий на вход ПЭТ-изображение, составленное специалистом, и возвращающий соответствующий набор линий отклика, состоит из следующих модулей:

- преобразование Радона;
- scatter-коррекция – избавление от искажений, связанных с геометрией пациента;
- извлечение слайсов синограммы – установка соответствия между частями синограммы и положением кушетки;
- генерация случайных данных – генерация нормально распределенных данных и масштабирование;
- попиксельная сумма синограмм;
- ACF-коррекция – избавление от искажений, связанных с плотностью тканей;
- генерация событий из вероятностного распределения.

Симулятор ПЭТ-сканера реализуется на языках MATLAB и C++, использует ранее реализованные классы на языке MATLAB и взаимодействует с программным модулем аппарата лучевой терапии на языке C++. Совместимость кода MATLAB и кода C++ поддерживается при помощи MATLAB Engine API и технологии MEX-файлов.

В начале работы симулятора происходит соединение с базой данных (класс PlanningSession), после чего файл формата json (класс ImageSeries), ссылающийся на серию ПЭТ-изображений, извлекается в оперативную память или на диск. Симулятор работает в режиме реального времени, генерируя события (фиксацию линий отклика), при этом часть из преобразований, выполняемых модулями, можно вычислить заранее. Последний модуль получает на вход синограмму, каждое значение которой является параметром для распределения Пуассона. Эта синограмма дает представление о вероятностном распределении для каждого пикселя исходного ПЭТ-изображения. В результате работы программы, сгенерированные данные передаются дальше программному модулю аппарата лучевой терапии, которые возвращает данные, необходимые для формирования луча в определенные моменты времени.

ЛИТЕРАТУРА

1. Shervin M Shirvani, MD, MPH, Calvin J Huntzinger, MS, Thorsten Melcher, PhD, Peter D Olcott, PhD, Yevgen Voronenko, PhD, Judy Bartlett-Roberto, MBA And Samuel Mazin, PhD Biology-guided radiotherapy: redefining the role of radiotherapy in metastatic cancer // British Institute of Radiology. - 2020. - №94.
2. Eirik Ramsli Hauge Dose Painting by Numbers for Head and Neck Cancer -Photons versus Protons. - Oslo: Reprosentralen, University of Oslo, 2018. - 142 p.
3. Angela Lamarca, Jorge Barriuso, Amarjot Chander, Mairéad G McNamara, Richard A Hubner, Derek ÓReilly, Prakash Manoharan, Juan W Valle 18Ffluorodeoxyglucose positron emission tomography (18FDG-PET) for patients with biliary tract cancer: Systematic review and meta-analysis // Journal of Hepatology. - 2019. - №71. - P. 115-129.
4. Jarmo Teuvo, Virva Saunavaara, Tuula Tolvanen, Terhi Tuokkola, Antti Karlsson, Jouni Tuisku, and Mika Teräs Quantitative Evaluation of 2 Scatter-Correction Techniques for 18F-FDG Brain PET/MRI in Regard to MR-Based Attenuation Correction // THE JOURNAL OF NUCLEAR MEDICINE. - 2017. - №58. - P.1691-1698.

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ ИССЛЕДОВАНИЯ СВОЙСТВ СТЕКЛА

В современном мире существует постоянная потребность в улучшении и оптимизации процессов производства, а также в создании новых материалов с определенными свойствами, отвечающими требованиям различных отраслей промышленности. Стекло – один из самых универсальных и широко используемых материалов в современной науке и технике. Его свойства зависят от химического состава и условий производства, что делает исследование стекол важной задачей для множества отраслей – от оптики до биомедицины.

Ранее для изучения стекол широко применялась система SciGlass [1] – обширная база данных, включавшая информацию о составе, свойствах и методах производства стеклянных материалов. Однако в последние годы SciGlass устарела и перестала поддерживаться, что привело к утрате удобного инструмента для анализа стекол.

Таким образом, целью данной работы является реализация платформы, способной заменить SciGlass и предоставить исследователям удобный инструмент для работы с данными о стеклах. Программно-аппаратный комплекс представляет собой сочетание аппаратной и программной части. Программная часть состоит из специально разработанного под конкретные нужды программного обеспечения, серверная часть устанавливается на сервер. Программная часть оснащена хранилищем данных.

Для достижения цели необходимо решение следующих задач:

1. Проектирование схемы базы данных.
2. Проектирование архитектуры платформы.
3. Предварительная обработка данных.
4. Реализация API платформы.
5. Реализация пользовательского интерфейса платформы.
6. Тестирование платформы.
7. Демонстрация полученного результата.

Перед разработкой новой платформы необходимо было провести предварительную обработку данных. Исторически информация в SciGlass хранилась в базе данных Microsoft Access, что создавало определённые ограничения по объёму данных и скорости работы. На первом этапе все данные из SciGlass были извлечены, очищены от возможных дубликатов и ошибок, а затем приведены к единой структуре. После этого они были загружены в современную реляционную базу данных PostgreSQL [2], обеспечивающую надёжное хранение, быстрый поиск и возможность масштабирования. Этот шаг позволил не только сохранить ценные научные данные, но и создать основу для дальнейшей разработки новой платформы, которая обеспечит удобный доступ к информации.

Архитектура системы ведения учета представлена на рисунке 1. На ней видно, что платформа разрабатывается на основе клиент-серверной архитектуры, что позволяет разделить обработку данных и пользовательский интерфейс, обеспечивая гибкость и масштабируемость системы.

Backend часть платформы реализована с использованием языка программирования Python и библиотеки Django Rest Framework [3] – мощного инструмента для создания API, который обеспечивает удобную обработку запросов, управление пользователями и доступ к базе данных PostgreSQL. Frontend часть построена на NextJS [4], что позволяет создать удобный и интерактивный интерфейс для работы с данными.

Архитектурные решения основаны на результатах нашего предыдущего исследования [5], где была проанализирована необходимость перехода на веб-

ориентированную платформу и преимущества отдельного подхода к разработке клиентской и серверной частей.

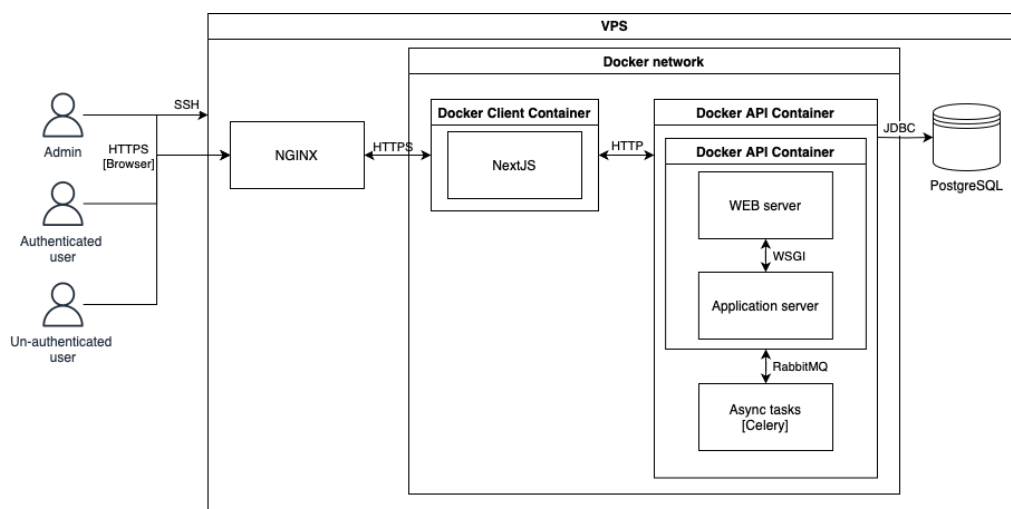


Рисунок 1 – Архитектура системы

Общий архитектурный стек системы включает несколько ключевых компонентов. Клиентский код на React будет размещён на сервере вместе с Nginx [6], который также будет выполнять роль прокси-сервера, перенаправляя запросы между клиентом и сервером, а также обеспечивая балансировку нагрузки и кеширование. Серверный код, основанный на Django Rest Framework, будет работать в изолированном контейнере Docker [7], что обеспечит удобное развертывание и поддержку. Взаимодействие между клиентом и сервером будет осуществляться через API [8], реализованный в Django. Для хранения данных будет использоваться база данных PostgreSQL, размещённая на сервере, обеспечивая надёжное хранение и быстрый доступ к информации.

Такая архитектура позволяет создать высокопроизводительную и масштабируемую систему, где каждый компонент выполняет свою задачу и может быть легко заменен или модифицирован в дальнейшем.

ЛИТЕРАТУРА

1. База данных о свойствах стекла SciGlass. [Электронный ресурс] Режим доступа: <https://github.com/epam/SciGlass>
2. Реляционная база данных PostgreSQL. [Электронный ресурс] Режим доступа: <https://www.postgresql.org/>
3. Django Rest Framework documentation. [Электронный ресурс] Режим доступа: <https://www.django-rest-framework.org>
4. NextJS documentation. [Электронный ресурс] Режим доступа: <https://nextjs.org/>
5. Р. Р. Попов, Н. В. Воинов, Е. А. Павлов РАЗРАБОТКА СИСТЕМЫ УЧЕТА ПОКАЗАТЕЛЕЙ ДЕЯТЕЛЬНОСТИ НЦМУ СПбПУ // Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых, 26–27 апреля 2023 г. – СПб. : ПОЛИТЕХ-ПРЕСС, 2023. – 267 с. – С. 66-68.
6. HTTP-сервер Nginx. [Электронный ресурс] Режим доступа: <https://nginx.org/ru/>
7. Docker containerization. [Электронный ресурс] Режим доступа: <https://www.docker.com/>
8. Бойков, К. М. Разработка клиент-серверной архитектуры для сервиса по управлению интерактивными подписками / К. М. Бойков, О. С. Командина, Н. В. Воинов // 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. 113-114. – EDN DTYFAF.

ПИЛОТНЫЙ ПРОЕКТ СИСТЕМЫ КОНТРОЛЯ ВЫХОДНОЙ ПРОДУКЦИИ ДЛЯ ЛИНИИ ОИНКТ 3, ЛИНИЯ 3

Сейчас подобные системы контроля производимой продукции внедрены во всех средних, крупных и даже некоторых мелких предприятиях. Их полезность и даже необходимость обусловлены большими объемами производственных процессов, большим количеством людей, участвующих в формировании продукта и не только. Некоторые производственные процессы разбиты помимо участков еще и территориально, из-за чего процесс анализа и контролирования всех процессов становится тяжелым.

Целью проекта является создание системы и модернизация существующих решений, охватывающих все этапы производственного процесса, для всеобъемлющего контроля произведенной продукции на отдельной взятой линии, или же участка – ОиНКТ 3. Для достижения целей проекта выдвинуты задачи:

- Обновление сайта «Статистики» на актуальный стек технологий [3][4][5].
- Установка недостающих производственных постов и последующее их добавление на сайт «Статистики» (непосредственная установка не относится к моим задачам).
- Формирование структур данных для передачи их в 1C ERP для хранения.
- Разработка программных продуктов для каждого нового производственного поста [5].

Архитектура проекта состоит из нескольких подсистем:

- Сайт «Статистики», работающий по принципу клиент-серверного взаимодействия.
- Отдельные приложения для работы на каждом производственном участке (рисунок 1).

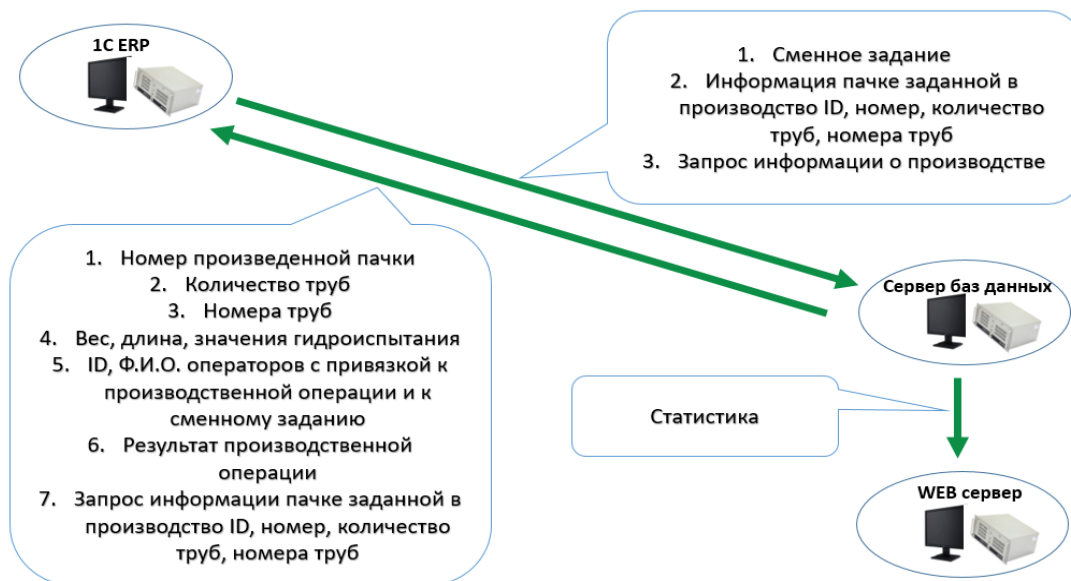


Рисунок 1 – Схема взаимодействия Программных приложений

Автоматизированные системы позволяют в разы быстрее обрабатывать полученные в ходе производства данные и точнее следить за всеми этапами этого производства. Также желание внедрить системы контроля обусловлены необходимостью поддерживать высокое качество производимой продукции, чтобы иметь возможность быть конкурентоспособными на рынке.

Данный проект на линии ОиНКТ 3 на ИТПЗ поможет тщательней прослеживать такие параметры как: время прохождения трубами всех производственных постов на линии 3, причастность всех работников к конкретной произведенной трубе, а также ведение статистики

брака. Полученные данные помогут определить наиболее частые причины возникновения бракованных труб, дабы оптимизировать соответствующие детали процессов и увеличить прибыль предприятия.

ЛИТЕРАТУРА

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.: ил. — (Серия «Мастер-класс»), ISBN 978-5-496-00433-6.
2. Доклады с конференций DotNext. (год обращения: 2025) [Электронный ресурс]. URL: <https://dotnext.ru/archive/>
3. Документация Microsoft по ASP.NET Core. (год обращения: 2025) [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0>
4. Документация Microsoft по Blazor WASM. (год обращения: 2025) [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/fundamentals/?view=aspnetcore-8.0>
5. Документация Microsoft по .NET 8. (год обращения: 2025) [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/core/introduction>

УДК 004.4

Д.С. Пятов (2 курс магистратуры),
Т.В. Леонтьева, к.т.н, доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ 3D ПЕЧАТИ ПО ТЕХНОЛОГИИ СЕЛЕКТИВНОГО ЛАЗЕРНОГО СПЕКАНИЯ

SLS, или селективное лазерное спекание – это метод 3D-печати, разработанный и запатентованный докторами Техасского университета (Остин, США) Карлом Декардом и Джо Биманом в середине 1980-х годов. Он заключается в нанесении порошкообразного материала на поверхность подложки, его равномерного распределения, и сканировании поверхности лазером. Под влиянием лазера порошок спекается и таким образом формируется слой изготавливаемого объекта, соответствующий текущему срезу изделия. После сканирования поверхности подложка опускается на толщину слоя изделия, при необходимости добавляется и распределяется порошок, и проводится печать следующего среза. После печати всех слоев готовый объект очищается от неиспользованного порошка и остывает. Преимуществами данной технологии являются отсутствие необходимости в дополнительном опорном материале, возможность одновременной печати нескольких изделий и многообразие возможных материалов для использования [1].

Большая часть данных принтеров производится зарубежными компаниями, при этом сложность их устройства и коммерческие интересы этих компаний зачастую ведут к тому, что для обслуживания этих инструментов необходимо вызывать их сотрудников. В связи с уходом из России многих зарубежных производителей и действующими санкциями очевидно, что доступность приобретения и обслуживания зарубежных принтеров в нашей стране снизилась. Учитывая это и взятый курс на импортозамещение, остро встает вопрос о производстве отечественных 3D-принтеров и программного обеспечения для них. При этом проблемой является недостаток свободных для использования программных средств такой направленности. Ранее был подготовлен прототип программы для работы с устройствами такого типа, при этом в нем остались места для улучшений в контексте работы и процесса разработки. Таким образом, целью данной работы является упрощение создания 3D-принтеров путем усовершенствования прототипа необходимого для их работы программного обеспечения с открытым исходным кодом.

Для достижения этой цели необходимо решить следующие задачи:

1. Составление требований к проектируемому программному обеспечению.
2. Обзор существующих программных средств для 3D-печати.

3. Сравнительный анализ найденных средств.
4. Поиск необходимых для реализации инструментов и теоретических сведений.
5. Усовершенствование реализации программного средства. (Использование более современных библиотек, оптимизация работы, организация процесса разработки качественного программного продукта).

Предложенное программное обеспечение является модульным, соответственно есть возможность использовать только необходимые его компоненты. Модули подключаются к программе в следующем порядке:

1. Модуль расчета маршрута лазера на слое.
2. Модуль разделения 3D-фигуры на слои.
3. Модуль графического интерфейса
4. Модуль авторизации пользователей.

Таким образом, возможны 4 конфигурации программного обеспечения.

Модуль расчета маршрута лазера на слое является самым важным, так как производит необходимую для печати информацию. Для заданных набора многоугольников, начальной точки и толщины линии вычисляет оптимальный путь заполнения слоя в виде последовательного набора линий с информацией о состоянии лазера (включен/выключен). Для этого необходимо разделить полигоны на внутренние и внешние, чтобы получить область заполнения, закрасить контуры, и затем перейти к заполнению областей внутри контуров. Для оптимизации маршрута используется алгоритм ближайших соседей, что позволяет минимизировать холостые проходы лазера. Также необходимо правильно выбрать угол заполнения и учесть особые случаи, такие как острые края и узкие места, где расстояние между контурами меньше, чем две толщины лазера.

Модуль разделения фигуры на слои вычисляет разрезы объекта, описанного STL-файлом [2], в горизонтальной проекции, в зависимости от заданной толщины слоя.

Модуль графического интерфейса позволяет визуализировать разрезы фигуры, маршрут лазера на каждом слое, а также выбирать STL-файл и задавать параметры печати удобным для пользователя способом.

Модуль авторизации предназначен для ограничения доступа к управлению принтером неавторизованных пользователей.

Разработка ведется для ОС Linux на языке C++. Для написания кода используется IDE CLion [3]. Для реализации графического интерфейса используется библиотека Qt [4]. Для организации процесса разработки качественного программного обеспечения используются следующие методы и инструменты:

1. Статический анализ [5] - CodeChecker, Codacy.
2. Динамический анализ - Valgrind, Google Sanitizers.
3. Модульное тестирование – Gtest [6].
4. Непрерывная интеграция / поставка (CI/CD) – Teamcity [7].

При выборе инструментов используется система поддержки принятия решений.

ЛИТЕРАТУРА

1. Султанова Ф. Р., Нам И. Э., Мирзахакимов С. Б. Технология селективного лазерного спекания (SLS) // Международный научный журнал «Инновационная наука» №10–2/2016 ISSN 2410–6070
2. STL (file format). [Электронный ресурс] Режим доступа: [https://en.wikipedia.org/wiki/STL_\(file_format\)](https://en.wikipedia.org/wiki/STL_(file_format))
3. CLion. A cross-platform IDE for C and C++. [Электронный ресурс] Режим доступа: <https://www.jetbrains.com/clion/>
4. Qt | Tools for Each Stage of Software Development Lifecycle. [Электронный ресурс] Режим доступа: <https://www.qt.io/>
5. Курочкин М. А.; Глазунов В. В.; Котлярова Л. П.; Моторин Д. Е.; Александрова О. В. Основы тестирования программного обеспечения: учебное пособие. Ч. 1.

6. GoogleTest - Google Testing and Mocking Framework (github.com) [Электронный ресурс] Режим доступа: <https://github.com/google/googletest>
7. TeamCity: the Hassle-Free CI/CD Tool by JetBrains. [Электронный ресурс] Режим доступа: <https://www.jetbrains.com/teamcity/?ysclid=lq4s90wgas454066488>

УДК 004.8

А.В. Примак, Е.М. Гадаев (1 курс магистратуры),
К.В. Пшеничная, к.т.н., доцент

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ ВЫБОРА ОПТИМАЛЬНЫХ ПАРАМЕТРОВ МАЛОМЕРНОГО СУДНА

В настоящее время актуально исследование возможности применения методов искусственного интеллекта при проектировании судов. В работе рассматривается задача выбора оптимальных параметров маломерного судна с использованием генетического алгоритма и искусственных нейронных сетей для аппроксимации целевой функции.

Задача оптимального выбора параметров маломерного судна формулируется следующим образом:

$$\begin{aligned} F(X, Fr) &\rightarrow \min, \\ X &= (x_1, x_2, x_3, x_4, x_5), \\ x_i &\in [a_i; b_i], i = 1, \dots, 5; \end{aligned}$$

где F – целевая функция (отношение остаточного сопротивления к весовому водоизмещению); X – вектор независимых переменных (параметров судна, подлежащих оптимизации); Fr – число Фруда (заданный параметр); $[a_i; b_i]$ – заданный диапазон изменения x_i ($i = 1, \dots, 5$). Составляющие вектора X включают следующие параметры судна: x_1 – коэффициент продольной полноты; x_2 – абсцисса центра величины; x_3 – отношение ширины к осадке; x_4 – отношение длины к ширине; x_5 – отношение длины к объемному водоизмещению в степени $1/3$.

Целевая функция данной задачи является сложной и нелинейной. Она представляется в виде аппроксимации зависимости остаточного сопротивления судна от его размеров, таких как длина, ширина, осадка и других параметров. В качестве модели аппроксимации была принята модель искусственной нейронной сети (НС). Для обучения НС использовались экспериментальные данные, полученные в результате проведения 308 полномасштабных экспериментов в лаборатории гидромеханики Делфтского технического университета [1]. Фрагмент экспериментальных данных представлен на рисунке 1.

-2.3	0.568	4.78	3.99	3.17	0.125	0.11
-2.3	0.568	4.78	3.99	3.17	0.150	0.27
-2.3	0.568	4.78	3.99	3.17	0.175	0.47
-2.3	0.568	4.78	3.99	3.17	0.200	0.78
-2.3	0.568	4.78	3.99	3.17	0.225	1.18
-2.3	0.568	4.78	3.99	3.17	0.250	1.82
-2.3	0.568	4.78	3.99	3.17	0.275	2.61
-2.3	0.568	4.78	3.99	3.17	0.300	3.76

Рисунок 1 – Фрагмент экспериментальных данных

Программная реализация НС осуществлялась на языке Python с применением фреймворка Tensorflow [2] и библиотеки Keras [3]. Рассматривалась полносвязная нейронная сеть прямого распространения, на вход которой подавались вектор X и число Фруда Fr , а на выходе определялось отношение остаточного сопротивления к весовому водоизмещению. В результате рассмотрения различных вариантов топологии НС была выбрана нейронная сеть [6-9-6-1], имеющая два скрытых слоя, девять нейронов в первом скрытом слое и шесть нейронов во втором скрытом слое.

Для определения оптимальных значений $x_1, \dots, x_5$, соответствующих наименьшему остаточному сопротивлению, в работе использовался один из самых известных эволюционных алгоритмов – генетический алгоритм (ГА) [4]. ГА имитирует процессы естественного отбора и эволюции с применением генетических операторов, в ходе выполнения которых преобразуется популяция, состоящая из возможных решений. Процесс преобразования популяции развивается в сторону улучшения значения функции приспособленности (целевой функции).

Генетический алгоритм был реализован в виде программы на языке Python, которая позволяет находить оптимальные параметры маломерного судна для заданного числа Фруда. При реализации алгоритма использовалась библиотека DEAP [5]. Общая структура программы представлена на рисунке 2.

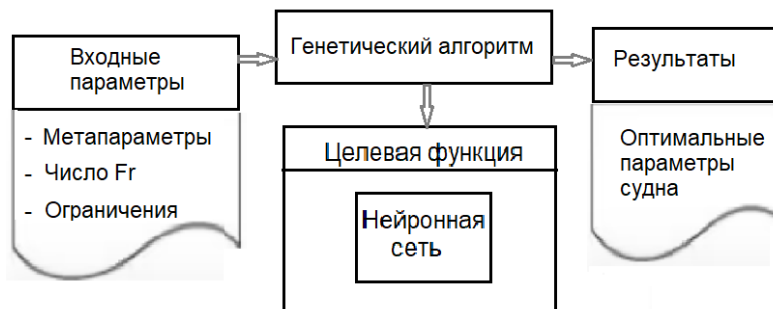


Рисунок 2 – Общая структурная программы

Фиксирование числа Фруда связано с тем, что это – наиболее сильно влияющий на сопротивление параметр, на фоне которого влияние остальных параметров значительно слабее. Оптимальные параметры судна были определены для следующих вариантов значений числа Фруда: 0,125; 0,150; 0,175; 0,200; 0,225; 0,250; 0,250; 0,275; 0,300; 0,325; 0,350; 0,375; 0,400; 0,425; 0,450.

Использование разработанной программы позволяет определять оптимальные характеристики маломерных судов на ранних стадиях проектирования.

ЛИТЕРАТУРА

1. Гидродинамика яхты. [Электронный ресурс]. Режим доступа: <http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics#> (дата обращения 12.03.2025).
2. Комплексная платформа для машинного обучения. Режим доступа <https://www.tensorflow.org/?hl=ru> (дата обращения 12.03.2025).
3. Keras: библиотека глубокого обучение на Python. [Электронный ресурс]. Режим доступа: <https://ru-keras.com/home/> (дата обращения 12.03.2025).
4. Вирсански Э. Генетические алгоритмы на Python / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020.
5. DEAP documentation. [Электронный ресурс]. Режим доступа: <https://deap.readthedocs.io/en/master/> (дата обращения 12.03.2025).

УДК 004.4

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

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ УПРАВЛЕНИЯ ЗАДАЧАМИ ПО РАБОТЕ С ДАННЫМИ В РАСПРЕДЕЛЕННЫХ ХРАНИЛИЩАХ

Организации различных отраслей промышленности заинтересованы в решении задач выявления ценной информации и скрытых зависимостей из большого массива данных для увеличения прибыли, снижения рисков и убытков [1-3]. Для решения этой задачи нужно выбрать подходящее распределенное хранилище данных. Затем разработчики ПО должны

реализовать интеграцию приложения с выбранными хранилищами. Также усложняется развертывание и поддержка сервиса – нужно думать не только о самом приложении, но и о дополнительной инфраструктуре.

Для эффективной разработки ПО необходим инструмент, позволяющий легко хранить, обрабатывать и анализировать данные. В данной статье описывается разработка платформы, позволяющая легко осуществить сторонним приложениям интеграцию с распределенными хранилищами.

Ключевые особенности:

- Поддержка хранения и поиска документов в распределенной файловой системе для хранения данных с потоковым доступом hdfs.
- Поддержка хранения и выполнения запросов в распределенной системе полнотекстового поиска Apache Solr.
- Поточковая обработка, которая включает в себя анализ, фильтрацию и валидацию данных по заданным заранее пользователем предикатам с использованием библиотеки Kafka streams.
- Возможность создания задачи мониторинга по заданным условиям поступающих данных с возможностью оповещения пользователя в случае нарушения этих условий.

Приложение представляет собой микросервисное приложение на языке Java [4,5]. В качестве основного фреймворка был выбран Spring Framework. Spring Framework предоставляет комплексную модель программирования и настройки для современных корпоративных приложений на базе Java на любой платформе развертывания [6].

Каждый микросервис выполняет свою роль. Список реализованных микросервисов, а также их взаимодействие между собой представлены на рисунке 1.

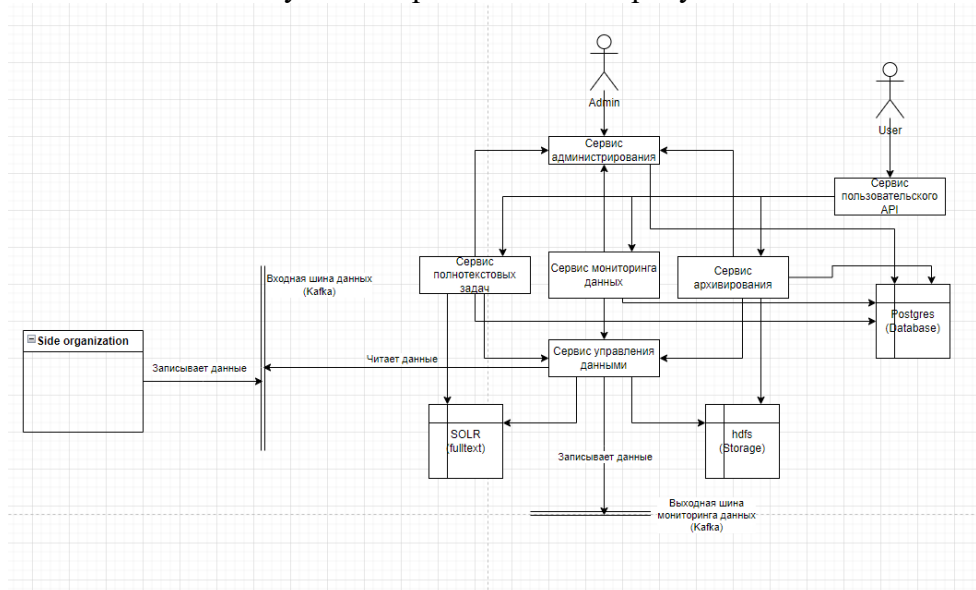


Рисунок 1 – Архитектура разрабатываемой платформы

Прототип системы успешно реализован и протестирован. По результатам можно сделать следующие выводы:

- Данная платформа не подразумевает графический интерфейс, однако правильно спроектированное пользовательское API позволяет легко использовать весь предоставляемый функционал.
- Данные, проходящие через платформу, и созданные в ней задачи надежно защищены настроенной при помощи Spring Security авторизацией и аутентификацией с использованием JWT токенов.
- Платформа надежно выполняет интеграцию с распределенными хранилищами.

Разработанная платформа упрощает разработку программного обеспечения для взаимодействия с большим потоком данных, который необходимо обрабатывать и хранить в

различных хранилищах. Разработчикам систем не нужно заботиться о развертывании, поддержке и разработке интеграции с хранилищами данных. С использованием платформы достаточно создать необходимую задачу.

ЛИТЕРАТУРА

1. Методы, алгоритмы и архитектуры распределенной обработки больших данных: учебное / [И. В. Никифоров, О. А. Юсупова, Н. В. Воинов, А. Д. Ковалев]; Санкт-Петербургский политехнический университет Петра Великого, Институт компьютерных наук и технологий, Высшая школа программной инженерии. — Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2024
2. Толстиков, Г. Н. Разработка Android приложения системы поддержки принятия решений / Г. Н. Толстиков, В. В. Амосов // 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. 130-132. – EDN GUXTHF.
3. Абисалов, Г. Р. Разработка Android приложения сервиса доставки и сбора продуктов / Г. Р. Абисалов, В. В. Амосов // 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. 4-5. – EDN DHWJZR.
4. Microservice Architectures from a Dependability Perspective. [Электронный ресурс] Режим доступа: https://www.researchgate.net/publication/389616502_SoK_Microservice_Architectures_from_a_Dependability_Perspective.
5. Большой гайд. Пишем микросервисы на Java и Spring Boot, заворачиваем в Docker, запускаем на EKS, мониторим на Grafana [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/682720/>
6. Документация по Spring Framework [Электронный ресурс]. Режим доступа: <https://spring.io/projects/spring-framework>

УДК 004.057.3

Р.А. Рагимов (3 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

СРАВНЕНИЕ СТАНДАРТОВ СЖАТИЯ ВИДЕО И АУДИО КОНТЕНТА ДЛЯ ПОТОКОВОЙ ПЕРЕДАЧИ И ВОЗМОЖНОСТИ, ОТКРЫВАЕМЫЕ СТАНДАРТОМ Wi-Fi 7 ДЛЯ ПОТОКОВОЙ ПЕРЕДАЧИ ИНТЕРАКТИВНОГО КОНТЕНТА В РЕАЛЬНОМ ВРЕМЕНИ.

Централизация вычислительных ресурсов снижает издержки в IT, чему способствует виртуализация и использование тонких клиентов (Thin client), сочетающих экономию с гибкостью. Однако проводные подключения и старые беспроводные технологии ограничивают мобильность, и тем самым гибкость, особенно для ресурсоемких задач.

Какие возможности предоставляет Wi-Fi 7 и какие методы сжатия информации можно использовать для эффективной передачи данных беспроводным путем в контексте тонких клиентов?

Краткий обзор стандарта IEEE 802.11be (Wi-Fi 7)

– Каналы шириной 320 МГц позволяют достичь теоретической пропускной способности вплоть до 46 Гбит/с по сравнению с 9.6 Гбит/с 160 МГц канала Wi-Fi 6 [2][3].

– 4K-QAM (Quadrature amplitude modulation) является нововведением стандарта IEEE 802.11be. Повышает плотность модуляции, позволяя передавать больше данных в том же диапазоне частот, что увеличивает эффективность передачи.

– Multi-Link Operation (MLO) в IEEE 802.11be позволяет устройствам одновременно использовать несколько частотных диапазонов для двусторонней передачи. Это увеличивает пропускную способность, снижает задержку и повышает надежность.

– Multiple Resource Unit (MRU) Обеспечивает более гибкое и эффективное распределение радио ресурсов, что улучшает использование спектра и снижает перегрузки в сети.

– 1024 Aggregate MAC Protocol Data Units (A-MPDU) Увеличенный размер A-MPDU снижает служебные накладные расходы, повышая общую эффективность передачи данных.

В целом стандарт нацелен на удовлетворение повышенных требований к беспроводной передаче медиа контента высокого качества, такие как lossless Аудио и 4/8K видео. Также данные нововведения нацелены на сферу виртуальной реальности, которой требуется быстрый отклик.

Сравнение с предыдущим стандартом в таблице 1.

Таблица 1 – Сравнение Wi-Fi 6 и Wi-Fi 7

	Wi-Fi 6	Wi-Fi 7
Стандарт	802.11ax	802.11be
Частоты (ГГц)	2.4, 5 (6 with WiFi 6E)	2.4, 5, 6
Пропускная способность (теор. макс.) (Гбит/с)	9.6	46
Ширина канала (МГц)	160	320
Метод модуляции	1024-QAM	4096-QAM
Поддержка Multi-Link Operation (MLO)	Нет	Есть

Сравнение в табличном виде представлено в таблице 2 для видео и 3 для аудио кодеков.

– h.264 (AVC) – Advanced Video Codec является высокоэффективным кодеком со степенью сжатия до 2000:1 (на практике меньше). Максимальное разрешение – 8K UHD) [4]. Использует микроблоки 16x16 (Y) и 8x8 (U/V) [1]. Широкая поддержка аппаратного ускорения.

– h.265 (HEVC) - High Efficiency Video Coding. Преемник h.264. Вдвое меньший размер данных при аналогичном качестве. Использует Coding Tree Unit (CTU) с размерами Luma 16, 32, 64.

– VP9 – открытый и бесплатный кодек от Google. До 50% лучше сжатие, чем H.264 (но на 10–20% медленнее). Поддерживает до 8K UHD.

– AV1 – современный, открытый и бесплатный кодек AOM. На 25–50% лучше сжатие, чем HEVC, и на 20–30% лучше, чем VP9. Поддерживает до 8K. Использует продвинутое разбиение блоков, улучшенное предсказание.

Таблица 2 – сравнение видео кодеков.

Кодек	Эффективность (против H.264)	Типичная задержка кодирования	Аппаратное ускорение
H.264	Средняя	Низкая	Обширное
H.265	Высокая (до 50% лучше)	Высокая	Ограниченное
VP9	Высокая (до 50% лучше)	Средняя	Ограниченное
AV1	Отличная (50% лучше)	Высокая	Низкая

Opus – Открытый и бесплатный кодек IETF. Очень низкая задержка, хорошее качество при низких битрейтах [5].

AAC (Advanced Audio Coding) – проприетарный кодек. Лучшее качество, чем MP3 при том же битрейте.

MP3 (MPEG-1 Audio Layer III) – старый и распространенный кодек. Приемлемое качество при невысоких битрейтах, но уступает AAC и Opus.

Таблица 3 – сравнение аудио кодеков.

Кодек	Эффективность (Кб/с)	Типичная задержка кодирования	Качество
Opus	6-510	2.5–50 мс	Отличное
AAC	16-320	20–100 мс	Отличное
MP3	32-320	60–100 мс	Хорошее

В результате – MLO в Wi-Fi 7 в сочетании с h.265 или AV1 и Opus может значительно расширить возможности тонких клиентов за счет увеличения портативности. Данный набор кодеков является наиболее привлекательным ввиду баланса задержки, качества, и сжатия.

ЛИТЕРАТУРА

1. Jayaratne, M., Gunawardhana, L. K. P. D., & Samarathunga, U. (2022), Comparison of H.264 and H.265. In: Engineering and Technology Quarterly Reviews, Vol.5, No.1, 17-24. ISSN 2622-9374. – Режим доступа: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4104336 (Дата обращения 15.03.2025)
2. Xiaoqian Liu, Yuhan Dong, Yiqing Li, Yousi Lin, Xun Yang and Ming Gan. IEEE 802.11be Wi-Fi 7: Feature Summary and Performance Evaluation [Электронный ресурс] // DOI: 10.48550/arXiv.2309.15951. – Режим доступа: <https://arxiv.org/abs/2309.15951> (Дата обращения 15.03.2025)
3. What is Wi-Fi 6? [Электронный ресурс] // Режим доступа: <https://www.intel.com/content/www/us/en/gaming/resources/wifi-6.html> (Дата обращения 15.03.2025)
4. Juan Gutiérrez-Aguado, Raúl Peña-Ortiz, Miguel García-Pineda, Jose M. Claver. (2020), Cloud-based elastic architecture for distributed video encoding: Evaluating H.265, VP9, and AV1 [Электронный ресурс] // DOI: <https://doi.org/10.1016/j.jnca.2020.102782> – Режим доступа: <https://www.sciencedirect.com/science/article/abs/pii/S1084804520302563> (Дата обращения 16.03.2025)
5. Opus (audio format) [Электронный ресурс] // Режим доступа: [https://en.wikipedia.org/wiki/Opus_\(audio_format\)](https://en.wikipedia.org/wiki/Opus_(audio_format)) (Дата обращения 16.03.2025)

УДК 004.92:796.332

Д.А. Рыженко (4 курс бакалавриата),
С.Э. Сараджишвили, к.т.н., доцент

РАЗРАБОТКА ПРОТОТИПА ИГРЫ: КОНТРОЛЛЕРА ПЕРСОНАЖА И ОСНОВНЫХ МЕХАНИК.

Современные видеоигры представляют собой сложные системы, требующие тщательной проработки как визуальной составляющей, так и игровых механик. Одной из ключевых задач на ранних стадиях разработки является создание интуитивно понятного и отзывчивого управления персонажем и основных механик. В данной работе представлен процесс разработки прототипа 3D футбольной игры, где игрок управляет одним персонажем, используя курсор мыши для определения направления движения.

Первым шагом в разработке управления персонажем стало преобразование координат курсора мыши в точку на плоскости игрового мира. Для этого был реализован алгоритм ray casting [1], который преобразует экранные координаты мыши в координаты на плоскости земли в 3D пространстве. Это достигается через вычисление матриц преобразования и пересечения луча, выпущенного из камеры, с плоскостью земли. Данный подход позволяет определить точку, указанную курсором, в направлении которой будет двигаться персонаж.

Для управления состояниями персонажа был реализован конечный автомат [2], представленный на рисунке 1. Персонаж может находиться в нескольких состояниях: бег, прыжок, подкат и удар головой в падении. Переход между состояниями осуществляется в зависимости от нажатия клавиш и текущего состояния персонажа. Например, при нажатии клавиши прыжка персонаж переходит в состояние "прыжок", а после завершения прыжка возвращается в состояние "бег". Такой подход позволяет легко управлять сложными анимациями и физическими взаимодействиями персонажа с окружением.

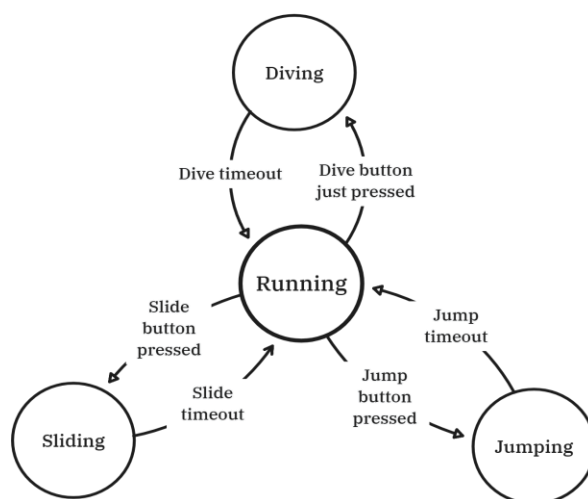


Рисунок 1 – Конечный автомат контроллера персонажа

Одной из ключевых механик игры является взаимодействие персонажа с мячом. Для этого используется физический движок Jolt [3], который обрабатывает столкновения между коллайдерами [4] персонажа и мяча. При пересечении коллайдеров персонаж может выполнить удар по мячу. Направление удара определяется положением курсора мыши относительно персонажа. Для реалистичного поведения мяча был применен эффект Магнуса, который учитывает вращение мяча и его влияние на траекторию полета.

Для удобства подбора внутриигровых параметров был реализован Immediate-mode GUI [5]. Этот интерфейс позволяет изменять параметры (сила удара, скорость персонажа, масса мяча, степень гашения угловой скорости и т.п.) в реальном времени без необходимости перезагрузки игры. Это значительно ускоряет процесс тестирования и балансировки игры.

Следующими шагами станут проведение плейтестов для выявления оптимальных внутриигровых параметров и дальнейшая доработка игрового процесса.

Разработка программы осуществлена с помощью языка Lua [6] и фреймворка LÖVR.

ЛИТЕРАТУРА

1. Lengyel, E. Mathematics for 3D Game Programming and Computer Graphics. – 3rd ed. – Boston: Cengage Learning, 2015. – P. 131-143.
2. Nystrom, R. Game Programming Patterns. – Genever Benning, 2014. – P. 87-104.
3. Jolt Physics Engine [Электронный ресурс]. Режим доступа: <https://jrouwe.github.io/JoltPhysics/index.html>
4. LÖVR [Электронный ресурс]. Режим доступа: <https://lovvr.org/docs/v0.18.0/World:overlapShape>
5. Immediate-mode GUI [Электронный ресурс]. Режим доступа: https://caseymuratori.com/blog_0001
6. Lua programming language. [Электронный ресурс]. Режим доступа: <https://lua.org/>

УДК 004.514

Н.Э. Сеферян (3 курс бакалавриата),
А.П. Маслаков, ст. преподаватель

РАЗРАБОТКА ДЕСКТОПНОГО ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ ИНФОРМАЦИЕЙ ПО КОМИТЕТАМ И УПРАВЛЕНИЯМ

В современных условиях государственные учреждения и крупные организации сталкиваются с необходимостью эффективного управления большими объемами данных, связанных с комитетами, управлениями, закрепленной техникой и сотрудниками. Такие данные включают информацию о расположении оборудования, его технических характеристиках, ответственных лицах и других важных аспектах. Однако существующие

решения для управления этими данными часто оказываются либо слишком дорогими, либо не адаптированными под специфические нужды государственных структур, особенно в условиях импортозамещения и ограниченного доступа к зарубежному программному обеспечению.

Исходя из вышесказанного, актуальность работы обусловлена необходимостью импортозамещения в условиях ограниченного доступа к зарубежному программному обеспечению. Приложение будет разработано с учетом требований государственных учреждений, что позволит использовать его без юридических ограничений. В настоящее время на рынке отсутствуют бесплатные решения, которые могли бы удовлетворить потребности государственных структур в управлении данными, что делает разработку данного приложения особенно важной [1, 2].

Целью работы является создание десктопного приложения для ведения и актуализации информации по комитетам и управлениям. Перед началом разработки, к приложению были сформулированы функциональные требования, основные из них:

- Возможность создания, обновления и удаления данных по комитетам и управлениям
- Поддержка древовидной структуры для организации информации по адресам, этажам и закреплённой технике
- Интеграция с ручными сканнерами для автоматизации ввода данных.
- Формирование отчётов для аналитики и принятия управленческих решений.
- Поддержка различных уровней доступа для обеспечения безопасности данных.

В качестве нефункциональных требований были выделены:

- Удобная навигация и интуитивность пользовательского интерфейса.
- Высокая производительность при работе с большими объёмами данных.
- Гибкость в настройке и адаптации под нужды пользователей.

Архитектурно, приложение представляет из себя клиент-серверное приложение, разбитое на несколько модулей (рисунок 1)



Рисунок 1 – Архитектура системы

Приложение разрабатывается на языке Java 21 и фреймворке Spring Boot, брокер сообщений RabbitMQ, система сборки – Maven, для тестов JUnit 5 и Mockito, СУБД – PostgreSQL, среда разработки - IntelliJ IDEA.

Рассмотрим немного подробнее каждый модуль системы.

Для хранения информации используется реляционная СУБД PostgreSQL. Данные организованы в виде таблиц, связанных через внешние ключи, что обеспечивает целостность данных и удобство их обработки [5].

Модуль сервера является сердцем приложения, отвечающим за взаимосвязь всех компонентов системы, и предоставляет REST API для передачи необходимых данных между ними.

Модуль пользовательского интерфейса: реализован с использованием JavaFX, что позволяет создать интуитивно понятный интерфейс с поддержкой древовидной структуры для отображения данных.

Модуль интеграции с ручными сканерами: для автоматизации ввода данных используется API сканеров, поддерживающих протокол USB HID. Данные со сканеров передаются в приложение в реальном времени и автоматически заносятся в базу данных.

Модуль формирования отчетов: на основе данных из базы данных формируются отчеты в формате PDF с использованием библиотеки Apache PDFBox. А также предоставляя гибкость в настройке и адаптации под нужды пользователей.

В результате, было разработано приложение, которое упростило управление данными в государственных учреждениях, позволило повысить производительность сотрудников за счет автоматизации процессов ввода и обработки данных, а также добиться улучшения качества данных за счет использования ручных сканеров и автоматизированных механизмов проверки. Гибкость и лёгкость формирования отчётов для аналитики позволили ускорить принятие управленческих решений.

Реализация проекта позволила не только упростить управление данными, но и создать инструмент, который может быть адаптирован под нужды различных организаций. В перспективе возможно расширение функционала приложения, включая интеграцию с другими системами и API, что сделает его еще более универсальным и полезным для пользователей.

ЛИТЕРАТУРА

1. Брюс Эккель. Философия Java. – СПб.: Питер, 2024. – 1168 с.
2. Мартин Роберт. Чистый код: создание, анализ и рефакторинг. – СПб.: Питер, 2022. – 464 с.
3. Кей Хорстманн. Библиотека профессионала. Том 2. Расширенные средства программирования. – М.: Лабиринт, 2020. – 864 с.
4. Джошуа Блох. Java. Эффективное программирование. – М.: Диалектика, 2022. – 464 с.
5. Евгений Моргунов. PostgreSQL. Основы языка SQL. – СПб.: Питер, 2019. – 336 с.
6. Иванов И.И. Современные подходы к разработке программного обеспечения. – М.: Издательство "Наука", 2021. – 320 с.
7. Петров П.П. Информационные системы в управлении. – СПб.: Издательство "Техносфера", 2023. – 280 с.

УДК 004

С.А. Свириденко (4 курс бакалавриата),
Е.А. Павлов, ст. преподаватель

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ДЛЯ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ УПРАВЛЕНИЯ СПОРТИВНОЙ ДЕЯТЕЛЬНОСТЬЮ

Современный спорт требует эффективного управления большими объемами данных, таких как результаты матчей, показатели тренировок, медицинские метрики спортсменов и многое другое. Существующие решения часто фрагментированы и не предоставляют гибких инструментов для анализа и генерации отчетов. Разрабатываемая система позволит централизованно хранить и агрегировать данные, что упростит их анализ и принятие решений. Это особенно важно для тренеров, менеджеров и других заинтересованных лиц, так как качество принимаемых решений напрямую влияет на результаты спортивной деятельности. Система может применяться в спортивных клубах, учебных заведениях и исследовательских учреждениях, что делает её разработку актуальной и востребованной.

Целью данной работы является разработка серверной части информационно-аналитической системы, которая централизует хранение данных, автоматизирует их обработку и предоставляет инструменты для анализа.

Задачи для решения поставленной цели:

1. Разработка модуля для централизованного хранения данных
2. Реализация механизмов динамической фильтрации данных
3. Автоматизация генерации отчетов в форматах Excel (журнал консультаций, результаты соревнований) и Word (данные по каждому спортсмену, спортивная результативность, обобщение по команде)
4. Обеспечение высокой производительности системы
5. Реализация модели аутентификации и авторизации

Технологии:

- **Java** (строгая типизация, экосистема библиотек).
- **Spring Framework**[1] (модульность, безопасность, интеграция с ORM).
- **PostgreSQL** (высокая производительность при работе с большими данными).
- **Hibernate ORM**[2] (удобство взаимодействия с базой данных).
- **QueryDsl**[4] (типо-безопасное построение динамических запросов для фильтрации данных)
- **Redis** (in-memory база данных для кеширования)
- **Liquibase** (миграция данных для БД)
- **Apache POI**[3] (генерация Excel отчётов)
- **Docker** (контейнеризация приложения для быстрого развёртывания)

Существующие решения.

Среди существующих решений можно выделить SAP Sports One [5] и Hudl [6]. Однако эти системы имеют ряд ограничений, таких как фрагментированность данных, сложность в настройке и отсутствие гибких инструментов для динамической фильтрации и автоматической генерации отчетов. Это делает их менее эффективными для оперативного управления спортивной деятельностью.

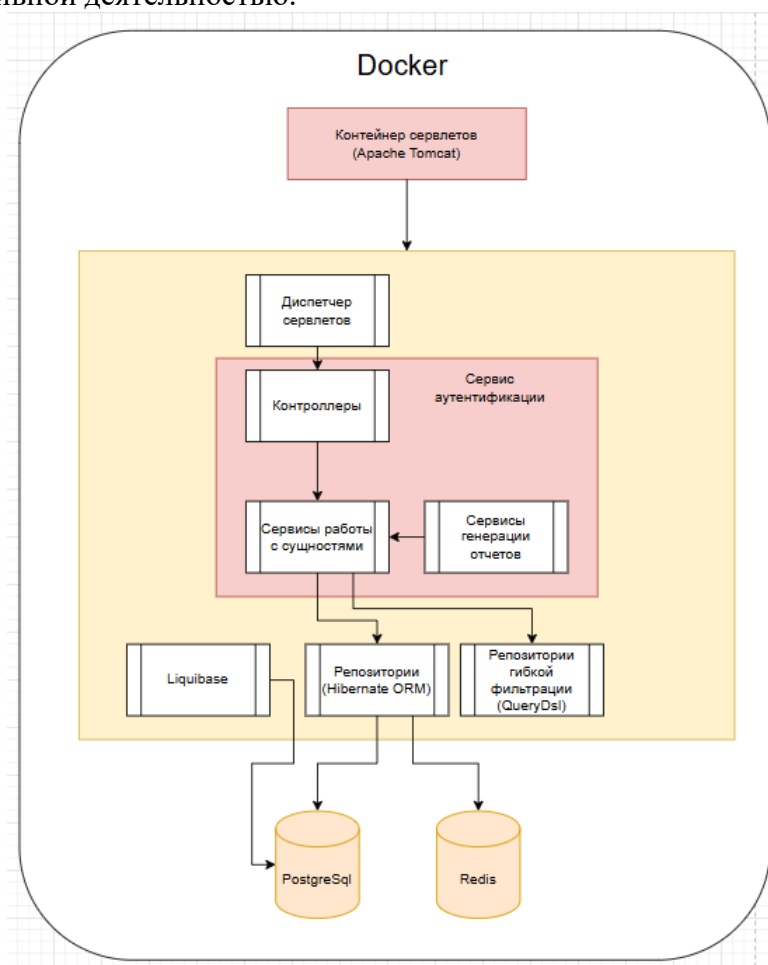


Рисунок 1 –Архитектура серверной части

Текущие результаты.

На текущий момент проведен сравнительный анализ существующих решений, спроектирована общая архитектура серверной части системы, реализована модель аутентификации и авторизации, а также разработаны репозитории для динамической фильтрации данных.

ЛИТЕРАТУРА

1. Spring Boot documentation. [Электронный ресурс] Режим доступа: <https://docs.spring.io/spring-boot/index.html>
2. Hibernate documentation. [Электронный ресурс] Режим доступа: <https://hibernate.org/orm/documentation/6.6/>
3. Apache POI documentation. [Электронный ресурс] Режим доступа: <https://poi.apache.org/apidocs/>
4. QueryDsl documentation. [Электронный ресурс] Режим доступа: http://querydsl.com/static/querydsl/3.4.2/reference/pdf/Querydsl_Reference.pdf
5. SAP Sport one. [Электронный ресурс] Режим доступа: <https://www.sap.com/products/technology-platform/sports-one.html>
6. Hudl. [Электронный ресурс] Режим доступа: https://www.hudl.com/en_gb/

УДК 004.932

А.М. Смола (2 курс бакалавриата),
Ю.В. Литвинов, к.т.н., ст. преподаватель

РАСПОЗНАВАНИЕ НАПРАВЛЕНИЯ ДОРОГИ ВДАЛИ ПО ВИДЕО В ADAS

Всё чаще для повышения безопасности на дороге и удобства управления автомобилем используются автоматизированные системы помощи водителю. ADAS (англ. Advanced Driver Assistance Systems) — комплекс систем помощи водителю, включающий, в частности, системы экстренного торможения, удержания в полосе, поддержания заданной скорости и дистанции до автомобиля впереди, а также оповещения о выходе из полосы.

Такие важные системы ADAS, как адаптивный круиз-контроль и система удержания в полосе, используют информацию о направлении дороги вдали. Причём важно знать направление дороги на как можно большем расстоянии, поскольку движение автомобиля по трассе происходит с большой скоростью, и в системе ADAS имеет место задержка, из-за чего данные с камеры являются немного устаревшими, когда доходят до контроллера. Поэтому для применения в реальных дорожных условиях алгоритм распознавания направления дороги вдали должен работать в режиме реального времени.

На сегодняшний день был достигнут заметный прогресс в задаче обнаружения дорожной разметки и сегментации дороги, в том числе и в режиме реального времени [1]. На это во многом повлияло то, что на смену традиционным подходам преимущественно на основе компьютерного зрения пришли методы глубокого обучения [2]. Однако авторы систем обнаружения дорожной разметки в основном делают упор менее чем на 40 ближайших к автомобилю метрах.

Таким образом, целью работы является реализация прототипа системы распознавания направления дороги на расстоянии не менее 40 метров. Для её выполнения были поставлены следующие задачи.

1. Выполнить обзор методов распознавания дорожной разметки и технологий для их модификации.
2. Реализовать алгоритм распознавания направления дороги вдали.
3. Провести тестирование алгоритма.

Реализованный алгоритм распознавания дороги вдали состоит из следующих этапов: выделения области интереса (дальней части дороги), применения алгоритма обнаружения дорожной разметки и получения вектора пикселей, соответствующих дорожной разметке.

Также была реализована версия алгоритма с программным увеличением изображения перед передачей в алгоритм обнаружения разметки.

Для реализации прототипа системы были использованы язык C++ и популярная библиотека компьютерного зрения OpenCV.

Область интереса в начале движения выставляется вручную, а затем автоматически смещается при повороте автомобиля. Для вычисления вектора смещения области интереса используется информация о перемещении точек на видео, отслеживаемых при помощи функции OpenCV для вычисления оптического потока `calcOpticalFlowPyrLK` [3].

Для распознавания дорожной разметки была использована TwinLiteNet — легковесная модель нейросети на основе архитектуры Encoder-Decoder [4].

Для улучшения качества изображения была применена модель ESPCN — модель с наилучшей скоростью работы среди реализованных в OpenCV [5].

Тестирование проводилось на 16560 изображениях датасета ApolloScapes. Тестовый стенд был оборудован процессором AMD Ryzen 5 5500U с интегрированным графическим процессором Radeon Graphics 12 и 16 ГБ оперативной памяти (RAM). В качестве базового решения был рассмотрен результат работы модели TwinLiteNet на всём изображении. Было установлено, что модификация системы путем улучшения качества изображения — даже с применением ESPCN (наиболее быстродействующей модели из рассмотренных) — значительно уменьшает скорость работы алгоритма. Так, алгоритм без улучшения качества изображения выдал результат по метрике FPS 7,5, а алгоритм с применением ESPCN — 2,5. В связи с этим в качестве финального решения был выбран алгоритм без этапа улучшения изображения. Метрики IoU и Accuracy измерялись на области интереса. Результаты по данным метрикам представлены в таблице 1, где M — математическое ожидание, σ — среднеквадратическое отклонение.

Таблица 1 – Результаты замера метрик

	IoU		Accuracy	
	M	σ	M	σ
Предлагаемое решение	31,5%	20,1%	94,4%	3,1%
Базовое решение	21,2%	16,7%	92,3%	4,6%

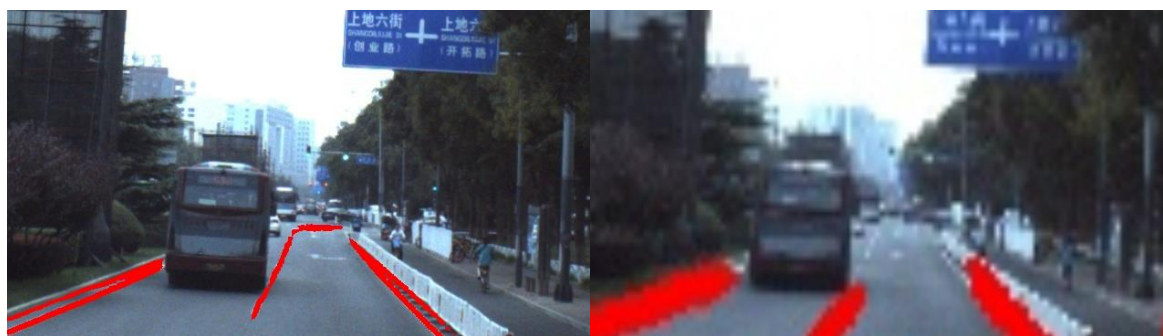


Рисунок 1, 2 – Результат предлагаемого алгоритма (слева) и базового (справа)

В результате было определено, что передача на вход модели распознавания дорожной разметки только области интереса улучшает точность определения направления дороги вдали по сравнению с обработкой всего изображения.

ЛИТЕРАТУРА

1. Kai Li Lim, Thomas Braunl. A Methodological Review of Visual Road Recognition Procedures for Autonomous Driving Applications — URL:

- https://www.researchgate.net/publication/332897463_A_Methodological_Review_of_Visual_Road_R_recognition_Procedures_for_Autonomous_Driving_Applications (дата обращения: 16. 03 .2025)
2. Lane Detection: The 3 types of Deep Learning (non-OpenCV) algorithms. [Электронный ресурс] Режим доступа: <https://www.thinkautonomous.ai/blog/lane-detection/> (дата обращения: 16. 03 .2025)
 3. OpenCV: Optical Flow. [Электронный ресурс] Режим доступа: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html (дата обращения: 16. 03 .2025)
 4. Quang-Huy Che, Dinh-Phuc Nguyen, Minh-Quan Pham, Duc-Khai Lam. TwinLiteNet: An Efficient and Lightweight Model for Driveable Area and Lane Segmentation in Self-Driving Cars. — URL: https://www.researchgate.net/publication/372487595_TwinLiteNet_An_Efficient_and_Lightweight_Model_for_Driveable_Area_and_Lane_Segmentation_in_Self-Driving_Cars (дата обращения: 16. 03 .2025)
 5. Super Resolution in OpenCV. [Электронный ресурс] Режим доступа: <https://learnopencv.com/super-resolution-in-opencv/> (дата обращения: 16. 03 .2025)

УДК 004.716

П.О. Сиротин (2 курс магистратуры),
В.Э. Шмаков, к.т.н., доцент

МЕХАНИЗМЫ ОПТИМИЗАЦИИ WAN

При проектировании топологии корпоративной компьютерной сети крупным компаниям, владеющим несколькими филиалами, которые находятся на большом отдалении друг от друга (в разных регионах), необходимо учитывать способ связи между данными филиалами для организации возможности передачи файлов между сотрудниками, централизованного доступа к сети Интернет, проведения видеоконференций и почтовых пересылок. Поскольку технология локальных сетей (LAN) не позволяет объединить в одну сеть устройства, что физически разделены большими расстояниями, приходится обращаться к применению глобальных сетей (WAN).

Стоит учитывать, однако, что WAN имеет ряд существенных недостатков и проблем, из-за которых рабочий процесс значительно усложняется (и замедляется), а именно: низкая пропускная способность сети (в среднем 10-20 Мбит/с, тогда как современные LAN имеют пропускную способность около 1 Гбит/с), высокие значения задержки (десятки миллисекунд), а также значительный процент потерь пакетов.

Целью данной работы является определение механизмов оптимизации WAN, которые должны быть встроены в прототип отечественного оптимизатора.

Для решения проблемы оптимизации WAN было разработано несколько механизмов, в дальнейшем объединенных в устройство (ПАК, программно-аппаратный комплекс), называемое оптимизатором WAN, тогда как компания HUAWEI (занимающаяся в том числе и выпуском сетевого оборудования), встроила механизмы оптимизации WAN в собственные маршрутизаторы (HUAWEI NetEngine AR600, AR6100 и др. [1]).

На рисунке 1 изображен возможный вид корпоративной сети для двух филиалов компании с использованием оптимизаторов WAN.

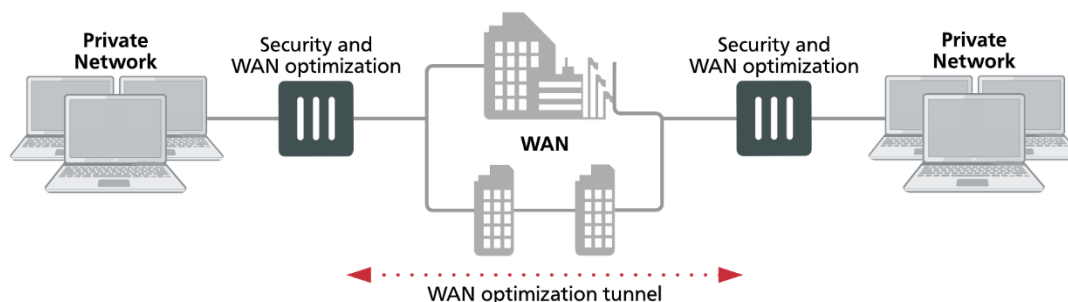


Рисунок 1 – Вариант реализации корпоративной сети с использованием оптимизаторов WAN [2]

Основными механизмами оптимизации являются дедупликация данных, TCP-оптимизация и оптимизация приложений.

Механизм дедупликации данных подразумевает использование ассоциативных массивов (с парами типа «ссылка – сегмент»), хранящихся на каждом из оптимизаторов. Перед передачей данных выполняется сегментирование (средний объем сегмента – 100 байт) и затем каждый сегмент индексируется. Далее оптимизатор проводит сравнение проиндексированных сегментов с уже имеющимися на собственном носителе данными. Возникновение совпадения при сравнении означает, что через данный оптимизатор соответствующий сегмент данных уже передавался, поэтому такой сегмент заменяется компактной ссылкой, чтобы снизить нагрузку на глобальную сеть. Уникальные сегменты (те, что раньше не передавались через данный оптимизатор) сжимаются, после чего происходит передача по WAN. В оптимизаторах SteelHead [3], выпускаемых компанией Riverbed (является одним из крупнейших вендоров на рынке оптимизаторов WAN), дедупликация данных реализована с применением алгоритма сжатия LZ [4]. Компания Riverbed заявляет, что посредством дедупликации данных использование полосы пропускания WAN сокращается на 60-95%, а объем передаваемых данных сокращается в 100 и более раз. Дедупликация является необходимым механизмом оптимизации WAN, поскольку через глобальную сеть регулярно передается избыточная информация (например, сотрудник А получает файл от сотрудника В, а затем отправляет обратно этот же файл с внесенными изменениями).

Стоит учесть, что изменение коэффициента сжатия или замена алгоритма сжатия возможны, но проводить их следует в соответствии с требованиями компании-заказчика, поскольку подобные модификации влияют на время, затрачиваемое на передачу данных через глобальную сеть.

Транспортный протокол передачи данных TCP слабо приспособлен для использования в глобальных сетях из-за присущей ему чувствительности к потерям пакетов и высокой задержке. В контексте использования протокола в глобальных сетях среди его недостатков принято выделять неэффективное использование пропускной полосы, медленный старт и механизмы контроля и избегания перегрузок.

Компания Cloudflare предложила собственную модификацию протокола TCP [5] для передачи данных в глобальных сетях, при использовании которой производительность протокола значительно возрастает (более, чем в 20 раз) в сравнении с традиционной реализацией протокола TCP. Таких результатов удалось добиться с помощью нескольких модификаций, а именно: замены алгоритма контроля перегрузок (был использован разработанный компанией Google алгоритм BBR [6] вместо стандартного для Linux алгоритма CUBIC), выборочных подтверждений (при потере пакетов передаваться заново будут не все пакеты, а лишь потерянные), динамического управления размером окна TCP (количества данных, которое может быть передано без подтверждения).

Оптимизация работы приложений заключается в оптимизации конкретных протоколов (как правило, прикладного уровня: HTTP, HTTPS, CIFS, NFS, TDS), что позволяет снизить количество операций по передаче данных, а также повысить производительность приложений.

Прототип отечественного оптимизатора WAN разрабатывается на основе размещенного в свободном доступе проекта SoloWAN [7] и механизмами оптимизации, описанными выше.

ЛИТЕРАТУРА

1. NetEngine AR600, AR6100, AR6200, and AR6300 V300R019 CLI-based Configuration Guide - WAN Optimization. [Электронный ресурс] Режим доступа: <https://support.huawei.com/enterprise/en/doc/EDOC1100189918/c51882e4/wan-optimization-configuration>
2. WAN optimization. [Электронный ресурс] Режим доступа: https://help.fortinet.com/fortiproxy/10/Content/Admin%20Guides/FPX-AdminGuide/200_Concepts/204_WANopt-topology.htm
3. Riverbed Optimization System (RiOS) 9.1 [Электронный ресурс] Режим доступа: <https://www.netwell.ru/upload/iblock/ea5/ea53bf438691689906fb89b7f13e22fe.pdf>

4. Алгоритм Зива-Лемпеля [Электронный ресурс] Режим доступа: https://www.opennet.ru/docs/RUS/inet_book/2/26/ziv_261.html
5. Freemon M. Optimizing TCP for high WAN throughput while preserving low latency. // The Cloudflare Blog, 2022. [Электронный ресурс] Режим доступа: <https://blog.cloudflare.com/optimizing-tcp-for-high-throughput-and-low-latency/>
6. bbr // GitHub. URL: <https://github.com/google/bbr>
7. SoloWAN Project. [Электронный ресурс] Режим доступа: <https://github.com/solowan/solowan?tab=readme-ov-file>

УДК 004.42

Н.А. Соколов (2 курс магистратуры),
С.А. Молодяков, д.т.н., профессор

РАЗРАБОТКА АЛГОРИТМА И ПРОГРАММЫ ДЛЯ ВЫЯВЛЕНИЯ ЧИТЕРСТВА В ОНЛАЙН-ШАХМАТАХ

Онлайн-шахматы, ставшие особенно популярными в последние годы благодаря доступности платформ и развитию технологий, сталкиваются с серьезной проблемой читерства. Читерство – это нелегальное использование шахматных программ в ходе игры, являющееся мошенничеством. Читерство наносит вред честной игре, снижает доверие к платформам и может разрушить репутацию шахматистов. Существует множество причин, вызвавших эту нехорошую тенденцию в мире шахмат. Современные шахматные программы, такие как Stockfish [1] и AlphaZero [2], обладают невероятной точностью и легко переигрывают даже сильнейших гроссмейстеров мира. Они доступны бесплатно и легко интегрируются с играми. Использование движков позволяет игрокам находить оптимальные ходы, что дает несправедливое преимущество.

Борьба с читерством длится уже очень долго. Игровые платформы внедряют защиту от использования движков, используя античит-системы [3]. На самых популярных шахматных платформах, таких как chess.com [4] и lichess.org [5], античит-системы работают достаточно эффективно, но, к сожалению, в большинстве неочевидных случаев требуют человеческого вмешательства, то есть ручной проверки модераторами. Это существенно замедляет процесс проверки пользователей и, соответственно, принятие мер по отношению к нечестным игрокам.

Целью работы является разработка алгоритма и программы, позволяющей автоматизировать выявление неочевидных случаев читерства.

Для достижения поставленной цели необходимо решение следующих задач:

1. Исследование существующих алгоритмов определения читерства и выявление их недостатков.
2. Разработка собственного алгоритма, обрабатывающего неочевидные случаи читерства.
3. Экспериментальное исследование созданных программных средств на реальных партиях.

Требования к программным средствам включают:

- загрузку партии в формате PGN, отображение списка ходов и анализ партии на читерство;
 - вывод статистики пользователя платформы lichess.org по нескольким последним играм (1, 5, 10, 50, 100 игр);
 - анализ последних сыгранных партий конкретного пользователя на читерство;
 - отображение результатов античит-проверки отдельной игры и пользователя;
- Признаки неочевидных случаев читерства включают:
- Стабильно одинаковое время, затрачиваемое на каждый ход независимо от сложности позиции
 - Резкая, нечеловечески быстрая реакция на неожиданный шах или нападение на фигуру

- Нахождение сильнейшего хода в каждый решающий момент в партии (особенно в сложных для человека позициях)
- Умышленно сделанные глупые ходы в начале партии, после которых начинается игра сильнейшими ходами
- Большая трата времени на очевидные для человека ходы
- Сильная игра при наличии времени, а с приходом цейтнота – ошибки (признак «ручного» читерства)

Структура разрабатываемых программных средств приведена на рисунке 1. Разработанные программные средства состоят из двух парсеров данных, двух анализаторов и графического интерфейса.

Для анализа отдельной партии достаточно прикрепить PGN-файл. Он попадет в парсер отдельной игры, после чего в удобном формате отправится в анализатор партии, где произойдет проверка на читерство. Результаты проверки будут отображены на графическом интерфейсе.

Для анализа пользователя необходимо ввести его ник и выбрать количество партий, которые необходимо проанализировать. Парсер пользователя отобразит статистику на экране, а также обработает все партии с помощью парсера отдельной игры. После этого анализатор пользователя создаст список «подозрительных» партий пользователя, которые можно будет проверить анализатором отдельной партии.

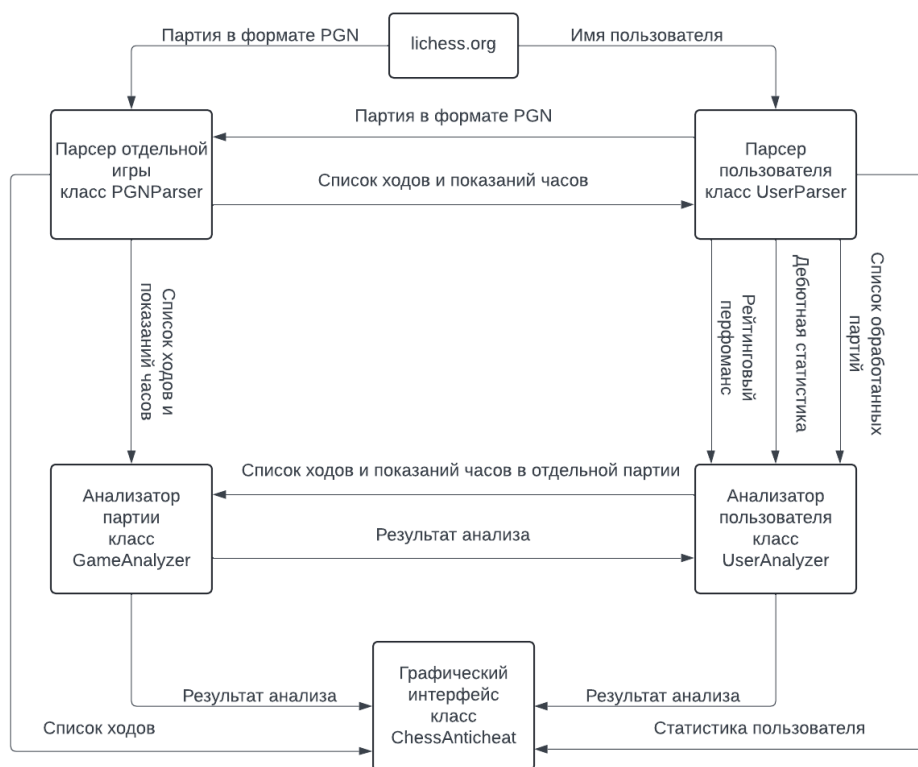


Рисунок 1 – Структура разрабатываемых программных средств

Программное обеспечение разработано на языке программирования C++ в среде разработки Microsoft Visual Studio с использованием фреймворка Qt.

Разработанные программные средства будут использованы при разработке античит-систем на онлайн-платформах для автоматизации выявления неочевидных случаев читерства.

ЛИТЕРАТУРА

1. Stockfish. Strong open-source chess engine [Электронный ресурс] URL: <https://stockfishchess.org/>
2. AlphaZero [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/AlphaZero>

3. Giovanna Ariel Ventre Computer move? Chess cheaters and the limits of algorithmic detection
<https://www.oxera.com/insights/agenda/articles/computer-move-chess-cheaters-and-the-limits-of-algorithmic-detection/>
4. Игровая платформа Chess.com [Электронный ресурс] URL: <https://chess.com/>
5. Игровая платформа Lichess.org [Электронный ресурс] URL: <https://lichess.org/>

УДК 004.42

Н.В. Сорин (2 курс магистратуры),
И.А. Селин, к.т.н., доцент

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИМУЛЯТОРА РАБОЧЕГО МЕСТА ШТУРМАНА

В любой профессии существует необходимость обучать новое поколение, так как люди уходят с работы, появляется что-то новое, да и сами средства обучения устаревают.

Это относится также к судоходству, существует необходимость обучать студентов управлению судном. Однако эта профессия специфична в том смысле, что слишком дорогостояще обучать всему студентов на настоящих судах. Это также не столь эффективно, так как тяжело воспроизвести различные окружающие условия, тем более что некоторые из них могли бы быть опасными для студентов. Например, попадание в шторм как тяжело воспроизводимо в реальных условиях, так и опасно. Таким образом по различным причинам обучения в различных училищах и учебных центрах производится на специальных компьютерных тренажерах[1], имитирующих реальные рабочие места экипажа корабля.

Такие тренажеры, как и почти любое программное обеспечение устаревают. Соответственно возникает необходимость в разработке нового тренажера, который позволил бы обучать всему на данный момент необходимому. Который бы был разработан на современном языке программирования, имел бы более современный интерфейс. И, в отличие от других существующих тренажеров, таких как NAUTIS Maritime Simulator[2], K-SIM Maritime Simulation[3], или Navi trainer[4], позволил бы обучать управлению конкретными моделями кораблей. Причиной является то, что в учебном центре, которому требуется тренажер, обучают управлению определенным моделям кораблей, что требует знания как внешнего вида приборов, так и их особенностей. Также стоит добавить, что существующие на рынке тренажеры распространяются на коммерческой основе, что также подтверждает необходимость в реализации нового тренажера. Данная работа посвящена разработке рабочего места штурмана корабля[5].

Целью работы является реализация рабочего места штурмана, которая позволила бы обучать штурмана задачам, которые он будет решать в будущем на реальном рабочем месте.

Для достижения этой цели необходимо решение следующих задач:

1. Изучить предметную область и составить требования к рабочему месту штурмана.
2. Спроектировать архитектуру рабочего места штурмана с учетом его взаимодействия с остальными частями тренажера.
3. Выбрать средства и технологии, необходимые для реализации рабочего места.
4. Реализовать вспомогательный проект с приборами, для встраивания их в рабочее место штурмана.
5. Реализовать рабочее место штурмана.
6. Провести тестирование разработанного продукта.
7. Написать инструкцию пользования рабочим местом.

Для разработки программного продукта стоит выбрать инструменты, при помощи которых он будет разрабатываться. В данном случае ими стали следующие средства и технологии:

Для реализации программного продукта был выбран язык C++[6], так как, с одной стороны, он достаточно высокоуровневый, поддерживает различные парадигмы, в том числе

объектно-ориентированное программирование. С другой стороны, язык имеет набор готовых библиотек, подходящих для работы с предметной областью.

Программный продукт реализуется при помощи фреймворка Qt[7]. Qt является кроссплатформенным фреймворком для создания графических приложений, что позволяет не сильно беспокоиться о зависимости от платформы. Еще одним преимуществом является то, что на нем создано огромное количество приложений и они имеют очень похожий интерфейс, из-за чего пользователи обычно привычны к нему.

В качестве IDE была выбрана IDE Qt Creator[8], поскольку данная IDE была специально разработана для работы с Qt.

Также была выбрана база данных PostgreSQL[9]. Данная база является одной из самых популярных. Так как данная задача не имеет каких-либо особенностей, требующих специальную базу, таких как быстрота запросов, особый вид хранения данных, или их очень большой объем, имеет смысл использовать самую распространенную базу, в работе с которой имеется наибольший опыт.

На рисунке 1 представлена концептуальная схема решения. Рабочее место штурмана включает в себя два окна: основное, на котором находится карта и интерфейс для взаимодействия с ней и второстепенное, содержащее панель с приборами, таблицу для работы с объектами на карте и фильтр по этой таблице. Приборы берутся из доп. библиотеки, подключаемой в виде подмодуля.



Рисунок 1 – концептуальная схема решения

ЛИТЕРАТУРА

1. Todor Coritarov, Assistant Professor, PhD, Nikola Vaptsarov, Naval Academ, Fostering learning innovations in maritime education: a synthesis of blended, project-based, and immersive learning approaches, Varna, Republic of Bulgaria 2024, DOI: 10.5281/zenodo.14531655
2. Hellenic shipping news VSTEP's Newly Launched Maritime Simulator Makes Ship Simulation Accessible for Everyone [Электронный Ресурс] Режим доступа: <https://www.hellenicshippingnews.com/vsteps-newly-launched-maritime-simulator-makes-ship-simulation-accessible-for-everyone/>
3. K-SIM maritime simulation [Электронный Ресурс] Режим доступа: <https://k-fleet.kongsbergdigital.com/products/k-sim/>
4. Навигационные тренажеры «Транзас» [Электронный Ресурс] Режим доступа: https://transas.ru/wp-content/uploads/2023/09/ntpro5000_prew.pdf
5. Zakirul Bhuiyan, Bridge Simulator as a Training platform for Future Remotely Controlled MASS Navigators, 2024, DOI: 10.1007/978-3-031-69437-0_15.
6. Документация по C++ [Электронный ресурс] Режим доступа: <https://en.cppreference.com/w/>
7. Junfeng Zhu, Research and Application of Qt Graphic View Framework, Advances in Engineering Technology Research, March 2023, DOI: 10.56028/aetr.4.1.189.2023

8. Sergey Bakurey, Svetlana Senotova, Development of mobile applications using the Qt framework, Modern Technologies and Scientific and Technological Progress, May 2023, DOI: 10.36629/2686-9896-2023-1-99-100
9. Jorge Domínguez Chávez, Fundamentos de PostgreSQL, November 2019, IEASS Editores, ISBN: 9789806366077

УДК 004.4

Д.М. Стретенский (4 курс бакалавриата),
Н.В. Воинов, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ РАБОТЫ С ПЕРВИЧНЫМИ БУХГАЛТЕРСКИМИ ДОКУМЕНТАМИ С ИСПОЛЬЗОВАНИЕМ JOOQ

Автоматизация обработки первичных бухгалтерских документов - важная задача для предприятий, поскольку ручная обработка документов отнимает много времени и чревата ошибками [1,2]. Для эффективного управления данными необходимы инструменты, позволяющие легко хранить, обрабатывать и генерировать документы в электронном формате. В данной статье описывается разработка программы для ведения первичного учета с использованием JOOQ [3].

Разрабатываемая программа предназначена для работы с такими бухгалтерскими документами, как счета-фактуры, накладные и акты выполненных работ. Ключевые особенности:

- Создание, редактирование и удаление документов, хранение данных в базе данных.
- Формирование отчетов и экспорт документов в PDF.
- Уведомление пользователя о незаконченном формировании

Приложение было разработано на языке Java с использованием библиотеки JOOQ для работы с базой данных SQLite, так как приложение распространяется локально на ПК в формате .exe. JOOQ позволяет создавать SQL-запросы на уровне кода [4], обеспечивая безопасность и производительность при работе с данными.

Использование JOOQ обеспечивает:

- Генерацию типизированных SQL-запросов.
- Автоматическое создание моделей на основе структуры базы данных.

Для создания пользовательского интерфейса используется JavaFX [5]. Формирование и экспорт документов в PDF осуществляется с помощью библиотеки iText [6].

Чтобы пользователь мог получать уведомления о забытом заполняемом документе, реализован микросервис [7,8], отправляющий сообщения-напоминания в мессенджере «Telegram».

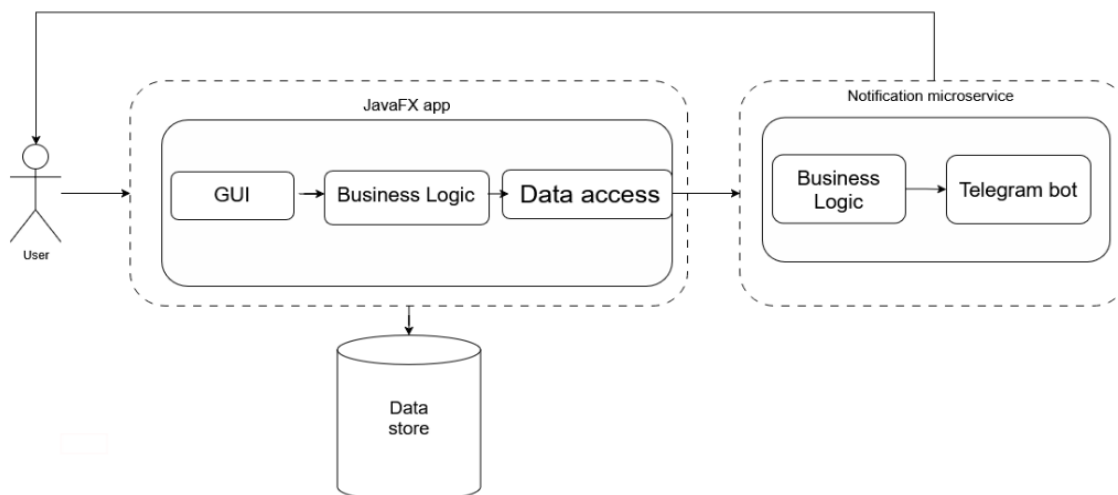


Рисунок 1 – Архитектура разрабатываемой системы

Прототип системы успешно реализован и протестирован. Оценка характеристик включает:

- Удобство использования: интуитивно понятный интерфейс и поддержка горячих клавиш ускоряют работу пользователя, а уведомления позволяют не забывать о заполнении важных документов.

- Безопасность: JOOQ позволяет создавать безопасные SQL-выражения.

Разработанная программа позволяет автоматизировать процесс работы с первичными бухгалтерскими документами, снизить риск ошибок и повысить эффективность бухгалтерского учета. Использование JOOQ дает гибкость в управлении базой данных и упрощает разработку. В перспективе возможна интеграция с внешними API и расширение функционала.

ЛИТЕРАТУРА

1. Толстиков, Г. Н. Разработка Android приложения системы поддержки принятия решений / Г. Н. Толстиков, В. В. Амосов // 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. 130-132. – EDN GUXTHF.
2. Абисалов, Г. Р. Разработка Android приложения сервиса доставки и сбора продуктов / Г. Р. Абисалов, В. В. Амосов // 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. 4-5. – EDN DHWJZR.
3. Васильев А. Н. Java. Объектно-ориентированное программирование: учебное пособие. СПб.: Питер, 2014. 400 с.
4. Anghel Leonard, jOOQ Masterclass: A Practical Guide for Java Developers To Write SQL Queries for Complex Database Interactions.
5. Прохоренко Н.А. JavaFX. Более полное руководство. 2020, 768 с.
6. Документация по iText PDF Documentation. [Электронный ресурс] Режим доступа: <https://api.itextpdf.com/iText5/java/5.5.13.4/>.
7. Средство автоматизации разработки CRUD микросервисов по спецификации REST интерфейса. [Электронный ресурс] Режим доступа: <https://elib.spbstu.ru/dl/3/2019/vr/vr19-907.pdf/view>
8. Microservice Architectures from a Dependability Perspective. [Электронный ресурс] Режим доступа: https://www.researchgate.net/publication/389616502_SoK_Microservice_Architectures_from_a_Dependability_Perspective.

УДК 004.42

Э.А. Старикиевич (4 курс бакалавриата),
А.В. Самочадин, к.т.н., доцент

АВТОМАТИЗАЦИЯ УЧЁТА И ПРОДАЖ ТОВАРОВ НА БОРТУ ПАССАЖИРСКИХ ПОЕЗДОВ

Пассажирские перевозки предъявляют высокие требования к оперативности и точности учёта товарных операций, необходимых для эффективного обслуживания пассажиров. Традиционные методы бумажного учёта нередко приводят к временным задержкам, увеличению числа ошибок и затрудняют аналитическую обработку данных [1], [2]. В связи с этим разработка специализированного мобильного приложения становится актуальной задачей.

Основой концепции приложения «Чайка» послужил анализ бизнес-процессов, выявивший реальные потребности проводников и железнодорожных компаний. Приложение

предназначено для интуитивного и быстрого ведения учёта товаров, регистрации продаж и формирования оперативных отчётов, что позволяет существенно сократить временные затраты и минимизировать риск ошибок [3]. Важным техническим решением стало обеспечение работы приложения как в условиях стабильного интернет-соединения, так и в офлайн-режиме с последующей автоматической синхронизацией данных.

Ключевым элементом разработки является современная архитектура, основанная на принципах микросервисного подхода. Реализована модульная система, в которую входят:

- мобильный клиент с локальным хранилищем данных (SQLite) для обеспечения непрерывной работы;
- серверная часть, осуществляющая обработку запросов и централизованное хранение информации (с использованием реляционных и аналитических баз данных);
- административный веб-интерфейс для управления и мониторинга процессов.

Схема архитектурного решения представлена на рисунке 1. В ней разграничены функциональные блоки, обеспечивая высокую связанность внутри модулей и минимальную – между ними. Такой подход гарантирует надёжность и простоту дальнейшей интеграции с существующими информационными системами железнодорожных компаний, на что, в конечном счёте, и направлен проект «Чайка» [4].

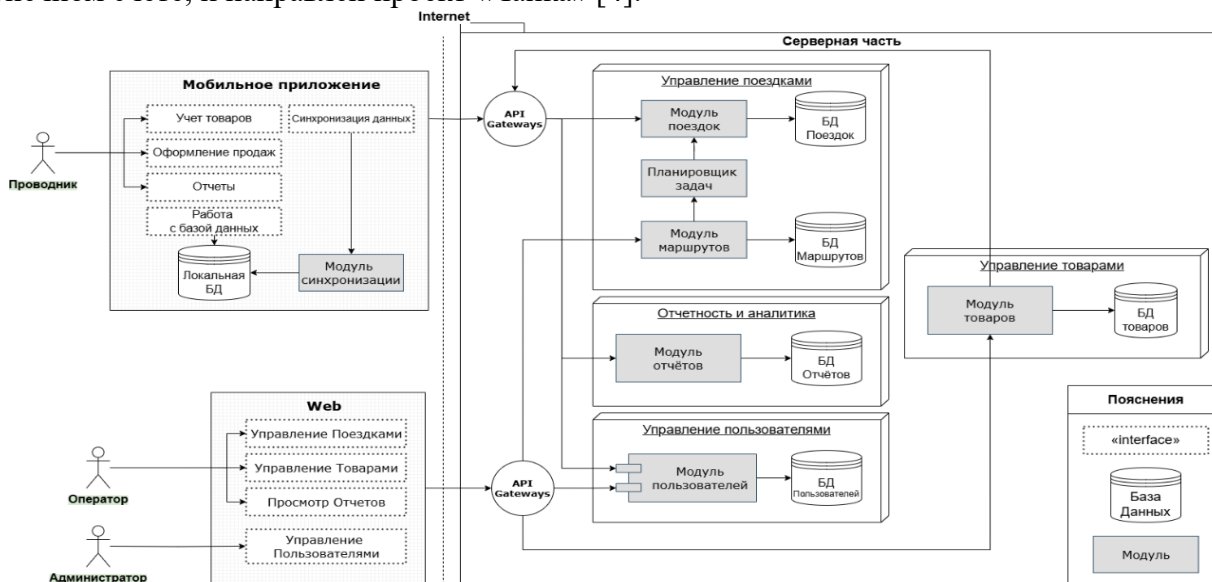


Рисунок 1 – Функциональная схема проекта «Чайка»

Для обеспечения целостности данных и устойчивости системы при работе в условиях ограниченной связи применяются механизмы локального кэширования Android, автоматической синхронизации и резервного копирования позволяют сохранять операции при отсутствии интернета. Это обеспечивает непрерывность бизнес-процессов и позволяет оперативно генерировать промежуточные и итоговые отчёты, необходимые для анализа данных и принятия управленческих решений. Оперативный сбор данных открывает возможности статистического моделирования, создания аналитических инструментов, способствующих оптимизации процессов и стратегическому планированию [5].

При разработке приложения «Чайка» учитывается важность создания интуитивно простого пользовательского интерфейса, который позволит минимизировать время обучения и повысить продуктивность проводников. Приложение должно обеспечить возможность быстро ориентироваться в функциональных модулях: от первоначальной загрузки данных и ведения товарного учёта, до оформления продаж и формирования детальных отчётов.

Для повышения надёжности системы и комфорта в её использовании, все основные сценарии работы приложения должны быть предельно автоматизированы. Так, следующая функциональность выполняется с минимальным участием пользователя:

- Обновление списка предлагаемых ЖД компанией товаров;
- Актуализация данных о текущем состоянии «склада» на борту отдельно взятого вагона;

- Составление промежуточных и конечного отчётов о поездке. Вычисление остатков, регистрация продажи, а также списания и добора товара. Подсчёт выручки;
- Отправление отчёта на сервис для аналитики по завершении рейса.

Другие сценарии использования приложения, такие как «Регистрация проводника» или «Регистрация на текущий рейс» реализованы таким образом, чтобы свести действия пользователя к интуитивно понятным и однозначным операциям.

Основной стек технологий проекта «Чайка» включает следующие компоненты:

- **Мобильное приложение с поддержкой офлайн-режима:** Разработка клиентского приложения **Kotlin** для платформы Android, который обеспечивает и упрощает локальное хранение данных (SQLite) и синхронизацию с сервером;
- **Серверная инфраструктура:** REST API для синхронного обмена данными. Разработка серверной части на **Golang** обеспечивает высокую производительность и масштабируемость;
- **Базы данных:** **PostgreSQL** для хранения структурированных данных. **Cassandra** – для обработки и хранения больших объёмов аналитической информации;
- **Система обмена сообщениями:** **RabbitMQ** – для асинхронной обработки и передачи сообщений между сервисами;
- **Управление доступом:** **ZITADEL** – для аутентификации и авторизации пользователей через протокол OAuth с расширением PKCE.

Являясь в перспективе корпоративным приложением, «Чайка» разрабатывается как современный и масштабируемый инструмент цифровой трансформации в сфере пассажирских перевозок. Оно сочетает в себе анализ бизнес-процессов, передовые технологии и модульную архитектуру, что позволит существенно повысить эффективность работы проводников и оптимизировать товарооборот железнодорожных компаний.

Проект выполняется при поддержке Фонда Содействия Инновациям.

ЛИТЕРАТУРА

1. Как вести учет транспортных перевозок. Электронный ресурс [Режим доступа]: <https://omegafuture.ru/resheniya-i-keysy/news/153/> (Дата обращения: 08.03.2025).
2. Основные принципы учета человеческого фактора в руководстве по проведению проверок безопасности полетов / Международная организация гражданской авиации. – М.: Международная организация гражданской авиации, 2010. – 147 с.
3. Виноградов Д. В. Разработка мобильных приложений и облачные сервисы. – Владимир: Изд-во ВлГУ, 2022. – 235 с.
4. Fowler M. Microservices [Электронный ресурс]. – [martinfowler.com](https://martinfowler.com/articles/microservices.html), 2014. – Режим доступа: <https://martinfowler.com/articles/microservices.html> (Дата обращения: 09.03.2025).
5. A Survey of Process-Oriented Data Science and Analytics for Supporting Business Process Management / A. Khan, A. Ghose, H. Dam, A. Syed. – arXiv preprint arXiv:2301.10398, 2023. – Режим доступа: <https://arxiv.org/abs/2301.10398> (Дата обращения: 11.03.2025).

УДК 004.453

Е.А. Суханова (4 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ПО ПОДБОРУ ФИЛЬМОВ

Разработка рекомендательных систем стала важной областью исследований в области науки о данных и искусственного интеллекта, поскольку эти системы играют решающую роль в персонализации пользовательского опыта в различных областях. В частности, киноиндустрия в значительной степени полагается на рекомендательные системы, помогающие пользователям находить релевантный контент из обширной библиотеки фильмов.

Актуальность работы заключается в растущем спросе на персонализированные рекомендации в киноиндустрии [1; 2], что можно объяснить общим ростом популярности

онлайн-кинотеатров. Несмотря на прогресс, существующие системы рекомендаций фильмов сталкиваются с несколькими нерешенными проблемами, которые требуют дальнейшего изучения [3]. Изучение и решение этих проблем имеет решающее значение для повышения удовлетворенности и вовлеченности пользователей в работу платформ, рекомендующих фильмы. Также из-за существования различных методик их выбор становится сложной задачей при построении рекомендательных систем (далее – РС), ориентированных на конкретную область. Кроме того, каждый метод имеет свой набор особенностей, преимуществ и недостатков, что поднимает еще больше вопросов, которые необходимо решить.

Целью является анализ и выявление наиболее эффективных подходов и инструментов для разработки системы рекомендаций фильмов и реализация рекомендательной системы, способной удовлетворить потребность пользователей в подборе контента. Для достижения поставленной цели были определены следующие задачи:

1. Проведение обзора литературы по рекомендательным системам, включающий в себя анализ и выбор существующих методов и инструментов для разработки РС.
2. Выбор, анализ и подготовка наборов данных.
3. Проектирование и создание рекомендательной системы, включающее в себя выбор и комбинирование нескольких алгоритмов рекомендаций.
4. Оценка эффективности и тестирование рекомендательной системы на разных наборах данных.

Для работы выбраны общедоступные наборы данных, отражающие взаимодействие пользователей с киноконтентом. Их анализ позволяет учесть разнообразие предпочтений и характеристик фильмов, что необходимо для обучения алгоритмов. Также наборы различаются между собой объемом и плотностью, что позволит протестировать разработанную систему при разных условиях.

Выделяют три типа рекомендательных систем: рекомендательные системы на основе содержания, рекомендательные системы на основе коллаборативной фильтрации и гибридные рекомендательные системы. Каждый из этих подходов имеет свои уникальные особенности, преимущества и недостатки, что делает их применимыми в различных сценариях [4].

Методы были проанализированы на их способность решения наиболее распространенных проблем в области рекомендательных систем (таблица 1), где значения «+» и «-» означают соответственно высокую производительность и неудовлетворительную. На основе анализа предпочтение отдано гибридной модели, так как она обеспечивает баланс между персонализацией и разнообразием рекомендаций.

Таблица 1 – Чувствительность методов рекомендательных систем к распространенным недостаткам РС [5; 6]

Метод РС	Холодный старт	Разреженность данных	Масштабируемость	Пузырь фильтров
Контентные системы	+	-	+	-
Методы коллаборативной фильтрации	-	+	+	-
Гибридные модели	+	+	+	-

По результатам предварительных экспериментов выявлено, что комбинация контентно-ориентированных и коллаборативных методов улучшает качество рекомендаций, использование кластеризации и инструментов распределенного вычисления улучшает

производительность рекомендательной системы, использование кластеризации в гибридной системе улучшает рекомендации за счет группировки похожих пользователей и объектов.

ЛИТЕРАТУРА

1. Мамиконян О. Количество подписчиков онлайн-кинотеатров в России растёт третий квартал подряд // ForbesLife / под ред. Forbes. — 2024.
2. Video Streaming Services Stats (2024). — URL: <https://explodingtopics.com/blog/video-streaming-stats>.
3. Karthikeyan P., Tejasvini C. Review of Movie Recommendation System // 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS).—Coimbatore, India, 2022.—С. 1538—1543.—DOI 10.1109/ICACCS54159.2022.9785014.
4. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions // IEEE Transactions on Knowledge and Data Engineering.—2005.—Т. 17, № 6.—С. 734—749.—DOI 10.1109/TKDE.2005.99.
5. Content-Based Recommender Systems Taxonomy / Н. Papadakis [и др.] // Foundations of Computing and Decision Sciences.—2023.—Т. 48, №2.—С. 211—241. — DOI 10.2478/fcds-2023-0009. — URL: <https://doi.org/10.2478/fcds-2023-0009>.
6. Collaborative Filtering Recommender Systems Taxonomy / Н. Papadakis, А. Papagrigoriou, С. Panagiotakis [и др.] // Knowledge and Information Systems.—2022.—Т. 64, № 1.—С. 35—74.—DOI 10.1007/s10115-021-01628-7.—URL: <https://doi.org/10.1007/s10115-021-01628-7>.

УДК 004.047

И.А. Соколов (1 курс магистратуры),
Е.С. Максина (2 курс магистратуры),
Е.И. Старунская (1 курс магистратуры),
И.В. Никифоров, к.т.н., доцент

ПРИМЕНЕНИЕ NLP ДЛЯ РЕШЕНИЯ ЗАДАЧИ АВТОМАТИЗАЦИИ СБОРА И ОБРАБОТКИ ОБРАТНОЙ СВЯЗИ

Обратная связь играет ключевую роль в развитии продуктов, сервисов и бизнес-процессов. Однако традиционный процесс сбора и обработки обратной связи сопряжен с рядом проблем [1, 2]:

- низкая вовлеченность – респонденты не заинтересованы в заполнении длинных опросов, что требует периодического напоминания со стороны организатора о необходимости пройти опрос.

- отсутствие гибкости – респонденты должны пройти опрос за один раз, что заставляет их откладывать прохождение.

- ограниченность форматов – для крупных опросов обратная связь фиксируется только в виде кратких текстовых ответов для возможности проведения аналитики.

- трудности анализа – ответы в свободной форме сложно структурировать и использовать для анализа.

Современные технологии, включая искусственный интеллект и автоматизированные коммуникационные платформы, позволяют решить обозначенные проблемы и повысить эффективность данного процесса за счет персонализированного взаимодействия с респондентами, гибкости ответов и автоматизированной обработки естественного языка. Разработка интеллектуальной системы сбора обратной связи позволит повысить полноту и качество данных, а также сделать процесс сбора и обработки обратной связи более удобным и ценным для бизнеса.

Целью работы является снижение трудоемкости процесса сбора и обработки обратной связи за счет автоматизации и использования NLP-технологий. Ручной анализ развернутых ответов занимает около недели работы (40 часов) из-за необходимости чтения,

структурирования и анализа текста. Разработанное решение позволяет сократить это время до 2 часов, увеличив скорость обработки данных в 20 раз.

Для достижения данной цели были поставлены следующие задачи, перечисленные ниже.

1. Рассмотреть существующие решения для сбора и обработки обратной связи, проанализировать их преимущества, недостатки и ограничения.

2. Выявить проблемы, возникающие при сборе, обработке и анализе обратной связи.

3. Изучить требования бизнеса к инструментам сбора обратной связи, определить ключевые критерии, необходимые для успешного внедрения решения в корпоративную среду.

4. Проанализировать существующие инструменты обработки естественного языка (NLP), оценить их применимость для анализа развернутых текстовых и голосовых ответов респондентов, выбрать оптимальные модели и методы для разрабатываемого инструмента.

5. Разработать интеллектуальное решение, включающее Telegram-бота для сбора обратной связи, систему обработки развернутых ответов, а также инструменты автоматизированного анализа данных.

6. Оценить эффективность разработанного решения, провести тестирование в реальной среде, проанализировать вовлеченность пользователей, точность обработки данных и применимость результатов анализа.

Сбор данных организован с помощью Telegram-бота, который упрощает процесс сбора обратной. Ключевые особенности:

1. Гибкость прохождения опроса – пользователи могут проходить опрос частями в течение заданного периода.

2. Автоматические напоминания – организаторам не нужно напоминать респондентам о необходимости прохождения опроса, так как бот периодически присылает напоминания, если опрос не завершен.

3. Поддержка голосовых развернутых ответов – респонденты тратят меньше времени на ответ, чем при наборе текста, при этом голосовые сообщения автоматически транскрибируются с помощью предобученных NLP-моделей.

Для анализа развернутых ответов были использованы предобученные NLP-модели [3-5], которые не требуют большого объема ресурсов, их можно установить локально и использовать без доступа к интернету, что позволяет разворачивать инструмент в корпоративной сети в соответствии с политиками информационной безопасности. При этом стоит учитывать, что многие модели анализа текста обучены на английском языке и не работают с русским языком, поэтому для обработки текстов на русском необходимо выполнять предварительный перевод на английский, это возможно сделать также при помощи предобученных моделей. Основные функциональные особенности разработанного инструмента представлены ниже.

1. Разделение отзыва на блоки – метод структурирования, который делит каждый отзыв с помощью ключевых слов на три смысловых блока: ожидания, фактический опыт, пожелания.

2. Анализ настроения отзыва – позволяет понять эмоциональную окраску высказываний пользователей для оценки их удовлетворенности. Анализ настроения реализован с помощью библиотеки VADER (предварительный перевод текста на английский MarianMTModel).

3. Классификация по категориям – автоматическая классификация отзывов по ключевым категориям реализована с помощью модели BART-large-MNLI. На вход принимается отзыв на русском, и возвращается словарь с классификацией.

4. Извлечение имен – автоматическое извлечение имен и фамилий, упомянутых в отзыве на русском языке с использованием библиотеки Natasha. На вход принимается текст отзыва и обрабатывается с помощью библиотеки Natasha. Используется модель NewsNERTagger для распознавания именованных сущностей. Функция возвращает список имен, найденных в отзыве.

5. Результаты анализа записываются в базу данных и имеют четкую структуру, что позволяет строить статистику и выявлять закономерности в обратной связи [5].

Опираясь на анализ актуальности автоматизации настройки инфраструктуры ИТ-проекта [7] при развертывании решения, для обеспечения стабильной работы системы сбора обратной связи можно использовать подходы автоматической балансировки нагрузки, описанные в работе [6]. Это позволяет эффективно распределять ресурсы между задачами обработки данных и внутренними процессами кластера.

Взаимосвязь всех компонентов итоговой системы для сбора и обработки обратной связи можно описать UML диаграммой последовательностей (рисунок 1).

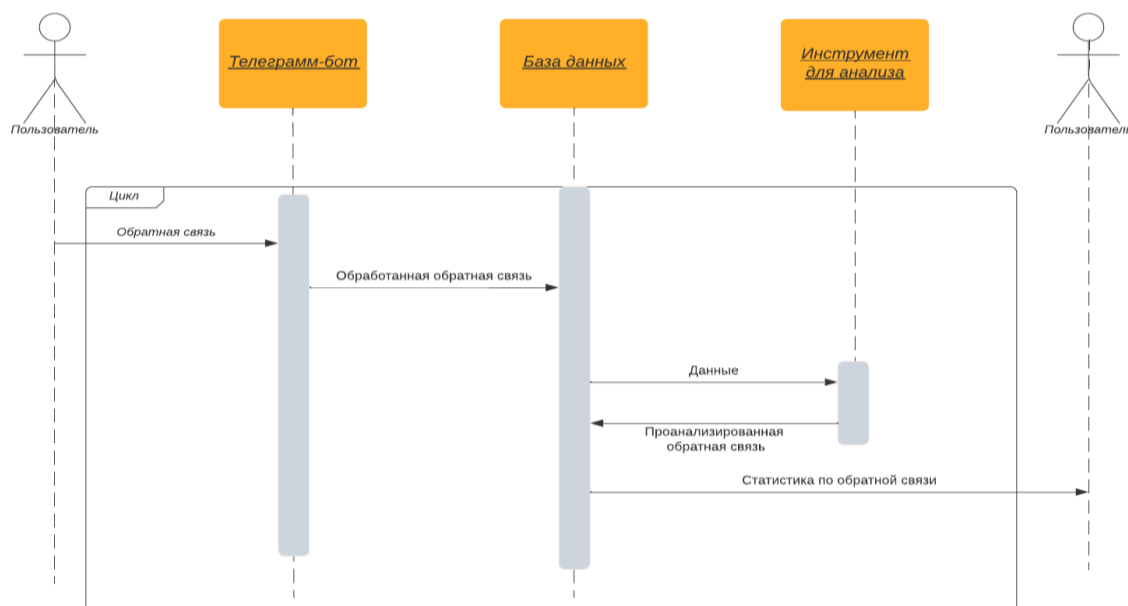


Рисунок 1 – Взаимосвязь компонентов

На данный момент система находится на этапе разработке. Работа выполняется при постановке задачи от группы компаний Газпромнефть.

ЛИТЕРАТУРА

1. Wu J. et al. Recursively summarizing books with human feedback // arXiv preprint arXiv:2109.10862. – 2021.
2. Cavalcanti A. P. et al. Automatic feedback in online learning environments: A systematic literature review // Computers and Education: Artificial Intelligence. – 2021. – Т. 2. – С. 100027.
3. Pérez J. M. et al. pysentimiento: A python toolkit for opinion mining and social nlp tasks // arXiv preprint arXiv:2106.09462. – 2021.
4. Hardeniya N. et al. Natural language processing: python and NLTK. – Packt Publishing Ltd, 2016.
5. Thanaki J. Python natural language processing. – Packt Publishing Ltd, 2017.
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 : 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. 79-80. – EDN QQAVXT.

ИСПОЛЬЗОВАНИЕ ЯЗЫКОВОЙ МОДЕЛИ ДЛЯ ГЕНЕРАЦИИ СИНТЕТИЧЕСКИХ ДАННЫХ ДЛЯ ДООБУЧЕНИЯ МОДЕЛИ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ПЛОЩАДКИ ПО ПРОДАЖЕ ТУРОВ ДЛЯ ПУТЕШЕСТВИЙ

Современные платформы сталкиваются с проблемой недостаточного объёма реальных данных для обучения рекомендательных систем [1, 2]. Это ограничивает их способность предоставлять персонализированные и точные рекомендации пользователям. Для решения этой проблемы предлагается использовать языковые модели (Language Models, LM), способные генерировать синтетические данные, такие как описания туров, действия пользователей и их предпочтения [4-6], что позволит расширить и улучшить обучающий набор данных для рекомендательной системы.

Целью проекта является решение проблемы нехватки данных для обучения рекомендательной системы, посредством генерации синтетических данных с использованием языковой модели, на пример T-lite-it-1.0 так как она имеет преимущества в рамках использования на русском языке [3, 5].

Для достижения этой цели необходимо решить следующие задачи:

1. Провести обзор существующих подходов к генерации синтетических данных с использованием языковых моделей [2, 5].
2. Настроить языковую модель для генерации синтетических данных, включающих описания туров, действия пользователей и их предпочтения [6, 9].
3. Адаптировать LLM для повышения точности и соответствия специфике данных [7, 9].
4. Оценить качество сгенерированных данных и их влияние на обучение рекомендательной системы [8, 9].

Архитектура системы машинного обучения, представленная на рисунке 1 включает несколько ключевых компонентов, ориентированных на использование языковой модели.

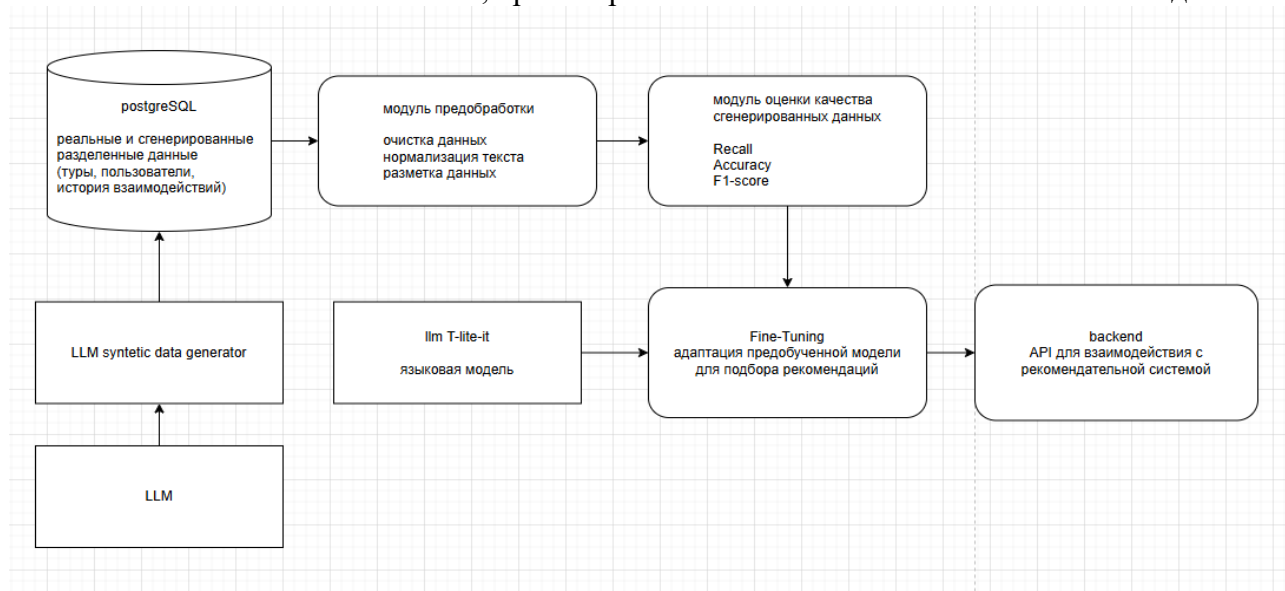


Рисунок 1 – Архитектура системы

Для генерации данных используется языковая модель. Эти данные включают в себя описания туров, действия пользователей на платформе и их предпочтения. Синтетические данные позволяют расширить обучающий набор и улучшить качество рекомендательной системы, особенно в условиях недостатка реальных данных.

После генерации данных проводится их оценка с использованием метрик, таких как точность (accuracy), полнота (recall) и F1-мера (F1-score) [8, 9], чтобы убедиться в их качестве и пригодности для обучения рекомендательной системы.

Таким образом, использование языковой модели для генерации синтетических данных позволяет не только расширить объём обучающих данных, но и повысить качество рекомендаций [3, 8], что в конечном итоге способствует улучшению пользовательского опыта и увеличению конверсии на платформе.

ЛИТЕРАТУРА

1. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems // IEEE Computer Society. – 2009. – Vol. 42. – No 8. – P. 30-37. – DOI 10.1109/MC.2009.263.
2. Sarwar B., Karypis G., Konstan J., Riedl J. Item-Based Collaborative Filtering Recommendation Algorithms // Proceedings of the 10th International Conference on World Wide Web. – 2001. – P. 285-295. – DOI 10.1145/371920.372071.
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. – DOI 10.1145/2959100.2959190.
4. Wu X., Zhang T., Rush A. M. Data Augmentation for Text Generation // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2020. – P. 6043-6054. – DOI 10.18653/v1/2020.emnlp-main.487.
5. Yoon J., Drumond L. N., Ruff M., Rendle S. Generating Synthetic Data for Recommender Systems with GANs // Proceedings of the 14th ACM Conference on Recommender Systems. – 2020. – P. 528-533. – DOI 10.1145/3383313.3418486.
6. Kumar R., Gupta A., Bengio Y. Synthetic Data Generation Using Large Language Models for Improving Recommendation Systems // Proceedings of the 2023 Conference on Neural Information Processing Systems (NeurIPS). – 2023. – P. 1-12. – DOI 10.48550/arXiv.2305.07623.
7. Sun F., Liu J., Zhang J., Xie X. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP). – 2019. – P. 1441-1451. – DOI 10.18653/v1/D19-1151.
8. Wu S., Zhang Y., Li C., Wang D. Language Models as Recommender Systems: Evaluating the Efficacy of Generative Approaches // Proceedings of the 2022 Conference on Artificial Intelligence (AAAI). – 2022. – P. 1-9. – DOI 10.1609/aaai.v36i4.20444.
9. Li J., Ren P., Chen Z., Ren Z., Wen J. R. Recommendation as Language Processing (RLP): A Unified Generative Framework // Proceedings of the 2022 Conference on Web Search and Data Mining (WSDM). – 2022. – P. 1-9. – DOI 10.1145/3488560.3498390.

УДК 004.3

К.И. Трубач, М.И. Кулик (3 курс бакалавриата),
Н.В. Малявко, П.А. Шлык (4 курс бакалавриата),
А.А. Иванюк, д.т.н., профессор

РЕАЛИЗАЦИЯ TRNG НА ОСНОВЕ DRAM PUF

В ходе данной работы был спроектирован TRNG, основанный на эффекте инициализации памяти случайными значениями при включении питания. После подачи питания был собран и проанализирован набор данных, который подтвердил наличие слабого DRAM PUF [1]. На основе этой PUF была создана программная модель TRNG, способная читать значения из памяти и осуществлять их постобработку, которая и перевела её в класс сильных PUF. Также произведена оценка характеристик (степень случайности и производительность) TRNG при различных методах постобработки.

Физически неклонировуемая функция (PUF) — это цифровая схема, позволяющая отображать множество запросов CH во множество ответов R уникальным и невоспроизводимым образом $CH \rightarrow R$. PUF широко предлагаются как средства безопасности

для идентификации и аутентификации устройств. Для DRAM PUF CH – это адрес, а R – значение по адресу CH при инициализации устройства. DRAM PUF основан на факте, что каждая ячейка DRAM-памяти после изготовления обладает уникальными характеристиками, которые приводят к случайным значениям при включении питания [2]. Последовательное включение и выключение памяти приводят к изменению начальных состояний некоторых ячеек памяти, что также даёт эффект невоспроизводимости [3]. В подтверждение гипотезы о DRAM PUF с помощью языка Python и окружения PYNQ был собран и проанализирован набор «сырых» данных объёмом $768\text{к} \times 8$ бит, для которых энтропия по Шеннону составила 7.39 бит, что является положительным результатом и обосновывает возможность построения TRNG на таком источнике.

TRNG (рисунок 1) был реализован программно и имеет состояния: запись в память (генерирование случайного числа), ожидание команды (от Cortex A9), ошибка (результат тестов на живучесть). Кроме этого, был введён набор команд: начало записи, переинициализация источника. Для реализации был выбран язык программирования Rust. Специфика его синтаксиса обеспечила исправление типичных ошибок ещё на этапе компиляции.

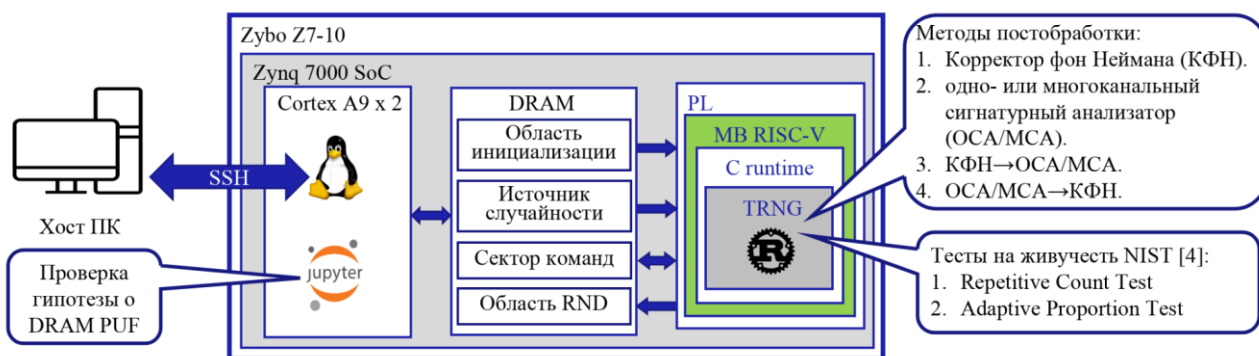


Рисунок 1 – Схема окружения, разработанного TRNG

По итогам анализа перечисленных комбинаций постобработки было определено следующее: КФН [5] значительно улучшает случайность выходной последовательности (до 7.9979 бит энтропии по Шеннону), но его использование не даёт переинициализировать TRNG без перезапуска; ОСА и МСА показали минимальную разницу в энтропии (7.9999 бит); при этом сочетание КФН и ОСА в любой комбинации практически не дало разницы по значению энтропии, как и само присутствие корректора. В итоге был сделан вывод о том, что использование схемы на рисунке 2, в реализации TRNG – это наиболее оптимальный подход, поскольку он предоставляет баланс между производительностью и качеством генерируемой последовательности (итоговое значение энтропии – 7.9999 бит, рисунок 3), и возможность восстанавливать работу системы, если последовательность вырождается, путём переинициализации ОСА вместо полной перезагрузки.

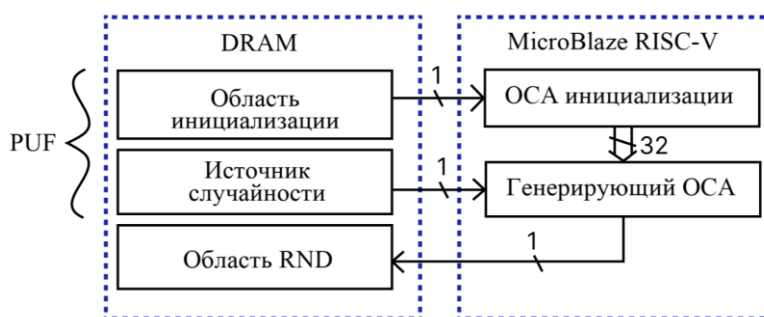


Рисунок 2 – Схема генерации случайных чисел в разработанном TRNG для случая постобработки с ОСА

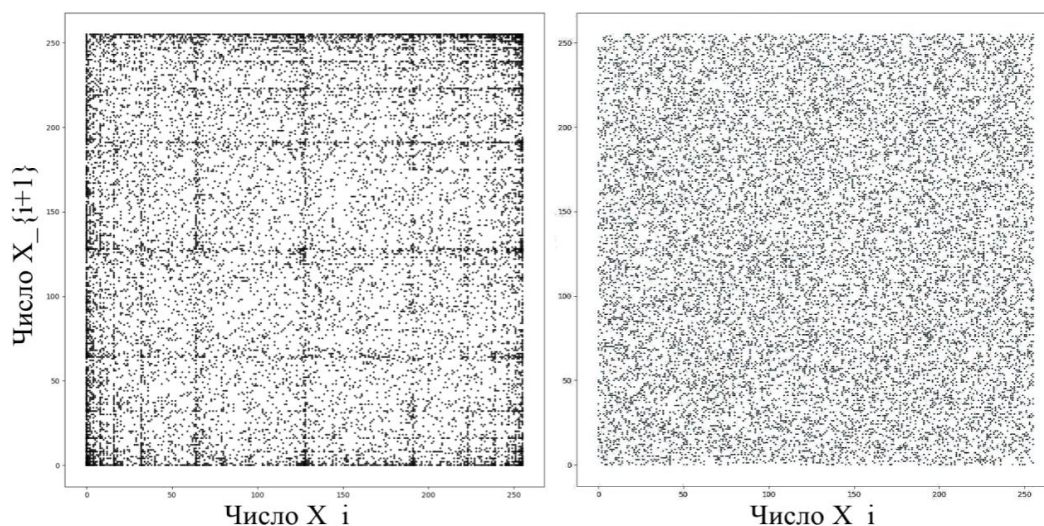


Рисунок 3 – Тест «распределение на плоскости» на основе первичных данных из DRAM (слева) и постобработанных данных (справа)

Для оценки производительности разработанного TRNG время измерялось посредством CRS-регистров (*time*, *timeh*) процессора RISC-V. Измеренная производительность составила 105 Кбит/с.

ЛИТЕРАТУРА

1. Пучков А. В., Иванюк А. А. Применение запоминающих устройств в качестве криптографических примитивов для интегральных схем программируемой логики // Информационные технологии и системы 2016 (ИТС 2016). – 2016. – С. 210-211.
2. Tehranipoor F. et al. DRAM-based intrinsic physically unclonable functions for system-level security and authentication // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. – 2016. – Т. 25. – №. 3. – С. 1085-1097.
3. Embedded Hardware // Ganssle, Jack; Tammy Noergaard; Fred Eady; Lewin Edwards; David J. Katz // Newnes – 2007 –. p. 106.
4. Sönmez Turan M. et al. Recommendation for the entropy sources used for random bit generation. – National Institute of Standards and Technology, 2016. – №. NIST Special Publication (SP) 800-90B.
5. Jun B., Kocher P. The Intel random number generator // Cryptography Research Inc. white paper. – 1999. – Т. 27. – №. 1-8. – С. 66.

УДК 004.94

А. Фернандез (2 курс магистратуры),
А.М. Газизова (4 курс бакалавриата),
А.В. Самочадин, к.т.н., доцент

ПЛАТФОРМА ДЛЯ ПАРАЛЛЕЛЬНОГО ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ

Дискретно-событийное моделирование (discrete-event simulation, DES) — имитационное моделирование, в котором функционирование системы представляется как последовательность событий во времени. При этом, не предполагается, что между последовательными событиями происходят какие-либо изменения в системе [1]. Событие, происходящее в определенный момент времени, приводит к изменению состояния системы. Основным недостатком дискретно-событийного моделирования является время выполнения программы на одном компьютере, с использованием последовательных алгоритмов [2]. Поэтому возникает необходимость в использовании нескольких процессоров для сокращения времени, затрачиваемого на моделирование системы. Параллельное дискретно-событийное

моделирование (parallel discrete-event simulation, PDES) представляет собой выполнение программы моделирования дискретных событий на нескольких процессорах с использованием параллельных алгоритмов.

Существуют два основных подхода преобразованию последовательной модели в множество взаимодействующих параллельных логических процессов: консервативный и оптимистический [3]. При консервативном подходе [4] не допускается нарушение причинно-следственных связей между событиями в процессе моделирования, что приводит к увеличению времени моделирования как за счет ожидания событий. Такой подход исключает возможность возникновения ошибок в процессе моделирования, однако накладывает ограничения на производительность. С другой стороны, оптимистический подход [5] действует более гибко. События обрабатываются сразу, даже если это может привести к ошибкам в причинно-следственной связи. Когда ошибки обнаруживаются, система выполняет откат, возвращая себя в корректное состояние. Такой подход позволяет процессам работать асинхронно и предоставляет больше возможностей для параллельной обработки. Существуют также гибридные алгоритмы [4], которые объединяют элементы консервативного и оптимистического подходов. Например, стратегии с ограничением агрессивности (агрессивные алгоритмы позволяют выполнение событий даже при недостатке нужной информацией [6]) (Reduced Aggressiveness) снижают вероятность откатов за счет контроля спекулятивных событий (события, которые были выполнены до синхронизации глобального времени), что уменьшает накладные расходы.

Целью данной работы является создания платформы, где одновременно могут использоваться несколько алгоритмов с разными подходами, таких как Time Warp (оптимистический подход), Window Racer (с ограничением агрессивности) [7] и Null Messages (консервативный подход). С помощью которой сможем оценить уровень производительности каждого алгоритма и классифицировать их.

Для разработки платформы дискретно-событийного параллельного моделирования был выбран язык C++ в сочетании с библиотекой OpenMPI, обеспечивающей межпроцессное взаимодействие через передачу сообщений.

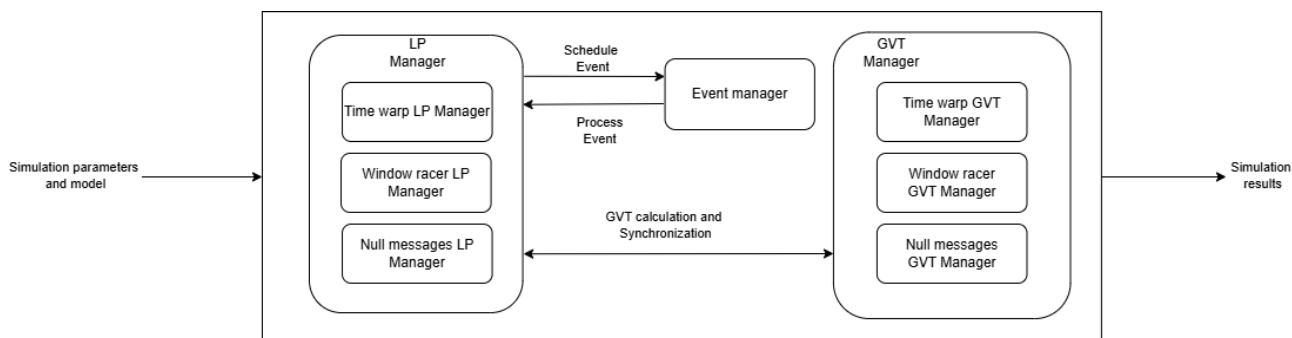


Рисунок 1 – Архитектура системы

Архитектура системы, представленная на рисунке 1, включает три ключевых модуля: LP Manager (менеджер логических процессов), GVT Manager (менеджер глобального виртуального времени) и Event Manager (менеджер событий). Взаимодействие между модулями происходит следующим образом: сначала платформа получает параметры моделирования и загружает модель. Затем LP Manager выбирает соответствующий алгоритм синхронизации (Time Warp, Window Racer или Null Messages) в зависимости от настроек.

Event Manager отвечает за планирование и обработку событий в соответствии с выбранным алгоритмом. Для обработки временных аномалий в оптимистическом подходе реализованы механизмы сохранения состояний и откатов с использованием анти-сообщений. GVT Manager выполняет расчет глобального виртуального времени (GVT), синхронизируя процессы и обеспечивая корректность выполнения моделирования.

В ходе реализации были проведены исследования, в рамках которых были собраны метрики для оценки производительности алгоритмов в различных условиях. В частности, анализировались сценарии с изменением расстояния между событиями и количества операций, выполняемых в каждом событии. Собранные данные позволяют провести сравнительный анализ и определить, при каких параметрах каждый из алгоритмов демонстрирует наилучшую эффективность. Это дает возможность оптимально выбирать подход в зависимости от специфики задачи, обеспечивая баланс между точностью, скоростью обработки и ресурсозатратностью.

ЛИТЕРАТУРА

1. Obermaier, C., Riebl, R., Al-Bayatti, A.H., Khan, S. and Facchi, C., "Measuring the Realtime Capability of Parallel-Discrete-Event-Simulations," *Electronics* 10(6), 636 (2021). DOI: <https://doi.org/10.3390/electronics 10060636>
2. Tanniru, Govardhan, "Parallel Discrete Event Simulation on Many Core Platforms Using Parallel Heap Event Queues." Thesis, Georgia State University, 2014. doi: <https://doi.org/10.57709/5544668>
3. Carothers, C.D., and Perumalla, K.S., "On deciding between conservative and optimistic approaches on massively parallel platforms," *Proc. Winter Simulation Conference*, 678–687 (2010). DOI: <https://doi.org/10.1109/WSC.2010.5679119>
4. Ferscha, A. and Tripathi, S., "Parallel and Distributed Simulation of Discrete Event Systems," *Handbook of Parallel and Distributed Computing*, McGraw-Hill, 15-50 (1995).
5. Pellegrini, A. and Quaglia, F., "A fine-grain time-sharing time warp system," *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 27(2), Article 10 (2017). DOI: <https://doi.org/10.1145/3013528>
6. Piccione, A., Andelfinger, P., & Pellegrini, A. (2023). Hybrid Speculative Synchronisation for Parallel Discrete Event Simulation. In *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '23)*, June 21–23, 2023, Orlando, FL, USA (pp. 1–12). ACM. <https://doi.org/10.1145/3573900.3591124>
7. Andelfinger, P., Köster, T., and Uhrmacher, A., "Zero Lookahead? Zero Problem. The Window Racer Algorithm," *Proc. ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS '23)*, 1–11 (2023). DOI: <https://doi.org/10.1145/3573900.3591115>

УДК 004.453

С.В. Фирсов (2 курс аспирантуры),
А.А. Теренин, к.т.н, доцент

LIVE-INDEPENDENT DYNAMIC PATCHING (LIDP): МЕТОДОЛОГИЯ ДИНАМИЧЕСКОГО ПАТЧИНГА .NET ПРИЛОЖЕНИЙ В КОНТЕЙНЕРИЗИРОВАННЫХ СРЕДАХ С ПРИМЕНЕНИЕМ IL WEAVING.

Контейнеризированные среды и системы их управления заняли свою нишу в разворачивании, управлении, и разработке программного обеспечения. Угрозы и уязвимости, которым подвержены такие среды имеют свою специфику в силу архитектуры построения систем [1]. Одной из таких угроз является – неизменяемость образа контейнера после разворачивания приложения без его пересборки [2], публикации в реестре образов и в следствие этого - проблема обновления работающих приложений без значительного времени простоя. Традиционные методы обновления, такие как постепенное обновление (rolling updates) или сине-зеленое развертывание (blue-green deployment), хотя и эффективны для многих сценариев, не всегда оптимальны при необходимости быстрого исправления критических ошибок или внедрения небольших изменений функциональности в действующие системы. В таких случаях требуется перезапуск контейнеров, что может привести к потере текущего состояния, временной недоступности сервиса и увеличению времени отклика.

В зависимости от сложности приложения и его размеров, процессов Software Development Life Cycle (SDLC) и внутренних процедур по Quality Assurance в компании, время

выпуска новой версии образа контейнера может достигать часа. Данный аспект никак не позволяет своевременно среагировать на обнаруженные CVE, кроме как полностью прекратить работу приложения или сервиса, подверженного угрозе компрометации.

Целью данной работы является разработка архитектуры и подхода доставки патча приложения на платформе .NET в контейнеризированную среду и применение его в процессе выполнения (runtime), которое не требует пересборки контейнера и его перепубликации.

Layer-Independent Dynamic Patching (LIDP) представляет собой инновационный подход, разработанный для устранения этих ограничений путем динамического изменения исполняемого кода .NET приложений в runtime. LIDP использует технологию IL weaving [3] (процесс модификации IL (Intermediate Language) кода, который является промежуточным представлением .NET программ перед компиляцией в машинный код) для внесения изменений в работающий процесс без его перезапуска, что особенно ценно в высоконагруженных системах и критически важных сервисах. Архитектура LIDP подхода представлена на рисунке 1.

Для реализации данной цели были выполнены следующие задачи:

1. Проанализированы текущие доставки изменений в контейнеризированные среды и разработана новая архитектура для размещения и применения информации о патче в интеграции с оркестратором контейнеров Kubernetes.

2. Исследован и разработан механизм реализации технология IL Weaving осуществляющий применение кода патча и целевое изменение приложение в runtime.

3. Реализован данный подход как прототип компонента для системы предотвращения вторжений (IPS)

Особенности IL Weaving в LIDP:

– **Runtime Weaving.** В отличие от традиционного IL weaving, происходящего на этапе сборки, LIDP применяет weaving непосредственно в runtime.

– **Гранулярный подход.** Изменения вносятся на уровне отдельных методов или даже инструкций IL.

– **Безопасное внедрение.** Архитектура гарантирует проверку совместимости перед применением.

– **Аудит изменений и возможность отката.** Помимо модификации IL, система производит аудит всех операций, сохраняет инструкции оригинального метода и позволяет выполнять откат изменений.

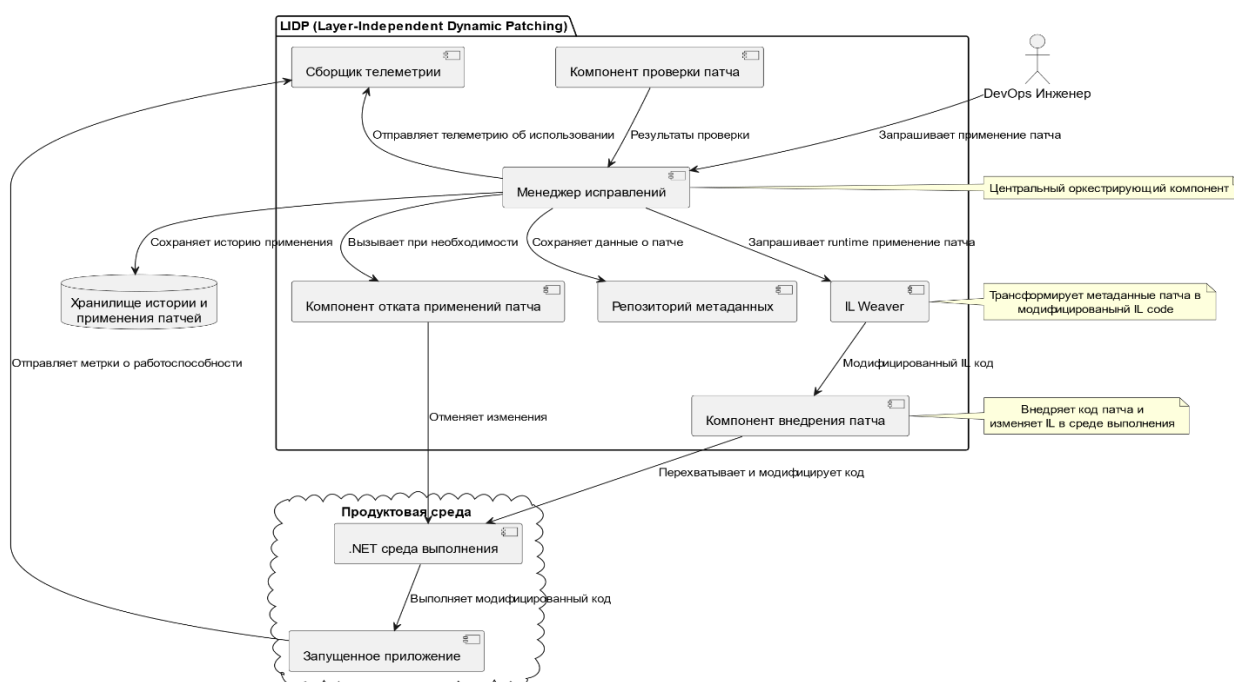


Рисунок 1 – Архитектура Layer-Independent Dynamic Patching системы

Для практического использования разработанных архитектуры и подхода, целевые приложения, желающие поддерживать LIDP должны быть интегрированы с предоставляемой библиотекой, реализующей LIDP контроллер [4].

Далее, процесс доставки изменений до работающего приложения в продуктовой среде с использованием оркестратора Kubernetes [5] представлен на рисунке 2.

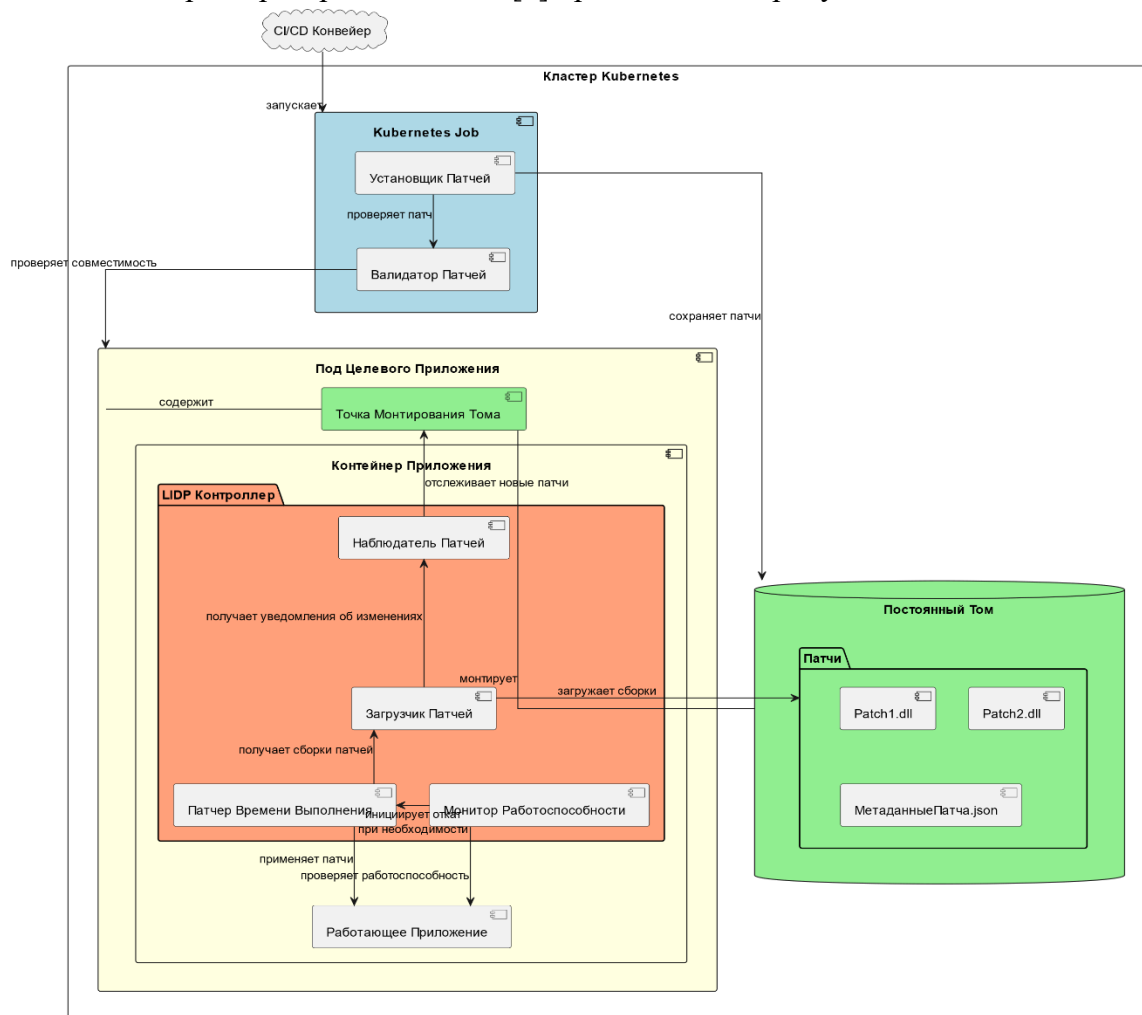


Рисунок 2 – Процесс доставки патча в архитектуре системы LIDP

ЛИТЕРАТУРА

1. Фирсов, С. В. Защита контейнеризованных сред. Анализ специфики уязвимостей, направленных на нарушение границ контейнера (Container Escape Attacks) и разработка методов их определения с использованием собственных реализаций Container Runtime Interface (CRI) и с Extended Berkeley Packet filter (EBPF) для ядра Linux / С. В. Фирсов // Современные сетевые технологии - 2024 (MoNeTeC) : Сборник трудов Пятой международной научно-технической конференции, Москва, 29–31 октября 2024 года. – Москва: Издательство Московского университета, 2025. – С. 68-74. – EDN NNUHZZ.
2. “Container Image Immutability and the Power of Metadata,” 2016. <https://www.cloudbees.com/blog/container-image-immutability-power-metadata>
3. “IL Weaving Technique,” 2023. <https://www.codecrafting.tips/code-chronicles-21-il-weaving-technique/>
4. Э. Лок, ASP.NET Core в действии. Третье изд. 2024. ISBN: 978-5-93700-183-2
5. J. Vyas, C. Love, Core Kubernetes. 2022. ISBN-13978-1617297557

ИСПОЛЬЗОВАНИЕ ЯЗЫКОВОЙ МОДЕЛИ В РАМКАХ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ТУРОВ ПОЛЬЗОВАТЕЛЮ НА ПЛОЩАДКЕ ПО ПРОДАЖЕ ТУРОВ ДЛЯ ПУТЕШЕСТВИЙ

Пользователи современных платформ по продаже туров для путешествий сталкиваются с проблемой сложности поиска и выбора подходящего тура из-за огромного количества предложений. Традиционные рекомендательные системы ограничены в возможностях анализа неструктурированных данных, таких как текстовые запросы, описания туров и отзывы [1]. Для решения этой проблемы предлагается использовать языковые модели (Language Models, LM), способные анализировать текстовую информацию и генерировать персонализированные рекомендации на основе контекста.

Целью проекта является повышение качества рекомендаций туров для путешествий за счёт интеграции языковой модели в рекомендательную систему.

Для достижения этой цели необходимо решить следующие задачи:

- Провести обзор существующих подходов к использованию языковых моделей в рекомендательных системах.
- Реализовать выбранный подход на платформе по продаже туров.
- Обучить модель на двух наборах данных: синтетически сгенерированных и реальных пользователей.
- Оценить эффективность системы и продемонстрировать улучшение качества рекомендаций.

Архитектура системы машинного обучения представлена на рисунке 1 включает несколько ключевых компонентов, ориентированных на реализацию языковой модели T-lite-it 1.0. Для повышения точности и адаптации модели к специфике данных, будет применяться процесс Fine-Tuning. Современные исследования показывают, что Fine-Tuning позволяет значительно улучшить качество рекомендаций за счёт адаптации модели к конкретной предметной области [2].

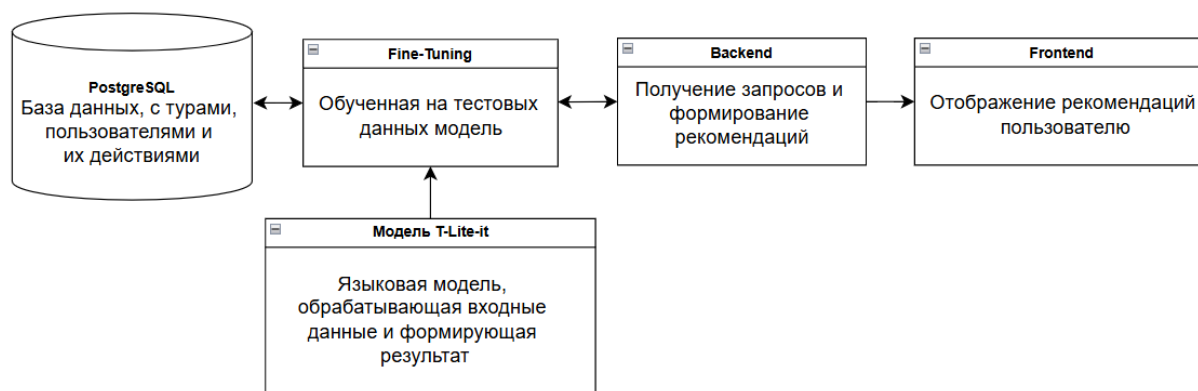


Рисунок 1 – Архитектура системы

Для обучения и тестирования модели используются два набора данных. Реальный набор данных, включающий запросы, отзывы и предпочтения пользователей платформы, слишком мал для полноценного обучения модели, что ограничивает её способность обобщать и адаптироваться к разнообразным сценариям [3]. Поэтому используется набор синтетически сгенерированных данных, который позволяет расширить охват обучения и улучшить качество модели.

После обучения модели на обоих наборах данных проводится сравнительный анализ её качества с использованием метрик, таких как точность (accuracy), полнота (recall) и F1-мера (F1-score). Исследования показывают, что комбинация синтетических и реальных данных позволяет достичь более высокой точности рекомендаций по сравнению с использованием только реальных данных [2].

Таким образом, использование языковой модели в рекомендательной системе туров позволяет не только улучшить качество персонализации, но и повысить удобство взаимодействия пользователей с платформой, что в конечном итоге способствует увеличению конверсии и удовлетворённости клиентов.

ЛИТЕРАТУРА

1. Иванов, Д. В., Петров, К. Л. Современные подходы к построению гибридных рекомендательных систем // Вестник компьютерных и информационных технологий. – 2019. – № 5. – С. 12–18.
2. Соколов, Е. А. Применение методов машинного обучения в рекомендательных системах // Информационные технологии. – 2020. – Т. 26, № 3. – С. 45–52.
3. Кузнецов, А. Н., Смирнов, И. В. Рекомендательные системы на основе глубокого обучения: принципы и практика. – М.: Издательство "Техносфера", 2021. – 320 с.

УДК 004.42

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

РАЗРАБОТКА СИСТЕМЫ МУЗЫКАЛЬНЫХ РЕКОМЕНДАЦИЙ

Наиболее распространенным способом прослушивания музыки в настоящее время являются стриминговые сервисы. Они предоставляют доступ к десяткам миллионов треков. Существующие системы музыкальных рекомендаций (MRS) [1] объединяют несколько актуальных подходов на основе совместной фильтрации и рекомендаций на основе контента.

Несмотря на то, что традиционные рекомендательные системы доказали свою эффективность, им присущи некоторые недостатки.

1. Проблема "холодного старта". Даная проблема возникает, когда у новых пользователей или объектов нет достаточной истории взаимодействия для точных рекомендаций. Традиционные системы с трудом предоставляют релевантные предложения в таких сценариях. Для решения этой проблемы используются вспомогательные источники данных или гибридные методы.

2. Проблемы масштабируемости. По мере расширения баз пользователей и объектов возникает проблема масштабируемости. Обработка больших массивов данных и поддержание скорости работы системы в реальном времени требуют эффективных алгоритмов и фреймворков распределенных вычислений.

3. Отсутствие персонализации. Применение фильтрации на основе контента может привести к чрезмерной персонализации, когда пользователи ограничиваются своими предпочтениями и не получают кардинально новые рекомендации. Коллаборативная фильтрация может не улавливать тонкие индивидуальные предпочтения, что приводит к менее точным рекомендациям. Поиск баланса между разнообразием и релевантностью остается постоянной проблемой для традиционных рекомендательных систем.

Таким образом, целью этой работы является построение системы музыкальных предпочтений, которая решает перечисленные недостатки.

Для достижения поставленной цели предлагается разработать систему музыкальных рекомендаций на основе запросов пользователя на естественном языке в форме диалога с чат-ботом. В общем виде работа программы представлена на рисунке 1.

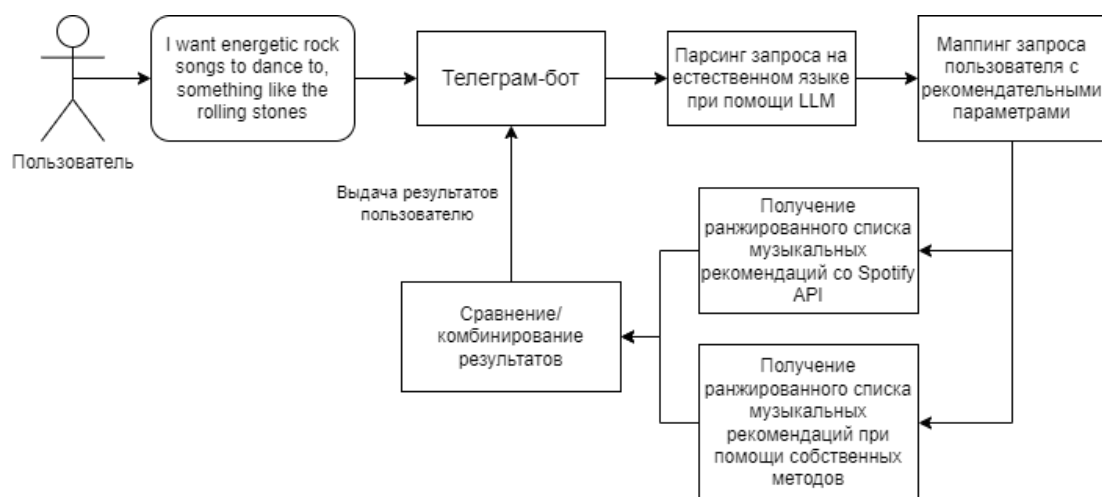


Рисунок 1 – Схема работы предложенной системы музыкальных рекомендаций

Пользователь на естественном языке формирует запрос системе, описывая какую музыку он хочет получить. Запрос пользователя обрабатывается, и при помощи технологий LLM [2] из него выделяются характеристики, на основании которых будет составляться список рекомендаций. По характеристикам, полученным из запроса пользователя, формируются два плейлиста из музыкальных рекомендаций: первый – с использованием API Spotify [3], второй – при помощи предлагаемого алгоритма ранжирования музыкальных треков. Полученные списки треков сравниваются, и проводится оценка релевантности рекомендаций.

Набор исходных данных, на основе которого обучается предлагаемый в работе алгоритм, состоит из сотен тысяч треков. Помимо информации о названии трека, альбома и исполнителя, он содержит аудио характеристики треков (танцевальность, вокальность, энергичность) [4].

Процесс выделения музыкальных характеристик из запроса пользователя для создания списка рекомендаций показан на рисунке 2.

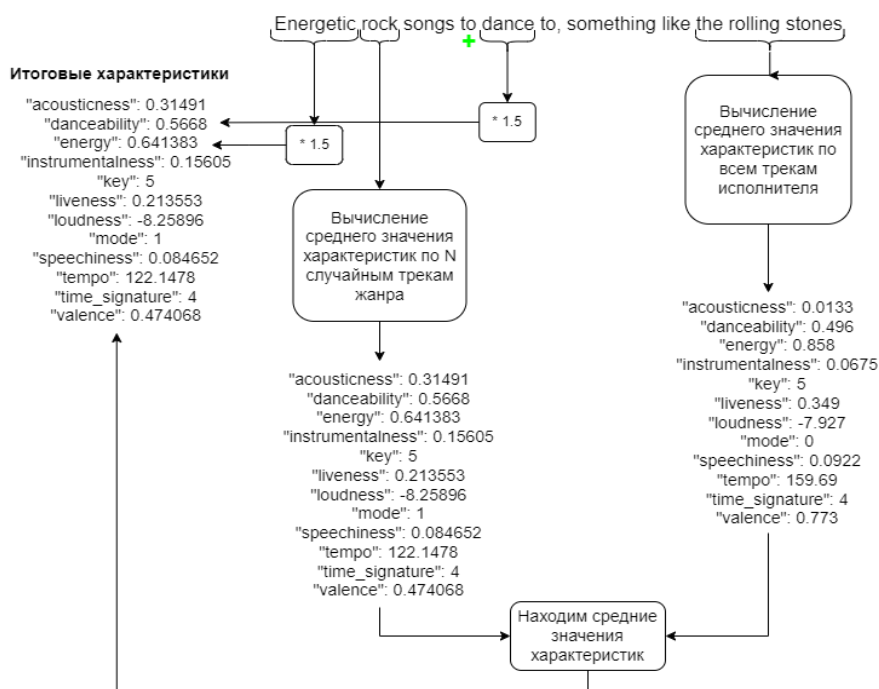


Рисунок 2 – Процесс выделения музыкальных характеристик из запроса пользователя

Используя полученные значения характеристик, ранжируется список треков [5], который и представляет собой конечные музыкальные рекомендации пользователю. Интерфейс

взаимодействия с построенной системой рекомендаций предлагается организовать в форме телеграм-бота. Реализованное программное средство успешно протестировано и показало высокую эффективность в задаче предоставления рекомендаций музыкального контента.

ЛИТЕРАТУРА

1. Deldjoo, Yashar, Markus Schedl, and Peter Knees. "Content-driven music recommendation: Evolution, state of the art, and challenges." Computer Science Review 51 (2024): 100618.
2. Wu, Likang, et al. "A survey on large language models for recommendation." World Wide Web 27.5 (2024): 60.
3. Pastukhov, D. "Inside Spotify’s Recommender System: A Complete Guide to Spotify Recommendation Algorithms." Music Tomorrow (2022): 3.
4. Zangerle, Eva, et al. "Leveraging affective hashtags for ranking music recommendations." IEEE Transactions on Affective Computing 12.1 (2018): 78-91.
5. Li, Hang. "A short introduction to learning to rank." IEICE TRANSACTIONS on Information and Systems 94.10 (2011): 1854-1862.

УДК 004.822

Б.Л. Цэнгэл (2 курс магистратуры),
Т.С. Горавнева, к.т.н., доцент

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ОСУЩЕСТВЛЕНИЯ НАВИГАЦИИ
ПО ПОМЕЩЕНИЯМ КРУИЗНОГО СУДНА

Целью работы является разработка мобильного кроссплатформенного приложения для осуществления навигации по помещениям круизного судна с учетом различных критериев. В условиях современного круизного туризма, где пассажиры сталкиваются с необходимостью быстро и эффективно перемещаться по большим и сложным пространствам, задача упрощения поиска оптимального пути становится особенно важной. Приложение, реализующее поиск маршрута, не только найдет оптимальный маршрут, но также позволит оптимизировать найденный маршрут под конкретные пользовательские критерии, такие как минимальное количество шагов, минимально затраченное время и использование лифтов или трапов.

Для достижения поставленной цели были рассмотрены существующие алгоритмы поиска [1], основное внимание было уделено возможности имплементации существующих алгоритмов в приложение для решения поставленной задачи (таблица 1).

Таблица 1

Алгоритмы	Достоинства	Недостатки
A Star	Подходит для плотных графов при правильной эвристике	Сильная зависимость от эвристики, в разреженных графах не дает преимуществ и требует дополнительной памяти для хранения эвристики
Беллмана-Форда	Способен работать с отрицательными весами	Высокая временная сложность
Флойда-Уоршелла	Вычисляет кратчайшие пути между всеми парами вершин	Высокая временная сложность, избыточен в текущей задаче
Джонсона	Вычисляет кратчайшие пути между всеми парами вершин	Высокая временная сложность, избыточен в текущей задаче
Дейкстры	Наименьшая временная сложность для разреженных графов	Не работает с отрицательными весами

Анализ существующих алгоритмов позволил выявить их достоинства и недостатки и в результате сравнения определить наилучший алгоритм – алгоритм Дейкстры [2].

Каждый этаж судна в рассматриваемой разработке представлен графом, граф в свою очередь – это совокупность вершин (стартовая и конечная точки, каюты, рестораны и другие ключевые места) и ребер (связи, соединяющие вершины, т.е. доступные пути от заданной точки до других точек). Алгоритм Дейкстры показывает наименьшую временную сложность для разреженных графов, а в текущей задаче граф является разреженным.

Формально определение разреженности выражается формулой:

$$D = \frac{2E}{V*(V-1)},$$

где E – количество ребер, а V – количество вершин.

Если $D \leq 0.1$ – граф считается разреженным.

Предположим, что каждый этаж имеет 100 вершин и 500 ребер (в действительности еще меньше), тогда

$$D = \frac{1000}{100 * (100 - 1)} = 0.1$$

Единственный недостаток алгоритма Дейкстры – это отсутствие возможности работать с отрицательными весами, но в контексте задач поиска маршрута этот недостаток игнорируется, так как расстояния не могут быть отрицательными.

Каждый этаж в мобильном приложении представлен в виде векторного изображения с каютами, барами и точками перехода в виде лифтов и лестниц (рисунок 1).

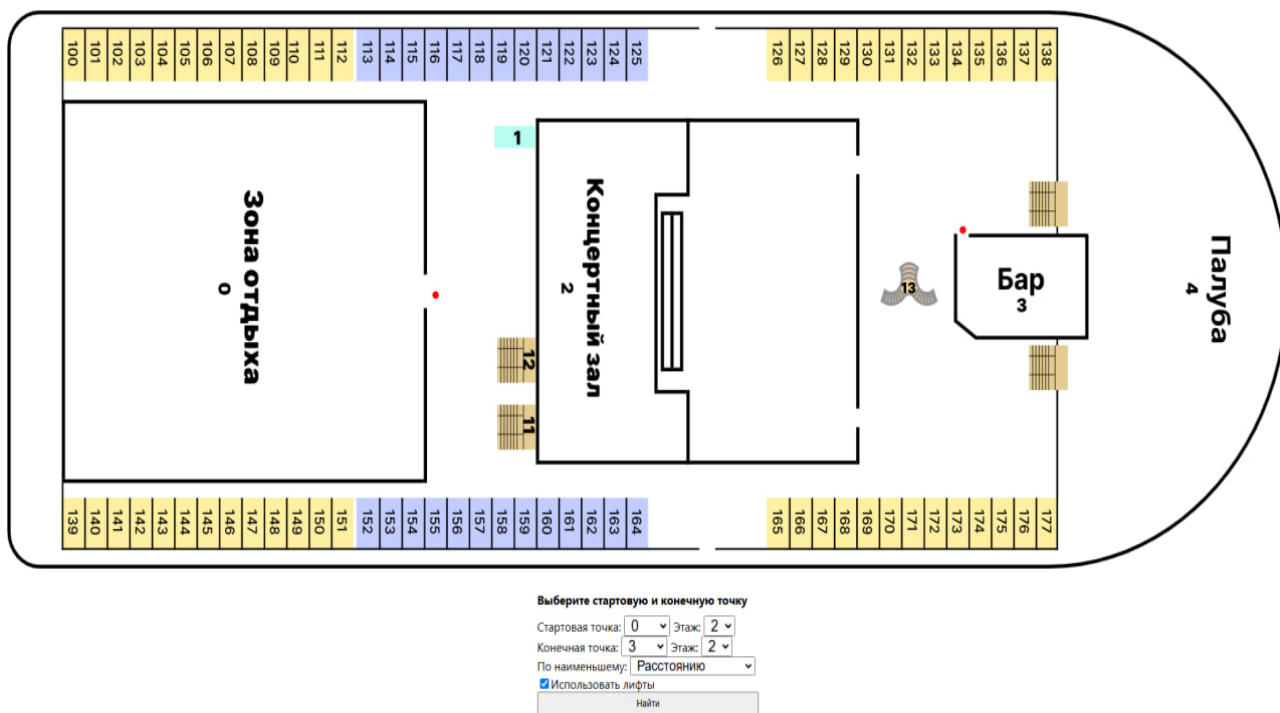


Рисунок 1 – Интерфейс приложения

Если стартовая и конечная точка находится на одном этаже, алгоритм Дейкстры отработывает один раз, так как не нужно строить маршрут до точки перехода.

В случае, если этажи стартовой и конечной точки не совпадают, алгоритм отработает дважды. Сначала алгоритм построит маршрут от стартовой точки до точки перехода на текущем этаже и после от точки перехода на конечном этаже до конечной точки (если выбран критерий по использованию лифтов).

В случае, если выбран критерий без лифтов, лифты перестанут работать в качестве точек перехода. Алгоритм будет строить маршрут до ближайшей лестницы и так до тех пор, пока не будет достигнута конечная точка.

После отображения найденного маршрута выводится информация о пройденном расстоянии, количестве шагов и затраченном времени.

В ходе разработки приложения было предложено направление для дальнейшего развития и совершенствования. Данным направлением является интеграция с картографическими сервисами, такими как Yandex Maps, Google Maps для динамической загрузки изменений объектов и путей, что позволит получать актуальную информацию.

Средства создания разработки:

1. Среда разработки Visual Studio Code 1.94.0.
2. Язык программирования JavaScript.
3. Библиотека React.

ЛИТЕРАТУРА

1. Алгоритм поиска на графах // URL: <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/> – (дата обращения 13.02.2025).
2. Алгоритм Дейкстры //URL: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/> – (дата обращения: 13.02.2025).

УДК 004.6+004.032

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

РАЗРАБОТКА РАСШИРЕНИЯ ДЛЯ СУБД POSTGRESQL ДЛЯ ИНТЕГРАЦИИ НЕЙРОННЫХ СЕТЕЙ С ИСПОЛЬЗОВАНИЕМ OPENNN

Современные задачи анализа данных требуют интеграции методов машинного обучения непосредственно в системы управления базами данных (СУБД) [1]. Однако существующие решения обладают недостатками:

- необходимость экспорта данных из СУБД для обучения моделей [2];
- отсутствие централизованного управления конфигурациями нейросетей;
- сложности в автоматизации обновления моделей на актуальных данных [3].

Целью работы является разработка расширения для PostgreSQL [4], позволяющего выполнять обучение, хранение и использование нейронных сетей непосредственно в СУБД. Для этого используются язык C++ и библиотека OpenNN [3].

Для достижения цели поставлены следующие *задачи*:

1. Анализ существующих расширений PostgreSQL для машинного обучения [2].
2. Исследование возможностей библиотеки OpenNN для интеграции с СУБД [3].
3. Проектирование архитектуры расширения с учетом схемы базы данных (рисунок 1).
4. Реализация функций для обучения моделей, сохранения весов и конфигураций.
5. Тестирование расширения на задачах классификации и регрессии.

Архитектура системы (рисунок 1) включает:

- Клиентскую часть: SQL-запросы для взаимодействия с расширением.
- Серверную часть:
 - расширение на C++ (интеграция через PGXS [4]);
 - использование OpenNN для обучения и инференса [3];
 - взаимодействие с таблицами БД (train_table, models_table).
- Хранилище: таблицы PostgreSQL для данных и моделей.

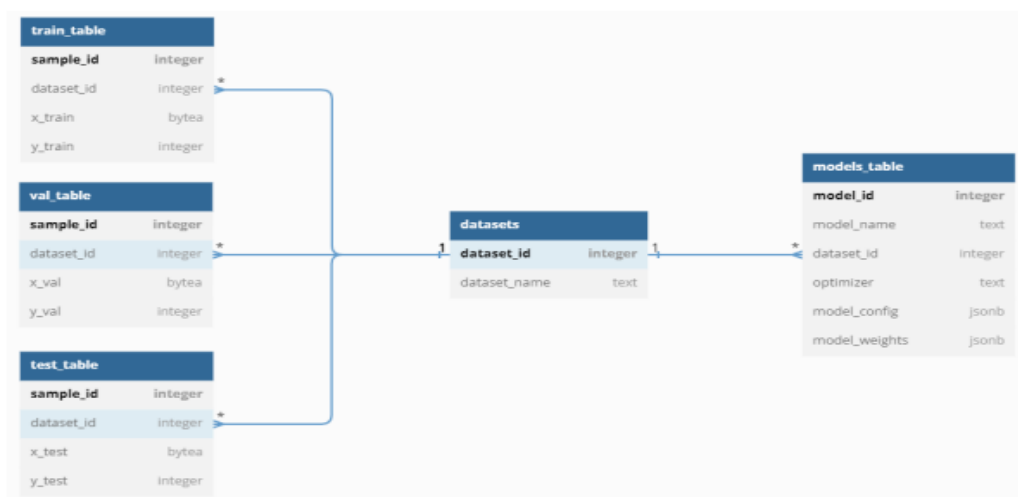


Рисунок 1 – Схема взаимодействия компонентов расширения с БД

Ключевые особенности:

- Обучение моделей без экспорта данных.
- Хранение весов в JSONB-полях (model_weights) [5].
- Поддержка оптимизаторов через поле optimizer.
- Генерация предсказаний через SQL-запросы [1].

Преимущества решения:

- Снижение затрат на передачу данных [2].
- Централизованное управление моделями.
- Интеграция в ETL-процессы [1].

Работоспособность проверена на тестовых датасетах. В перспективе — поддержка распределенного обучения.

ЛИТЕРАТУРА

1. Garcia R. Modern Approaches to ML Integration in Databases. *ACM Computing Surveys*, 2023. Vol. 55(2). P. 1–30.
2. Smith J. In-Database Machine Learning: Trends and Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2022. Vol. 34(5). P. 2100–2115.
3. Lopez M. OpenNN: An Open-Source Neural Networks Library. *Journal of Machine Learning Research*, 2021. Vol. 22. P. 1–6.
4. PostgreSQL Documentation. 2023. URL: <https://www.postgresql.org/docs/>
5. Brown A. JSONB and Its Applications in PostgreSQL. *Proc. of the 2020 International Conference on Data Science*, 2020. P. 45–50.

УДК 004.021

Д.Э. Шабинский (2 курс магистратуры),
Н.В. Воинов, к.т.н., доцент

РАЗРАБОТКА ANDROID-КЛИЕНТА ПРОЕКТА POLYNABR

Современные мобильные приложения играют важную роль в повседневной жизни пользователей, обеспечивая удобный доступ к сервисам и информации [1-3]. В условиях стремительного развития цифровых технологий и увеличения объема информации студенты сталкиваются с проблемой структурированного хранения и поиска учебных материалов [4]. Традиционные методы, такие как форумы, социальные сети и локальные файлы, часто оказываются неудобными из-за фрагментарности информации и отсутствия единого формата для представления знаний.

Проект PolyHabr ориентирован на поддержку студентов, предоставляя платформу для обмена учебными материалами, обсуждения научных вопросов и поиска информации по темам. Он решает проблему разрозненности знаний, предлагая централизованную платформу, где студенты могут публиковать, комментировать и оценивать материалы, а также получать рекомендации на основе интересов.

Цель работы – разработка Android-клиента [5] для сервиса PolyHabr, который обеспечит удобное взаимодействие пользователей с платформой. В отличие от традиционных решений, таких как закрытые студенческие группы в мессенджерах и файлообменники, PolyHabr предлагает удобную навигацию, персонализированный поиск и систему рейтинга материалов.

В рамках работы были рассмотрены существующие аналоги, проведен анализ архитектурных решений и выбраны оптимальные технологии. Основное внимание уделено удобству использования, безопасности данных и расширяемости функционала.

Разработка клиентского приложения выполнялась с применением Kotlin [6] и архитектурного подхода Clean Architecture. Для обеспечения модульности и тестируемости использовались MVVM, Dagger/Hilt [7] для внедрения зависимостей, Retrofit для сетевого взаимодействия и Room для работы с локальной базой данных. Навигация реализована на основе Jetpack Navigation Component [8].

Основной функционал приложения:

- Авторизация и регистрация пользователей.
- Просмотр, публикация и комментирование учебных материалов.
- Поиск и фильтрация контента по различным критериям.
- Управление профилем пользователя и настройками приложения.

Архитектура приложения основана на принципе разделения слоев (presentation, domain, data), что обеспечивает гибкость и поддержку расширяемости системы. Взаимодействие с серверной частью построено на REST API с использованием PostgreSQL [9] в качестве СУБД.

Результатом работы является Android-клиент, обеспечивающий удобный доступ к сервису PolyHabr. В перспективе планируется интеграция с дополнительными сервисами, улучшение пользовательского интерфейса и внедрение механизмов машинного обучения для персонализации контента.

ЛИТЕРАТУРА

1. Толстиков, Г. Н. Разработка Android приложения системы поддержки принятия решений / Г. Н. Толстиков, В. В. Амосов // 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. 130-132. – EDN GUXTHF.
2. Абисалов, Г. Р. Разработка Android приложения сервиса доставки и сбора продуктов / Г. Р. Абисалов, В. В. Амосов // 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. 4-5. – EDN DHWJZR.
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 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. 126-127. – EDN DUTYHC.
4. Новые технологии вычислительной техники в учебных планах кафедры информационных и управляющих систем СПбГПУ / В. В. Амосов, Т. А. Вишневская, С. А. Молодяков [и др.] // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. – 2013. – № 2(171). – С. 322-331. – EDN QYOIRF.
5. Android Developers Documentation. URL: <https://developer.android.com/docs>
6. Kotlin Language Documentation. URL: <https://kotlinlang.org/docs/home.html>

7. Dagger Dependency Injection. URL: <https://dagger.dev/dev-guide/android.html>
8. Jetpack Navigation Component. URL: <https://developer.android.com/guide/navigation/navigation-getting-started>
9. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/>

УДК 004.453

Е.Г. Шапочкин, Н.А. Дементьев (4 курс бакалавриата),
И.В. Шошмина, к.т.н., доцент

РАЗРАБОТКА ВЕБ-СЕРВИСА ДЛЯ АНАЛИТИКИ И УПРАВЛЕНИЯ РЕКЛАМНЫМИ КАМПАНИЯМИ НА МАРКЕТПЛЕЙСАХ.

В работе рассматривается проблема обеспечения информативного анализа статистических данных и эффективного управления рекламными кампаниями продавцами. Для информативного анализа требуется подобрать ключевые метрики продаж, а также предоставить инструменты для их мониторинга. Эффективное управление рекламными кампаниями состоит в регулировании затрат на рекламу, настройке рекламных активностей и параметров кампаний, управлении рекламными кампаниями на нескольких маркетплейсах из одной точки.

Для создания такого сервиса мы выбрали веб-архитектуру, для получения информации с маркетплейсов используются их API. Наш сервис был разработан для маркетплейсов Wildberries и Ozon. Взаимодействие с этими маркетплейсами реализовано через использование открытого API Wildberries [1] и Ozon [2], где присутствуют методы, которые мы используем для сбора статистики и управления рекламными кампаниями.

Целью данной работы является создать сервис, автоматизирующий управление рекламными кампаниями и обеспечивающий аналитику в кабинете продавца.

Для достижения этой цели были поставлены следующие задачи:

1. Разработать модуль регистрации пользователей и привязки магазинов.
2. Разработать модуль работы с базами данных.
3. Разработать модуль работы с API.
4. Разработать модуль автоматического сбора статистики.
5. Разработать модуль управления рекламными кампаниями.
6. Разработать модуль взаимодействия frontend и backend.
7. Разработать визуальную часть сервиса.

Наш веб-сервис построен на клиент-серверной архитектуре (рисунок 1) и организован по модульному принципу (рисунок 2). Клиентская часть (фронтенд) отвечает за пользовательский интерфейс и обработку взаимодействий, а серверная часть (бэкенд) реализует бизнес-логику, управляет данными и обрабатывает запросы. Взаимодействие между клиентом и сервером осуществляется через REST API. Для хранения данных используется база данных MySQL, с которой бэкенд взаимодействует через Doctrine ORM, предоставляющую уровень абстракции для работы с данными в фреймворке Symfony. Для управления и анализа данных в базе используется DBeaver.

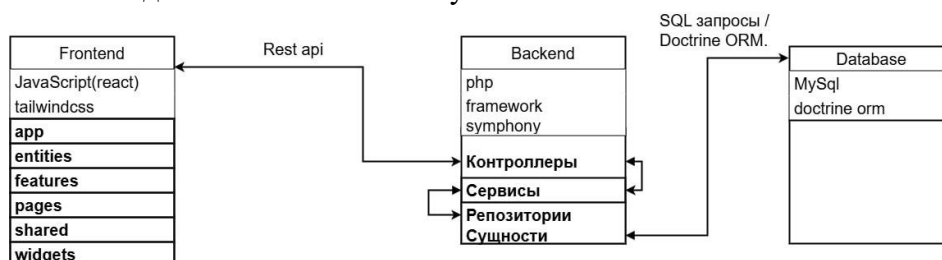


Рисунок 1 – Архитектура веб-сервиса

Модульность обеспечивает разделение функциональности на независимые компоненты, что упрощает поддержку, масштабирование и развитие системы. Модули взаимодействуют друг с другом. Сервис реализован на языке PHP с использованием IDE PhpStorm. В качестве серверного фреймворка используется Symfony [3], обеспечивающий обработку запросов, бизнес-логику и работу с базой данных. Фронтенд построен на библиотеке React [4] и использует HTML, CSS и JavaScript.

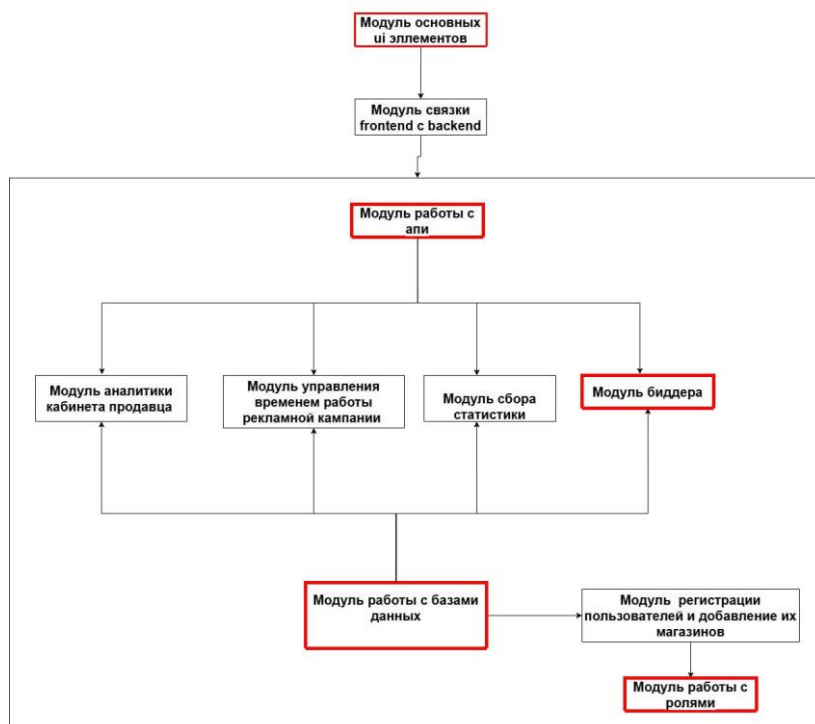


Рисунок 2 – Модульная архитектура системы

Для обеспечения информативной аналитики мы внедрили динамическое управление периодом статистических данных, сортировку по метрикам, настройку таблиц, фильтрацию и группировку данных, а также возможность вывода данных в формате графиков и сравнение данных за разные периоды. Для эффективного управления рекламными кампаниями мы предложили настраиваемые параметры для регулирования затрат и сформировали набор «быстрых» действий для группового управления несколькими рекламными кампаниями.

На рынке присутствуют схожие инструменты, например MarketGuru [5], MPSTATS [6], Mr Manager.[7] В сравнении с аналогами наш сервис обладает следующими преимуществами: расширенный период доступности статистических данных, удобный и интуитивно понятный интерфейс, возможность выгрузки данных в формате Excel для дальнейшего анализа, работа с несколькими маркетплейсами.

ЛИТЕРАТУРА

1. Документация по API методам Wildberries [Электронный ресурс] Режим доступа: <https://dev.wildberries.ru/>
2. Документация по API методам Ozon [Электронный ресурс] Режим доступа: <https://dev.ozon.ru/>
3. Документация Symfony [Электронный ресурс] Режим доступа: <https://symfony.com/doc/current/index.html>
4. Документация React [Электронный ресурс] Режим доступа: <https://reactjs.org/docs/getting-started.html>
5. MarketGuru [Электронный ресурс] Режим доступа: <https://marketguru.io/>
6. MPSTATS [Электронный ресурс] Режим доступа: <https://mpstats.io/>
7. Mr Manager [Электронный ресурс] Режим доступа: <https://mpmgr.ru/>

РАЗРАБОТКА TELEGRAM-БОТА С ИНТЕГРАЦИЕЙ С ОБЛАЧНЫМ СЕРВИСОМ
АМОСРМ ДЛЯ ОПТИМИЗАЦИИ РАБОТЫ МЕЖДУНАРОДНОЙ ДЕТСКОЙ ШКОЛЫ
ПРОГРАММИРОВАНИЯ «АЛГОРИТМИКА»

«Алгоритмика» — международная школа программирования для детей. В каждом ее филиале обучается от 100 детей. Связь с родителями поддерживают менеджеры. В среднем около трети рабочего времени каждый менеджер тратит только на коммуникацию с родителями, отвечая на ежедневные вопросы: уточнение расписания, данные для входа ребенка, консультации по курсам и уточнение данных. Информация о клиентах школы хранится в облачном сервисе amoCRM. Все данные добавляются в систему менеджерами вручную. Таким образом, существенную часть работы менеджера можно было бы автоматизировать, чтобы или переключить его на более сложные задачи управления, или же сократить штат.

Существующие решения обладают рядом недостатков, среди которых:

- Высокая стоимость.
- Предоставление ограниченного функционала.
- Необходимость использования дополнительного ПО.

Цель работы следующая: создать решение, которое позволило бы оптимизировать работу менеджеров и упростило бы взаимодействие родителей учеников со школой, а также повысило бы осведомленность о школе у людей, чьи дети ранее не обучались с «Алгоритмике».

Чтобы поддерживать хорошие отношения с клиентами, необходимо улучшить качество обслуживания информационных систем с целью создания лояльности клиентов к компании [5]. Было решено создать чат-бота, который будет предоставлять возможность взаимодействия с данными из amoCRM прямо в интерфейсе социальной сети, а также позволит объединить функционал для различных сторон процесса коммуникации между школой и родителем в одном месте. Реализовано это будет с помощью возможности взаимодействовать с ботом под разными ролями. Для разработки решения был выбран мессенджер Telegram [1]. Основная информация о школе и курсах, которую бот будет предоставлять пользователям по запросу, хранится в базе данных SQLite [2]. Бот разработан на языке Python с использованием библиотеки Aiogram [3]. Отметим, что решения, существующие на рынке, не реализуют представленный функционал в полной мере.

Задачи, которые необходимо решить для достижения поставленной цели:

1. Выявление особенностей в работе школы, которые можно было бы улучшить посредством использования бота.
2. Интеграция бота с amoCRM для возможности взаимодействия с данными.
3. Разработка базы данных для хранения информации о школе, необходимой для работы бота [2].
4. Разделение ролей в боте: обычный пользователь, зарегистрированный пользователь (родитель), администратор [3].
5. Оптимизация функционала каждой роли [3][4].

В перспективе стоит задача интеграции бота в работу «Алгоритмики» и его последующая поддержка.

ЛИТЕРАТУРА

1. Официальная страница FAQ Telegram [Электронный ресурс] – URL: <https://telegram.org/faq> (дата обращения: 03.03.2025).

2. Документация SQLite [Электронный ресурс] – URL: <https://www.sqlite.org/index.html> (дата обращения: 13.02.2025).
3. Документация Aioqram [Электронный ресурс] – URL: https://aioqram.readthedocs.io/_/downloads/en/latest/pdf/ (дата обращения: 10.02.2025).
4. Путра Ф. П, Созтанто Х, Сабинин С. Прототип системы управления на основе Интернета вещей с использованием Telegram и метода Bot API // ResearchGate. 2025. URL: https://www.researchgate.net/publication/389506926_Prototype_of_Internet_of_Things-Based_Control_System_Using_Telegram_with_Bot_API_Method (дата обращения: 01.03.2025).
5. Разак Дж. А, Приямбодо Д. В. Разработка CRM-системы с Telegram API в сервисной информационной системе // ResearchGate. 2023. URL: https://www.researchgate.net/publication/368400030_RANCANG_BANGUN_CRM_DENGAN_API_TELEGRAM_PADA_SISTEM_INFORMASI PELAYANAN (дата обращения: 01.03.2025).

УДК 004.457

И.К. Шульгин (4 курс бакалавриата),
Б.М. Медведев, к.т.н., доцент

ПРОГРАММНОЕ СРЕДСТВО ВАЛИДАЦИИ РАСЧЁТОВ ЗА РОУМИНГОВЫЕ УСЛУГИ

Крупные операторы сотовых сетей связи активно развивают системы конвергентного биллинга [1], обеспечивающие единое управление всеми пользователями и услугами, независимо от категории и способа предоставления услуг. Одной из актуальных задач создания таких систем является расчет за предоставление услуг вне зоны обслуживания домашней сети (роуминг), который, как правило, выполняют сторонние клиринговые дома (DCH, Data Clearing House), обеспечивающие взаимодействие между операторами сотовой связи для обмена данными расчётов за роуминг.

Для реализации взаиморасчётов за услуги роуминга операторы заключают между собой RAEX IOT (Roaming Agreement Exchange Inter-Operator Tariff) соглашения, в которых описаны различные параметры вызова и соответствующие им ставки, по которым происходит тарификация. Для формализации данных используются стандарты GSMA, определяющие бизнес-требования для роуминговых соглашений [2], а также логическую и физическую структуру данных TAP (Transferred Account Procedure) и RAP (Returned Accounts Procedure) файлов [3]. Взаимодействие между операторами-партнёрами при роуминге заключается в обмене TAP файлами с информацией о предоставленных услугах и RAP файлами, в которых содержатся данные о найденных в TAP файлах ошибках [4]. В связи с тем, что операторы могут по-разному интерпретировать бизнес-требования для роуминговых соглашений, расчет за роуминг существенно усложняется и требует учета специфических особенностей множества соглашений RAEX IOT операторов-партнеров.

Целью данной работы является создание программных средств валидации расчетов за роуминговые услуги, обеспечивающих проверку TAP-файлов на основе RAEX IOT соглашений между операторами-партнерами.

Для достижения цели мною были решены следующие задачи в процессе разработки программного средства:

1. Разработка алгоритмов поиска группы направлений и ставок на основе данных TAP-файлов и RAEX IOT соглашений.
2. Разработка программных средств валидации TAP-файлов на основе RAEX IOT соглашений.
3. Тестирование разработанных программных средств.

Одним из основных параметров расчетов является группа направлений, к которой принадлежит телефонный номер адресата. Группы направлений представляют собой множества телефонных номеров, заданных в RAEX IOT соглашении в виде полного номера,

префикса или диапазона номеров. Большое количество направлений делает неэффективным поиск перебором. Для уменьшения времени поиска по номерам и префиксам разработан алгоритм поиска на базе структуры данных типа префиксное дерево с поддержкой диапазонов номеров.

TAP-файлы в соответствии со стандартом TAP3 содержат информацию о нескольких вызовах и имеют вложенную древовидную структуру. Каждому вызову может соответствовать множество отдельных начислений, имеющих частично общие параметры. Был разработан алгоритм обхода данных TAP-файла и поиска ставок на основе RAEX IOT соглашений, позволяющий минимизировать количество обращений к данным соглашений и использовать найденные параметры ставок для всех соответствующих начислений.

Валидация начислений сводится к повторной тарификации использованных услуг. Тарификация каждой услуги может происходить с использованием нескольких найденных ставок, интервалов начисления и других параметров, заданных в RAEX IOT соглашениях. Для того чтобы начисление прошло валидацию, его значение должно совпасть с результатом повторной тарификации, иначе формируется ошибка, которая в последствии записывается в RAP-файл.

Разработка программных средств проверки TAP-файлов на основе RAEX IOT соглашений выполнена в виде компонента Check IOT Extension для системы RAPTAP@Reactor, входящей в состав продукта Roaming System компании Bercut [5]. Система RAPTAP@Reactor реализована на базе сервера исполнения сценариев на языке программирования Lua, компонентами системы являются библиотеки, написанные на языке программирования C++. Архитектура системы RAPTAP@Reactor представлена на рисунке 1.

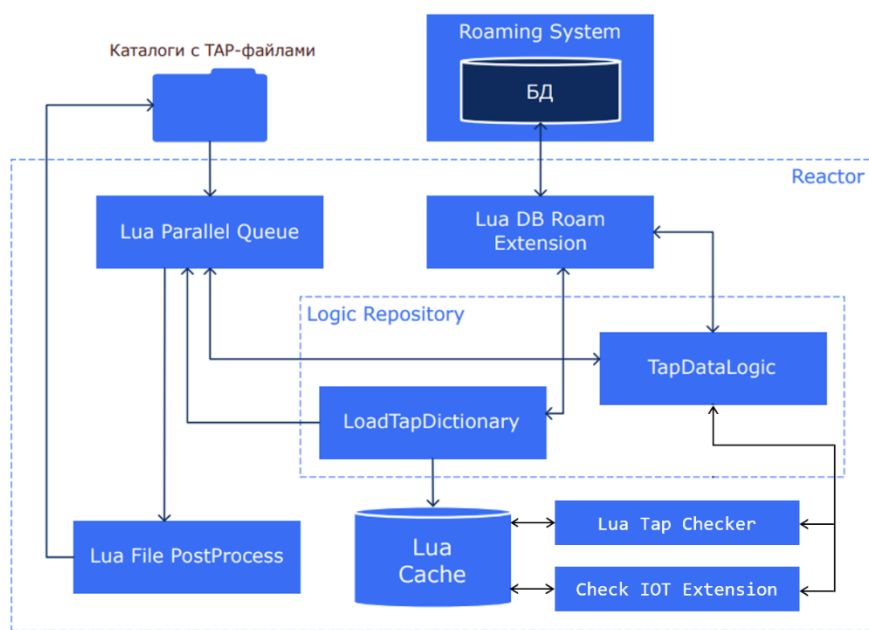


Рисунок 1 – Архитектура системы RAPTAP@Reactor

Lua логики LoadTapDictionary и TapDataLogic осуществляют заполнение справочников в кэше данными из БД и исполнение проверок TAP-файла. Компонент Lua Parallel Queue обеспечивает формирование очереди обработки файлов, Lua DB Roam Extension взаимодействует с БД, Lua Tap Checker и Check IOT Extension отвечают за проверки TAP-файла, Lua File PostProcess обеспечивает обработку файлов по результатам проверок.

Компонент Check IOT Extension получает на вход данные вызовов из TAP-файла, не имеющих синтаксических и логических ошибок. В процессе проверки используются данные роуминговых соглашений, заранее записанные в кэш. Поиск ставок и валидация начислений сопровождаются подробным логированием, упрощающим понимание причин, из-за которых валидация не была пройдена.

Для тестирования системы использовалось более 300 автотестов, покрывающих различные комбинации проверяемых данных TAP-файлов и данных роуминговых соглашений. На основе результатов проведения нагрузочного тестирования можно сделать вывод, что использование разработанных алгоритмов поиска группы направлений и поиска ставок позволяет выполнить требования к высоконагруженной системе, которая должна обрабатывать информацию о 20 млн. вызовов в сутки менее чем за 5 часов.

ЛИТЕРАТУРА

1. Ericsson. Prepaid Postpaid Convergent Charging – 2005.
2. GSMA. Official Document BA.29 – RAEX IOT Business Requirements Version 5.0 – 2024
3. GSM Association. Official Document TD.57 – TAP 3.12 Format Specification Version 36.5 – 2019
4. Tutorials Point (I) Pvt. Ltd. Telecom Billing Tutorial – 2015
5. Bercut Ltd. Прямой обмен TAP- и RAP-файлами с операторами-партнерами – 2024

УДК 004.415.2

Ш.К. Якубов (4 курс бакалавриата),
В.Э. Шмаков, к.т.н., доцент

ПРОЕКТИРОВАНИЕ ПЛАТФОРМЫ ДЛЯ СОЗДАНИЯ NFT И ЖЕТОНОВ НА БЛОКЧЕЙНЕ TON

В последние годы технологии блокчейн и невзаимозаменяемые токены NFT (non-fungible token) стали важными компонентами цифровой экономики. NFT [1] представляют собой уникальные цифровые активы, которые невозможно заменить или подделать благодаря использованию цифровых подписей и технологии блокчейн. В блокчейне TON (The Open Network) [2] также существует возможность создания собственных монет, называемых жетонами (Jettons) [3], которые функционируют как полноценная криптовалюта.

Однако, несмотря на растущую популярность NFT и блокчейн-технологий, существует ряд проблем, связанных с доступностью и удобством использования этих технологий для широкой аудитории. Современные платформы для создания и управления NFT часто требуют от пользователей глубоких технических знаний и навыков программирования, что создаёт барьеры для новичков. Отсутствие интуитивно понятных интерфейсов также затрудняет процесс создания и управления токенами.

Целью данной работы является разработка платформы, которая позволит пользователям создавать собственные NFT и жетоны на блокчейне TON без обладания глубокими техническими знаниями. Платформа должна предоставлять удобный интерфейс для управления токенами и коллекциями, а также обеспечивать высокую скорость обработки транзакций и низкие комиссии благодаря использованию блокчейна TON.

Для достижения поставленной цели были выполнены следующие задачи:

1. Изучение блокчейна TON. Проведён анализ технической реализации блокчейна TON, включая особенности работы смарт-контрактов и транзакций. Блокчейн TON был выбран благодаря его высокой скорости обработки транзакций, низким комиссиям и масштабируемости.
2. Изучение стандартов NFT и жетонов. Исследованы стандарты смарт-контрактов для NFT (например, TON NFT Standard) и жетонов (Jettons). Эти стандарты обеспечивают уникальность и функциональность токенов.
3. Разработка UI/UX дизайна. Создан интуитивно понятный интерфейс для платформы, ориентированный на пользователей без технической подготовки. Дизайн включает в себя удобные формы для создания токенов, управления коллекциями и просмотра транзакций.
4. Разработка клиентской и серверной частей. Клиентская часть платформы реализована с использованием React [4], что обеспечивает высокую производительность и отзывчивость

интерфейса. Серверная часть разработана на Node.js [5] и взаимодействует с блокчейном TON через TON SDK [6].

Архитектура платформы включает следующие компоненты:

- Клиентская часть. Пользовательский интерфейс для создания и управления NFT и жетонами.
- Серверная часть. Обработка запросов пользователей, взаимодействие с блокчейном TON и управление смарт-контрактами.

На рисунке 1 слева изображена структура взаимодействия смарт-контрактов при реализации жетонов, а справа при реализации NFT

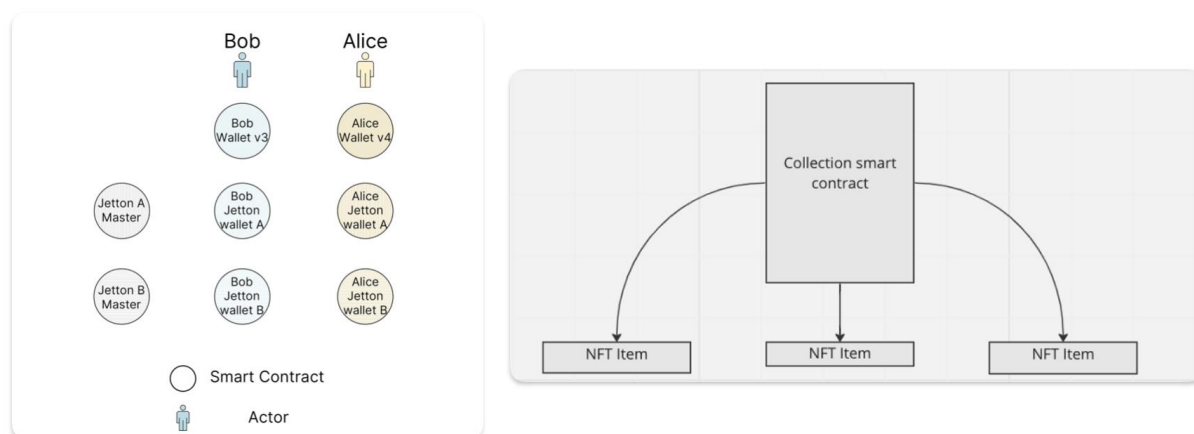


Рисунок 1 – Структура взаимодействия между смарт-контрактами

В результате проведённой работы была разработана платформа, которая позволяет пользователям создавать и управлять NFT и жетонами на блокчейне TON. Платформа предоставляет удобный интерфейс, что делает её доступной для широкой аудитории. Благодаря использованию блокчейна TON, платформа обеспечивает высокую скорость обработки транзакций и низкие комиссии.

Разработанная платформа упрощает процесс создания и управления NFT и жетонами на блокчейне TON, что делает эти технологии более доступными для широкой аудитории. Благодаря использованию современных технологий и интуитивно понятного интерфейса, платформа станет удобным инструментом для создания цифровых активов и коллекций.

ЛИТЕРАТУРА

1. NFT Standards and Implementation. <https://github.com/ton-blockchain/TEPs/blob/master/text/0062-nft-standard.md>
2. TON Blockchain Documentation. [Электронный ресурс] Режим доступа: <https://ton.org/docs>
3. Jettons Standards and Implementatio <https://github.com/ton-blockchain/TEPs/blob/master/text/0074-jettons-standard.md?ysclid=m7ixzicpgy255956895>
4. React.js Official Documentation. [Электронный ресурс] Режим доступа: <https://reactjs.org/docs/getting-started.html>
5. Node.js Official Documentation. [Электронный ресурс] Режим доступа: <https://nodejs.org/en/docs/>
6. TON SDK Documentation. [Электронный ресурс] Режим доступа: <https://github.com/tonlabs/TON-SDK>

УДК 004.453

Н.З. Айналияев (3 курс бакалавриата),
А.В. Шукин, к.т.н., доцент,
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА ПЛАГИНА ДЛЯ ВЫЯВЛЕНИЯ ПРИЧИН FLAKY-ТЕСТОВ В PLAYWRIGHT

Flaky-тесты – это автоматизированные тесты, которые при повторных запусках могут давать разные результаты, могут как проходить, так и падать в одной и той же конфигурации и среде без изменений в коде [1]. Нестабильные тесты создают серьезные проблемы для процессов CI/CD, так как могут приводить к ложным срабатываниям, увеличивать время на поиск и исправление ошибок и снижать доверие разработчиков к тестам, а также негативно влиять на общую стабильность процесса разработки. В контексте использования Playwright [5], популярного инструмента для автоматизации тестирования веб-приложений, проблема flaky-тестов становится особенно актуальной из-за сложности взаимодействия с асинхронными операциями, сетевыми задержками, динамическим контентом и зависимостью от внешних факторов [2]. Рассмотрим примеры flaky-тестов [1]:

1. Асинхронные ожидания. Основная причина 45% нестабильных тестов — проблемы с асинхронным ожиданием [1]. Тесты часто пытаются взаимодействовать с элементами, которые ещё не загружены. Это приводит к ошибкам, так как действия выполняются в неправильном порядке, вызывая исключения. Например, в библиотеке `ring-ui` [6], которая предоставляет реакт-компоненты, некоторые тесты зависят от загрузки изображений через сеть. Если изображение не успевает загрузиться, тест может завершиться с ошибкой, так как компонент не отображает изображение.

2. Проблема с выбором элементов DOM. Некоторые тесты не могут корректно выбрать нужный элемент из-за различий в браузерах или устаревших элементов, блокирующих фокус. Например, в проекте `react-datepicker` [7] (компонент выбора даты для React) код в тесте неправильно устанавливает два элемента на странице для автоматической фокусировки, вызывая скачок на странице, который приводит к визуальному мерцанию.

Целью данной работы является разработка плагина для Playwright на TypeScript, который позволит выявлять и анализировать причины flaky-тестов.

Для реализации плагина используются TypeScript, Playwright API. Плагин разработан для интеграции в существующие проекты на Playwright и предоставляет следующие возможности:

1. Сбор и анализ данных о выполнении тестов, включая логи, временные метки, сетевые запросы, ошибки рендеринга, скриншоты, видео и метрики производительности.

2. Генерация детализированных отчетов с указанием потенциальных причин flaky-тестов, таких как нестабильные сетевые запросы, проблемы с таймаутами, изменяющийся контент или ошибки в логике тестов.

Существуют коммерческие решения для управления flaky-тестами, такие как Cypress Cloud (Flaky Test Management) и DataDog Test Optimization. Помимо того, что они требуют действующей подписки, они имеют ряд ограничений:

1. Cypress Cloud предоставляет инструменты для выявления flaky-тестов, но ориентирован исключительно на Cypress и не поддерживает Playwright. Кроме того, Cypress Cloud требует использования их облачной инфраструктуры.
2. DataDog Test Optimization предлагает расширенные возможности анализа производительности и выявления flaky-тестов, но требует интеграции с их облачной инфраструктурой.

Таблица 1 – Сравнение аналогичных инструментов

Характеристика	Разрабатываемый плагин	Cypress Cloud	DataDog Test Optimization
Поддержка Playwright	Да	Нет	Да
Поддержка Cypress	Нет	Да	Да
Сбор логов	Да	Да	Да
Сбор скриншотов и видео	Да	Да	Нет
Генерация отчетов	Да	Да	Да
Стоимость	Бесплатно	Платная подписка	Платная подписка

В рамках исследования получены следующие *результаты*:

1. Разработан плагин для выявления причин flaky-тестов.
2. Проанализирован набор с программными ошибками, проявляющимися в flaky-тестах.
3. Сформулированы дальнейшие шаги по улучшению системы:
 - 3.1. Оптимизация производительности плагина для снижения нагрузки на систему.
 - 3.2. Расширение функциональности для детального анализа flaky-тестов.

ЛИТЕРАТУРА

1. Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, Darko Marinov: An empirical analysis of flaky tests. SIGSOFT FSE 2014: 643-653
2. Owain Parry, Gregory M. Kapfhammer, Michael Hilton, Phil McMinn: A Survey of Flaky Tests. ACM Trans. Softw. Eng. Methodol. 31(1): 17:1-17:74 (2022)
3. Jonathan Bell, Owolabi Legunsen, Michael Hilton, Lamyaa Eloussi, Tiffany Yung, Darko Marinov: DeFlaker: automatically detecting flaky tests. ICSE 2018: 433-444
4. Wing Lam, Kivanç Muslu, Hitesh Sajani, Suresh Thummalapenta: A study on the lifecycle of flaky tests. ICSE 2020: 1471-1482
5. Playwright Docs / [Электронный ресурс] Режим доступа: <https://playwright.dev/docs/intro>
6. chore: use Base64 URI decoded avatar to avoid flaky UI tests if image loads slow 2020. [Электронный ресурс]. Режим доступа: <https://github.com/JetBrains/ring-ui/commit/f3482d9dd17ecda5ced0c3f2cf237a13bc13c338>
7. Hacker0x01. react-datepicker. Remove second autoFocus example. 2018. [Электронный ресурс]. Режим доступа: <https://github.com/Hacker0x01/react-datepicker/pull/1390>

РАЗРАБОТКА ЦЕПОЧКИ ИНКРЕМЕНТАЛЬНОЙ ОБРАБОТКИ БОЛЬШИХ ДАННЫХ И
СРАВНИТЕЛЬНАЯ ОЦЕНКА ETL-РЕШЕНИЙ ДЛЯ ЕЁ РЕАЛИЗАЦИИ

Современные задачи обработки и анализа больших данных требуют эффективных решений для извлечения, преобразования и загрузки данных (далее – ETL). Инкрементальный анализ данных становится ключевым подходом для работы с постоянно обновляющимися источниками информации, что требует использования специализированных инструментов, способных автоматизировать и оптимизировать эти процессы [3]. Одним из основных направлений в области управления ETL-процессами является использование оркестраторов, которые позволяют описывать, планировать и контролировать выполнение задач с помощью программного кода [4].

Среди таких инструментов выделяются Dagster, Apache Airflow, Apache Nifi и Prefect, которые предоставляют гибкие возможности для создания и управления ETL-пайплайнами. Эти инструменты позволяют описывать сложные workflows, обеспечивают воспроизводимость и масштабируемость процессов. Однако с ростом объемов данных и сложности пайплайнов возникает необходимость в инструментах, способных эффективно тестировать, мониторить и оптимизировать эти процессы.

Таким образом, *целью* данной работы является анализ и экспериментальное сравнение комплексных средств ETL для инкрементального анализа больших данных на примере Dagster, Apache Airflow, Nifi и Prefect, а также выявление их преимуществ и недостатков в контексте современных требований к обработке данных [1,5].

Для достижения этой цели необходимо решение следующих *задач*:

1. Изучить принципы инкрементального анализа данных и особенности ETL-процессов в контексте больших данных.
2. Рассмотреть современные инструменты оркестрации ETL-пайплайнов: Dagster, Apache Airflow и Nifi, Prefect.
3. Разработать тестовые ETL-пайплайны для инкрементальной обработки данных с использованием каждого из инструментов.
4. Провести экспериментальное тестирование производительности и удобства использования указанных инструментов.
5. Сравнить подходы, выявить их сильные и слабые стороны.

Результаты исследования позволяют сделать выводы о применимости рассмотренных инструментов в различных условиях и помогут выбрать наиболее подходящее решение для задач инкрементального анализа больших данных. Рассмотренные инструменты ETL-оркестрации Dagster, Apache Airflow, Apache NiFi и Prefect обладают рядом схожих черт, таких как поддержка open-source, возможность описания пайплайнов с использованием кода, а также интеграция с популярными системами хранения и обработки данных.

Для проведения эксперимента проведено развертывание среды:

1. Установка и конфигурация Dagster, Apache Airflow, Apache NiFi и Prefect.
2. Настройка интеграции с системой хранения данных HDFS.
3. Разработка тестовых ETL-пайплайнов.
4. Создание идентичных пайплайнов для каждого инструмента, включающих этапы извлечения, преобразования и загрузки данных.
5. Реализация инкрементальной обработки данных с использованием механизмов, предоставляемых каждым инструментом.

Эксперимент проводился на тестовых ETL-пайплайнах. Основное внимание уделялось сравнению производительности и удобства использования инструментов. В ходе тестирования измерялись такие параметры, как время выполнения задач, потребление

ресурсов (оперативная память, загрузка CPU), а также сложность настройки и отладки пайплайнов. В ходе тестирования отметим, что некоторые инструменты демонстрируют преимущество в скорости выполнения задач, однако это часто сопровождается повышенным потреблением ресурсов. Отдельные решения требовали значительных усилий для достижения оптимальной конфигурации.

Также проведен обзор схожих работ. В статье, "A Survey of Big Data Pipeline Orchestration Tools from the Perspective of the DataCloud Project,"[6] авторы акцентируют внимание на важности эффективной управления потоками данных в современном контексте. Исследование выявляет ключевые характеристики, которые должны иметь инструменты оркестрации.

В статье "Comparative Analysis of Open-Source ML Pipeline Orchestration Platforms"[7], выполняется сравнительный анализ платформ оркестрации пайплайнов машинного обучения. В исследовании систематически оцениваются несколько платформ, таких как Apache Airflow, Kubeflow, Dagster, Metaflow, MLflow, Prefect и ZenML, на основании набора критериев, адаптированных к специфической бизнес-среде.

ЛИТЕРАТУРА

1. Что такое ETL: процесс, инструменты, подходы [Электронный ресурс]. – URL: <https://practicum.yandex.ru/blog/chto-takoe-etl/> (дата обращения: 21.02.2025).
2. Dagster Documentation [Электронный ресурс]. – URL: <https://docs.dagster.io/> (дата обращения: 23.02.2025).
3. Dagster vs Airflow: Orchestration in Big Data [Электронный ресурс]. – URL: <https://bigdataschool.ru/blog/dagster-vs-airflow-dag-orchestration-in-big-data.html> (дата обращения: 23.02.2025).
4. Prefect: Python library for workflow automation [Электронный ресурс]. – URL: <https://pypi.org/project/prefect/> (дата обращения: 27.02.2025).
5. Ong, T. C. Dynamic-ETL: a hybrid approach for health data extraction, transformation and loading [Электронный ресурс] / T. C. Ong, M. G. Kahn, B. M. Kwan [и др.] // BMC Medical Informatics and Decision Making. – 2017. – Т. 17. – Статья 134. – URL: <https://link.springer.com/article/10.1186/s12911-017-0532-3> (дата обращения: 01.03.2025).
6. A Survey of Big Data Pipeline Orchestration Tools from the Perspective of the DataCloud Project / Matskin M., Tahmasebi S., Layegh A [и др.] – 2023. – URL: <https://payberah.github.io/files/download/papers/damdid.pdf> (дата обращения: 03.03.2025).
7. Comparative Analysis of Open-Source ML Pipeline Orchestration Platforms: Bachelor Thesis for obtaining the academic degree Bachelor of Science (B.Sc.) [Электронный ресурс] / N. Grigoryan; University of Trier, – 2024. – URL: https://www.researchgate.net/profile/Narek-Grigoryan-3/publication/382114154_Comparative_Analysis_of_Open-Source_ML_Pipeline_Orchestration_Platforms/links/668e31d6b15ba559074d9a4b/Comparative-Analysis-of-Open-Source-ML-Pipeline-Orchestration-Platforms.pdf (дата обращения: 03.03.2025).

УДК 004.58:004.657

Б.Е. Арбузов (2 курс магистратуры),
О.Н. Петров, к.т.н., доцент

РАЗРАБОТКА ШАБЛОНА ВИРТУАЛЬНОГО ИНСТРУКТОРА ДЛЯ СИСТЕМ РАСШИРЕННОЙ РЕАЛЬНОСТИ

Целью работы является изучение и разработка рекомендаций по созданию виртуального инструктора, который информирует пользователя об объекте исходя из запросов, поступающих с клиентской части разработанного XR-приложения [1]. Наибольший интерес при разработке представляет использование устройств дополненной реальности (AR) и смешанной реальности (MR). Основное внимание при разработке шаблона уделяется методу формирования запросов на необходимую информацию посредством сканирования QR-кодов и штрих-кодов, а также методам хранения информации, отображаемой в результате запроса.

Виртуальный инструктор позволяет ускорить процесс передачи информации пользователю. Приложение разработано с использованием клиент-серверной архитектуры и включает приложение на конечном устройстве (мобильном устройстве, очках расширенной реальности) и размещённую на сервере базу данных, однако допускается также хранение базы данных на том же устройстве, что и приложение. Гибкость наполнения базы данных позволяет адаптировать виртуального гида под различные задачи: обучение рабочего персонала на объекте, получение расписания занятий в конкретной аудитории учебного заведения, представление информации о музейных экспонатах [2] и т. д. Информация передается пользователю как в текстовом, так и в аудиоформате.

Проблематика данной работы заключается в исследовании и применении новых инструментов и методов разработки виртуального инструктора. Разработка приложения осуществлена в среде разработки Unreal Engine 5 [3]. В качестве языка разработки выбрано совместное использование языка высокого уровня C++ и скриптового языка Blueprints.

Тестирование методов, реализованных в шаблоне, для конкретных устройств и данных осуществлено на платформе iOS. Одним из преимуществ Unreal Engine является возможность интеграции сторонних плагинов, расширяющих требуемую функциональность. Например, плагин IOSQRCodeReader [5] предоставляет классы для обработки QR-кодов. Узел Get QRCode Url из этого плагина возвращает строку, содержащую прочитанный QR-код (рисунок 1).

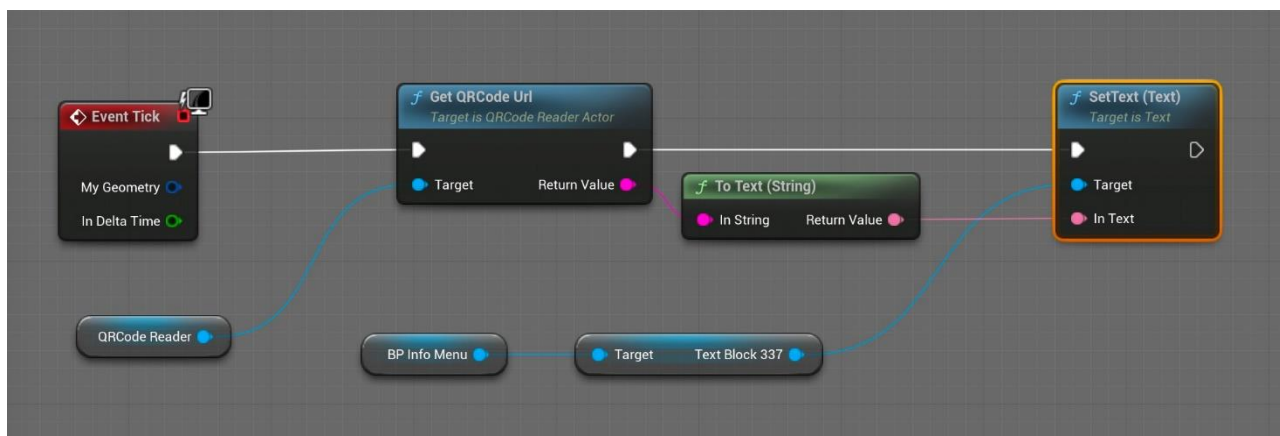


Рисунок 1 – Blueprint-реализация QR-сканера.

Сохраненные строки могут содержать ссылки на значения в локальной или удаленной базе данных. Получив запрос, система передает соответствующие данные в пользовательский интерфейс. Таким образом, каждый QR-код хранит уникальный запрос к базе данных. Такие коды можно размещать на объектах или рядом с ними, чтобы предоставлять пользователю связанную информацию.

В целях обеспечения безопасности работы с базой данных (в том числе для предотвращения SQL-инъекций подменой QR-кода) реализуется шифрование запросов перед формированием QR-кода на их основе и последующее дешифрование после считывания QR-кода камерой устройства.

Практическое использование заполненного шаблона тестируется на задачах предоставления информации по инвентарным объектам и использованию помещений.

ЛИТЕРАТУРА

1. Кузнецов В. А., Руссу Ю. Г., Куприяновский В. П. Об использовании виртуальной и дополненной реальности // International Journal of Open Information Technologies. 2019. Т. 7. № 4. С. 75–84. eISSN 2307-8162.
2. Чеснова Е. Н. Современные AR/VR/MR-технологии и оборудование в экскурсионной деятельности // Физическая культура. Спорт. Туризм. Двигательная рекреация. 2024. Т. 9. № 4. С. 115–122. ISSN 2500-0365. DOI: 10.47475/2500-0365-2024-9-4-115-122

3. Николаев И. С., Сагаева И. Д. Применение инструментария Unreal Engine для создания мобильного приложения // Тенденции развития науки и образования. 2024. № 110-17. С. 162–168. DOI: 10.18411/trnio-06-2024-956.
4. Арбузов Б. Е., Пономарёв А. А. Моделирование поведения искусственного интеллекта с применением технологии Blueprints в Unreal Engine // Неделя науки Санкт-Петербургского государственного морского технического университета. 2023. Т. 1. № 1. С. 48–52.
5. MatthewDZane. Unreal Engine iOS QR Code Reader Example Project [Электронный ресурс]. URL: <https://github.com/MatthewDZane/UnrealEngineIOSQRCodeReaderExampleProject> (дата обращения: 14.03.2025).

УДК 004

М.А. Барковская (4 курс бакалавриата),
Д.С. Шалымов, к.ф.-м.н., доцент

CODE SEARCH TOOLKIT: РАЗРАБОТКА СИСТЕМЫ ПОИСКА ФРАГМЕНТОВ ПРОГРАММНОГО КОДА НА ОСНОВЕ МЕТОДОВ ОБНАРУЖЕНИЯ ДУБЛИКАТОВ

Современные формальные методы обнаружения клонов кода (code clone detection), такие как CCAAligner [1], NIL [2] и CCSTokener [3], успешно решают задачу поиска дублирующихся фрагментов в пределах одной кодовой базы. Однако их потенциал для решения задачи точечного поиска клонов — когда требуется найти все вхождения конкретного фрагмента кода в заданном репозитории — остаётся нераскрытым. Также существующие формальные методы испытывают сложности с поиском weak clones — семантически схожих, но структурно различных фрагментов кода. Целью работы является создание REST-сервиса [8], который комбинирует адаптированные методы обнаружения клонов для решения задачи code search, то есть для поиска заданного фрагмента кода в заданном GitHub-репозитории [5].

Ключевая проблема существующих подходов:

1. Большинство методов ориентировано на глобальный анализ кодовой базы, а не на поиск клонов для заданного фрагмента.
2. Отсутствие инструментов, позволяющих комбинировать различные алгоритмы обнаружения клонов с настройкой параметров (пороги сходства). Это ограничивает гибкость при работе с разнородными объектами [4].
3. Большинство методов не приспособлены к встраиванию в рабочие процессы разработчиков, что затрудняет их практическое использование.

Разработанная система решает описанные выше проблемы, предоставляя возможность:

1. Передавать искомый фрагмент кода и ссылку на GitHub-репозиторий [5], в котором требуется найти похожие фрагменты кода, в REST-сервис.
2. Использовать адаптированные и оригинальные методы обнаружения клонов (CCAAligner [1], NIL [2]) для задачи поиска:
 - 2.1. Модификация алгоритмов методов с открытой реализацией позволяет системе искать клоны только для входного фрагмента вместо полного сканирования кодовой базы.
 - 2.2. Добавление параметров настройки позволяет пользователю самостоятельно настраивать пороги сходства клонов и выбирать весовые коэффициенты для результатов выбранных методов, что повышает точность (включая обнаружение weak clones).
3. Выбирать несколько методов (например, NIL [2] и CCSTokener [3]) и комбинировать результаты их работы с возможностью последующего ранжирования результатов по степени их схожести.

Сервис [8] написан на языке программирования Python, а взаимодействие с GitHub [5] реализовано с помощью Git [6], что позволяет системе автоматически получать доступ к содержимому репозитория. Для обеспечения кроссплатформенности и устранения зависимостей от окружения система развёртывается в Docker-контейнерах [7]. Это позволяет запускать её на любой архитектуре без необходимости ручной настройки сред выполнения.

Для дальнейшего развития системы рассматривается реализация механизма инкрементальной индексации, который позволит эффективно работать с изменяющимися кодовыми базами, что особенно актуально для последующей интеграции с IDE.

Разработанный REST-сервис [8] не только оптимизирует рутинные задачи разработки, но и открывает новые возможности для управления качеством кода. Например, при интеграции с IDE он позволит программистам мгновенно находить необходимые фрагменты кода прямо в редакторе — это сократит время на рефакторинг и снизит риск пропустить критически важные дубликаты. В образовательной сфере сервис может стать инструментом для проверки уникальности студенческих работ, выявляя не только прямые заимствования, но и семантически схожие решения.

Разработанная система [8] предоставляет инструмент для точечного поиска клонов, сочетающий академические достижения в области обнаружения клонов с практическими требованиями индустрии. Использование открытых методов и интеграция с GitHub [5] делают её практичным решением для задач поддержки качества имеющегося кода.

ЛИТЕРАТУРА

1. P. Wang, J. Svajlenko, Y. Wu, Y. Xu, and C. K. Roy, “CCAligner: a token based large-gap clone detector,” in Proceedings of the 40th International Conference on Software Engineering, pp. 1066–1077. doi: 10.1145/3180155.3180179.
2. T. Nakagawa, Y. Higo and S. Kusumoto, “NIL: large-scale detection of large-variance clones”, in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 830 – 841. doi: 10.1145/3468264.3468564
3. W. Wang, Z. Deng, Y. Xue and Y. Xu, “CCStokener: Fast yet accurate code clone detection with semantic token”, in Journal of Systems and Software, Volume 199. doi: 10.1016/j.jss.2023.111618
4. M. Zakeri-Nasrabadi, S. Parsa, M. Ramezani, C. Roy and M. Ekhtiarzadeh, “A systematic literature review on source code similarity measurement and clone detection: : Techniques, applications, and challenges”, in Journal of Systems and Software, Volume 204. doi: 10.1016/j.jss.2023.111796
5. Github. [Электронный ресурс] Режим доступа: <https://github.com>
6. Git. [Электронный ресурс] Режим доступа: <https://git-scm.com>
7. Docker. [Электронный ресурс] Режим доступа: <https://www.docker.com>
8. Code Search Toolkit (REST-сервис). [Электронный ресурс] Режим доступа: https://github.com/mmmmmarrryuy/code_search_toolkit.git

УДК 004.416

Д.С. Беляев (2 курс магистратуры),
Б.М. Медведев, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ МОРСКОЙ НАВИГАЦИОННОЙ СИСТЕМЫ НА БАЗЕ ПРОТОКОЛОВ NMEA

Протокол NMEA (National Marine Electronics Association) [1] передачи навигационных данных широко применяется для мониторинга транспортных средств в судоходстве, авиации и наземных грузоперевозках [2]. Увеличение интенсивности трафика судов в акватории портов, а также усиление требований к безопасности требуют разработки новых систем контроля, способных обрабатывать большие объемы данных в реальном времени.

Целью работы является разработка кроссплатформенного веб-приложения для приема, парсинга и хранения NMEA-сообщений, обеспечивающего обработку до 5000 сообщений в минуту и визуализацию навигационных данных в реальном времени.

Для достижения поставленной цели были решены следующие задачи:

1. Разработка алгоритмов побитового парсинга данных и архитектуры программных средств на основе микросервисов.

2. Разработка программных средств сервисов приема данных, очереди и парсинга сообщений, веб интерфейса.
3. Модульное и системное тестирование разработанных программных средств и исследование производительности.

Система построена по микросервисной архитектуре, приведенной на рисунке 1.

Система принимает NMEA-сообщения от навигационных устройств и систем с использованием стека протоколов TCP/IP.

Сервис приложений извлекает из NMEA-сообщений данные о координатах, скорости, времени, курсе и других параметрах, преобразует их в структурированный JSON-формат. Также осуществляется валидация данных и обработка возможных ошибок для последующего хранения в базе данных.

Сервис обработки сообщений передает обработанные данные в базу данных и веб-интерфейс, обеспечивая асинхронную обработку и балансировку нагрузки. RabbitMQ используется для организации надежного и масштабируемого обмена данными между сервисами.

Хранилище данных содержит структурированные данные о местоположении, скорости, времени и других параметрах навигации.

Web-сервер отображает данные в таблицах с возможностью фильтрации и сортировки. Реализована поддержка передачи командных сообщений, для управления навигационными устройствами.

Программные средства Backend разработаны с использованием ASP.NET Core и Node.js для обеспечения высокой производительности и кроссплатформенности [3].

Программные средства Frontend реализованы на основе Vue, что обеспечивает динамическое обновление данных и взаимодействие с серверной частью [4].

Система управления базами данных реализована на MongoDB, которая поддерживает хранение временных рядов и геоданных, а также предоставляет масштабируемые механизмы обработки данных [5].

Система очередей сообщений основана на RabbitMQ, что позволяет организовать асинхронную обработку данных, балансировку нагрузки и повышение отказоустойчивости [6].

Средства контейнеризации реализованы с использованием Docker, что обеспечивает изоляцию сервисов, управление зависимостями и развертывание в различных средах [7].

Средства контроля качества кода включают SonarLint и ESLint, которые выполняют статический анализ кода и выявляют ошибки на этапе разработки [8].

Исследование производительности системы показало, что:

Время обработки одного NMEA-сообщения — 1–5 мс (в зависимости от объема данных и сложности парсинга).

Задержка передачи данных через RabbitMQ — 2–10 мс (при высоконагруженной системе с оптимизированными настройками).

Время отклика API — 10–50 мс (при стандартных запросах).

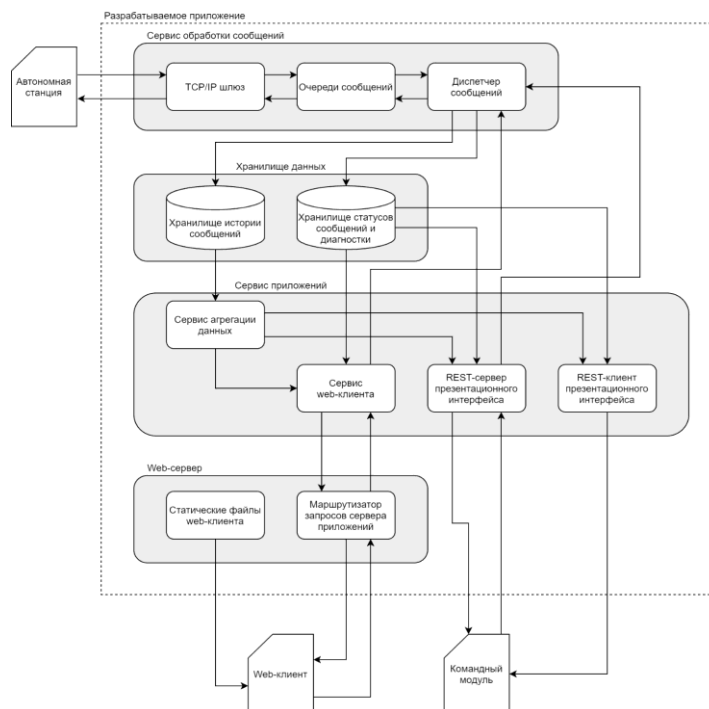


Рисунок 1 – Микросервисная архитектура приложения

Разработанные программные средства морской навигационной системы обеспечивают обработку до 50 типов NMEA-сообщений, поступающих с интенсивностью до 5000 сообщений в минуту, а также отображение навигационных данных и необходимых характеристик судна в графическом интерфейсе пользователя.

Созданное программное обеспечение будет использовано в составе ПО навигационного радиомодуля для управления движением судов.

Будущие исследования могут быть направлены на интеграцию с системами машинного обучения для прогнозирования маршрутов судов и аномалий в данных.

ЛИТЕРАТУРА

1. B. Johnson, "Протокол NMEA для морской навигации," в *IEEE Xplore*, 2019. Доступно: <https://ieeexplore.ieee.org/document/8713943>.
2. Smith и J. Doe, "Анализ производительности навигационных систем на основе NMEA," *ScienceDirect*, т. 45, № 3, стр. 215-228, 2019. Доступно: <https://www.sciencedirect.com/science/article/pii/S2405896319319526>.
3. Williams, "Эталонные тесты производительности ASP.NET Core," *arXiv preprint*, 2022. Доступно: <https://arxiv.org/abs/2205.05642>.
4. Miller и K. Brown, "Сравнение производительности фреймворков React и Vue," *ResearchGate*, т. 12, № 4, стр. 311-325, 2020. Доступно: https://www.researchgate.net/publication/336232123_Performance_Comparison_of_React_and_Angular.
5. Robinson и M. Taylor, "Эффективное хранение и запрос временных рядов в MongoDB," *Springer Journal of AI & Data Science*, т. 38, № 5, стр. 521-538, 2021. Доступно: <https://link.springer.com/article/10.1007/s10462-020-09856-5>.
6. Harris и L. Garcia, "Масштабируемость и производительность RabbitMQ в распределенных системах," *ScienceDirect*, т. 29, № 7, стр. 1103-1115, 2021. Доступно: <https://www.sciencedirect.com/science/article/abs/pii/S1383762121000476>.
7. Anderson, "Анализ производительности контейнеризации с Docker," в *IEEE Xplore*, 2021. Доступно: <https://ieeexplore.ieee.org/document/9474472>.
8. Lee и J. Kim, "Статический анализ кода с SonarLint и ESLint," *ACM Digital Library*, т. 27, № 6, стр. 899-912, 2020. Доступно: <https://dl.acm.org/doi/10.1145/3381004.3381012>.

УДК 004.04

О.В. Батушева (4 курс бакалавриата),
О.В. Александрова, ст. преподаватель

АВТОМАТИЗИРОВАННОЕ СРЕДСТВО РАСПОЗНАВАНИЯ И ПРОВЕДЕНИЯ АНАЛИЗА ЗАЯВОК КЛИЕНТОВ

Автоматизация закупок особенно важна в условиях глобальной конкуренции и перехода экономики в цифровой формат. Потребность автоматизации закупок объясняется следующими причинами:

- Корректный прогноз закупок влияет на формирование цен на товары
- Излишняя трудоемкость, ресурсоемкость и длительность закупочного процесса приводит к дополнительным издержкам и росту вероятности отказов от сотрудничества потенциальных заказчиков
- Наибольшую часть рабочего времени у менеджеров по закупкам отнимают ручные процессы сведения и обсуждение условий с поставщиками.

Целью данной работы является разработка системы автоматизации закупок, которая решает задачу: снижение трудозатрат на обработку и анализ данных.

Для достижения цели необходимо решить следующие задачи:

1. Интеграция с почтовыми сервисами для автоматического получения и фильтрации писем.
2. Анализ текста писем с выделением ключевых данных (название товара, количество, требования).
3. Интеграция с 1С КА для получения актуальной информации о товарах.
4. Проверка наличия товаров в базе данных и автоматическая генерация заказов.
5. Автоматизация отправки уведомлений клиентам.
6. Тестирование всех компонентов системы.

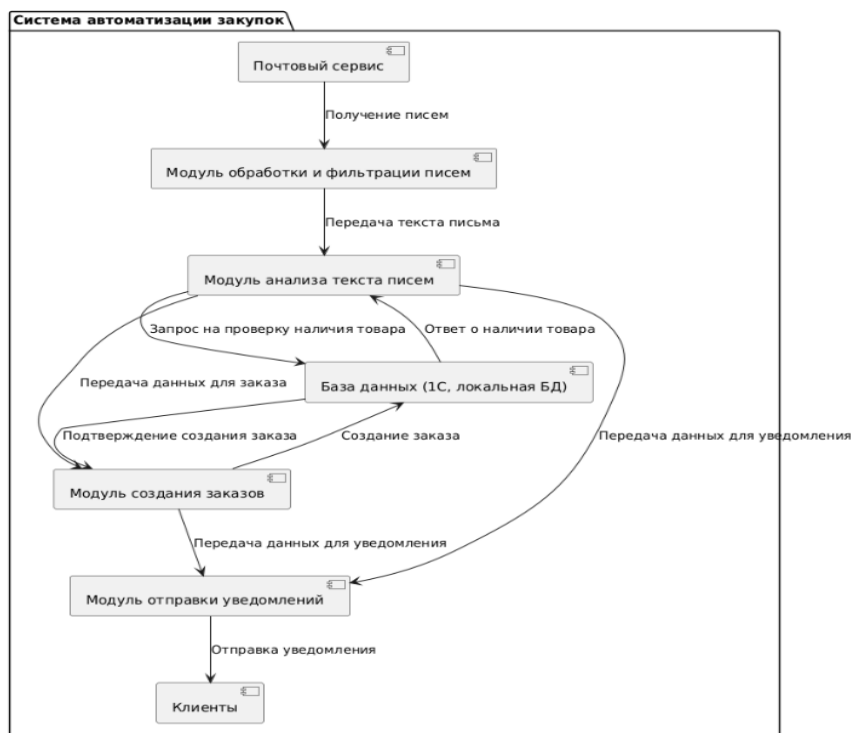


Рисунок 1 – Архитектура системы

Система автоматизации закупок состоит из нескольких взаимосвязанных компонентов. Почтовый сервис получает письма от заказчиков, которые затем обрабатываются модулем фильтрации и анализа текста для извлечения ключевых данных. Наличие товаров автоматически запрашивается у базы данных, а модуль создания заказов автоматически генерирует заказы на основе полученной информации. Модуль отправки уведомлений информирует клиентов о статусе заказов или отправляет коммерческие предложения обратно на почтовый сервис. Взаимодействие этих элементов, представленное на рисунке 1, обеспечивает эффективную автоматизацию закупок.

Описание решения:

1. Разработка NLP-модели: Извлечение ключевых данных из текста писем: название товара, количество, требования. Обучение модели на реальных письмах заказчиков.
2. Интеграция с почтовым сервисом: Автоматическое получение писем через Яндекс.Почту (POP3/IMAP). Фильтрация писем и получение информации о новых заказах.
3. Интеграция с 1С КА: Получение актуальной информации о товарах (наличие, цена). Автоматическое создание заказов и резервирование товаров.
4. Обработка данных и создание заказа: Создание счета и отправка информации клиенту на электронную почту.
5. Тестирование системы: Проверка работы каждого модуля: подключения к почтовому сервису через POP3/IMAP, NLP-модели, интеграции с 1С КА. Оценка точности извлечения данных и корректности обработки.

Система автоматизации закупок предлагает ключевые преимущества, такие как снижение трудозатрат за счет автоматизации рутинных операций, включая обработку писем и создание заказов, что также уменьшает ошибки ручного ввода. Она повышает скорость обработки заказов благодаря мгновенному анализу данных из писем и быстрому формированию заказов и отправки уведомлений. Решение легко интегрируется с существующими системами, такими как 1С КА, а также электронной почтой.

Система автоматизации закупок снижает трудозатраты, ускоряет обработку заказов и минимизирует ошибки за счет автоматизации рутинных процессов. Ее интеграция с почтовыми сервисами и 1С КА, а также модульная архитектура, обеспечивают гибкость и масштабируемость, повышая эффективность закупок на предприятии.

ЛИТЕРАТУРА:

1. Перелыгин И.М. Анализ программных продуктов и изучение автоматизации процессов в сфере мониторинга закупок и товаров // Вопросы безопасности. — 2024. — № 3. — С. 45-52.
2. Агентство Agora. Автоматизация закупок: современные подходы и решения [Электронный ресурс]. — URL: https://www.agora.ru/avtomatizaciya-zakupok/?utm_source=chatgpt.com (дата обращения: 10.10.2023).
3. Comindware. Цифровизация закупок: тренды и технологии [Электронный ресурс]. — URL: https://www.comindware.ru/blog/cifrovizacija-zakupok/?utm_source=chatgpt.com (дата обращения: 10.10.2023).
4. Сбоев, А. Г. Нейросетевое моделирование и машинное обучение на основе экспериментальных и наблюдательных данных : диссертация на соискание ученой степени доктора наук : 05.13.18 / Сбоев Александр Георгиевич. — Москва : Национальный исследовательский ядерный университет «МИФИ», 2021. — 389 с.

УДК 004.42

Д.Е. Булатов, А.А. Вяземский, А.А. Шожат (1 курс магистратуры),
Т.В. Коликова, ст. преподаватель

АНАЛИЗ СВЕДЕНИЙ О ШАХМАТНЫХ ПАРТИЯХ И ИГРОКАХ LICHESS.ORG

Шахматы — это не просто игра, а сложная стратегическая дисциплина, которая требует от игроков глубокого анализа, планирования и предвидения. С развитием технологий и доступностью онлайн-платформ, таких как Lichess.org, шахматы стали более доступными и популярными среди широкой аудитории. Этот сайт предоставляет обширную базу данных о партиях, игроках и их рейтингах.

В данной статье представлен анализ данных [1] о шахматных партиях и игроках с сайта Lichess.org. Основные аналитические задачи исследования включают в себя анализ следующих аспектов:

- Соотношение вариантов завершения игры.
- Соотношение побед, поражений и ничьих в зависимости от типа завершения игры.
- Зависимость количества игр от времени суток.
- Соотношение побед, поражений и ничьих в зависимости от разницы в рейтинге.
- Соотношение побед, поражений и ничьих в зависимости от времени, потраченного на игру.

Для скачивания данных используется Lichess API [2]. С помощью него можно получить информацию обо всех шахматных партиях любого игрока. Полученная информация по каждому игроку содержится в отдельных json-файлах. В целом, можно задавать фильтры для партий (по дате, цвету фигур, режиму игры и т.п.). Можно указывать данные, которые будут получены в результате. Были использованы stream-запросы, и была загружена вся доступная информация о партии.

Была разработана следующая архитектура [3] проекта, представленная на рисунке 1

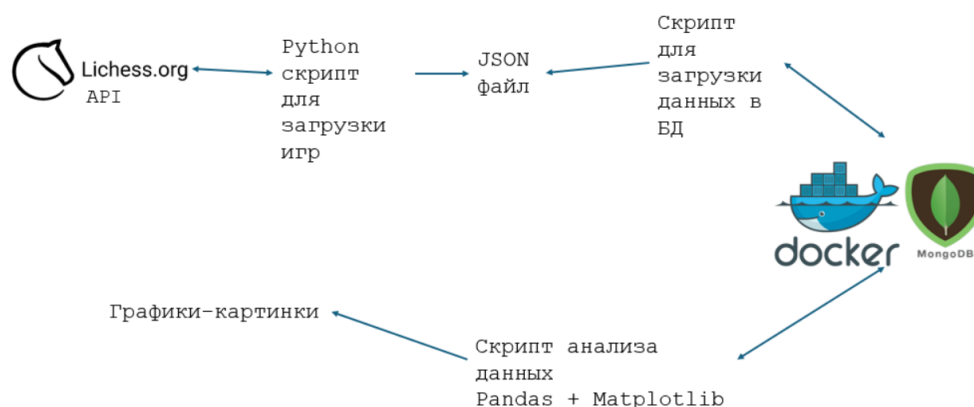


Рисунок 1 – Архитектура проекта

Для анализа партий одинакового типа применяется фильтрация набора данных по режиму игры.

В итоге данные, которые были получены и проанализированы, были подвергнуты статистическим методам оценки полноты. Полученное распределение [3] говорит о том, что те данные, которые мы брали для анализа в нашем исследовании являются полными. График распределения приведен на рисунке 2.

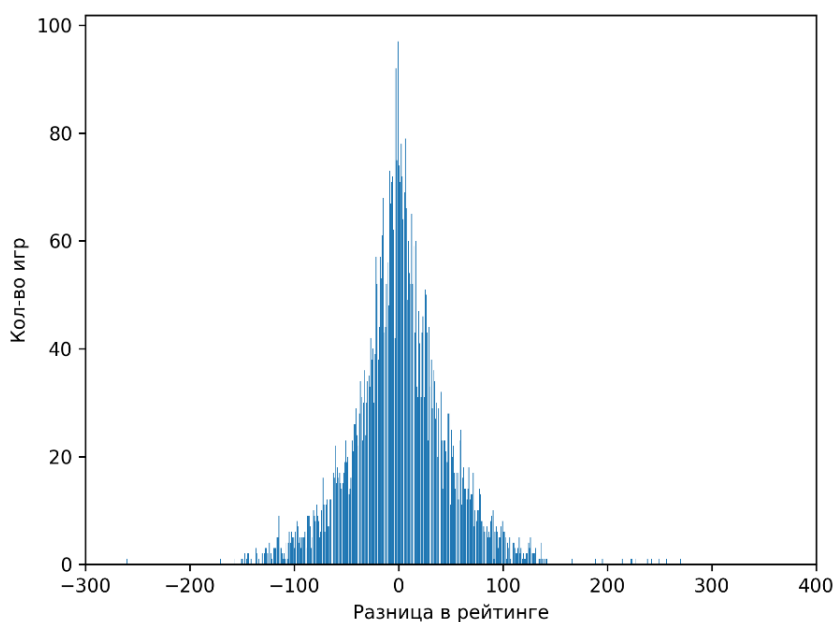


Рисунок 2 – Распределение на основе партий одного игрока

В данном исследовании был проведен анализ игроков в шахматы и их партий с сайта Lichess.org. Все поставленные аналитические задачи были выполнены, что позволило научиться работать с данными, а также использовать всю доступную информацию, которую можно переработать из них. Результаты исследования и практическая реализация данного проекта могут быть полезны для людей, которые увлечены шахматы, для самих игроков, которым интересен подробный анализ своих партий для улучшения своих стратегий.

ЛИТЕРАТУРА

1. Анализ данных с использованием Python. [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/353050/>
2. Lichess API. [Электронный ресурс] Режим доступа: <https://lichess.org/api#section/Introduction>

3. Роберт Мартин, Чистая архитектура. Искусство разработки программного обеспечения. 2022, - с. 232-235
4. Нормальное распределение. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Нормальное_распределение

УДК 004.65

А.А. Борисов (4 курс бакалавриата),
С.А. Нестеров, к.т.н., доцент

УТИЛИТА МИГРАЦИИ БАЗ ДАННЫХ ИЗ ORACLE DATABASE В РЕД БАЗУ ДАННЫХ

В настоящее время многие российские компании переходят на отечественное программное обеспечение (ПО). Это происходит в силу разных обстоятельств: прекращение лицензирования со стороны зарубежных поставщиков, а также рекомендации для государственных корпораций и компаний с государственным участием о переходе на отечественное ПО [1].

Oracle Database – это объектно-реляционная система управления базами данных (СУБД) от компании Oracle. Данная СУБД использует процедурный блочно-структурированный язык PL/SQL, разработанный компанией Oracle для написания хранимых в базе данных подпрограмм [5].

Ред База Данных – объектно-реляционная система управления базами данных от российской компании Ред Софт. Ядро Ред Базы Данных построено на основе СУБД с открытым исходным кодом Firebird [2]. Для написания алгоритмов обработки данных в хранимых процедурах, функциях и триггерах, в Ред Базе Данных используется процедурное расширение языка SQL – PSQL.

Целью представляемой работы является разработка утилиты для автоматизации миграции базы данных с Oracle Database Enterprise Edition 21с на Ред База Данных 5.0.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Анализ проблем миграции с Oracle Database в Ред Базу Данных и поиск их решения.
2. Определение методов для устранения обнаруженных проблем, поиск и разработка аналогов для реализации функциональных возможностей СУБД Oracle Database, которые отсутствуют в СУБД Ред База Данных.
3. Реализация утилиты, которая преобразует метаданные СУБД Oracle Database в СУБД Ред База Данных.

В 2017 году вышла статья, где автор также предпринял попытку разработки подобной утилиты [6]. Однако с того времени команды разработчиков обеих СУБД проделали большую работу и сильно улучшили свои продукты. Стала актуальной разработка новой утилиты миграции для последних версий Oracle Database и Ред База Данных.

Общий принцип работы утилиты:

1. Синтаксический анализ скрипта создания базы данных в СУБД Oracle Database.
2. Построение синтаксического дерева (Abstract Syntax Tree).
3. Обход синтаксического дерева с преобразованием отдельных узлов и формированием выходного скрипта для создания базы данных в СУБД Ред База Данных.

Утилита использует библиотеку ANTLR (ANother Tool for Language Recognition). ANTLR – это генератор синтаксических анализаторов для чтения, обработки, выполнения или перевода структурированного текста или двоичных файлов [4].

Принцип работы ANTLR основан на синтаксическом анализе входного потока данных, построения абстрактного синтаксического дерева, обхода этого дерева и предоставления, сгенерированного API для внедрения логики для каждого из событий обхода. API генерируется на основе грамматики. Грамматика описывается в виде расширенной формы Бэкуса – Наура и состоит из лексических и синтаксических правил [3].

ANTLR позволяет генерировать анализаторы для многих языков программирования: C#, C++, Java. Для написания утилиты миграции был выбран язык программирования Java, что обеспечивает программе надёжность, безопасность и кроссплатформенность.

Отладка и написание грамматики будет происходить в специальном расширении для среды разработки IntelliJ Idea – ANTLR v4.

По заданной грамматике ANTLR генерирует методы согласно двум паттернам проектирования: Visitor и Listener [7]. Для данной утилиты был выбран подход на основании шаблона Listener.

На данный момент утилита реализует следующий функционал:

1. Преобразование оператора create function, create procedure, create trigger, create package.
2. Преобразование оператора create table и основных типов данных.
3. Преобразование оператора create view и оператора select.

4. Преобразование основных конструкций PL/SQL: циклы, оператор IF-ELSE, обработка исключений, объявление курсора.

Пример конвертации оператора создания процедуры представлен в таблице 1.

Таблица 1 - Пример конвертации оператора создания процедуры

Oracle Database
<pre>CREATE OR REPLACE PROCEDURE "HR"."ADD_JOB_HISTORY" (p_emp_id job_history.employee_id%type , p_start_date job_history.start_date%type , p_end_date job_history.end_date%type) IS BEGIN INSERT INTO job_history (employee_id, start_date, end_date) VALUES(p_emp_id, p_start_date, p_end_date); END add_job_history;</pre>
Ред База Данных
<pre>CREATE OR ALTER PROCEDURE "ADD_JOB_HISTORY" (p_emp_id TYPE OF COLUMN job_history.employee_id , p_start_date TYPE OF COLUMN job_history.start_date , p_end_date TYPE OF COLUMN job_history.end_date) AS BEGIN INSERT INTO job_history (employee_id, start_date, end_date) VALUES(:p_emp_id, :p_start_date, :p_end_date); END;</pre>

ЛИТЕРАТУРА

1. Министерство цифрового развития, связи и массовых коммуникаций РФ [Электронный ресурс] Режим доступа: <https://digital.gov.ru/ru/documents/7342/>
2. Ред База Данных 5.0. Руководство администратора [Электронный ресурс] Режим доступа: https://reddatabase.ru/ru/pdfjs/?file=/media/documentation/RedDatabase-5.0.0-Admin_Guide-ru.pdf
3. Методы трансляции: метод. пособие для студ. спец 1–31 03 04 “Информатика” всех форм обуч. / И.И. Пилецкий, В.В. Шиманский. – Минск: БГУИР, 2012. – 88с.
4. Terence Parr. The Definitive ANTLR4 Reference, The Pragmatic Bookshelf, 2012. – 322 с.
5. Oracle Database Documentation 21с [Электронный ресурс] Режим доступа: <https://docs.oracle.com/en/database/oracle/oracle-database/index.html>

6. Симаков Р.А., Еремин И.А. Разработка утилиты миграции БД с Oracle на Ред База Данных // Алгоритмы, методы и системы обработки данных, 2017. –№1(35). –с. 45-57
7. Wengner, M. Practical Design Patterns for Java Developers, 2023. — 243 с.

УДК 004.42:004.415:004.738.5

И.Н. Булгаков (4 курс бакалавриата),
А.В. Петров, ст. преподаватель

АДАПТАЦИЯ АВТОМАТИЗИРОВАННОГО СЕРВИСА ДЛЯ АНАЛИЗА РЕЗУЛЬТАТОВ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ В СИСТЕМЕ ОБЛАЧНОГО ГЕЙМИНГА

Современные компании, работающие с облачными технологиями и потоковыми сервисами (в том числе облачным геймингом), всё чаще сталкиваются с необходимостью автоматизировать анализ больших объёмов данных по нагрузочным тестам [1]. В условиях микросервисной архитектуры, где каждый сервис отвечает за определённую функцию (авторизацию, обработку платежей, трансляцию видеопотока и т. д.), важно быстро определять, где возникают узкие места и какова динамика их изменения [2]. Отслеживание этих метрик вручную требует существенных временных затрат. Поэтому создание автоматизированного решения, позволяющего собирать, хранить и визуализировать результаты нагрузочных тестов, даёт существенную экономию усилий и помогает оперативно реагировать на возможную деградацию производительности [3].

С течением времени нагрузочное тестирование может расширяться на всё большее количество сервисов, а их результаты часто складываются в произвольных местах: в артефактах CI/CD-систем, на локальных машинах разработчиков или вообще остаются в виде HTML-отчётов, которые никто не просматривает [4]. Таким образом, возрастает риск утраты исторических данных, необходимых для глубокого анализа и выявления тенденций.

Непосредственно в данной работе решается проблема адаптации специализированного микросервиса, который извлекает метрики из отчётов нагрузочного тестирования (формируемых, например, инструментом K6 в GitLab), и обеспечивает их дальнейшее хранение в базе данных для наглядной визуализации через Grafana [5]. Это позволяет не «изобретать велосипед» заново и не менять устоявшийся процесс тестирования, а лишь дополнить его модулем, автоматически собирающим и обрабатывающим результаты.

Целью работы является адаптация и доработка микросервиса таким образом, чтобы он мог:

1. Автоматически получать отчёты нагрузочного тестирования из артефактов GitLab;
2. Парсить различные форматы (HTML, JSON и т.д.), в которых генерируются метрики K6;
3. Записывать полученные данные в базу данных (развёрнутую, к примеру, в Docker);
4. Предоставлять Grafana удобный интерфейс для визуализации исторических данных по всем прогонам нагрузочных тестов [4].

После обновления конфигурации CI/CD в GitLab и развёртывания свежего окружения (например, с использованием Docker Compose) возникли проблемы совместимости микросервиса с изменённой структурой отчётов. Основные задачи сводились к следующему:

1. Обновление структуры проекта и зависимостей – нужно было настроить рабочую среду в Docker (и при необходимости docker-compose), включая парсеры для различных форматов отчётов и библиотеки для интеграции с GitLab.
2. Исправление ошибок, связанных с устаревшими методами извлечения метрик – некоторые отчёты содержат иные теги/JSON-поля, вследствие чего старый код парсинга выдавал ошибки и требовал доработки.
3. Обеспечение корректной работы микросервиса с обновлённой схемой базы данных – понадобилось расширить таблицы, отвечающие за хранение метрик, чтобы учитывать новый набор показателей (время отклика, перцентиль, количество ошибок и др.) [2].

В первую очередь была исследована структура создаваемых отчётов, а также настроен локальный стенд с помощью Docker Desktop. Было выполнено подключение к локальному GitLab Runner (или к тестовому pipeline), чтобы воссоздать полную цепочку: от генерации отчёта до его загрузки в микросервис.

Для решения проблем совместимости проведён рефакторинг парсера: часть устаревших методов чтения HTML- и JSON-файлов заменена на более надёжные механизмы извлечения данных, учитывающие специфику новых версий К6-отчётов. Также доработана логика записи в БД, включающая дополнительные проверки полей и автоматическое добавление временных меток каждого прогона теста.

Для успешной визуализации в Grafana настроен отдельный дашборд, который подтягивает метрики из базы данных. При этом в шаблонах графиков дополнительно добавлены фильтры по дате/времени, имени микросервиса и версии приложения, чтобы аналитики могли сравнивать результаты разных прогонов и видеть прогресс или деградацию работы системы [5].

Результат проделанной работы – система с адаптированным микросервисом, который автоматически обрабатывает результаты нагрузочных тестов для системы облачного гейминга, представлен на рисунке 1. Исправлены все ошибки, связанные с несовместимостью структур отчётов и базы данных, а все ключевые функции (сбор, парсинг, хранение, визуализация) работают без сбоев. Это существенно упрощает анализ производительности и позволяет команде разработчиков в любой момент отследить изменения метрик, выявить проблемные места и оперативно принять меры по оптимизации.

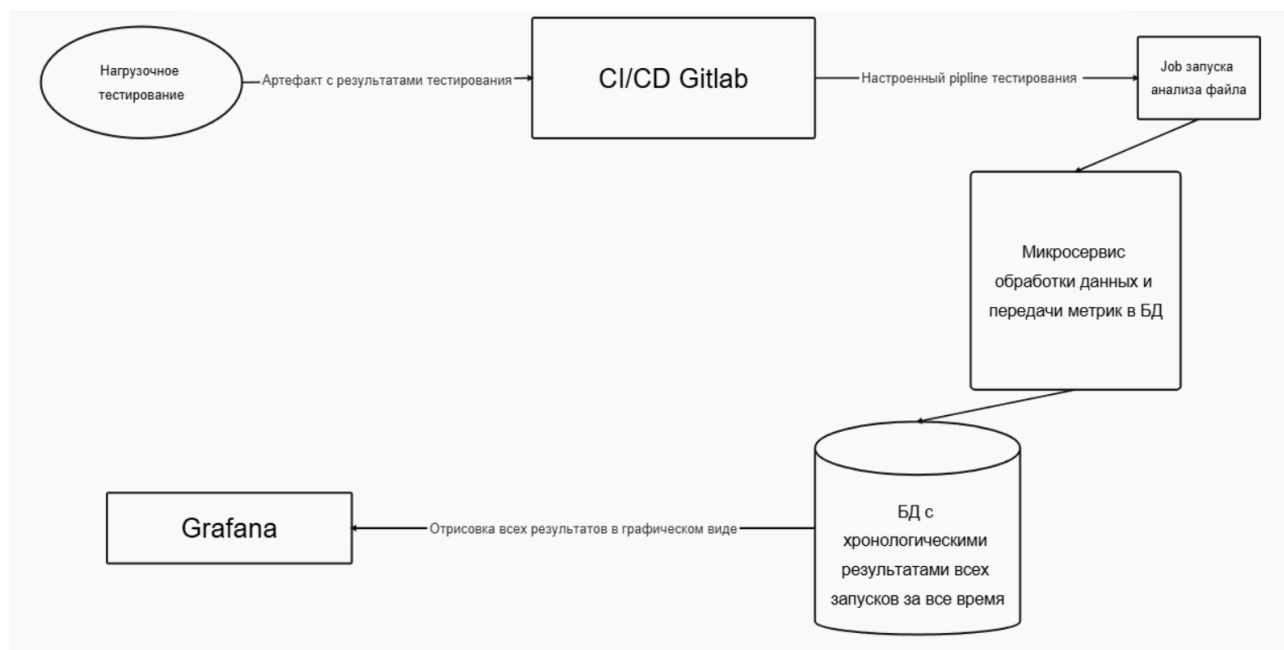


Рисунок 1 – Архитектура системы обработки результатов нагрузочного тестирования и их визуализации

ЛИТЕРАТУРА

1. Гук И.Ю., Новиков С.А. Методы обработки больших данных в облачных сервисах // Вестник компьютерных наук. – 2021. – №4. – С. 45-56.
2. Иванов П.Д. Анализ метрик производительности в микросервисной архитектуре. – Москва: Изд-во Техносфера, 2020. – 240 с.
3. Смирнов А.В. Автоматизированные системы мониторинга нагрузки. – СПб.: Питер, 2019. – 312 с.
4. Petrov K., Ivanov D. Performance Metrics and Load Testing in Cloud Services // International Journal of Data Science. – 2022. – Vol. 8, No. 2. – P. 112-125.

5. Brown J., Miller T. Continuous Integration and Performance Testing in DevOps // Journal of Software Engineering. – 2020. – Vol. 15, No. 3. – P. 89-104.
6. Docker Documentation. Using Docker for Load Testing. URL: <https://docs.docker.com/load-testing/> (дата обращения: 16.03.2025).

УДК 004.622

М.Г. Брежо (2 курс магистратуры),
Н.В. Воинов, к.т.н., доцент

РАЗРАБОТКА МЕХАНИЗМА ПЕРЕДАЧИ СОБЫТИЙ СИСТЕМЫ ЗАКРОМА.ХРАНИЕНИЕ ДЛЯ ИНТЕГРАЦИИ С SIEM-ПЛАТФОРМОЙ ARCSIGHT

Цель работы – создание гибкого инструмента для автоматизированного преобразования неструктурированных логов системы «Закрома.Хранение» [1] в стандартизированный формат CEF [2] и обеспечения их асинхронной передачи в SIEM-платформу ArcSight [3].

Для достижения поставленной цели необходимо решить следующие задачи:

1. Анализ структуры исходных логов системы «Закрома.Хранение» и требований ArcSight к формату CEF.
 2. Разработка модуля парсинга данных из syslog-файлов и БД системы.
 3. Реализация трансформатора событий с обязательными полями CEF (DeviceVendor, DeviceProduct, SignatureID).
 4. Интеграция с Apache Kafka [4] для буферизации данных и обеспечения отказоустойчивости.
 5. Настройка защищенного REST-канала передачи данных в ArcSight.
- Архитектура решения представлена на рисунке 1:

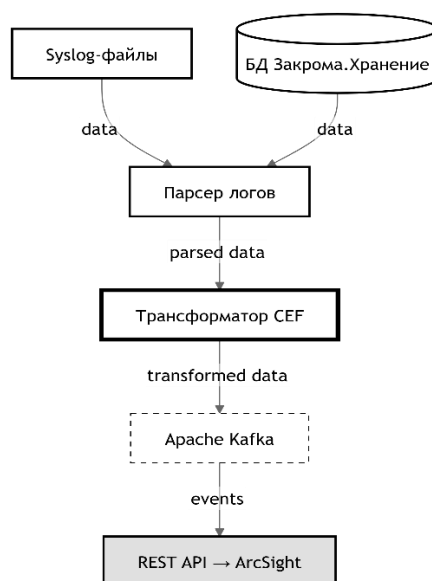


Рисунок 1 – Архитектура системы

К разрабатываемому инструменту предъявляются следующие основные требования:

1. Источники данных: syslog-файлы [5] и БД «Закрома.Хранение».
2. Функция модуля парсинга: извлечение ключевых параметров (ID пользователей, IP-адреса, типы операций).
3. Функция трансформатора CEF: формирование структурированных событий согласно документации ArcSight.
4. Применение Apache Kafka: промежуточный брокер сообщений для асинхронной обработки и балансировки нагрузки.
5. Механизм доставки данных: REST API.

Для реализации инструмента был выбран следующий технологический стек:

1. Язык программирования Python (библиотеки pandas для обработки данных, confluent-kafka для интеграции с Kafka).

2. Apache Kafka для буферизации событий.

3. REST API с использованием FastAPI для взаимодействия с ArcSight.

4. СУБД PostgreSQL для временного хранения логов.

Ожидаемые результаты:

1. Автоматизация конвертации логов в CEF-формат.

2. Снижение нагрузки на SIEM-платформу за счет асинхронной передачи через Kafka.

3. Повышение детализации данных безопасности (регистрация 100% событий системы).

ЛИТЕРАТУРА

1. Документация ПО Закрома.Хранение [Электронный ресурс] URL:

<https://zakroma.ru/knowledgeBase/storage> (дата обращения: 15.03.2023)

2. NXLog Documentation. Process ArcSight Common Event Format (CEF) logs [Электронный ресурс]

URL: <https://docs.nxlog.co/integrate/cef-logging.html> (дата обращения: 15.03.2023)

3. HPE ArcSight ESM Security Information and Event Management, SIEM [Электронный ресурс]

URL: [https://www.tadviser.ru/index.php/Продукт:HPE_ArcSight_ESM_\(Security_Information_and_Event_Management,_SIEM\)](https://www.tadviser.ru/index.php/Продукт:HPE_ArcSight_ESM_(Security_Information_and_Event_Management,_SIEM)) (дата обращения: 15.03.2023)

4. Kafka 3.9 Documentation [Электронный ресурс] URL: <https://kafka.apache.org/documentation/> (дата

обращения: 15.03.2023)

5. What is Syslog? Components, benefits, and best practices [Электронный ресурс] URL:

<https://dev.to/parseable/what-is-syslog-components-benefits-and-best-practices-4k4i> (дата обращения: 15.03.2023)

УДК 004.457

А.И. Возиян (2 курс магистратуры),

П.Д. Дробинцев, к.т.н., доцент

АВТОМАТИЗАЦИЯ ПРОВЕРКИ КОРРЕКТНОСТИ РАЗВЕРТЫВАНИЯ СЕРВЕРНОЙ ЧАСТИ 1С В ОБЛАЧНОЙ СРЕДЕ РАЗРАБОТКИ

В периметре ИТ-компании реализуется проект по автоматизации развертывания облачной платформы разработки, включающей облачную среду разработки и автоматизированное рабочее место (АРМ) разработчика 1С. Цель проекта состоит в снижении времени разработки платформенных решений на базе 1С с 3 месяцев до 1 недели. Развертывание приложений 1С выполняется с использованием плейбуков (playbooks) системы управления конфигурациями Ansible с включением модуля shell для запуска Bash-скриптов, отвечающих за конфигурирование серверной части 1С [1].

Применение данного модуля не удовлетворяет требованиям проекта, поскольку оно не обеспечивает гибкости в управлении кодом и имеет ограничения в параллельном выполнении скриптов, что замедляет процесс развертывания платформы. Bash-скрипты не обладают встроенными средствами для надежной обработки и исправления ошибок, возникающих при развертывании серверной части 1С, что приводит к некорректной работе плейбуков [2].

Возникающие неисправности приводят к увеличению времени предоставления облачного АРМ для 1С-разработчиков и росту трудозатрат команды сервисной поддержки на устранение некорректной работы ПО. Поэтому актуальной является задача по разработке метода автоматизированной проверки корректности развертывания серверной части 1С и его реализация в программном средстве.

Метод автоматизированной проверки корректности развертывания серверной части 1С должен соответствовать следующим требованиям:

1. Проверка состояния серверных компонентов 1С.

На каждом этапе развертывания необходимо контролировать работоспособность всех серверных компонентов в виде службы, кластера и инфобазы 1С путем проверки их состояния.

2. Автоматическое восстановление работоспособности.

В случае неуспешного запуска серверных компонентов метод должен обеспечивать автоматическое восстановление их работоспособности, например, посредством перезапуска.

3. Изменение настроек компонентов серверной части 1С.

Метод должен поддерживать изменение параметров кластера и инфобазы 1С для приведения к целевому состоянию, включая публикацию инфобазы на веб-сервере и управление сеансами пользователей.

4. Формирование человекочитаемых отчетов о неисправностях.

При возникновении нетипичных ошибок развертывания должен автоматически генерироваться человекочитаемый отчет об ошибке для специалистов сервисной поддержки, что особенно важно для администраторов, не знакомых с особенностями логирования 1С.

5. Классификация ошибок развертывания.

Метод должен обеспечивать классификацию типичных ошибок развертывания с возможностью их автоматического исправления, а в случае более сложных неисправностей – для формирования детализированных отчетов с целью их передачи сервисной команде.

В ходе сравнительного анализа подходов и инструментов проверки корректности развертывания 1С были рассмотрены средства управления и администрирования, системы логирования и диагностики, а также инструменты тестирования развертывания. На основе критериев совместимости с ОС Linux, поддержки автоматизации в скриптах, функциональности и удобства использования выбран перечень инструментов, формирующих основу реализации предлагаемого метода автоматизированной проверки. В этот перечень входят встроенные утилиты администрирования 1С RAS и RAC [3] наряду с системными утилитами ОС `systemctl` и `journalctl` [4].

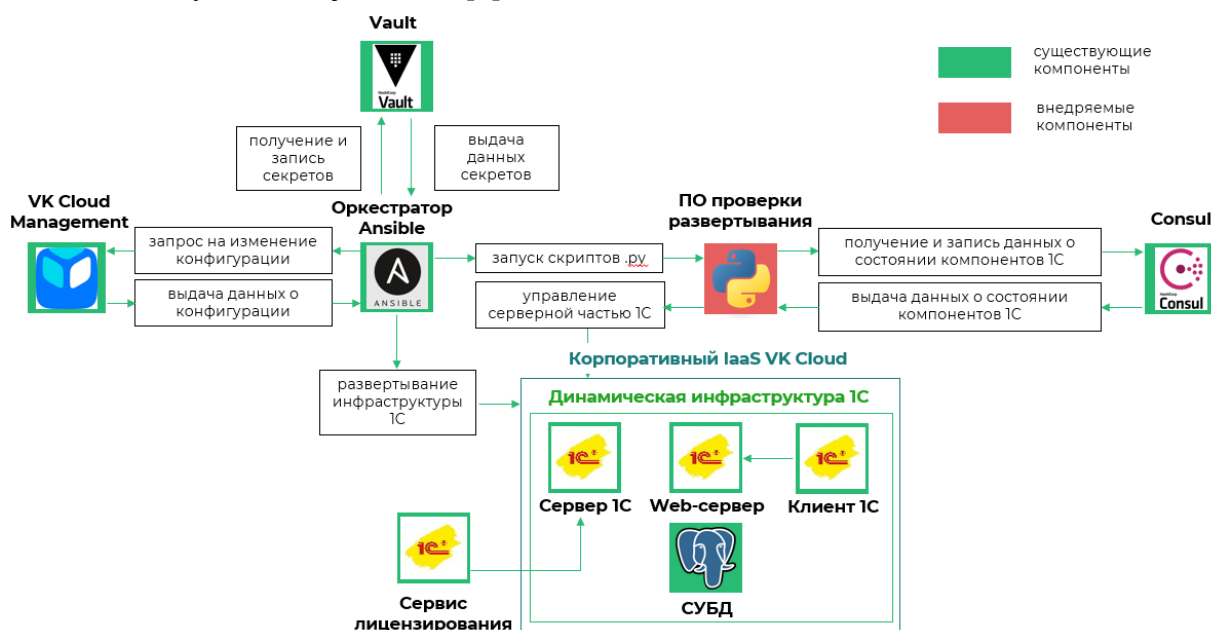


Рисунок 1 – Архитектура проекта облачной платформы разработки

Архитектура программного средства, представленная на рисунке 1, состоит из трех ключевых модулей: управления, логирования и отчетности. Модуль управления отвечает за развертывание серверных компонентов 1С, включая запуск службы, создание и настройку кластера серверов, а также создание и конфигурирование инфобазы на этом кластере. В процессе работы модуль выполняет проверку корректности развертывания каждого компонента и автоматически восстанавливает его работоспособность при выявлении неисправности. Модуль логирования фиксирует ключевые этапы установки и настройки

системы. В заключение, модуль отчетности формирует человекочитаемые отчеты об ошибках в случае возникновения неисправностей в ходе развертывания.

Программное средство реализовано на языке программирования Python и посредством модуля script встраивается в плейбуки Ansible, что позволяет автоматизировать процесс развертывания динамической инфраструктуры 1С. Дополнительно реализована интеграция с CMDB-системой Consul, в которой фиксируются статусы состояния серверных компонентов 1С [5].

Работа выполняется при поддержке компании «Газпромнефть-ИТО». За счет внедрения программной реализации метода автоматизированной проверки ожидается снижение времени предоставления облачного АРМ для 1С-разработчиков на 20% и сокращение времени устранения инцидентов в рамках проекта на 15% в течение первых трех месяцев внедрения.

ЛИТЕРАТУРА

1. ansible.builtin.shell module – Execute shell commands on targets [Электронный ресурс]. – URL: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/shell_module.html (дата обращения: 13.03.2025).
2. Horton E., Parnin C. Dozer: Migrating shell commands to Ansible modules via execution profiling and synthesis //Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice. – 2022. – С. 147-148.
3. Глава 5. Администрирование :: Клиент-серверный вариант. Руководство администратора :: 1С:Предприятие 8.3.7. Документация [Электронный ресурс]. – URL: <https://its.1c.ru/db/v837doc/bookmark/cs/TI000000136> (дата обращения: 13.03.2025).
4. Работа с systemctl и journalctl в Linux [Электронный ресурс]. – URL: <https://www.ispmanager.ru/knowledge-base/rabota-s-systemctl-i-journalctl-v-linux> (дата обращения: 13.03.2025).
5. Практическое руководство по HashiCorp Consul — Часть 1 [Электронный ресурс]. – URL: <https://habr.com/ru/articles/531602/> (дата обращения: 13.03.2025).

УДК 004.4'2

С.А. Дружков (4 курс бакалавриата),
Д.В. Кознов, д.т.н., профессор

ПОИСК КЛОНОВ И АВТОМАТИЧЕСКИЙ РЕФАКТОРИНГ LUA-ПРОГРАММ ДЛЯ УМЕНЬШЕНИЯ РАЗМЕРА КОДА СЕТЕВЫХ УСТРОЙСТВ

Широко известен скриптовый язык Lua. Этот язык часто используется в компьютерных играх, а также при разработке встроенных систем. Например, компании Cisco, Xiaomi и Huawei используют Lua в программном обеспечении для роутеров. Программное обеспечение роутеров крупной телекоммуникационной компании состоит из множества компонент, которые реализуют различные сетевые протоколы и интерфейсы, базу данных, драйвера устройств и пр. Функциональность многих компонент похожа, поэтому их разработка ведётся путём копирования и доработки уже существующих (copy-paste). Стоит также отметить, что роутеры и другие встроенные системы имеют существенные технические ограничения, и дублирование кода вынуждает производителей использовать дополнительную память, что снижает рентабельность итоговых изделий.

Проблема дублирования кода стоит еще более остро для приложений на языке Lua, т. к. этот язык используется на роутерах в режиме интерпретации (на это есть свои причины). Таким образом задача уменьшения объемов Lua-кода путём устранения дублирования является актуальной.

Существуют различные техники, позволяющие уменьшить итоговый размер системы. Начиная от использования классических алгоритмов сжатия данных до удаления отладочной информации из исполняемых файлов. Но для интерпретируемых языков обычно используются техники, основанные на преобразованиях исходного кода программ [1, 2]. Обычно этот

процесс включает удаление пробелов, комментариев, пустых строк и переименование локальных переменных. Наиболее продвинутые инструменты поддерживают удаление неиспользуемого и недостижимого кода, а также другие оптимизации. Однако все эти преобразования не решают проблему дублирования кода.

Целью данной работы является разработка индустриального решения для поиска клонов и автоматического рефакторинга Lua-программ, предназначенных для сетевых устройств, с последующей интеграцией в конвейер сборки крупной телекоммуникационной компании.

Для достижения этой цели в рамках работы были сформулированы следующие задачи.

1. Сформулировать требования к индустриальному решению.
2. Создать релевантный подход к поиску клонов и автоматическому рефакторингу, а также спроектировать архитектуру решения.

3. Реализовать поиск клонов и автоматический рефакторинг Lua-программ.

4. Провести тестирование решения и валидацию результатов работы.

В рамках работы над решением были выявлены следующие функциональные требования: возможность поиска клонов кода на языке программирования Lua, возможность автоматического рефакторинга групп клонов, возможность поиска и рефакторинга вложенных клонов, конфигурируемость поиска и рефакторинга клонов. Нефункциональные требования включает в себя поддержку больших проектов (> 10 миллионов строчек кода), минимальное число зависимостей от внешних библиотек и программ, детерминированность результатов работы решения, сохранение структуры кода при рефакторинге, безопасность всех преобразований кода.

Поиск клонов реализован на основе алгоритма Vuddy [5]. Выбор этого алгоритма обусловлен скоростью работы и ориентированностью на большие проекты. Выбранным методом рефакторинга является извлечение функции с последующей ее параметризацией. Аналогичный метод рефакторинга применяется в PyTeRor [3], но только для клонов тестов.

Архитектура решения представлена на рисунке 1. Решение состоит из трех основных компонент: нормализатор, детекторов клонов, движок рефакторинга. Нормализатор отвечает за предварительную обработку исходного текста программ для увеличения числа найденных клонов. Детектор клонов отвечает за поиск рефакто-ориентированных клонов. Движок рефакторинга реализует однопроходный алгоритм рефакторинга клонов.

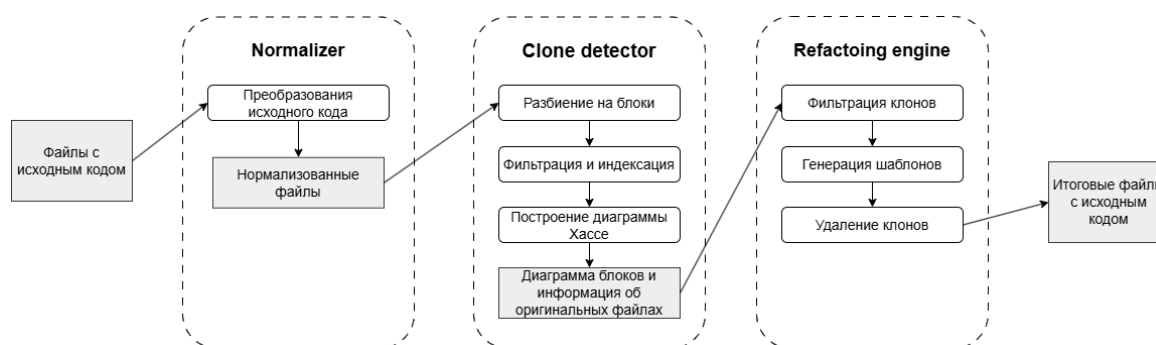


Рисунок 1 – Архитектура решения

Компонента нормализации основана на идеях из работы [4] и предоставляет следующие трансформации: удаление лишних return, необязательной инициализации, лишних «;», неиспользуемого кода; преобразование обращения по полю в обращение по ключу; переименование меток; объединение объявлений переменных. Их применение позволяет увеличить число найденных клонов на 1-2% в зависимости от проекта.

Детектор клонов использует лексический подход [5], адаптируя его для работы с Lua, а также в отличие от оригинала позволяет искать клоны на уровне блоков кода. Поскольку выбранным способом рефакторинга является извлечение функции, то уже в моменте поиска клонов проверяется возможность извлечь соответствующий фрагмент кода. При объединении клонов в классы проверяется согласованность имен локальных переменных и синтаксических

конструкций между фрагментами кода, что позволяет сильно упростить алгоритм рефакторинга.

Движок рефакторинга ориентирован на поддержку удаления сразу целых групп клонов и рефакторинг вложенных клонов. Как и исходные блоки кода, так и группы клонов образуют древовидную структуру. Поэтому имеется возможность учитывать вложенные клоны при рекурсивном обходе дерева при рефакторинге текущей группы.

В итоге удалось добиться уменьшения размера Lua-кода на 5-7%. Была также проведена выборочная ручная проверка сгенерированного кода на корректность.

Итак, все поставленные задачи были выполнены; в настоящее время осуществляется внедрение разработанного решения.

ЛИТЕРАТУРА

1. Terser. [Электронный ресурс] Режим доступа: <https://github.com/terser/terser>
2. UglifyJS. [Электронный ресурс] Режим доступа: <https://github.com/mishoo/UglifyJS>
3. Kingston S., I Pun V. K., Stolz V. Automated clone elimination in Python tests // International Symposium on Leveraging Applications of Formal Methods. – Cham : Springer Nature Switzerland, 2024. – С. 97-114.
4. Pizzolotto D., Inoue K. Blanker: a refactor-oriented cloned source codenormalizer // 2020 IEEE 14th International Workshop on Software Clones(IWSC). – IEEE, 2020. – С. 22-25.
5. Kim S. et al. Vuddy: A scalable approach for vulnerable code clone discovery // 2017 IEEE symposium on security and privacy (SP). – IEEE, 2017. – С. 595-614.

УДК 32.973

Е.Д. Доманцевич (4 курс бакалавриата),
Е.А. Муравьев, к.т.н., доцент

ОБРАБОТКА ОШИБОК В КОМПИЛЯТОРЕ С КОНТРОЛИРУЕМОГО ЕСТЕСТВЕННОГО ЯЗЫКА СПЕЦИФИКАЦИИ ТРЕБОВАНИЙ

В докладе обсуждается блок обработки ошибок, разработанный Доманцевичем Е.Д. в составе компилятора с контролируемого естественного языка CNL_Lab, разработанного руководителем - Муравьевым Е.А. Грамматика языка получена ограничением грамматики русского языка. Язык предназначен для формальной спецификации требований при бизнес моделировании предметных областей как начальных этапов жизненного цикла разработки программных систем. Предполагается также его применение в системах интеллектуального поиска в нормативных документах, таких как стандарты, справочники, разного рода руководства и др. Запрос специфицирует часть объекта хранения в базе данных, по которой выполняется выборка всех объектов, находящихся в определенном отношении к этой части. Такая операция генерации целого по части лежит в основе умозаключений, реализуемых поисковым алгоритмом.

Ниже приведены примеры спецификации фрагментов спецификаций, взятых из Российского Морского Регистра Судоходства[1] и стандарта 2.5.1 языка UML[2].

температурное условие надежной работы [системы автоматизации и [элементов и устройств] систем автоматизации] в закрытых помещениях.

температурное условие надежной работы [электронных элементов и электронных устройств] в распределительных щитах.

температурное условие надежной работы [электронных элементов и электронных устройств] в кожухах.

условие относительной влажности воздуха температуры надежной работы систем автоматизации.

вибрационные условия надежной работы систем автоматизации в отделении румпеля.

элемент это составная часть модели.

элемент владеет элементами.

комментарий это текстовая аннотация.

комментарий привязан к набору элементов.

кратность это определение инклюзивного интервала неотрицательных чисел.

нижняя граница кратности это начало интервала.

верхняя граница кратности это конец интервала.

чтобы пользователю указать допустимую мощность создания экземпляра элемента,

элемент внедряет кратность.

Компилятор реализован с применением компилятора компиляторов PLY[3], с использованием библиотеки rumporphy3[4]. Результатом компиляции предложения является фрейм, представляющий синтаксическую и семантическую структуру предложения. Структура фрейма определяется грамматикой языка CNL_Lab. Фрейм имеет **фиксированную** структуру слотов с небольшим числом контролируемых вариаций структуры фрейма. Вариации ограничены применением для определенных слотов возможности в качестве значения слота применять список однородных значений. Рассмотрим пример. Спецификация представлена предложением, приведенным ниже и выделенным курсивом.

Диспетчер формирует рейсы.

Во фрейме – результате компиляции приведенного предложения, незаполненные слоты имеют значение, представленное словом nil. Заполненные слоты групп подлежащего, сказуемого и дополнения выделены курсивом. Имена слотов выделены жирным шрифтом и представляют синтактико-семантические категории.

```
((:sentence (:aim_condition :nil)
  (:sub_pred
    (:sub
      (:prop_noun (:prop :nil) :диспетчер (:prop :nil))
      (:aggregate :nil)
      (:material :nil)
      (:loc :nil)
      (:part :nil)
      (:except :nil)
    )
    (:pred
      (:quality :nil)
      (:act (:act_VERB :формирует))
      (:obj_instr
        (:obj
          (:obj_name
            (:prop_noun (:prop :nil) :рейсы (:prop :nil))
            (:aggregate :nil)
            (:material :nil)
            (:loc :nil)
            (:part :nil)
            (:except :nil)
          )
        )
      )
      (:instr :nil)
    )
  )
))
```

В составе фрейма можно видеть такие синтаксические категории, как подлежащее (**sub**), сказуемое (**pred**), дополнение (**obj**), а также семантические категории материал (**material**), инструмент действия (**instr**), местоположение (**loc**) и др. В литературе для обозначения категорий применяются и в разное время применялись такие термины как *роли, концептуализации, семантические надежды, конститутивные отношения*, SLR – *Semantic Role Labeling*.

Фиксированная структура фрейма дает возможность алгоритмизации выравнивания вариаций нескольких термов по структуре и последующего поэлементного сравнения семантически идентичных слотов в процессе семантического поиска и выборки.

Ограничения, накладываемые грамматикой CNL_Lab, требуют достаточно высокой квалификации разработчика спецификаций. Снижение трудоемкости и сложности процесса спецификации достигается применением интеллектуального обработчика ошибок, генерирующего разработчику варианты решения возникающих проблем при разработке спецификации.

PLY предлагает мощный процессор обработки ошибок. Грамматика расширяется продукциями, содержащими токен *error*, появление которого в стеке запускает изолированный алгоритм обработки ошибок. Мощная функциональность процессора обработки ошибок PLY позволяет поднять обработку ошибок на уровень интеллектуального процесса, генерирующего рекомендации пользователю по вариантам принятия правильного решения по исправлению ошибки. Задача разработки такого интеллектуального обработчика ошибок, использующего функциональность PLY реализации известной утилиты Yacc, ставилась и решена в докладываемой работе. Подробнее этот вопрос рассмотрен в самом докладе.

ЛИТЕРАТУРА

1. 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/
2. <https://www.omg.org/spec/UML/2.5.1/PDF/changebar>
3. PLY (Python Lex-Yacc) — [ply 4.0 documentation](http://ply.readthedocs.io/en/latest/)
4. <https://github.com/no-plagiarism/pymorphy3>

УДК 004.453

В.Д. Емельянов (4 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА АНАЛИЗАТОРА КОМПОНЕНТОВ ИСХОДНОГО КОДА НА ПРИМЕРЕ WEB-СЕРВИСОВ, ИСПОЛЬЗУЮЩИХ SPRING BOOT

Компонентный анализ программного кода (SCA, от англ. Software Composition Analysis - анализ состава программного обеспечения) – важный этап в процессе разработки и сопровождения современного программного обеспечения. Данный подход направлен на детальное исследование структуры программного продукта с целью идентификации и оценки его составных частей, включая используемые библиотеки, фреймворки, компоненты и зависимости. Основная цель SCA состоит в обеспечении прозрачности и управляемости процесса разработки, повышении безопасности и надежности конечного продукта, а также минимизации юридических и технических рисков, связанных с использованием стороннего кода[1].

Цель исследования – проектирование и реализация программного решения, позволяющего автоматизировано определять участки исходного кода, заимствованные из

исходного кода, разработанного третьей стороной. Для достижения цели решены следующие задачи:

1. Исследовать подходы к компонентному анализу, а также к смежной проблеме выявления плагиата в исходном коде.
2. Спроектировать и реализовать средство автоматизированного поиска заимствований в составе программного кода.
3. Протестировать и апробировать разработанное SCA-решения.

Основа функционирования разработанного средства заключается в применении подхода, широко применяемого в смежной проблеме поиска плагиата в исходном коде[2], заключающегося в снятии «отпечатков» исходного кода с применением абстрактного синтаксического дерева и алгоритма Winnowing[3]. Данный подход обеспечивает высокую точность при малом количестве хранимой информации (сравнительно с объемом исходного кода) и показывает высокую устойчивость при изменениях, в том числе при обфускации (т.е. намеренного изменения исходного кода с целью сокрытия заимствования) и применении больших языковых моделей.

Разработанное решение разработано как Web-сервис с использованием фреймворка Spring, планируется реализация клиентских решений для упрощения интеграции с платформами CI/CD (такими как Jenkins, GitLab и т.д.) и упрощения последующего анализа сформированных отчетов. Общая архитектура разработанного решения приведена на рисунке 1. Результаты экспериментов подтверждают, что использованные методики действительно применимы в области компонентного анализа и позволяют решать проблемы устойчивости к изменениям при малом использовании вычислительных ресурсов (таких как процессорное время, постоянная и оперативная память).

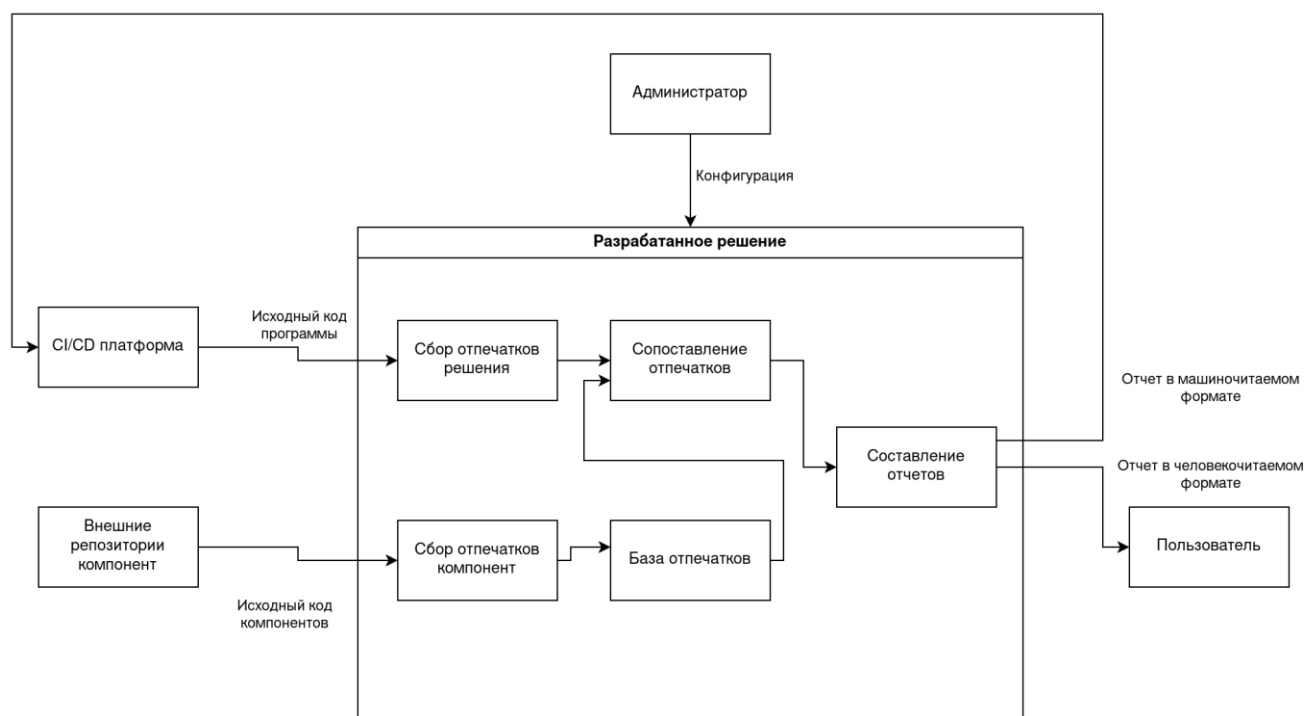


Рисунок 1 – Архитектура разработанного решения

Разработанный инструмент демонстрирует высокую эффективность в автоматизированном определении заимствованных компонент, что подтверждается экспериментами на примере web-сервисов на базе Spring boot, что имеет практическую целесообразность в процессах CI/CD[4].

ЛИТЕРАТУРА

1. The Dynamics of Software Composition Analysis / D. Foo [и др.] // CoRR. — 2019. — Т. abs/1909.00973. — arXiv: 1909.00973. — URL: <http://arxiv.org/abs/1909.00973>.
2. Dolos: Language-agnostic plagiarism detection in source code / R. Maertens [и др.] // Journal of Computer Assisted Learning. — 2022. — Т. 38, № 4. — С. 1046—1061. — DOI <https://doi.org/10.1111/jcal.12662>. — eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/jcal.12662>. — URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/jcal.12662>
3. Schleimer S., Wilkerson D. S., Aiken A. Winnowing: local algorithms for document fingerprinting // Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. — San Diego, California: Association for Computing Machinery, 2003. — С. 76—85. — (Сер.: SIGMOD '03). — DOI 10.1145/872757.872770. — URL: <https://doi.org/10.1145/872757.872770>.
4. Syed A. Best Practices for Application Security // Supply Chain Software Security: AI, IoT, and Application Security. — Springer, 2024. — С. 127—170.

УДК 004.453

И.А. Епишин (3 курс бакалавриата),
А.В. Щукин, к.т.н., доцент,
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ СРЕДСТВ ТРЕКИНГА ВЗГЛЯДА ПРИ ВЫПОЛНЕНИИ ЗАДАНИЙ ПО ПРОГРАММИРОВАНИЮ И РАЗРАБОТКА ПЛАГИНА ДЛЯ ВЫЯВЛЕНИЯ АНОМАЛЬНОГО ПОВЕДЕНИЯ

Актуальность исследования заключается в востребованности приложения для выявления списывания среди студентов при подготовке работ в удаленном режиме [1]. На наш взгляд, существующие технологии прокторинга либо слишком дороги, либо не имеют необходимого функционала. Основная мотивация разработки – усовершенствование прокторинг-систем.

Целью работы является разработка плагина для выявления аномального поведения (списывания) среди студентов при подготовке работ в удаленном режиме [2]. Для достижения этой цели необходимо решение следующих *задач*:

1. Проанализировать существующие решения, реализуемые различными прокторинг-системами.

2. Установить конкретные цели для разрабатываемого плагина.

3. Сравнить существующие языки и фреймворки для этих языков, которые реализуют необходимые функции отслеживания взгляда для наших целей.

4. Реализовать плагин для выявления аномального поведения при помощи технологий отслеживания взгляда.

5. Провести тестирования на пользователях в реальном времени.

Разрабатываемый плагин выполняет четыре основные задачи:

1. Отслеживает взгляд пользователя

2. Выявляет аномалии в поведении пользователей

3. В режиме реального времени предоставляет информацию об аномальном поведении

4. Формирует отчет о всех выявленных аномалиях в поведении пользователя

Для выполнения основных задач необходимо, чтобы выбранный фреймворк предоставлял возможность отслеживания взгляда пользователя в реальном времени, а также возвращал все основные данные о том, куда направлен взгляд пользователя, такие как координаты, направление и т.п. [3].

Разрабатываемый плагин позволяет улучшить существующие системы прокторинга дополнив их крайне важным функционалом с минимальными трудозатратами. По этой причине была выбрана разработка данного плагина [4].

Архитектура плагина представлена на рисунке 1.

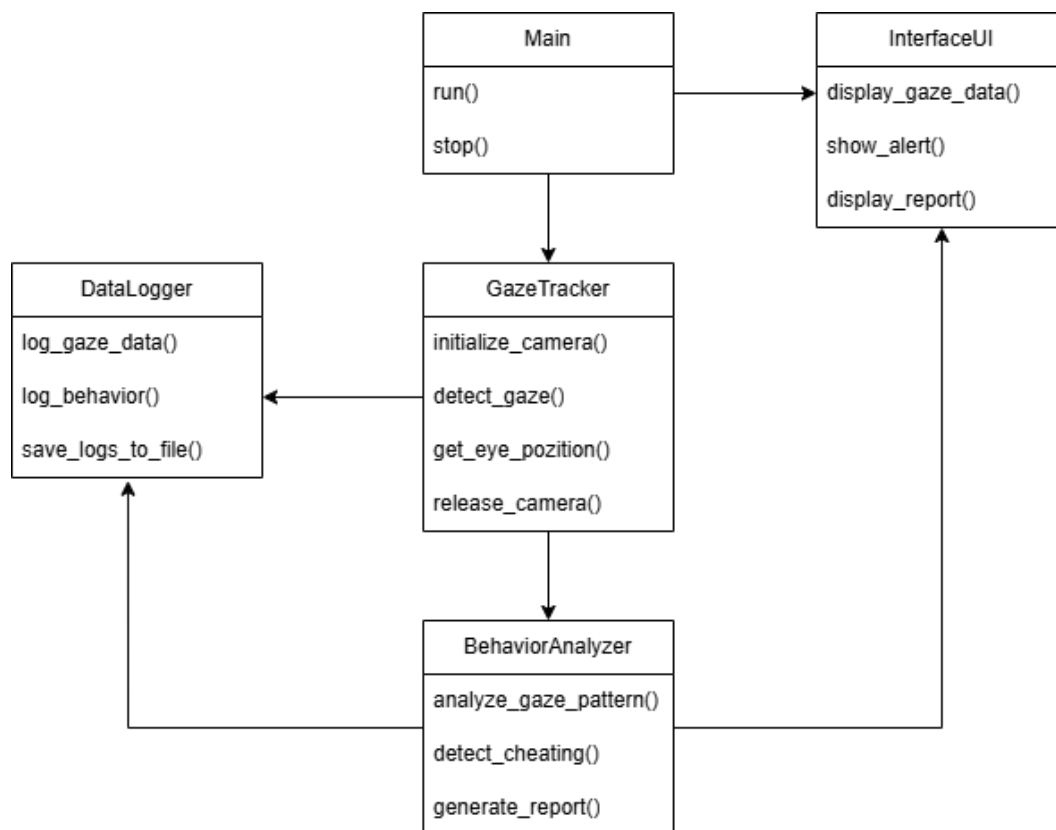


Рисунок 1 – Архитектура плагина

Подход реализован в программном средстве на языке Python. В качестве фреймворка был выбран GazeTracking [5].

Результатами экспериментов является отчёт выбранной группы пользователей о корректной работе плагина с высоким процентом выявленных аномалий поведения и низким процентом пропущенных попыток списывания.

ЛИТЕРАТУРА

1. Свиридова Н.В. Технология прокторинга в дистанционном образовании // Инновационные технологии в современном образовании – 2019 – С. 556-562.
2. Seyma Özdemir, Cemal Tosun. Investigation of Eighth-Grade Students' processes of solving skill-based science questions by eye tracking technique // Educ. Inf. Technol. – 30(2) – 2025 – P. 2237-2275.
3. Ana I. Molina, Crescencio Bravo, Jesús Gallardo, Carmen Lacave, Miguel A. Redondo. Awareness support in collaborative programming tools: An evaluation based on programmer's perception and eye tracking // J. Syst. Softw. – 112276 – 2025 – P. 220.
4. Kang-Il Park, Jack Johnson, Cole S. Peterson, Nishitha Yedla, Isaac Baysinger, Jairo Aponte, Bonita Sharif. An eye tracking study assessing source code readability rules for program comprehension. // Empir. Softw. Eng. – 29(6) – 2024 – 160.
5. Milan Segedinac, Goran Savic, Ivana Zeljkovic, Jelena Slivka, Zora Konjovic. Assessing code readability in Python programming courses using eye-tracking // Comput. Appl. Eng. Educ – 32(1) – 2024.
6. Gary Cheng, Di Zou, Haoran Xie, Fu Lee Wang. Exploring differences in self-regulated learning strategy use between high- and low-performing students in introductory programming: An analysis of eye-tracking and retrospective think-aloud data from program comprehension // Comput. Educ. – 104948 – 2024 – 208.

СРЕДСТВО СОЗДАНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ ЦЕНТРОВ ОБЛАЧНЫХ
ВЫЧИСЛЕНИЙ

С развитием цифровых технологий и увеличением объемов обрабатываемых данных облачные вычисления стали неотъемлемой частью современной информационной инфраструктуры [1]. Центры облачных вычислений обеспечивают доступ к масштабируемым ресурсам, позволяя эффективно распределять вычислительные мощности и снижать затраты на ИТ-инфраструктуру. Высокие затраты на исследования реальных центров требуют применения имитационного моделирования для анализа производительности и оптимизации работы центров облачных вычислений.

Существуют различные инструменты для моделирования облачных вычислений, однако ни один из известных распространяемых по open-source модели инструментов, не позволяет осуществлять тестирование алгоритмов прогнозирования и распределения ресурсов облачных вычислений. Они либо не обеспечивают удобства сбора необходимых метрик, либо требуют значительных вычислительных ресурсов [2]. Поэтому предлагаемое решение ориентировано на эффективный сбор ключевых показателей, критически важных для исследуемой задачи. Ключевая функциональная особенность предлагаемого решения – поддержка алгоритмов машинного обучения, позволяющих определять отношения физических и виртуальных машин согласно выбранному алгоритму распределения ресурсов. Также предлагаемое решение включает в себя генератор заявок, предназначенный для генерации псевдослучайных параметров виртуальных машин, получаемых с использованием выбранных законов распределений по каждому параметру.

Цель данной работы — разработка инструмента для имитационного моделирования работы центра облачных вычислений, адаптированного для интеграции со стендом тестирования алгоритмов прогнозирования и распределения ресурсов облачных вычислений.

Для реализации поставленной цели требуется решить следующие задачи:

1. Проектирование и разработка инструмента имитационного моделирования работы центра облачных вычислений
2. Создание системы логирования и сбора метрик.
3. Разработка генератора заявок.
4. Интеграция инструмента со стендом для тестирования алгоритмов прогнозирования и распределения ресурсов облачных вычислений.

Архитектура инструмента, представленная на рисунке 1, включает в себя 2 основных модуля. Модуль симуляции содержит в себе подмодуль статической симуляции, позволяющий осуществлять моделирование вычислительного центра с неизменной конфигурацией ресурсов и подмодуль динамического моделирования, обеспечивающий возможность изменения конфигурации ресурсов в ходе симуляции. Результаты симуляции регистрируются слушателем и записываются в базу данных временных рядов. Модуль генерации заявок, отвечает за генерацию синтетических данных для моделирования заявок (в виде виртуальных машин и облачных задач) на ресурсы вычислительного центра. Этот модуль использует модифицированный брокер CloudSimPlus для управления распределением задач между виртуальными машинами, отправляя в сервис распределения ресурсов результаты работы генератора и получая номера физических машин, в которые требуется встроить виртуальные. Архитектура разработана с использованием шаблона Model-View-Controller [3] для разделения логических частей инструмента.

Для разработки и тестирования инструмента используется следующий технологический стек. В основе проекта лежит Spring Boot 3[4] позволяющий создавать асинхронные веб-

приложения. Для моделирования облачных вычислений применяется фреймворк CloudSimPlus 8.5 [5]. Тестирование автоматизируется с помощью JUnit 5, Mockito и Hamcrest [6]. Для анализа покрытия кода тестами используется JaCoCo. Преобразование данных между моделями упрощается за счет MapStruct.

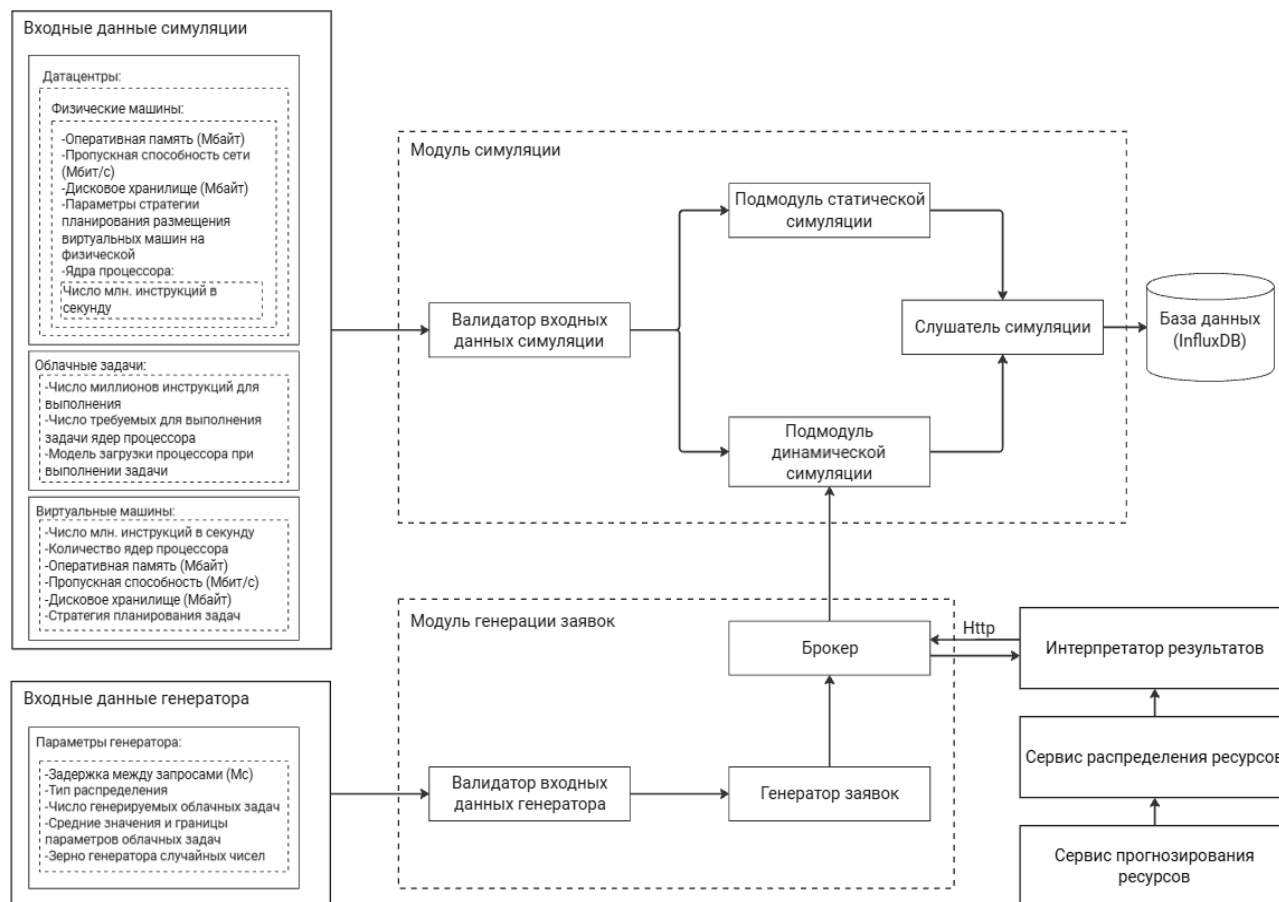


Рисунок 1 – Архитектура инструмента моделирования работы центра облачных вычислений

В результате работы был спроектирован и разработан инструмент для имитационного моделирования работы центра облачных вычислений. Также проведена интеграция инструмента со стендом для тестирования алгоритмов прогнозирования и распределения ресурсов облачных вычислений. Разработанный инструмент используется для дальнейших исследований задач оптимизации управления ресурсами в облачных средах.

ЛИТЕРАТУРА

1. R. Buyya, C.S. Yeo, S. Venugopal, J. Briberg, I. Brandic Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility - Future Generation Computer Systems 2009. - T.25:599-616
2. Uttar Pradesh Cloud computing tools: inside views and analysis - Procedia Computer Science (2020) - T.25: 382–391
3. Sabina Necula Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications - 2024.
4. S.C. Hubli, R.C. Jaiswal Efficient Backend Development with Spring Boot: A Comprehensive Overview – IJRASET 2023.
5. R. Andreoli, J. Zhao, T. Cucinotta, R. Buyya CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environments - Software: Practice and Experience 2025.
6. Dor D. Ma'ayan The quality of junit tests: an empirical study report - ACM/IEEE 1st International Workshop on Software Qualities and their Dependencies 2018

ОПТИМИЗАЦИЯ НАГРУЗКИ НА БЭКЕНД ПРИ МАССОВЫХ ИЗМЕНЕНИЯХ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ АСИНХРОННЫХ ТЕХНОЛОГИЙ И КЕШИРОВАНИЯ

Современные мобильные приложения в реальном времени обрабатывают огромные объёмы данных, что создаёт значительную нагрузку на серверные системы. Массовые операции записи могут приводить к увеличению времени отклика у пользователей, перегрузке серверов, а также блокировкам баз данных. Таким образом, для повышения производительности систем внедряются асинхронные технологии и механизмы кеширования для эффективного распределения нагрузки [1].

Одним из путей, при выборе которого можно снизить зависимость от аппаратных ресурсов и произвести распределение нагрузки, является использование асинхронного программирования и кеширования. Обработка данных асинхронной задачи в основном подразумевает использование брокеров сообщений, таких как RabbitMQ или Apache Kafka, что позволяет эффективно управлять производительностью данных, помогает избежать блокировок и продуманно распределить нагрузку между обработчиками. [2].

Целью данной работы является снижение нагрузки на бэкэнд-систему мобильного приложения за счёт внедрения кеширования на основе Redis и последующей асинхронной обработки данных с помощью брокера сообщений RabbitMQ.

RabbitMQ - популярный инструмент асинхронной обработки сообщений, применимый в системах с высокой и средней нагрузкой благодаря простоте интеграции, высокой производительности и удобной системе очередей. Apache Kafka также применим для распределённой обработки потоков данных с высокой пропускной способностью и способен к масштабированию, но требует более сложной настройки [3].

В базах данных кеширование данных активно применяется для снижения нагрузки. Redis является одним из популярных решений, обладает прекрасной пропускной способностью и легко интегрируется с другими системами, так как хранит данные в оперативной памяти, что позволяет существенно увеличить скорость обработки запросов пользователей, снизив таким образом нагрузку на источник данных [4].

На рисунке 1 представлена схема архитектуры, включающая асинхронную обработку с использованием брокеров сообщений и кеширование данных посредством Redis.

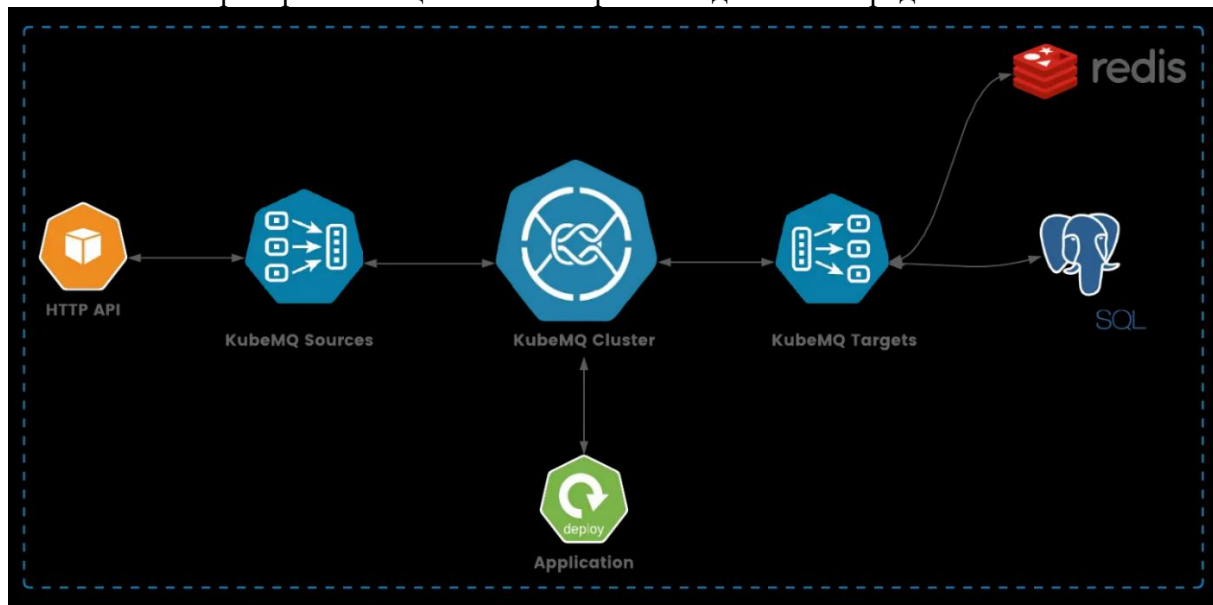


Рисунок 1 – Архитектура системы

Для достижения поставленной цели необходимо решение следующих *задач*:

1. Анализ существующих подходов к оптимизации нагрузки на бэкенд-системы при массовых изменениях данных.
2. Сравнительный анализ и выбор технологии кеширования данных, удовлетворяющей требованиям мобильного приложения.
3. Реализация механизма кеширования на основе Redis для снижения нагрузки на базу данных и повышения скорости отклика системы.
4. Оценка эффективности реализованного решения на основе кеширования.
5. Планирование и подготовка к внедрению асинхронной обработки данных с использованием брокера сообщений RabbitMQ для дальнейшего повышения производительности и масштабируемости приложения при росте нагрузки.

Таким образом, результаты исследования позволяют сделать следующие выводы:

- использование брокеров сообщений, таких как RabbitMQ или Kafka, позволяет оптимизировать процесс асинхронной обработки множества запросов и обеспечивать равномерное распределение нагрузки между серверными узлами [5,6];
- внедрение кеширования данных с помощью Redis позволяет существенно снизить нагрузку на базу данных, сокращая время отклика при выполнении массовых операций и ускоряя доступ к часто запрашиваемым данным, что особенно важно для мобильных приложений [4].

Один из предложенных подходов был реализован в рамках дипломной работы при разработке серверной части мобильного приложения, предназначенного для систематизации и отслеживания предметов. В качестве решения выбрано кеширование на основе Redis [4], что позволило снизить нагрузку на базу данных за счёт хранения промежуточных и часто запрашиваемых данных в оперативной памяти. Благодаря этому увеличилась скорость доступа к информации и улучшилось время отклика приложения. На следующих этапах разработки планируется внедрение асинхронного взаимодействия на базе RabbitMQ, что дополнительно повысит производительность и обеспечит масштабируемость при росте числа пользователей.

ЛИТЕРАТУРА

1. Асинхронная обработка в высоконагруженных системах. [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/733154/> (дата обращения: 06.03.2025).
2. Москвичева К. С., Долгачев М. В. Бумажная промышленность: Kafka против RabbitMQ. Сравнительное исследование двух отраслевых эталонных реализаций Publish/Subscribe [Электронный ресурс] Режим доступа: <https://clck.ru/3Hj3Zd> (дата обращения: 06.03.2025).
3. Обзор и сравнение брокеров сообщений Apache Kafka и RabbitMQ. [Электронный ресурс] Режим доступа: <https://www.elibrary.ru/item.asp?id=54217800> (дата обращения: 08.03.2025).
4. Improving web-oriented information systems efficiency using Redis caching mechanisms. [Электронный ресурс] Режим доступа: <https://www.elibrary.ru/item.asp?id=66052238> (дата обращения: 08.03.2025).
5. Применение Apache Kafka для работы с большими данными. [Электронный ресурс] Режим доступа: <https://www.elibrary.ru/item.asp?id=68537325> (дата обращения: 08.03.2025).
6. Тестирование производительности RabbitMQ. [Электронный ресурс] Режим доступа: <https://www.elibrary.ru/item.asp?id=48362229> (дата обращения: 06.03.2025).

РАЗРАБОТКА ИНКРЕМЕНТАЛЬНОГО ГЕНЕРАТОРА КОДА ДЛЯ ПОЛУЧЕНИЯ ИНФОРМАЦИИ О ТИПАХ В .NET ПРИЛОЖЕНИЯХ

Одной из частых операций в современных .NET приложениях [1] является поиск типов в сборке, удовлетворяющих предикату, который программист задает в своем коде. Такой подход обеспечивает хорошую расширяемость приложения, убирает избыточный код и облегчает разработку, снижая когнитивную нагрузку на программиста. Для реализации такого функционала используется стандартный механизм рефлексии .NET [2], однако он создает некоторые ограничения [3]:

1. Ошибки времени исполнения при сборке в виде нативных бинарных файлов.
2. Выделение динамической памяти.
3. Ухудшение пользовательского опыта разработчика.

Цель работы – разработать инструмент, позволяющий обойти все вышеперечисленные ограничения, обеспечивая при этом удобный и интуитивный интерфейс для программиста. Для достижения этой цели необходимо решение следующих *задач*:

1. Обзор существующих реализаций замены рефлексии в .NET.
2. Проведение сравнительного анализа найденных вариантов.
3. Предложение унифицированного инструмента оптимизации, покрывающего задачи, представленные в существующих подходах.
4. Реализация инструмента с автоматизацией тестирования и публикации в репозитории пакетов.
5. Демонстрация оптимизации на основе бенчмарков [4].

Для *реализации* используется Roslyn Compiler Platform [5]. Это достаточно новый SDK для разработчиков, позволяющий встраиваться в процесс компиляции .NET сборки. Начиная с .NET 6 (2021 год) Microsoft представила инструмент инкрементальных генераторов кода [6], который базируются на этой платформе. Генераторы на основе анализа предоставленных синтаксических деревьев могут создавать файлы с кодом, добавляя их к компиляции сборки. Таким образом процесс сбора информации о типах можно перенести из стадии времени выполнения программы на стадию времени компиляции.

Архитектура разработанного инкрементального генератора изображена на рисунке 1. Разработчик создает метод, помечает его атрибут `AotAssemblyScan` и атрибутами фильтрации, а метод возвращающий список подходящих типов. Далее хост (IDE или компилятор) вызывает код в `AssemblyScanGenerator`, который получает синтаксические деревья, фильтрует типы, используя критерии, определенные разработчиком, генерирует код и кэширует результат. Критерии фильтрации также определяются на основе атрибутов, а за их обработку отвечают наследники `ITypeCriteria`.

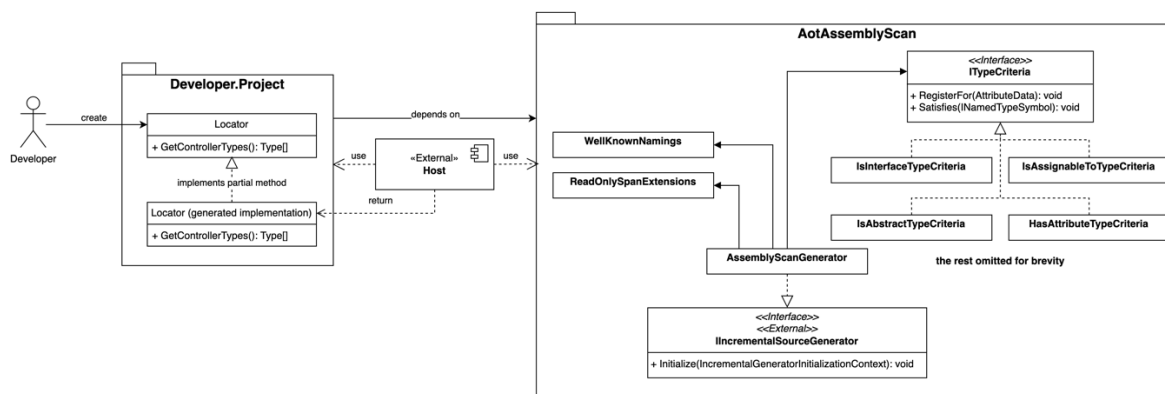
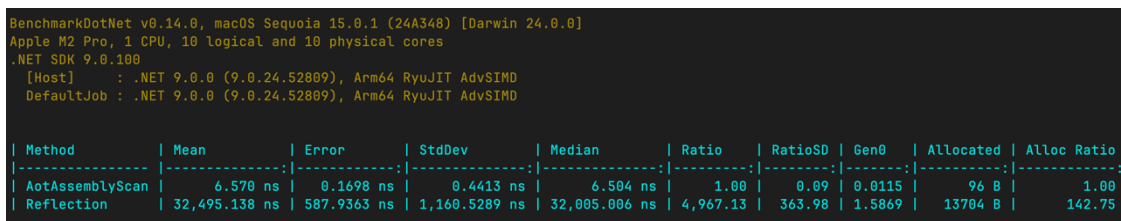


Рисунок 1 – Архитектура инструмента

Автоматическое тестирование генерируемого кода использует подход snapshot тестирования на основе библиотек xUnit и Verify.SourceGenerators. Запуск тестов происходит автоматически в GitHub при создании запроса на слияние кода.

Публикация пакета на NuGet и GitHub Container Registry происходит автоматически при создании тега с номером версии на main ветке.

Результат был проверен на реальном коммерческом приложении из 40 тысяч строк. На рисунке 2 можно увидеть прирост по времени в 4967 раз и уменьшение количества затраченной памяти в 142 раза.



Method	Mean	Error	StdDev	Median	Ratio	RatioSD	Gen0	Allocated	Alloc Ratio
AotAssemblyScan	6.570 ns	0.1698 ns	0.4413 ns	6.504 ns	1.00	0.09	0.0115	96 B	1.00
Reflection	32,495.138 ns	587.9363 ns	1,160.5289 ns	32,005.006 ns	4,967.13	363.98	1.5869	13704 B	142.75

Рисунок 2 – Сравнение производительности рефлексии и разработанной библиотеки

Таким образом, была разработана библиотека, которая в декларативном стиле позволяет находить типы в сборке, обеспечивая намного большую производительность, чем стандартный подход на основе рефлексии.

ЛИТЕРАТУРА

1. .NET. [Электронный ресурс] Режим доступа: <https://dotnet.microsoft.com/en-us/>
2. Slimák Ondřej Source Generators in C#: дис. канд. инф. наук: Brno, 2022. - 77 с.
3. Stanislav Lukeš's API for C# Code Generation: дис. канд. инф. наук: Prague, 2020. - 55 с.
4. Akinshin. A Pro .NET Benchmarking: The Art of Performance Measurement. - 1-е изд. - New York: Apress, 2019. - 687 с.
5. Roslyn Compiler Platform. [Электронный ресурс] Режим доступа: <https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/>
6. Incremental Source Generators. [Электронный ресурс] Режим доступа: <https://github.com/dotnet/roslyn/blob/main/docs/features/incremental-generators.md>

УДК 004.4'6

С.С. Карпова (4 курс бакалавриата),
О.В. Александрова, ст. преподаватель

АДАПТАЦИЯ АВТОМАТИЗИРОВАННОГО TASK-ТРЕКЕРА ДЛЯ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ

Современные компании часто сталкиваются с проблемой управления проектами и систематизации задач. Безусловно автоматизированные task-трекеры, которые закрывают основные нужды команды сильно экономят силы, энергию и время, которое можно потратить на реализацию проекта, а не на планирование и отслеживание сроков решения текущих проблем.

Redmine — веб-приложение для управления проектами с открытым исходным кодом, которое активно используется в разработке программного обеспечения [1]. Одним из главных достоинств системы является очень простой в использовании Agile плагин. Он помогает управлять проектами, разбивая их на более мелкие части, размещенные на Agile доске. Такой зрительный ход позволяет легко отслеживать выполнение задач, а также эффективнее распределять их между членами команды. Более того, task трекер совместим с такими распространенными методами, как Kanban, Scrum или их гибридные виды. Первоначально задуманный как трекер задач, Redmine может быть дополнен разнообразными плагинами, существенно расширяющими его функциональные возможности. Помимо традиционного

управления задачами, ролевого контроля доступа, форумов и вики, он легко настраивается и адаптируется к потребностям команды.

С течением времени обновления Redmine могут приводить к несовместимости существующих плагинов, что создает сложности для пользователей, зависящих от кастомного функционала [2]. Непосредственно я решаю проблему адаптации уже созданного плагина (который добавляет подзадачи в виде таблицы к каждой задаче в task-трекере), к версии Redmine.6.0.3, что позволит сохранить его, обеспечивая его корректную работу без потери функциональности и отменяет разработку нового плагина для решения тех же видов задач.

При обновлении Redmine до версии 6.0.3 возникли проблемы с совместимостью существующего плагина. Основные задачи:

1. Обновление структуры проекта и зависимостей [3]
2. Исправление ошибок, связанных с устаревшими библиотеками [4]
3. Обеспечение корректной работы плагина с новой версией Redmine

В первую очередь необходимо было исследовать файловую структуру сервера и проверить функционирование плагина. Было выполнено подключение к серверу через SSH и анализ содержимого, а также развертывание локального сервера с использованием Docker Desktop.

Для решения проблем совместимости была исправлена ошибка в чтении библиотек: устаревшее подключение файлов через `require` заменено на `require_relative` [4]

Настройка локального окружения включала развертывание сервера вручную с помощью `docker-compose up`, импорт базы данных через `phpMyAdmin` и воссоздана структура папок Redmine и плагинов на локальной машине [1].

В контроллерах добавлены `permitted_parameters` в `customer_request_controller.rb`, исправлена передача атрибутов `line_parameters`. В моделях удалены `attr_accessor`, дублирующие поля базы данных. Исправлено назначение даты и статуса: `issue.start_date`. Обновлено шаблоны, изменено расширение файлов и исправлены вызовы в `views/customer_request` [3].

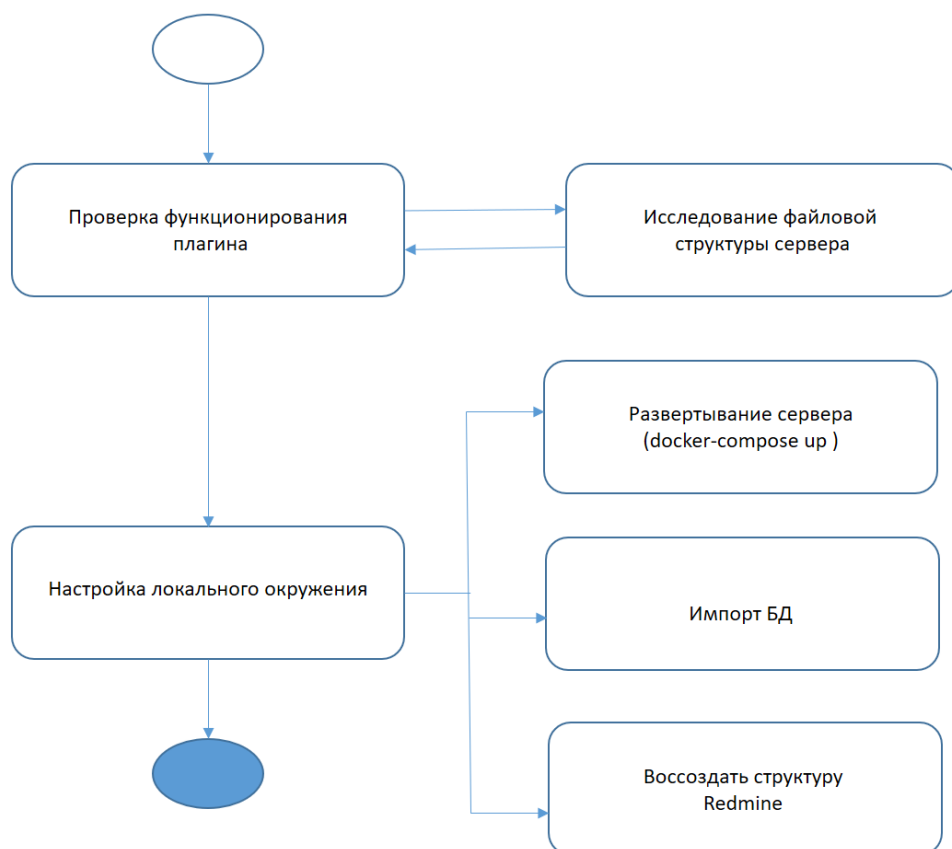


Рисунок 1 – Этапы работы по адаптации плагинов к новым версиям Redmine

В результате проделанной работы удалось адаптировать старый плагин к новой версии Redmine, устранив ошибки совместимости и обеспечив корректную работу всех функций.

ЛИТЕРАТУРА

1. Андриенко, Д. Развертывание и адаптация Redmine под нужды страховой компании / Д. Андриенко // Системный администратор. – 2013. – № 12(133). – С. 30-33. – EDN RNHFFR.
2. Ачилов, Р. Установка Redmine и интеграция его с SVN / Р. Ачилов // Системный администратор. – 2014. – № 4(137). – С. 10-14. – EDN RYZESV.
3. Федорцова, Н. А. Анализ системы управления проектами Redmine и ее роль в автоматизации производства / Н. А. Федорцова, С. А. Кадушкина // Приоритетные научные направления: от теории к практике. – 2016. – № 30. – С. 120-125. – EDN WMZHET.
4. Черемухин, А. В. Ruby On Rails - инновационная технология разработки ИНТЕРНЕТ-приложений / А. В. Черемухин, А. А. Потапкин, Л. С. Зеленко // Труды международного симпозиума "Надежность и качество". – 2009. – Т. 1. – С. 191. – EDN NDKLMP.

УДК 004.8

Р.И. Кривошапкина (4 курс бакалавриата),
Т.В. Леонтьева, к.т.н., доцент

ИНТЕГРАЦИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ПРИЛОЖЕНИЕ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ

Современные системы управления проектами, такие как Asana или Trello, предоставляют широкий функционал для организации командной работы, однако сложность их интерфейса может затруднять использование для начинающих пользователей. В условиях роста популярности и доступности удалённой работы возникает потребность в простых и интуитивных инструментах с интеллектуальными функциями.

Целью данной работы является разработка концепции веб-приложения для управления командными проектами с минималистичным дизайном и встроенным искусственным интеллектом (ИИ), поддерживающим формат свёрточного чата, который будет генерировать идеи для проектов и предоставлять рекомендации по задачам. Проект направлен на упрощение планирования и повышение эффективности командной работы.

Для достижения поставленной цели определены следующие задачи:

- Провести анализ существующих решений в области управления проектами и выявить их ключевые недостатки.
- Спроектировать минималистичный и интуитивно понятный интерфейс приложения, ориентированный на удобство пользователя.
- Разработать концепцию ИИ-компонента, который будет интегрирован в интерфейс в виде свёрточного чата для интеллектуальной поддержки.
- Оценить потенциальные преимущества предлагаемого решения по сравнению с существующими аналогами.

Концепция приложения предполагает использование доступных технологий. Для реализации интерфейса предполагается использование Vue.js – лёгкого и производительного фреймворка для создания динамичных веб-интерфейсов [1]. Интерфейс приложения (см. рисунок 1) включает боковую панель навигации с разделами, а также основную рабочую область с различными вкладками. Эти вкладки позволяют пользователям переключаться между различными форматами отображения задач: списком, канбан-доской, календарём или временной шкалой. Для стилизации интерфейса планируется применение Tailwind CSS, который можно использовать локально, что минимизирует зависимость от внешних ресурсов [2].

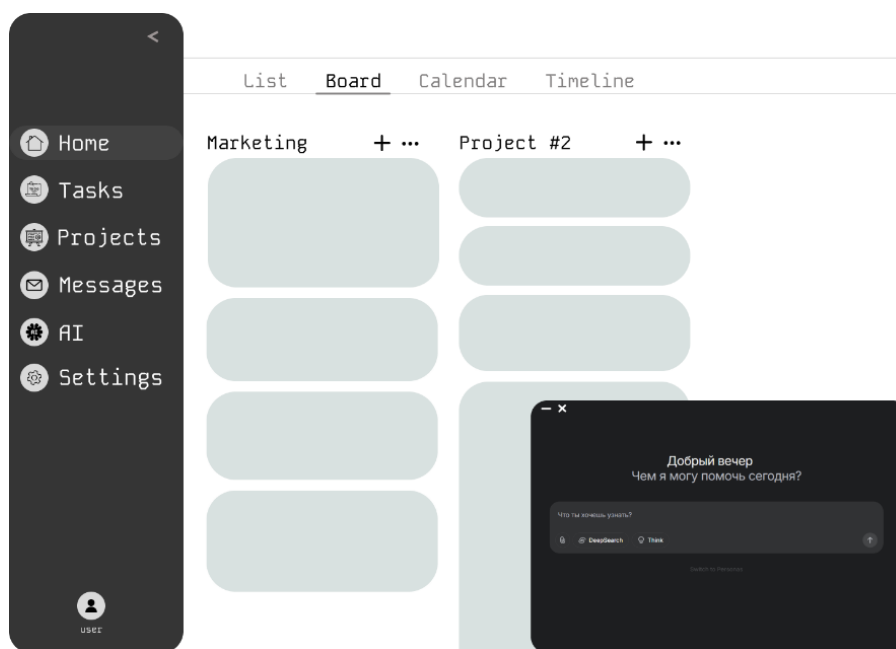


Рисунок 1 – Концепция интерфейса веб-приложения с ИИ-чатом

Для серверной части предполагается использование Python с фреймворком Flask, который отличается простотой настройки и гибкостью. Flask будет отвечать за обработку API-запросов между фронтендом и ИИ-компонентом, а также за взаимодействие с базой данных [3]. Для хранения данных о задачах, проектах и пользователях планируется использование PostgreSQL — открытой и широко применяемой СУБД, доступной через российские облачные провайдеры, такие как Yandex Cloud или VK Cloud. Это обеспечивает надёжность и масштабируемость системы, а также соответствие требованиям локализации данных.

ИИ-компонент для свёрточного чата предполагается реализовать с использованием отечественной модели обработки естественного языка, например, RuGPT от SBER, что обеспечит генерацию идей и рекомендаций на русском языке [4]. Интеграция ИИ с бэкендом будет выполнена через FastAPI, обеспечивающий высокую производительность API [5].

Проект направлен на упрощение процессов планирования, снижение порога вхождения для пользователей и повышение общей эффективности командной работы за счёт применения современных технологий. На этапе реализации стек технологий будет уточнён с учётом возможных ограничений доступа.

В заключение, разработанная концепция демонстрирует возможность создания удобного инструмента управления проектами с использованием ИИ. Перспективы включают интеграцию анализа производительности команды и адаптацию ИИ под специфические задачи.

ЛИТЕРАТУРА

1. Vue.js [Электронный ресурс] Режим доступа: <https://ru.vuejs.org/guide/introduction.html>
2. Берьянов М.С., Салахов И.Р., Иванов М.Д. ИССЛЕДОВАНИЕ СОВРЕМЕННОГО СТЕКА REACT РАЗРАБОТКИ В 2022 ГОДУ// Научный сетевой журнал «Столыпинский вестник» №8/2022.
3. Волеводз Д. А. ВОСТРЕБОВАННЫЕ BACK-END-ТЕХНОЛОГИИ ДЛЯ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА // Международный научный журнал «Инновационная наука» ISSN 2410-6070 №6-1 / 2024
4. RuGPT [Электронный ресурс] Режим доступа: <https://rugpt.io>
5. FastAPI [Электронный ресурс] Режим доступа: <https://fastapi.tiangolo.com/>

РЕКОМПИЛЯЦИЯ С ПОМОЩЬЮ GHIDRA

Обратная разработка программного обеспечения включает в себя решение задачи декомпиляции. Задача декомпиляции заключается в следующем: имея исполняемый файл, скомпилированный из кода, написанного на каком-либо высокоуровневом языке, произвести программу на высокоуровневом языке, которая выполняет те же действия, что и изначальный исполняемый файл [1].

Существует ряд решений для обратной разработки программного обеспечения, таких как IDA и Binary Ninja. Среди самых популярных из них открытый исходный код имеет Ghidra — это фреймворк для обратной разработки программного обеспечения, предоставляющий возможность декомпиляции бинарных файлов и экспорта кода [2].

Большинство технологических решений с возможностью декомпиляции, в том числе Ghidra, не гарантирует отсутствие синтаксических и семантических ошибок в получаемом исходном коде, поскольку создатели видят разрабатываемые декомпиляторы как инструменты, предназначенные для анализа полученного кода, а не его компиляции и запуска. По тем же причинам разработчики Ghidra не ведут работу по направлению устранения возникающих ошибок в декомпилированном коде. В том числе, некоторые некорректные синтаксические конструкции вставляются декомпилятором специально, чтобы пользователю было проще понять смысл конструкции. Рекомпиляция — это разновидность декомпиляции, результатом которой является код на языке высокого уровня, который может быть скомпилирован и запущен. Технологии рекомпиляции могут способствовать портированию приложений с одной архитектуры на другую, а также анализу поведения декомпилированного кода (можно его менять, компилировать и запускать заново). Сообщество заинтересовано в рекомпиляции с помощью Ghidra, ведутся соответствующие обсуждения (<https://github.com/NationalSecurityAgency/ghidra/issues/236>). Таким образом, в работе рассматривается проблема рекомпиляции с помощью Ghidra.

Целью работы является приближение кода, декомпилируемого с помощью Ghidra, к рекомпилируемому — такому, что его можно скомпилировать и запустить без ошибок.

Для достижения этой цели были поставлены следующие задачи:

1. Обзор существующих решений в области декомпиляции.
2. Изучение декомпиляции с помощью Ghidra на примерах и выбор затруднений для последующего исправления.
3. Предложение подхода для устранения ошибок из декомпилированного кода.
4. Реализация данного подхода.
5. Сбор сравнительной статистики количества ошибок компиляции в зависимости от использования предложенного подхода на наборе исполняемых модулей.

В ходе изучения декомпиляции с помощью Ghidra на примерах был обнаружен ряд различных затруднений.

Сначала был выбран подход, заключающийся в постобработке декомпилированного кода. Был написан скрипт на Python, который работает с Ghidra по её API с помощью библиотеки `ruhidra` [3], передаёт ей бинарный файл и обрабатывает полученный декомпилированный код.

Позднее было решено продолжать исследование в формате изменения кодовой базы самого проекта Ghidra. Основным результатом на данный момент является наш форк (<https://github.com/VyacheslavIurevich/recompilation-ghidra>), в котором мы вносим изменения в модули, отвечающие за декомпиляцию в Ghidra.

В декомпилированном коде встречаются не только функции, написанные пользователем, но и служебные функции — те, которые транслятор добавляет для своих целей. Например, это функции библиотеки C Runtime [4] или PLT-трамплины [5] для подключения разделяемых

библиотек. Было решено выявлять такие функции по шаблонным инструкциям или названию и удалять их из декомпилированного кода. Данное затруднение было разрешено в постобработчике и в кодовой базе Ghidra, однако эвристики для обнаружения служебных функций требуют существенной доработки.

Анализатор типов данных сопоставляет типы данных в декомпилированном коде с типами данных из архивов типов данных. Но в экспортированном коде на языке C заголовочные файлы, содержащие используемые типы данных, не подключаются. При экспорте можно выбрать опцию имитации типов данных, но используемые функции из заголовочных файлов не будут выгружены. Данное затруднение было исправлено в постобработчике и в кодовой базе проекта, причём в ходе работы над изменениями в Ghidra было выявлено и исправлено несколько недочётов в её реализации. В том числе, была исправлена проблема, описанная в Issue официального репозитория (<https://github.com/NationalSecurityAgency/ghidra/issues/4740>).

В постобработчике был исправлен ряд других затруднений, на данный момент ведётся работа по переносу данных исправлений в код самого приложения Ghidra. В декомпилированном коде можно встретить нигде не определённые нестандартные типы, использование служебных функций декомпилятора, некорректные обращения к переменным и многие другие затруднения, мешающие рекомпиляции.

Был подготовлен набор бинарных файлов, состоящий из пользовательских программ `hello_world` (вывод сообщения в `stdout`) и `bmp` (чтение заголовка BMP) и бенчмарков `linpack` и `dhrystone`, скомпилированных с помощью `gcc-13` под архитектуру `x86_64`, а также системных утилит Linux `echo`, `pwd` и `cat`, взятых из каталога `/bin` операционной системы Ubuntu 24.10. К пользовательским программам не применялась утилита `strip`. При экспорте декомпилированного кода были выбраны стандартные опции выгрузки, кроме вариации выбора опции «Emit Data-type Definitions». В таблице 1 приведена статистика количества ошибок компиляции после декомпиляции перечисленных бинарных файлов с помощью разных инструментов. Можно сделать вывод, что мы добились рекомпиляции достаточно простых примеров, и существенно приблизили к рекомпилируемости более сложные.

Таблица 1 – Статистика количества ошибок компиляции после декомпиляции бинарных файлов

Бинарный файл	Ghidra без изменений	Ghidra с внесёнными изменениями без имитации типов данных	Ghidra с внесёнными изменениями и имитацией типов данных	Постобработчик
<code>hello_world</code>	7	1	0	0
<code>bmp</code>	38	9	2	0
<code>linpack</code>	15	57	6	0
<code>dry2</code>	44	32	38	7
<code>echo</code>	172	64	71	12
<code>pwd</code>	219	93	156	27
<code>cat</code>	256	82	143	33

ЛИТЕРАТУРА

1. Cifuentes Cristina, Gough K. John. Decompilation of binary programs // Software: Practice and Experience. — 1995. — . — Vol. 25, no. 7. — P. 811–829. — URL: <https://doi.org/10.1002/spe.4380250706>.
2. NSA. Описание Ghidra. [Электронный ресурс] — Режим доступа: <https://github.com/NationalSecurityAgency/ghidra/blob/master/README.md> (дата обращения: 12 марта 2025 г.)
3. DoD Cyber Crime Center (DC3). Описание библиотеки `pyhidra`. [Электронный ресурс] — Режим доступа: <https://github.com/dod-cyber-crime-center/pyhidra> (дата обращения: 12 марта 2025 г.)

4. Horgan Patrick. Debugging. [Электронный ресурс] — Режим доступа: <http://dbp-consulting.com/tutorials/debugging/linuxProgramStartup.html> (дата обращения: 12 марта 2025 г.).
5. Drepper Ulrich. How To Write Shared Libraries. — 2005. — URL: <https://www.cs.dartmouth.edu/sergey/cs258/ABI/UlrichDrepper-How-To-Write-Shared-Libraries.pdf>

УДК 004.896

А.А. Киселева (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент

ПОДХОД АВТОМАТИЗИРОВАННОГО РАЗВЕРТЫВАНИЯ ОБЛАЧНЫХ СЕРВИСОВ «Р7-ОФИС» В ИНФОРМАЦИОННО-ТЕХНОЛОГИЧЕСКОЙ ИНФРАСТРУКТУРЕ ПРЕДПРИЯТИЯ С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА ANSIBLE

В крупных индустриальных компаниях широко используются офисные приложения [1]. Важной для проектных команд является возможность не только локального редактирования файлов, но и совместной работы с ним, поскольку это помогает экономить время на пересылку документов и синхронизацию между ними [2].

Существует ряд продуктов, предоставляющий услуги совместного редактирования: Microsoft Office, OpenOffice, Google Workspace, Яндекс 360 и др. Все аналоги обладают рядом преимуществ и недостатков [3]. Немаловажно и то, какое количество времени, средств и усилий необходимо приложить для внедрения того или иного офисного пакета, так как это влияет на сроки и стоимость конечного продукта. Следовательно, актуален вопрос и выбора способа развертывания пакета [5-6].

Целью работы является снижение стоимости и трудоемкости развертывания программно-аппаратного комплекса предоставления услуг многопользовательского доступа корпоративной платформы совместной работы за счет создания автоматизированной информационной инфраструктуры и развертывания на ней отечественного офиса.

Для выбора офисного пакета и способа автоматизации были проведены исследования, в ходе которых различные сервисы сравнивались между собой по определенным заранее критериям. В результате офисным пакетом для внедрения был выбран «Р7-офис» [7], а инструментом автоматизации приложение Ansible [8].

Концепция предложенного в работе подхода представлена на рисунке 1. Он состоит из следующих основных этапов.

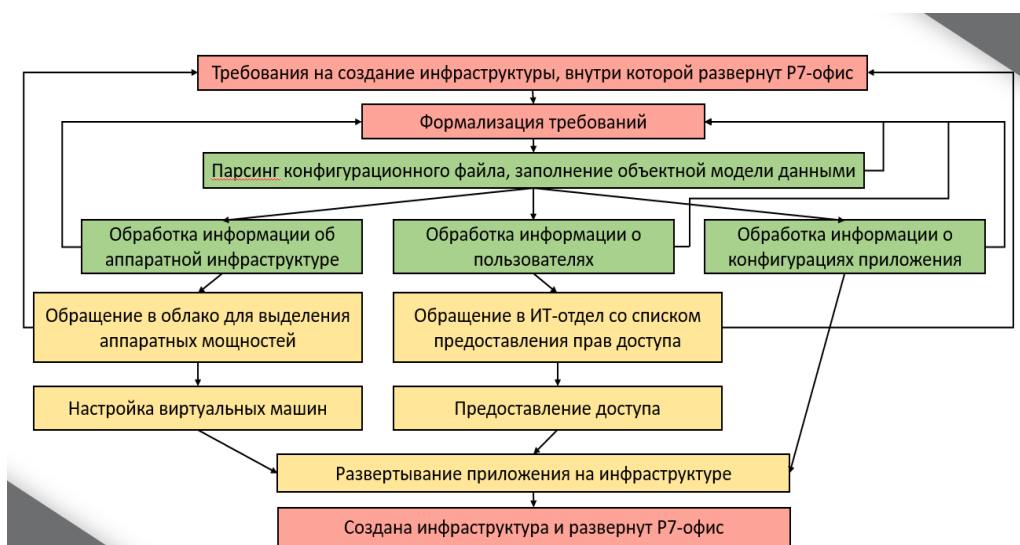


Рисунок 1 – Схема подготовки ИТ-инфраструктуры и развертывания ПО

1. Сбор требований к программному обеспечению, инфраструктуре для развертывания, а также списка пользователей с указанием ролей для предоставления доступов;

2. Формализация требований в файле .csv в установленном формате с использованием инструкции;

3. Использование скриптов python для парсинга файла с требованиями и получения конфигурационных файлов для настройки:

1) uml-файлы для настройки аппаратных частей;

2) bash-скрипт для развертывания приложения;

3) данные для создания заявок на выделение мощностей и выдачу прав пользования системой пользователям.

4. Подготовка кластера и применение Ansible для конфигурирования серверов, настройки системы и развертывания приложения.

Предложенный подход реализован в программном средстве, для которого была предложена архитектура информационной системы для развертывания решения. Программное средство состоит из следующих элементов:

– сформированный файл .csv, который содержит требования к инфраструктуре и список пользователей системы с ролями для предоставления доступа;

– программное средство, разработанное на языке программирования python, которое нацелено на автоматический разбор требований и формирование выходных файлов для дальнейшей работы с ними;

– OTRS-система [9], настроенная с целью обработки заявок на выдачу прав доступа к офисному пакету сотрудникам и выделения мощностей;

– настроенный Ansible-кластера для непосредственного конфигурирования серверов в соответствии с требованиями.

На данный момент работа проходит проверку с помощью экспериментального метода. Гипотеза заключается в том, что автоматизированный подход позволит сократить временные затраты внедрения решения на 45%, что является эффективным для компаний.

ЛИТЕРАТУРА

1. Горкавченко А.А., Залукаева Е.А., Попеня И.В., Терехов Я.А. Возможности онлайн офисов // Формирование и развитие новой парадигмы науки в условиях постиндустриального общества. Сборник статей по итогам Международной научно-практической конференции. Стерлитамак. 2023. С. 89-93.
2. Кирвас В. А. Современные облачные информационные технологии совместной работы // Экспертные оценки элементов учебного процесса: программа и материалы XIX межвузовской научно-практической конференции. Харьков. Т. 25. С. 35-43.
3. Никоненко А. Н. Методы и критерии выбора поставщиков // WORLD SCIENCE: PROBLEMS AND INNOVATIONS. 2016. С. 60-62.
4. В. А. Ивлев, И. В. Никифоров Актуальность автоматизированной настройки инфраструктуры ит-проекта // Современные технологии в теории и практике программирования. Сборник материалов конференции. Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого". 2022. С. 79-80. EDN QQAVXT.
5. В. А. Ивлев, Г. В. Мироненков, И. В. Никифоров, А. Д. Ковалев Генерация информационно-технологической инфраструктуры проекта на основе неформализованных требований // 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. 242-244. EDN CENYIP.
6. V. A. Ivlev, I. V. Nikiforov, O. A. Yusupova Automation method for configuring IT infrastructure for IT projects // International Conference on Digital Transformation: Informatics, Economics, and Education (DTIEE2023): Proceedings of SPIE. The International Society for Optical Engineering.

Fergana. Uzbekistan. Washington: SPIE-SOC PHOTO-OPTICAL INSTRUMENTATION ENGINEERS. 2023. P. 126370D. DOI 10.1117/12.2680779. EDN BYZHИK.

7. Заславский, А. А. P7 офис как отечественная альтернатива пакету Microsoft Office // Интерактивное образование. 2022. № 2. С. 25-27. EDN QRDBKM.
8. Hochstein L., Moser R. Ansible: Up and Running: Automating configuration management and deployment the easy way // O'Reilly Media, Inc. 2017.
9. Яремчук С. Система управления запросами OTRS // Системный администратор. 2007. №. 9. С. 18-22.

УДК 004.942

Н.А. Кубов (2 курс магистратуры),
А.М. Сабуткевич (2 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент

ПРИМЕНЕНИЕ МЕТОДА ГОРИЗОНТАЛЬНОГО МАСШТАБИРОВАНИЯ ДЛЯ ИСПОЛНЕНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ В ОБЛАЧНОЙ СРЕДЕ

Современные задачи моделирования сложных систем требуют высокой вычислительной мощности и гибкости инфраструктуры [1, 2]. Имитационные модели, широко используемые в различных областях — от инженерных расчетов до анализа социально-экономических процессов, — зачастую сталкиваются с ограничениями традиционных подходов к выполнению расчетов [3, 4]. Рост сложности моделей и объемов обрабатываемых данных делает актуальным поиск эффективных методов масштабирования исполнения имитационных моделей.

Горизонтальное масштабирование исполнения задач является важным инструментом для повышения производительности и адаптивности вычислений в облачной среде [5]. Методы горизонтального масштабирования в контексте имитационного моделирования позволяют обрабатывать большие объемы данных и поддерживать высокую производительность при работе с высоконагруженными процессами, которые невозможно эффективно реализовать на одном сервере; обеспечивать масштабируемость ресурсов, что важно для адаптивного выделения мощностей в зависимости от текущих нужд для запуска исполнения имитационной модели, делая процесс моделирования экономически эффективным; повышать надежность и отказоустойчивость, поскольку распределение нагрузки между узлами минимизирует риски отказов, обеспечивая устойчивость процессов симуляции.

Целью работы является повышение производительности платформ имитационного моделирования за счет применения метода горизонтального масштабирования исполнения имитационных моделей в облачной среде с использованием средств контейнеризации

Для достижения этой цели необходимо решение задач, описанных ниже:

1. Провести исследование техник и методов поддержки горизонтального масштабирования в облачной среде.
2. Провести исследование и представить сравнительный анализ существующих платформ имитационного моделирования.
3. Предложить метод горизонтального масштабирования исполнения имитационных моделей в облачной среде.
4. Реализовать предложенный метод в программном средстве с использованием средств контейнеризации.
5. Оценить эффективность применения разработанного метода и программного средства в ходе экспериментальных исследований для сферы облачных вычислений.

После анализа теоретических подходов для разработки методов горизонтального масштабирования [6] было принято решение разработать метод горизонтального

масштабирования исполнения имитационных моделей в облачной среде с использованием применения статических правил на основе пороговых значений.

На рисунке 1 представлена общая схема метода. На первом шаге сервер управления экспериментом получает запрос от пользователя на исполнения имитационной модели. После чего в системе создаётся сущность «Эксперимент». Далее из базы данных читаются данные о имитационной модели и конфигурации эксперимента. На втором этапе происходит генерация исполняемой задачи (имитационной модели, готовой для исполнения) в виде jar-архива. После чего исполняемая задача загружается в облачное хранилище и на основе конфигурации эксперимента происходит запрос в облачное окружение на создание контейнеров с сервисом симуляции внутри. На третьем этапе создается запрос на исполнение задачи. После получения запроса на исполнение симуляции сервисы симуляции загружают jar-архив из облачного хранилища. Классы внутри jar-архива загружаются в ClassLoader. На четвертом этапе создается сессия, инициализируется контекст исполнения, и задача прикрепляется к обработчику. После чего происходит инициализация исполнения симуляции, в процессе которой сервис симуляции использует программный интерфейс исполняемой задачи для генерации событий.

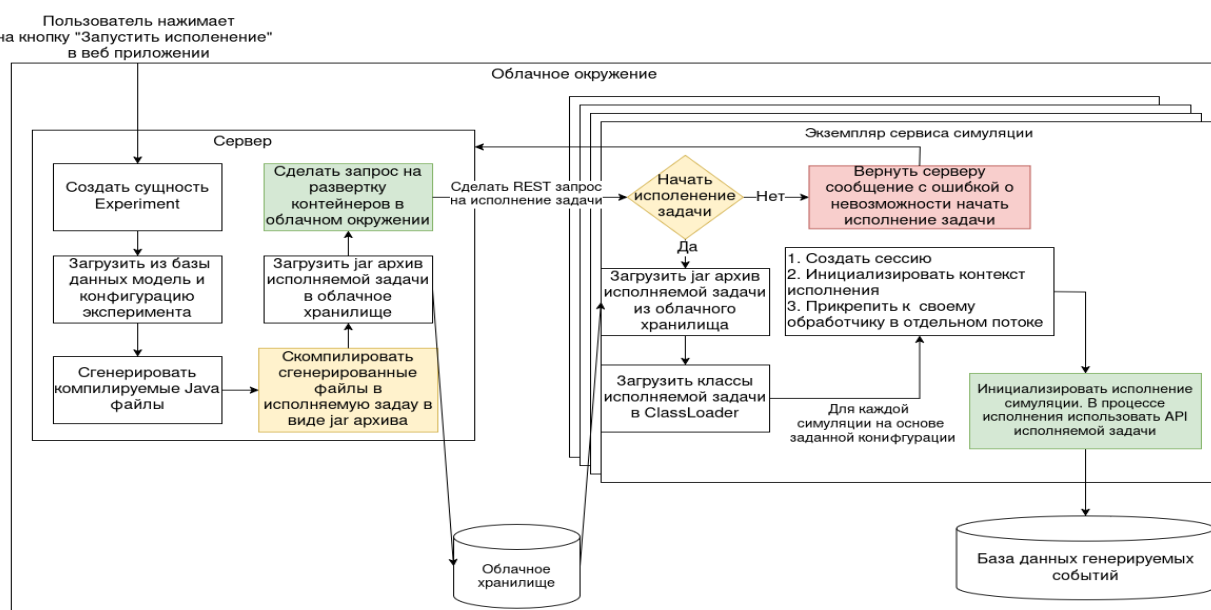


Рисунок 1 – схема предлагаемого метода горизонтального масштабирования исполнения имитационных моделей в облачной среде.

Предложенный метод был применен на практике в производственной компании. Метод был реализован в программном средстве в виде прототипа платформы имитационного моделирования в облачной среде с использованием средств контейнеризации. Для оценки эффективности метода было проведено экспериментальное исследование, в ходе которого проводилось сравнение результатов исполнения имитационных моделей, генерирующих разное количество событий в версии программной системы до внедрения метода и после внедрения метода. Экспериментальное исследование показало, что время исполнения моделей с использованием метода сократилось в среднем до 20%.

Таким образом, на основе полученных данных можно утверждать, что предлагаемый метод является эффективным для обеспечения горизонтального масштабирования исполнения имитационных моделей в облачной среде.

ЛИТЕРАТУРА

1. Исследование систем автоматизации настройки инфраструктуры ИТ-проекта / В. А. Ивлев, Ф. Ю. Чемашкин, И. В. Никифоров, А. Д. Ковалев // 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. 204–205.

2. Денисенко Б. А., Тянутов М. В., Кубов Н. А., Никифоров И.В Анализ открытых источников данных по нагрузкам на ЦОД // 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. 191–193.
3. Имитационное моделирование поведения сложных многоагентных систем с использованием вероятностной модели / А. М. Сабуткевич, Д. А. Вихляев, И. В. Никифоров, А. В. Самочадин // 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. 98–100.
4. Решение проблемы комбинаторного взрыва при генерации траекторий поведения агентов в процессе имитационного моделирования / А. М. Сабуткевич, Д. А. Вихляев, И. В. Никифоров, А. В. Самочадин // 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. 213–215.
5. Ермаков Н. В., Таленфельд Т. С., Никифоров И. В., Егоров П. Б. Исследование подходов к созданию высокопроизводительного масштабируемого облачного сервиса для дедупликации данных в хранилище // Информатика и кибернетика (ComCon-2016): сборник докладов студенческой научной конференции Института компьютерных наук и технологий, Санкт-Петербург, 04–09 апреля 2016 года, 2016. – С. 144–147.
6. Alharthi, S., Alshamsi, A., Alseiari, A., & Alwarafy, A. (2024). Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions. *Sensors*, 24(17), 5551. <https://doi.org/10.3390/s24175551>

УДК 004.453

А.В. Ларин (2 курс магистратуры),
В.В. Амосов, к.т.н., доцент

СБОРЩИК БАЗЫ ЗНАНИЙ И ИИ-АССИСТЕНТ ПО ВОПРОСАМ ПОЛЬЗОВАТЕЛЕЙ НА ОСНОВЕ WEB3

Современные системы ИИ-ассистентов, такие как IQ GPT и Cookbook.dev, сталкиваются с проблемами ограниченности источников данных, их актуальности и качества предоставляемых ответов. В условиях стремительного развития Web3-технологий и децентрализованных систем возникает необходимость в создании масштабируемой и адаптивной базы знаний, способной интегрировать данные из блокчейн-среды и традиционных источников. Цель работы – разработать сборщик базы знаний и ИИ-ассистент, обеспечивающий высокое качество ответов за счет использования современных облачных и Web3-технологий.

Проект представляет собой программный продукт, состоящий из двух основных компонентов:

1. Сборщик базы знаний, осуществляющий сбор данных через парсинг и скрепинг [1] открытых интернет-ресурсов, включая блокчейн-источники. Данные сохраняются в векторном хранилище OpenAI для последующего использования.

2. ИИ-ассистент, который использует агентов OpenAI для генерации ответов на запросы пользователей. Интеграция с мессенджерами Discord, Telegram и Slack обеспечивает удобство взаимодействия.

Архитектура системы реализована с применением фреймворка Serverless [2] на основе микросервисной модели [3], включающей:

- сервис парсинга и скрепинга данных;
- сервис интеграции ИИ-ассистента с мессенджерами;
- контейнеризация компонентов с использованием AWS ECS для обеспечения масштабируемости и отказоустойчивости [4].

Разработан прототип сборщика базы знаний на языке TypeScript [5], способного интегрировать данные из Web3-источников и традиционных ресурсов. Создан ИИ-ассистент, демонстрирующий высокую точность и скорость ответов на запросы пользователей. Реализована масштабируемая архитектура системы, обеспечивающая устойчивость и адаптивность к изменениям в среде Web3.

Предложенный подход демонстрирует возможность эффективного использования Web3-технологий для создания современных ИИ-ассистентов. Разработанная система обладает потенциалом для дальнейшего развития и применения в различных областях, требующих обработки больших объемов данных и предоставления актуальной информации.

ЛИТЕРАТУРА

1. M. Khder, "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application," International Journal of Advances in Soft Computing and its Applications, vol. 13, no. 3, pp. 145–168, Dec. 2021, doi: 10.15849/IJASCA.211128.11.
2. M. S. Aslanpour, A. N. Toosi, C. Cicconetti, S. Dustdar, "Serverless Edge Computing: Vision and Challenges," in Proceedings of the Australasian Symposium on Parallel and Distributed Computing (AusPDC 2021), Feb. 2021, Australia. doi: 10.1145/3437378.3444367.
3. Y. Gan and C. Delimitrou, "The Architectural Implications of Cloud Microservices," IEEE Computer Architecture Letters, vol. PP, no. 99, pp. 1–1, May 2018. doi: 10.1109/LCA.2018.2839189.
4. C. Pahl, "Containerisation and the PaaS Cloud," IEEE Cloud Computing, vol. 2, no. 3, pp. 24–31, Jun. 2015, doi: 10.1109/MCC.2015.51.
5. TypeScript: JavaScript With Syntax For Types. [Электронный ресурс] Режим доступа <https://www.typescriptlang.org/>

УДК 004.415.53

М.С. Лихачев (3 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ СРЕДСТВ PROPERTY-BASED ТЕСТИРОВАНИЯ НА JAVASCRIPT

Сегодня тестирование играет важную роль в обеспечении качества кода. Модульные тесты [4], будучи распространённым методом проверки, позволяют контролировать заранее известные сценарии, но их эффективность напрямую зависит от мастерства разработчика и конкретных входных данных. Хотя при наличии достаточного опыта можно продумать типичные случаи сбоев, на самом деле нецелесообразно рассматривать все возможные исходы, не проверив код строка за строкой.

Для решения этих проблем существует property-based тестирование (далее – PBT). Основная идея PBT заключается в генерации случайных входных данных и проверке того, выполняются ли заданные свойства при работе с этими входными данными [2]. Если этот процесс обнаруживает контрпример - входное значение, которое приводит к сбою, - разработчик уведомляется [5]. Свойство - это высокоуровневая спецификация ожидаемого поведения тестируемой функции, которое должно выполняться всегда [1]. Например, в отсортированном массиве каждый предыдущий член должен быть меньше либо равен последующему. Для проверки этого свойства, как правило, напишут три unit-теста, в то время как для PBT требуется всего лишь один [3].

Для языка программирования JavaScript существуют две основных библиотеки, реализующих принципы PBT – fast-check [7] и JSVerify [6], которые сравниваются в рамках данной работы.

Таким образом, целью данной работы является экспериментальное сравнение средств property-based тестирования на JavaScript на примере библиотек fast-check и JSVerify.

Для достижения этой цели необходимо решение следующих задач:

1. Изучить принципы PBT и его преимущества над классическими методами тестирования, в частности unit-тестами.
2. Рассмотреть в отдельности возможности каждой из библиотек fast-check и JSVerify.
3. Провести экспериментальное сравнение данных библиотек.

В ходе проведения сравнительного анализа библиотек fast-check и JSVerify выявлены следующие схожие характеристики: открытый исходный код, каждая из них является бесплатной и имеет открытую лицензию. Различия зафиксированы в таблице 1, где столбцы 3-6 сформированы авторами в рамках проведения экспериментов.

Таблица 1 – Сравнительная таблица библиотек fast-check и JSVerify.

Фреймворк	Язык реализации	Удобство	Время и память	Поддержка	Особенности поиска ошибок
fast-check	TypeScript	Современный API, подробная документация	Потребляет меньше памяти. Время приблизительно одинаковое.	Активная поддержка	Из-за расширенного функционала находит больше специальных случаев
JSVerify	JavaScript	API схож с fast-check, но документация менее подробная	Потребляет больше памяти. Время приблизительно одинаковое.	Перестал поддерживаться с конца 2018 года.	Не проверяет специальные случаи, которые находит fast-check

Предварительное сравнение проводилось с помощью запуска различных тестовых сценариев, проверяющих разные виды свойств с использованием входных данных разных типов. Для этого был создан отдельный класс для проверки потребления памяти, который принимает на вход два файла, содержащих тесты, написанные с использованием fast-check и JSVerify соответственно. Фреймворком для запуска тестов был выбран Mocha [8], который сам предоставляет информацию о времени выполнения тестов.

Опишем подробнее один из примеров.

Для проверки скорости работы и потребления памяти были написаны тесты на такие свойства, как обратимость операций с JSON, обратимости массива и строки, неизменности или уменьшения длины при фильтрации массива, а также некоторые другие. В результате на этих примерах время выполнения тестов оказалось примерно одинаковым, но потребление памяти у fast-check меньше.

Также при проверке такого свойства, что квадратный корень из числа всегда должен возвращать модуль числа, fast-check проверяет наличие NaN чисел, в то время как JSVerify этого не делает из-за отсутствия функционала, проверяющего наличие особого значения «не число».

Полученные в результате экспериментального сравнения результаты позволяют сделать вывод о том, что fast-check на данный момент является ведущей библиотекой для property-based тестирования на JavaScript.

ЛИТЕРАТУРА

1. Learning and statistical model checking of system response times / B. K. Aichernig, P. Bauerstätter, E. Jöbstl [et al.] // Software Quality Journal. – 2019. – Vol. 27, No. 2. – P. 757-795.

2. Karlsson, S. QuickREST: Property-based Test Generation of OpenAPI-Described RESTful APIs / S. Karlsson, A. Causevic, D. Sundmark // Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation, ICST 2020 : 13, Porto, 23–27 марта 2020 года. – Porto, 2020. – P. 131-141.
3. Our first Property-Based Test // fast-check Режим доступа: <https://fast-check.dev/docs/tutorials/quick-start/our-first-property-based-test/>
4. Кокотов, Д. В. Правила разработки эффективных модульных тестов / Д. В. Кокотов // XXI век: итоги прошлого и проблемы настоящего плюс. – 2023. – Т. 12, № 2(62). – С. 48-53.
5. Harrison Goldstein, Joseph W. Cutler, Daniel Dickstein, Benjamin C. Pierce, and Andrew Head. Property-Based Testing in Practice // In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24). – 2024. – Article 187, P. 1–13.
6. JSVerify. [Электронный ресурс] Режим доступа: <https://jsverify.github.io/>.
7. fast-check. [Электронный ресурс] Режим доступа: <https://fast-check.dev/>.

УДК 004

Я.П. Мазин (4 курс бакалавриата),
Е.А. Павлов, ст. преподаватель

ИСПОЛЬЗОВАНИЕ АРХИТЕКТУРЫ ТРАНСФОРМЕРОВ ДЛЯ ПРОГНОЗИРОВАНИЯ ПОКАЗАТЕЛЯ ПРЕЛОМЛЕНИЯ ХАЛЬКОГЕНИДНЫХ СТЁКОЛ

Халькогенидные стёкла – это стёкла, содержащие один или несколько халькогенных элементов, а именно серу (S), селен (Se), или теллур (Te), в сочетании с другими элементами из групп IV, V и VI периодической таблицы, за исключением кислорода.

Благодаря своей структуре халькогенидные стёкла обладают рядом уникальных физических и оптических свойств. Высокий показатель преломления позволяет использовать халькогенидные стёкла в лазерах, плоской оптике и фотонных интегральных схемах.

Однако, по сравнению с оксидными стёклами, свойства халькогенидных стёкол и связь этих свойств с композицией элементов химического состава стекла изучены плохо. Одним из таких свойств является показатель преломления. Из-за сложного химического состава халькогенидных стёкол, их аномальных структурных особенностей и недостатка экспериментальных данных, для большинства стёкол аналитическое вычисление показателя преломления не представляется возможным. Поэтому главным методом разработки новых стёкол является метод проб и ошибок.

Трансформер – архитектура нейронной сети, разработанная для задач обработки естественного языка. Основана на механизме внимания, который позволяет ей выявлять и учитывать сложные взаимосвязи между элементами входных данных [2]. Эта особенность трансформеров может быть использована для выявления связи между составами стёкол и их показателем преломления.

FT-Transformer – трансформер, который преобразует табличные данные в векторные представления, а затем обрабатывает их с использованием механизма внимания. Его архитектура представлена на рисунке 1.

Целью данной работы является создание и обучение нейронной сети, основанной на архитектуре трансформеров, для предсказания показателя преломления халькогенидных стёкол по их составу. Это позволит ускорить процесс разработки стёкол с заданными оптическими характеристиками.

Для достижения данной цели необходимо:

1. Проанализировать имеющиеся данные по составу халькогенидных стёкол и их показателям преломления.
2. Выполнить предобработку данных и составить датасет.
3. Предложить способы решения задачи, основанные на разных архитектурах трансформеров, и способы подачи данных на вход модели.

4. Провести обучение моделей и их валидацию.
5. Оценить качество работы моделей, сравнить их между собой и с существующими решениями, основанными на других алгоритмах.

В результате анализа выяснилось, что данные о показателе преломления доступны для 670 стёкол. Показатель преломления изменяется в пределах от 1,94 до 4,48. При этом для 75% стёкол этот показатель меньше 2.75. Это означает, что датасет разреженный и у модели могут возникнуть трудности с предсказанием больших значений показателя преломления.

Исследования [1][3] и анализ показали, что высокое содержание теллура (Te) и мышьяка (As) в стёклах увеличивает показатель преломления. Чтобы учесть это, в данные добавлены столбцы:

- HighAs = 1, если содержание As > 25%, иначе 0.
- HighTe = 1, если содержание Te > 25%, иначе 0.
- HighAsTe = 1, если оба столбца HighAs и HighTe равны 1.

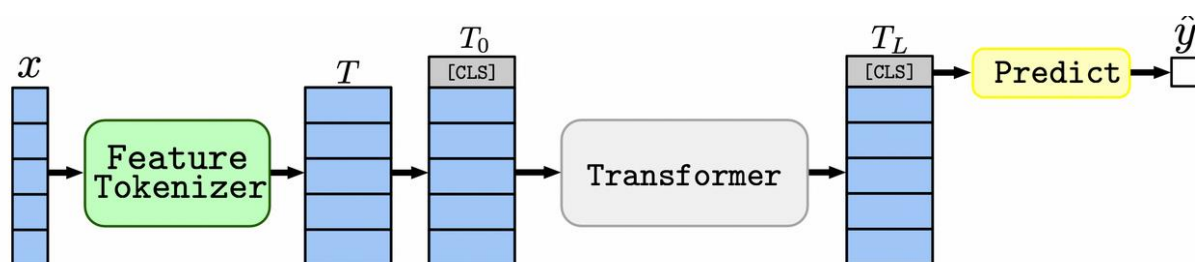


Рисунок 1 – Архитектура FT-Transformer

Для решения задачи применялась архитектура FT-Transformer. При обучении использовался алгоритм кросс-валидации KFold, для которого датасет был разделён на 5 частей. Для автоматического подбора гиперпараметров использовался фреймворк optuna, специально предназначенный для оптимизации параметров машинного обучения.

В итоге удалось получить модель, показывающую на тестовой выборке результаты, представленные в таблице 1. Данные результаты превосходят большую часть результатов, показанных моделями из статей [1][4].

Таблица 1 – Метрики качества для модели FT-Transformer

RMSE (Root Mean Squared Error)	0,11± 0,03
R ² (Coefficient of Determination)	0,89± 0,02

В дальнейшем планируется провести обучение трансформера, работающего с текстовыми данными. Для этого необходимо преобразовать табличные данные в предложения и провести аугментацию данных, используя разные формулировки. В этом может помочь одна из доступных больших языковых моделей (LLM).

ЛИТЕРАТУРА

1. Mastelini S. M., Cassar D. R., Alcobaca E., Botari T., de Carvalho A. C. P. L. F., Zanotto E. D. Machine learning unveils composition-property relationships in chalcogenide glasses // *Acta Materialia*. – 2022. – Т. 240. – С. 118302. – DOI: 10.1016/j.actamat.2022.118302.
2. Chen M., Liu B., Liu Y., Li T. Self-supervised Learning for Glass Property Screening // *arXiv preprint*. – 2023. – arXiv:2410.24083v1. – URL: <https://arxiv.org/abs/2410.24083>.
3. Cassar D. R., Mastelini S. M., Botari T., Alcobaca E., de Carvalho A. C. P. L. F., Zanotto E. D. Predicting and interpreting oxide glass properties by machine learning using large datasets // *Ceramics International*. – 2021. – Т. 47. – С. 23958–23972. – DOI: 10.1016/j.ceramint.2021.05.105.
4. Sayam Singla, Sajid Mannan, Mohd Zaki, Virok Sharma Accelerated design of chalcogenide glasses through interpretable machine learning for composition–property relationships // *Journal of Physics: Materials*. – 2023. – Т. 6. – С. 025003. – URL: <https://colab.ws/articles/10.1088%2F2515-7639%2Facc6f2>.

АВТОМАТИЧЕСКОЕ ВЫЯВЛЕНИЕ НЕНАВИСТНИЧЕСКИХ ВЫСКАЗЫВАНИЙ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

В настоящее время общение достигло беспрецедентного уровня активности благодаря онлайн-социальным платформам, которые преодолели географические и языковые барьеры. Однако, этот переход сопровождается распространением ненавистнических высказываний, которые негативно влияют на социальную среду этих платформ. В области обработки естественного языка ведутся исследования по разработке моделей для выявления и классификации ненавистнических высказываний, направленные на улучшение безопасности и качества онлайн-среды.

Ряд работ, таких как [1,2,3], и технологии выявления ненавистнических высказываний в Интернете, такие как Perspective, HateSonar и Moderate Hatespeech, являются подходами к решению проблемы распространения ненавистнических высказываний. Однако эти работы и технологии имеют ряд недостатков, таких как: разбалансировка по классам в наборах данных; погрешность в работе по отношению к одной группе людей (например, Moderate Hatespeech по отношению к темнокожим); несовместимость с новой грамматической динамикой; этические проблемы и вопросы конфиденциальности; вычислительные и латентные затраты; проблема «черного ящика», приводящая к отсутствию прозрачности.

Большинство недостатков вышеупомянутых подходов связано с типом наборов данных, которые являются оригинальными. Одним из способов преодоления этих недостатков является использование синтетических данных, которые генерируют более глобальный, сбалансированный спектр сценариев и ненавистнических комментариев. Синтетические данные содержат как исходные данные, так и возмущения, такие как ирония или сарказм, которые кардинально меняют смысл предложения.

Таким образом, целью данной работы является автоматическое выявление ненавистнических высказываний путем сравнительного анализа эффективности архитектур нейронных сетей на основе синтетического набора данных.

Для достижения этой цели необходимо решение следующих задач:

- Обзор существующих подходов к автоматическому выявлению ненавистнических высказываний
- Предварительная обработка используемого синтетического набора данных и векторное представление данных
- Обучение и тестирование ряд архитектур нейронных сетей
- Оценка характеристик эффективности наилучшей модели
- Создание простого API, интегрирующего разработанную модель и его упаковка в контейнере.

Синтетический набор данных составлен на английском языке и содержит более 40 000 высказываний. Предварительная обработка выполняется с помощью библиотеки NLTK на языке Python и включает такие операции как удаление стоп-слов и пунктуации, интерпретацию смайликов, токенизацию и лемматизацию. Векторное представление текстовых данных осуществлялось с помощью алгоритмов FastText и GloVe. На рисунке 1 изображена архитектура решения.

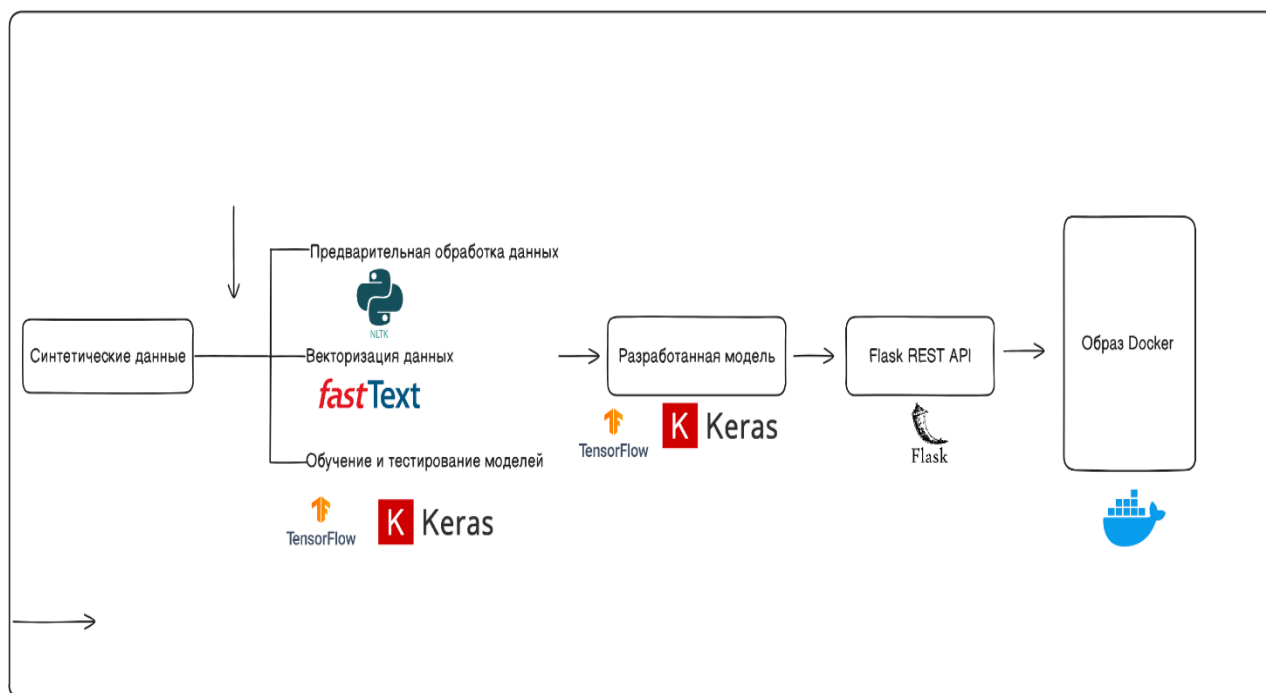


Рисунок 1 – Архитектура решения

В ходе обучения были использованы и реализованы с помощью библиотеки TensorFlow следующие архитектуры: модель CNN [4] - за качество извлечения признаков, и рекуррентные модели LSTM [5] и GRU [5] - за их внутренние механизмы обработки последовательных данных. Наилучшая модель стала двунаправленной версией модели GRU с механизмом внимания. Процессы обучения и тестирования моделей были реализованы в среде Google Colab Pro.

Разработанная модель показывает следующие характеристики и метрики эффективности: размер не более 12 мб, на тестовых наборах 72.68% ассурасу, 72.65% точности, 72.19% recall и 72.29% F1-оценка. Модель была интегрирована в API, разработанный на Flask [6], а затем упакована с помощью Docker [7].

ЛИТЕРАТУРА

1. El Koshiry A., Eliwa E., El-Hafeez T., Omar A., Arabic Toxic Tweet Classification: Leveraging the AraBERT Model, Big Data and Cognitive Computing. 2023, Vol 170, no. 4. <https://doi.org/10.3390/bdcc7040170>
2. Сметанин С.И. Автоматическая классификация токсичных комментариев на русском языке // Computational linguistics and intellectual technologies. 2020. № 19. Т. 26. С. 1149–1159. <http://dx.doi.org/10.28995/2075-7182-2020-19-1149-1159>
3. Ribeiro A., Silva N., INF-HatEval at SemEval-2019 Task 5: Convolutional Neural Networks for Hate Speech Detection Against Women and Immigrants on Twitter, In Proceedings of the 13th International Workshop on Semantic Evaluation. 6–7 June 2019, pp. 420–425.
4. Soni S., Chouhan S.S., Rathore S.S., TextConvoNet: a convolutional neural network based architecture for text classification, Appl Intell. 2023, Vol 53, pp. 14249–14268. <https://doi.org/10.1007/s10489-022-04221-9>
5. Будыльскийкий Д. В. GRU и LSTM: современные рекуррентные нейронные сети // Молодой ученый. 2015. № 15. Т. 95. С. 51–54.
6. Flask. [Электронный ресурс] Режим доступа: <https://flask.palletsprojects.com/en/stable/>
7. Docker. [Электронный ресурс] Режим доступа: <https://www.docker.com/>

РАЗРАБОТКА СИСТЕМЫ ИНТЕЛЛЕКТУАЛЬНОГО УПРАВЛЕНИЯ И БАЛАНСИРОВКИ
НАГРУЗКИ ДЛЯ API-СЕРВИСОВ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

В современных условиях организации, использующие API-сервисы искусственного интеллекта (ИИ), сталкиваются с отсутствием эффективных инструментов для централизованного управления и контроля использования различных ИИ-сервисов. Существующие решения не предоставляют возможности гибкого распределения нагрузки между API-ключами, что приводит к превышению лимитов запросов и неоптимальному использованию ресурсов. Кроме того, в условиях потенциальных геополитических ограничений доступа к международным ИИ-сервисам [1], организации нуждаются в надежном решении, способном обеспечить бесперебойный доступ к этим сервисам.

Целью данной работы является разработка программного обеспечения промежуточного уровня (middleware), выступающего в роли интеллектуального прокси-сервера для взаимодействия с различными провайдерами ИИ-сервисов. Для достижения этой цели необходимо решить следующие задачи:

1. Разработать систему управления API-ключами с возможностью балансировки нагрузки.
2. Реализовать многоуровневую систему ограничений для контроля расходов.
3. Внедрить систему мониторинга и логирования всех операций.
4. Обеспечить безопасность и отказоустойчивость системы.

Разрабатываемое программное обеспечение ориентировано преимущественно на средние и крупные компании, активно внедряющие технологии искусственного интеллекта в свои бизнес-процессы, а также на разработчиков программного обеспечения, интегрирующих ИИ-сервисы в свои продукты. Особый интерес решение представляет для компаний, находящихся в регионах с потенциальными ограничениями доступа к международным ИИ-сервисам.

В ходе исследования были проанализированы существующие решения для управления доступом к API искусственного интеллекта: OpenAI API [2], ProxyAPI [3] и LiteLLM [4]. Сравнительный анализ показал, что предлагаемое решение обладает рядом преимуществ перед аналогами (таблица 1).

Таблица 1 – Сравнение характеристик аналогов реализации

Характеристика	Наша реализация	OpenAI API	ProxyAPI	LiteLLM
Самостоятельный хостинг	ДА	НЕТ	НЕТ	ДА
Наценка за использование	НЕТ	НЕТ	ДА	НЕТ
Распределение нагрузки между ключами	ДА	НЕТ	НЕТ	НЕТ
Лимиты по токенам/денежным средствам	ДА	ДА	НЕТ	ДА
Логирование запросов	ДА	НЕТ	НЕТ	ДА
Настраиваемая модерация логов	ДА	НЕТ	НЕТ	НЕТ

Для реализации данного программного обеспечения разработаны следующие ключевые компоненты:

1. Система управления API-ключами на базе реляционной базы данных PostgreSQL. Все чувствительные данные хранятся в зашифрованном виде с использованием алгоритма AES-

256, а доступ к операциям с ключами строго контролируется через систему ролей и разрешений.

2. Балансировщик нагрузки, реализованный как интеллектуальная система распределения запросов между доступными API-ключами, использующая комбинированный подход на основе алгоритмов Round-robin и Weighted round-robin. Это позволяет динамически адаптировать распределение нагрузки в зависимости от текущей ситуации и установленных ограничений.

3. Система контроля и ограничений на основе высокопроизводительного механизма in-методу счетчиков. Данный подход позволяет эффективно отслеживать и контролировать множество параметров, включая денежные лимиты, количество использованных токенов, частоту запросов и активность с различных IP-адресов.

4. Система мониторинга и логирования, реализованная на базе локального механизма записи логов. Автоматическая ротация логов предотвращает чрезмерное использование дискового пространства и облегчает последующий анализ данных.

5. Административный интерфейс, представленный в двух взаимодополняющих форматах: веб-интерфейс на базе React и программный REST API интерфейс. Оба интерфейса используют единую систему аутентификации [5] на основе токенов и поддерживают детальное разграничение прав доступа.

6. Система автоматической обработки и модерации запросов, реализованная как многоуровневый механизм фильтрации и анализа контента с использованием регулярных выражений, локальных LLM-моделей и специализированных API модерации [6].

Основной стек технологий включает Python 3.11+, FastAPI, PostgreSQL и Docker для контейнеризации.

В ходе исследования была разработана концепция программного обеспечения, предоставляющего уникальное решение для централизованного управления доступом к различным провайдерам ИИ-сервисов. Анализ существующих аналогов показал, что предлагаемое решение обладает рядом преимуществ, включая возможность самостоятельного хостинга, отсутствие дополнительных наценок и наличие расширенных функций безопасности и мониторинга. Реализация данного проекта позволит организациям оптимизировать использование ИИ-сервисов, повысить безопасность и эффективность работы с API, а также обеспечить стабильный доступ к сервисам в условиях возможных региональных ограничений.

ЛИТЕРАТУРА

1. WANG S.H. OPENAI - EXPLAIN WHY SOME COUNTRIES ARE EXCLUDED FROM CHATGPT – 2023 – ISSN: 0028-0836e ISSN: 1476-4687
2. OpenAI API Документация [Электронный ресурс] Режим доступа: <https://platform.openai.com/docs/> – (дата обращения: 24.01.2025)
3. ПрохуAPI Документация [Электронный ресурс] Режим доступа: <https://proхuapi.ru/docs> – (дата обращения: 24.01.2025)
4. LiteLLM Документация [Электронный ресурс] Режим доступа: <https://docs.litellm.ai/docs/> – (дата обращения: 24.01.2025)
5. АНИКИН Д.А. АНАЛИЗ МЕТОДОВ АВТОРИЗАЦИИ И АУТЕНТИФИКАЦИИ REST API – 2023 – С. 122 – eISSN: 2500-1752
6. ДЕНИСОВ Д.А. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ КАК ИНСТРУМЕНТ МОДЕРАЦИИ КОНТЕНТА – 2024 – С. 19-22

РАЗРАБОТКА И ПРИМЕНЕНИЕ РАСПРЕДЕЛЁННОЙ МОДЕЛИ ВЫЧИСЛЕНИЙ ДЛЯ
МОДЕЛИРОВАНИЯ СИСТЕМЫ ДИПОЛЕЙ

Современные информационные технологии предоставляют большие возможности для моделирования сложных систем и решения широкого спектра задач. Распределённые вычисления стали ключевым инструментом для решения задач, требующих высокой производительности и масштабируемости. Данная тенденция не обходит стороной и область высокопроизводительных математических вычислений. В научно-ориентированных компаниях, где регулярно проводятся вычислительные эксперименты с заданными параметрами, уже используются специализированные распределённые системы управления заданиями. Однако такие системы часто разрабатываются под конкретные нужды и имеют ограниченную универсальность, что стимулирует создание новых, более гибких решений.

Цель данной работы – разработать масштабируемое программное решение, которое позволит моделировать коллективные эффекты в двухмерной системе диполей с запаздыванием, задействуя основные преимущества распределённых вычислительных узлов. Исследование данных коллективных эффектов представляет большой интерес для науки, в частности в области оптической микроскопии. В отличие от предыдущих исследований [1-3], фокусирувавшихся на локальных прототипах, данная работа предлагает масштабируемое решение, адаптированное для распределённых вычислений в условиях стохастической неопределённости.

Для достижения поставленной цели необходимо решить следующий набор задач:

1. Анализ аналогичных решений в области распределённых математических вычислений.
2. Сравнительный анализ методов распределения вычислительной нагрузки между узлами кластера с учётом специфики вычисляемой задачи.
3. Проектирование масштабируемой архитектуры для решения поставленной задачи.
4. Реализация предложенной архитектуры с использованием доступных средств.
5. Проведение ряда измерительных мероприятий для демонстрации эффективности предлагаемой архитектуры.

Исходная модель представляет из себя систему из N линейных обыкновенных дифференциальных уравнений второго порядка с запаздыванием, которые задают соотношения между дипольными моментами рассеивателей [3]. Сами координаты рассеивателей является случайными величинами, подчиняющимся двумерному гауссову распределению. Для данной системы удалось получить аналитическое представление для квазистационарного решения. Для описания коллективных характеристик данной системы была введена функция $I_n(\theta, \varphi)$ – мощность рассеянного света в единицу телесного угла. В рамках исследования представляла интерес зависимость между усреднённой по ансамблю функции $\bar{I}_n$ от числа рассеивателей.

Ввиду высокой вычислительной сложности расчётов и большого числа итераций в рамках одной серии экспериментов интерес представляет возможность распараллеливания и распределения вычислений на несколько устройств. Вычисление целевой функции входит в класс методов Монте-Карло. В работах [4-6] была продемонстрирована эффективность распараллеливания методов данной группы. В свою очередь метод распределения нагрузки, предложенный в работе [7], может оказаться эффективным при фрагментации выполняемых задач т. к. сама архитектура может быть построена на базе локальной сети персональных компьютеров. Существует множество разных способов организовать взаимодействие между узлами в вычислительном кластере. Парадигма издатель/подписчик является одним из способов организации подобных взаимодействий [8].

На рисунке 1 представлена схема взаимодействия компонентов при отправке серии экспериментов на исполнение.

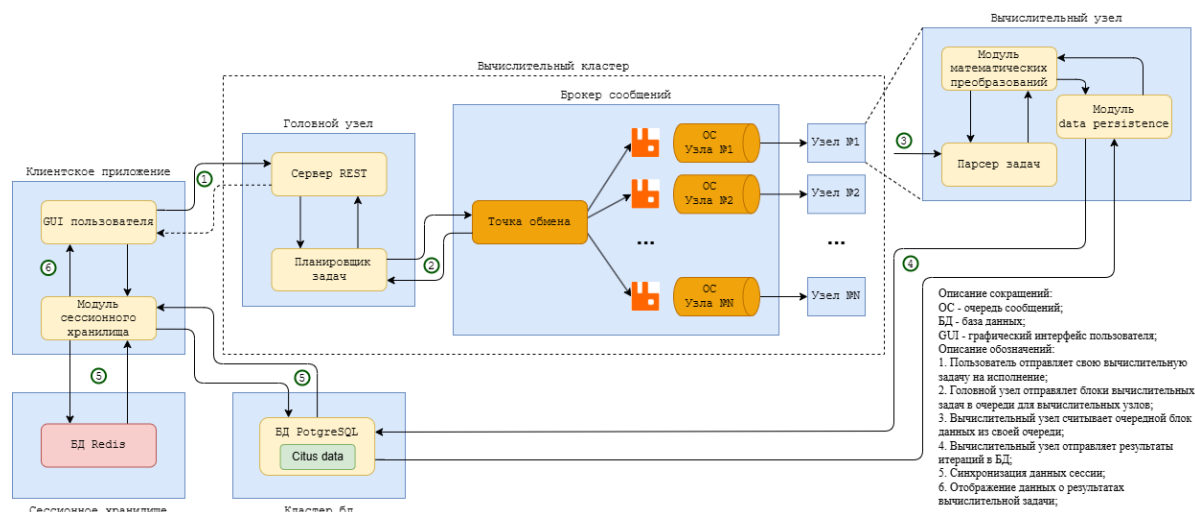


Рисунок 1 – Схема запуска серии экспериментов

Можно выделить следующие преимущества данной системы:

1. Возможность динамической конфигурации составляющих кластера.
2. Возможность возобновления вычислений с некоторой “контрольной точки”.
3. Линейный коэффициент горизонтального масштабирования в общем случае.
4. Возможность реализации более сложных планировщиков в рамках головного узла для решения своих задач.

Основные элементы данной архитектуры были реализованы на языке программирования C++ с использованием среды программирования Clion 2024.2.2. Для кэширования данных сессии использовалась БД Redis 8.0, а для хранения данных вычислительных экспериментов и логов применялась БД PostgreSQL 17.0 с расширением citus data. В качестве брокера сообщений был использован RabbitMQ 4.0.

ЛИТЕРАТУРА

1. Шалгуева С.Л., Воскобойников С.П., Попов Е.Н. Разработка программы для моделирования взаимодействия электромагнитных рассеивателей. В сборнике: Современные технологии в теории и практике программирования. Сборник материалов научно-практической конференции. Санкт-Петербург, 2021. С. 181-182.
2. Мейник А.В., Воскобойников С.П., Попов Е.Н. Разработка программы для моделирования взаимодействия цепочки диполей. В сборнике: Современные технологии в теории и практике программирования. Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых. Санкт-Петербург, 2023. С. 155-157.
3. Мейник А.В., Воскобойников С.П., Попов Е.Н. Разработка программы для моделирования взаимодействия двухмерного ансамбля диполей на плоскости. В сборнике: Современные технологии в теории и практике программирования. Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых. Санкт-Петербург, 2024. С. 64-65.
4. Марченко М.А. Комплекс программ MONC для распределенных вычислений методом Монте-Карло. // Сибирский журнал вычислительной математики. – 2004. –Т. 7. – С.43-55.
5. Camarasu-Pop S. et al. Monte Carlo simulation on heterogeneous distributed systems: A computing framework with parallel merging and checkpointing strategies //Future Generation Computer Systems. – 2013. – Т. 29. – №. 3. – С. 728-738.
6. Xiong L. et al. Distributed Least-Squares Monte Carlo for American Option Pricing //Risks. – 2023. – Т. 11. – №. 8. – С. 145.

7. Катусева Я.В. Балансировка загрузки несимметричного вычислительного комплекса при решении задачи статистического оценивания // Информатика и системы управления. - 2006. - № 2(12). - С. 88-93.
8. Pham V. N. et al. Efficient solution for large-scale IoT applications with proactive edge-cloud publish/subscribe brokers clustering //Sensors. – 2021. – Т. 21. – №. 24. – С. 8232.

УДК 004.42

М.А. Мосолов (4 курс бакалавриата),
Г.В. Смородников, Р.А. Золотарев (2 курс магистратуры),
А.В. Самочадин, к.т.н., доцент

СРЕДСТВА КРАТКОСРОЧНОГО ПРОГНОЗИРОВАНИЯ ДЛЯ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

Традиционные алгоритмы распределения ресурсов для облачных вычислений базируются на текущем состоянии системы, что ограничивает их способность адаптироваться к изменяющимся со временем нагрузкам[1]. В данной работе предлагается подход к динамическому распределению ресурсов, который использует прогнозирование нагрузки. В отличие от статических и реактивных методов, предложенная схема позволяет учитывать будущие изменения в загрузке системы, обеспечивая более сбалансированное и предсказуемое распределение задач.

Целью данной работы является разработка средств краткосрочного прогнозирования нагрузки на облачные вычисления для улучшения динамического распределения ресурсов.

Для достижения данной цели необходимо решить следующие задачи: провести анализ временных рядов нагрузки на облачные серверы, исследовать существующие алгоритмы прогнозирования, разработать модель прогнозирования нагрузки на основе машинного обучения[2], реализовать модуль предсказания в виде микросервиса, провести тестирование и оценку предложенной модели тестовых данных.

Схема предложенного подхода представлена на рисунке 1.

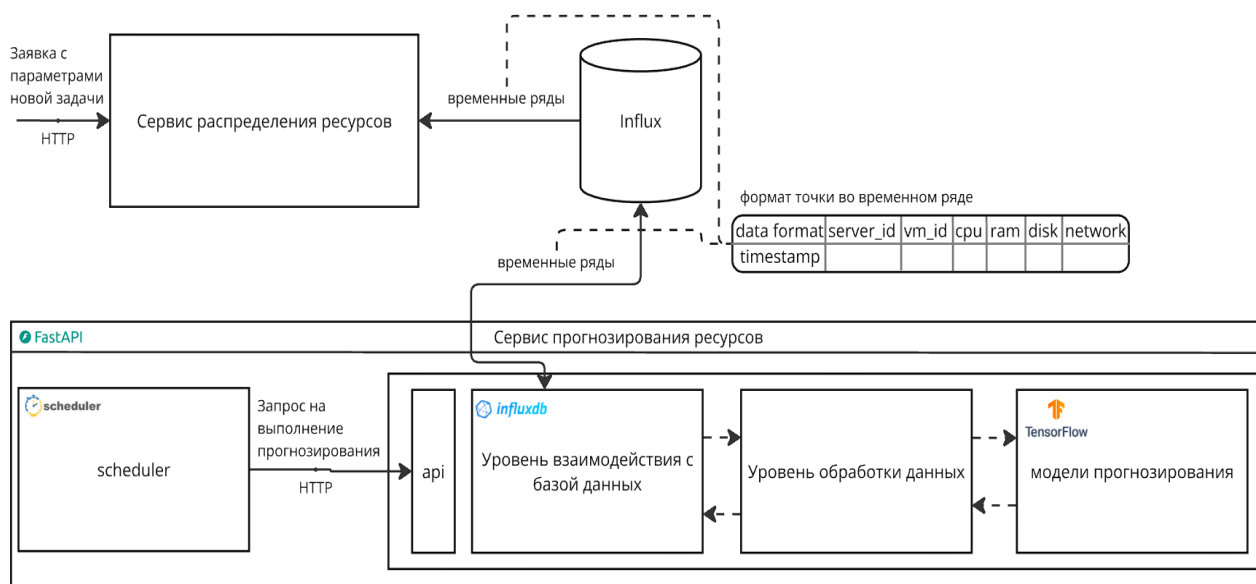


Рисунок 1 – Схема работы системы прогнозирования нагрузки и управления ресурсами

Источником входных данных для сервиса прогнозирования являются временные ряды, содержащие ключевые метрики загрузки облачных серверов, такие как использование ресурсов (cpu, ram, disk, network). Данные хранятся в формате временных

последовательностей. В дополнение к реальным данным в систему включён модуль генерации синтетических временных рядов для тестирования и валидации моделей.

Архитектура сервиса прогнозирования основана на многоуровневом подходе, обеспечивающем разделение приложения на несколько уровней: планировщик, API-интерфейс, уровень взаимодействия с базой данных, уровень обработки данных и уровень прогнозирования, использующий предварительно обученные модели машинного обучения. В рамках предобработки данные агрегируются и подготавливаются к подаче в модель. Прогнозирование выполняется по запросу через endpoint REST API, а для автоматического периодического запуска реализован планировщик.

Выходные данные сервиса представляют собой прогнозируемые временные ряды, содержащие те же ключевые метрики, но с предсказанными значениями на будущие временные точки. Таким образом, система получает предсказанные ряды, аналогичные входным, но с изменёнными значениями и временными метками, что позволяет использовать их в сервисе распределения ресурсов для оптимального управления вычислительными мощностями.

Для реализации сервиса прогнозирования нагрузки выбран стек, обеспечивающий высокую производительность, масштабируемость и удобную интеграцию. FastAPI используется для обработки асинхронных запросов, что позволяет эффективно управлять большим количеством подключений. Поддержка Pydantic облегчает валидацию данных, а автоматическая генерация API-документации (Swagger/OpenAPI) упрощает тестирование и интеграцию с другими сервисами. Кроме того, FastAPI поддерживает микросервисную архитектуру, что делает его удобным для масштабирования.

TensorFlow предоставляет инструменты для реализации рекуррентных нейронных сетей (например, LSTM)[3], которые эффективны для анализа временных рядов и прогнозирования нагрузки. Библиотека обеспечивает высокую производительность при обработке больших объемов данных, поддерживая работу на GPU, а также интегрируется с другими инструментами анализа данных[4].

InfluxDB[5] выбрана как специализированная СУБД для временных рядов, обеспечивающая высокую скорость записи, а также поддержку сжатия данных для оптимизации хранения.

Для автоматического запуска прогнозирования используется библиотека APScheduler, позволяющая гибко настраивать выполнение периодических задач. Она поддерживает асинхронное выполнение, обработку ошибок и возможность повторного запуска при сбоях, что повышает надежность всей системы. Такой стек технологий обеспечивает баланс между производительностью, удобством разработки и возможностью масштабирования.

Разработанное средство прогнозирования успешно реализовано и интегрировано в систему, однако его эффективность пока не оценена, в настоящее время проводится апробация предложенного подхода.

ЛИТЕРАТУРА

1. Armbrust, M., Fox, A., Griffith, R., et al. A View of Cloud Computing. [Электронный ресурс] Режим доступа: <https://dl.acm.org/doi/10.1145/1721654.1721672>
2. Gao, J., Wang, H., Shen, H. Machine Learning Based Workload Prediction in Cloud Computing. [Электронный ресурс] Режим доступа: https://www.researchgate.net/publication/347019069_Machine_Learning_Based_Workload_Prediction_in_Cloud_Computing
3. Hochreiter, S., Schmidhuber, J. Long Short-Term Memory. [Электронный ресурс] Режим доступа: <https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735>
4. Feng, B., Ding, Z. A Survey of Cloud Workload Prediction Methods. [Электронный ресурс] Режим доступа: <https://ieeexplore.ieee.org/abstract/document/10676402>
5. InfluxDB Documentation. [Электронный ресурс] Режим доступа: <https://docs.influxdata.com/>

РАЗРАБОТКА СЕРВИСА ПОДДЕРЖКИ КЛИЕНТОВ В СИСТЕМЕ ОБРАБОТКИ
ЗВОНКОВ

В работе рассматривается проблема ведения и поддержки обработки звонков и автоматизации ведения клиентской базы для компаний. Для эффективного управления данными необходимо создать инструмент, который позволит специалистам сократить время выполнения задач, снизить вероятность ошибок при операциях с клиентами и снизить количество сотрудников, занятых над одной задачей.

Целью данного проекта является разработка веб-приложения для обслуживания клиентов в системе обработки звонков (рисунок 1). Разрабатываемый сервис (рисунок 2, выделен красным) включает функционал управления клиентской базой, назначения тарифов, учёта платежей и предоставляет специализированный интерфейс для prompt-инженеров.

Система (рисунок 1), в которую нужно встроить сервис, работает следующим образом: клиент, который хочет получить аналитику, заводится в базе данных через CRM модуль или непосредственно разработчиком. Ему начисляется баланс и тариф, по которому будет проводиться анализ звонков. Затем они проходят фильтрацию, чтобы отбросить короткие, некачественные экземпляры, транскрибируют и хранят отдельно в Yandex Cloud. Для них составляют пайплайн, в котором описываются критерии, по которым нужно оценить эти звонки и отправляют в LLM модель. Конечный результат хранится в базе данных и отдается как отчет клиенту.

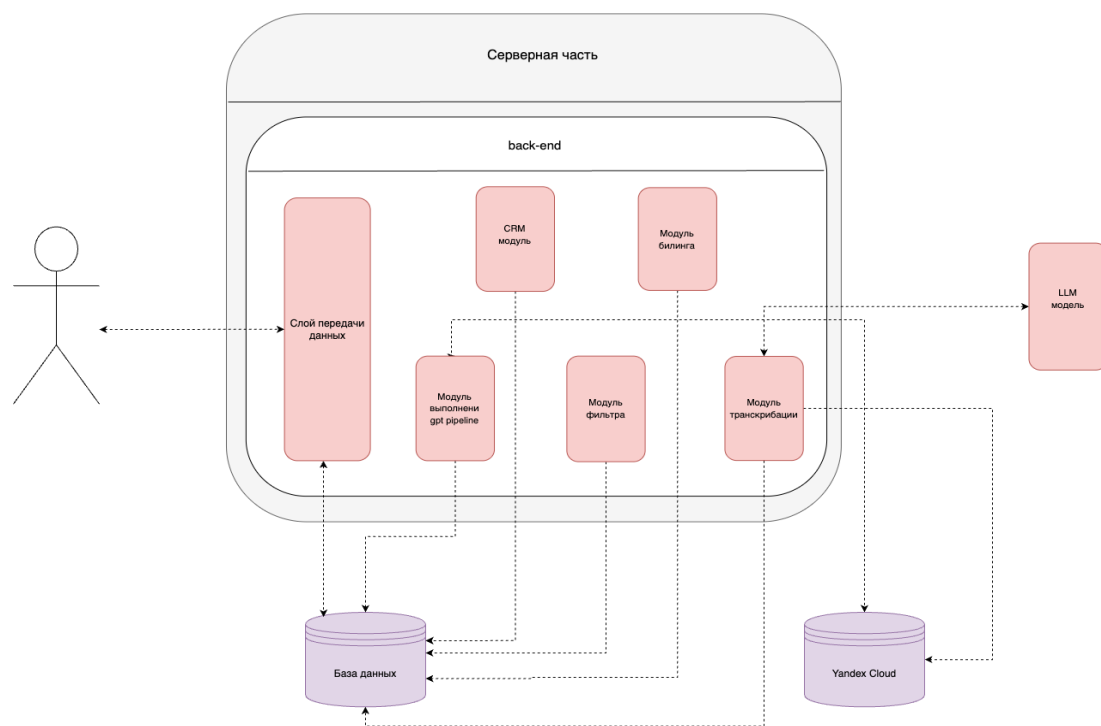


Рисунок 1 – Система обработки звонков

При проектировании веб приложения были выделены следующие задачи, выполнение которых позволило бы достигнуть поставленной цели:

- Реализация backend части приложения исходя из выбранного технологического стека.
- Реализация взаимодействие с базой данных и хранилищем файлов.
- Создание системы авторизации пользователей и разделение их на роли.
- Доработка базы данных путем добавления новых таблиц и изменения старых.

- Реализация frontend части приложения исходя из выбранного технологического стека.
- Реализация взаимодействия frontend и backend посредством API.
- Реализация поисковой строки клиентов.
- Тестирование ключевых модулей системы.

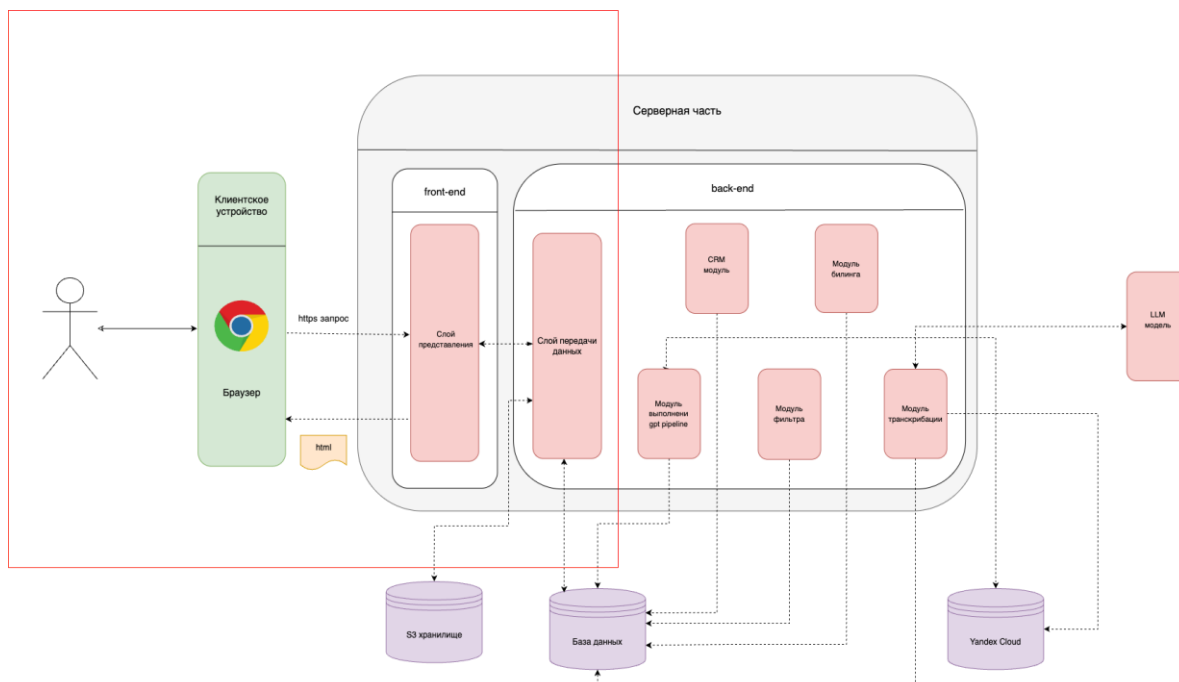


Рисунок 1 – Архитектура веб приложения

При проектировании сервиса была выбрана клиент-серверная (рисунок 2, выделен красным) архитектура, исходя из будущей загруженности, и разбит внутри серверной части (слой передачи данных) на многослойную архитектуру. Такой подход позволяет проще расширять систему, по мере роста основного сервиса, а также тестировать и поддерживать кодовую базу.

Серверная часть написана на python с применением фреймворка Fast API [1]. Также была использована orm SQLAlchemy [4], абстракция над базой данных, которая позволяет эффективно обрабатывать запросы и взаимодействовать с базой данных PostgreSQL. Клиентская часть реализована на Next.js [2] и Material UI [3]. Клиент и сервер связываются по протоколу REST API. Также реализовано подключение и взаимодействие с хранилищем Yandex S3, где хранятся файлы.

В сравнении с конкурентами, такими как UIS [6], PhoneUP 5.0 [7], наш сервис имеет преимущества, а именно настраиваемая и глубокая аналитика с помощью LLM модели, улучшенная работа с тарифами и балансом клиентов для автоматического пополнения и списания средств.

ЛИТЕРАТУРА

1. Документация по FastAPI [Электронный ресурс]. Режим доступа: <https://fastapi.tiangolo.com/>
2. Документация по Next.js [Электронный ресурс]. Режим доступа: <https://nextjs.org/docs>
3. Документация по Material UI [Электронный ресурс]. Режим доступа: <https://mui.com/material-ui/getting-started/>
4. Документация по SQLAlchemy [Электронный ресурс]. Режим доступа: <https://docs.sqlalchemy.org/en/20/>
5. Документация по PostgreSQL [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/>
6. UIS [Электронный ресурс]. Режим доступа: <https://www.uiscom.ru/academiya/spravochnyj-centr/dokumentatsiya-api/>
7. PhoneUP [Электронный ресурс]. Режим доступа: <https://aurus5.ru/phoneup/>

РЕАЛИЗАЦИЯ ИНСТРУМЕНТА СЖАТИЯ ФАЙЛОВ НА ОСНОВЕ
ОСОБЕННОСТЕЙ ЦВЕТОВОСПРИЯТИЯ

Около 5% населения Земли и примерно 8% мужчин страдает от цветовой слепоты, также известной как дальтонизм [1]. При таком недуге люди различают меньше цветов, эту особенность мы можем использовать для сжатия изображений. Проблема увеличения количества данных затрагивает умы многих разработчиков по всему миру, для примера, количество данных, созданных в мире в 2023 в 13 раз больше, чем в 2013 году [2].

Одним из способов сокращения информации является сжатие с потерями. Отличным примером послужит формат МР3, который основан на принципе сокращения избыточности данных [3]. По аналогии с ним предполагается создать инструмент, который игнорирует один из трёх оттенков основных цветов, по аналогии с восприятием людей с дихромазией. В последнее время разработчики социальных сетей часто задаются вопросом об оптимизации передачи медиафайлов пользователю [4]. Данный же инструмент позволит сократить количество трафика, отправляемого подобным способом.

Таким образом, *целью* данной работы является реализация инструментального средства, которое способно снизить размера изображений за счёт создания подхода оптимизации цветов взяв за основу восприятие цвета людей с дихромазией.

Для достижения этой цели необходимо решение следующих задач:

1. Обзор научной литературы по теме цветовой слепоты и изображению цветов в существующих программных форматах.
2. Предложение подхода для оптимизации цвета изображений.
3. Реализация данного подхода.
4. Демонстрация снижения размера файлов.

Архитектура инструмента представлена на рисунке 1. На вход программы подаётся изображение, которое необходимо оптимизировать, на выход возвращается оно же сжатым. Суть инструментального средства заключается в реализации алгоритма Бреттеля [5, 6]. За счёт того, что дихроматы не способны видеть один из трёх основных цветов (красный, зелёный или синий) мы перекодируем файл, избавляясь от ненужного. Изначально пользователь при использовании инструмента указывает тип цветовосприятия, далее цвета изображения при помощи матриц преобразуются из цветовой модели sRGB в цветовую модель LMS. После на основе векторов цветов LMS строится матрица преобразования. Далее уже изменённое фото преобразуется обратно из LMS в sRGB. В результате пользователь видит такую же картинку, как и до этого, но мы отдаём файл меньшего размера.

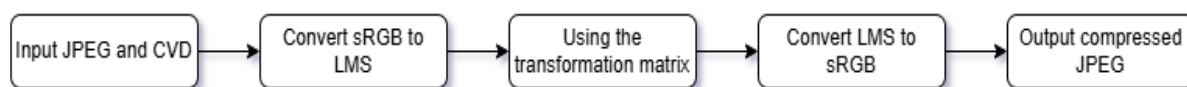


Рисунок 1 – Архитектура системы

Подход реализован в программном средстве на языке Java [7]. Данный подход позволяет сэкономить трафик, передаваемый пользователю, например, при передаче фото в социальной сети. Сэкономленный трафик может быть использован для улучшения качества другими способами, например, увеличением разрешения. Также он может позволить улучшить пользовательский опыт в виде увеличения скорости загрузки и уменьшения задержки. В

перспективе возможно расширение категорий цветовой слепоты, а также возможность работы с различными цветовыми пространствами, например в HDR-диапазоне.

ЛИТЕРАТУРА

1. H. Kolb, E. Fernandez, and R. Nelson (Eds.). 2011. WebVision: The Organization of the Retina and Visual System. John Moran Eye Center, University of Utah, Salt Lake City, UT. Retrieved from <http://webvision.med.utah.edu/>.
2. Данные о росте объема данных в мире с 2010 по 2028 год. Statista. URL: <https://www.statista.com/statistics/871513/worldwide-data-created/> (дата обращения: 03.03.2025).
3. Кенкишвили В. С. СЖАТИЕ ЦИФРОВОЙ АУДИОИНФОРМАЦИИ ПО МЕТОДУ МР3 // Вестник науки. 2024. №6 (75). URL: <https://cyberleninka.ru/article/n/szhatie-tsifrovoy-audioinformatsii-po-metodu-mp3> (дата обращения: 04.03.2025).
4. Лей Райан, Ши Хайся, Чен Хаотенг, Монафаред Али, Ши Чен. Как Meta внедрила AV1 в Reels [Электронный ресурс] / Райан Лей, Хайся Ши, Хаотенг Чен, Али Монафаред, Чен Ши. – Режим доступа: <https://engineering.fb.com/2023/02/21/video-engineering/av1-codec-facebook-instagram-reels/> (дата обращения: 04.03.2023).
5. H. Brettel, F. Viénot, and J. Mollon. 1997. Computerized simulation of color appearance for dichromats. J. Opt. Soc. Am. 14, 10 (1997), 2647–2655.
6. F. Vienot, H. Brettel, and J. Mollon. 1999. Digital video colourmaps for checking the legibility of displays by dichromats. Color Res. Appl. 24, 4 (1999), 243–252.
7. Java. [Электронный ресурс] Режим доступа: <https://www.java.com/>. (дата обращения 04.03.2025)

УДК 004.65

А.А. Палий (4 курс бакалавриата),
С.А. Нестеров, к. т. н., доцент

АНАЛИЗ МЕТОДОВ ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ POSTGRESQL ПРИ РАБОТЕ СО СЛОЖНЫМИ ТИПАМИ ДАННЫХ

Базы данных играют ключевую роль в современных информационных системах. PostgreSQL – одна из самых популярных СУБД, известная своей гибкостью и богатым функционалом. С ростом объемов данных и требований к скорости работы возникает задача оптимизации производительности базы данных. В последнее время особенно актуально хранение и обработка сложных типов данных – например, JSON-документов или пространственной (географической) информации – непосредственно в базе. Эти форматы данных удобны и гибки, однако могут негативно влиять на производительность без специальной настройки. Цель – проанализировать методы оптимизации СУБД при работе как с обычными реляционными данными, так и с JSON/JSONB и геоданными, и провести сравнительный анализ их эффективности. Это позволит оценить, как сохранить высокую производительность PostgreSQL при расширении её функционала до NoSQL-подходов и специализированных типов данных.

Индексы – один из инструментов ускорения запросов к базе. Индекс представляет собой структуру данных, которая позволяет быстро находить нужные строки без полного сканирования таблицы. Правильно настроенный индекс может значительно сократить число обращений к диску и времени поиска записи [4]. PostgreSQL поддерживает различные виды индексов: классические B-Tree (эффективны для сравнений =, >, <), хэш-индексы для поиска по равенству, а также специализированные индексы для сложных данных – GiST, SP-GiST, GIN, BRIN и др. [5]. Например, GIN предназначен для индексирования вложенных структур, таких как массивы или документы JSON, позволяя эффективно искать по содержимому этих полей [1]. За счёт использования соответствующих индексов PostgreSQL может значительно ускорить выборку из колонок со сложными типами данных.

Другой важный метод оптимизации – использование кеша и грамотная настройка памяти. PostgreSQL хранит часто запрашиваемые данные в буферном кеше (памяти). Это существенно повышает скорость, так как чтение из оперативной памяти намного быстрее дисковых операций [2]. Параметр `shared_buffers` определяет объём памяти под кеш страниц базы; увеличение этого параметра позволяет удерживать больше данных в памяти и реже обращаться к диску. Помимо этого, настройка параметра `work_mem` (память для сортировки и хеш-таблиц) помогает ускорить сложные операции за счёт выполнения их в памяти без промежуточных файлов на диске. Оптимизация этих параметров должна выполняться с учётом нагрузки: слишком маленький кеш замедлит систему, а чрезмерно большой может привести к неэффективному использованию ресурсов.

Производительность СУБД во многом зависит от эффективности самих SQL-запросов. Необходимо анализировать планы выполнения и избегать неоптимальных конструкций. Например, вместо тяжёлых вложенных подзапросов или последовательных сканирований таблиц стоит использовать индексы и подходящие операции соединения. Правильно спроектированные запросы в сочетании с индексами уменьшают количество обрабатываемых строк и операций ввода-вывода, что прямо ускоряет работу. Другая техника – материализованные представления – позволяет кешировать результаты тяжёлых агрегирующих запросов. Материализованное представление сохраняет вывод запроса как отдельную таблицу и при повторных обращениях сразу выдаёт готовые данные. Доступ к заранее вычисленным данным зачастую гораздо быстрее непосредственного выполнения сложного запроса к исходным таблицам [7]. Материализованные представления значительно снижают нагрузку на базу при регулярном выполнении одних и тех же аналитических запросов.

JSON/JSONB. PostgreSQL позволяет хранить JSON-документы двумя способами: как текстовый JSON или в двоичном формате JSONB. Прямое хранение JSON удобно, но при выборке такие данные парсятся заново для каждого запроса, что замедляет работу. В отличие от этого, `jsonb` сохраняется в разобранном бинарном виде, благодаря чему чтение и поиск по структуре документа выполняются гораздо быстрее (цена – чуть более медленная загрузка, т. к. при вставке происходит разбор JSON) [1]. Для часто изменяемых данных с чёткой структурой реляционная модель обычно эффективнее, тогда как для хранения логов, конфигураций или редко обновляемых сложных объектов JSONB предоставляет удобство без значительной потери производительности при грамотной настройке.

PostgreSQL (совместно с расширением PostGIS) позволяет работать с географическими объектами: точками, полигонами, линиями и т. д. Такие пространственные данные существенно отличаются от числовых или строковых, и операции над ними вычислительно более сложны. Без специальной оптимизации поиск по геоданным быстро становится узким местом: прямое сканирование большой таблицы с геометрией для каждой пространственной проверки крайне медленно. Решение — использование геоиндексов. PostgreSQL поддерживает R-деревья через механизм GiST/SP-GiST для индексирования геометрических типов [6]. Spatial-индексы разбивают пространство на области и хранят ограничивающие прямоугольники объектов, что позволяет быстро отсекаать заведомо не соответствующие условию записи. Практически невозможно эффективно использовать пространственную базу без индексации: поиск геообъекта по условию без индекса требует последовательного просмотра каждой записи, тогда как индекс даёт возможность сразу сузить поиск до небольшой части данных [3].

Сравнение подходов. Использование сложных типов данных даёт очевидные выигрыши в гибкости разработки и функциональности базы. Хранение JSONB внутри PostgreSQL позволяет отказаться от отдельной NoSQL-хранилища, объединив данные в одной системе, а хранение геоданных в базе даёт возможность выполнять пространственные запросы с использованием языка SQL. При этом цена такой универсальности – потенциальное снижение быстродействия, если не учитывать особенности этих типов. Реляционная модель при прочих равных обеспечивает максимальную производительность запросов: фильтрация и соединение

по обычным столбцам (INTEGER, TEXT и т. д.) выполняются быстрее, чем поиск внутри сложных структур. Однако современный PostgreSQL компенсирует издержки: благодаря поддержке JSONB и мощным механизмам индексирования разрыв в производительности существенно сокращается [1].

При правильном подходе PostgreSQL способен эффективно работать со сложными типами данных без критического ухудшения производительности. Таким образом, расширенные возможности PostgreSQL могут быть реализованы без серьезных потерь эффективности, если использовать богатый арсенал средств оптимизации. В перспективе дальнейшее развитие PostgreSQL продолжает снижать издержки работы со сложными типами.

ЛИТЕРАТУРА

1. PostgreSQL Documentation. JSON Types – The json and jsonb data types. PostgreSQL 17 Official Documentation [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/datatype-json.html#:~:text=The%20can%20be%20a%20significant%20advantage>.
2. Kothari N. Understanding the Importance of shared_buffers... – Fastware Blog. [Электронный ресурс] Режим доступа: <https://www.postgresql.fastware.com/pzone/2024-06-understanding-shared-buffers-work-mem-and-wal-buffers-in-postgresql#:~:text=Why%20is%20it%20important%3F>.
3. PostGIS Workshop. Chapter “Spatial Indexing” – PostGIS Documentation. [Электронный ресурс] Режим доступа: <https://postgis.net/>.
4. Timescale. PostgreSQL Performance Tuning: Optimizing Database Indexes. [Электронный ресурс] Режим доступа: <https://www.timescale.com/learn/postgresql-performance-tuning-optimizing-database-indexes#:~:text=If%20you%20notice%20higher%20values,may%20significantly%20improve%20query%20performance>.
5. Рогов Е. В. PostgreSQL 16 изнутри. – М.: Издательство ДМК Пресс, 2024. – 664 с.
6. Habr (PostgresPro). Индексы в PostgreSQL – Часть 5: GiST (Автор: Е. Рогов). [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/postgrespro/articles/333878/>.
7. PostgreSQL Documentation. Materialized Views – Access to data in a materialized view is faster.... PostgreSQL 17 Official Documentation [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/current/rules-materializedviews.html#:~:text=While%20access%20to%20the%20data,a%20table%20which%20records%20sales>.

УДК 004.4`2

И.А. Перов, Г.М. Жуков (4 курс бакалавриата),
П.Д. Дробинцев, к.т.н., доцент

ВЫБОР ФРЕЙМВОРКА ДЛЯ ИНЪЕКЦИИ ЗАВИСИМОСТЕЙ ПРИ РАЗРАБОТКЕ ANDROID-ПРИЛОЖЕНИЙ НА ОСНОВЕ ОЦЕНКИ ВЛИЯНИЯ НА ПРОИЗВОДИТЕЛЬНОСТЬ И СКОРОСТИ СБОРКИ

Современные Android-приложения характеризуются высокой сложностью архитектуры и большим числом зависимостей между компонентами. Для управления зависимостями активно применяются различные фреймворки инъекции зависимостей (Dependency Injection, DI). Выбор фреймворка существенно влияет на скорость работы приложения и сборки проекта.

Среди лидеров фреймворков для DI в Android выделяются Dagger, Koin и Hilt, каждый из которых использует свой подход к инъекции зависимостей [2,3,4].

Целью данной работы является исследование влияния фреймворков Dagger, Koin и Hilt на производительность Android-приложений и время сборки в мономодульных и многомодульных проектах.

Для достижения этой цели необходимо решение следующих задач:

1. Анализ подходов и инструментов внедрения зависимостей в Android-разработке.
2. Разработка тестовых приложений с использованием фреймворков Dagger, Koin и Hilt в моно- и многомодульной архитектуре.
3. Замер и сравнительный анализ времени сборки приложений («холодная» и «горячая» сборки).
4. Анализ полученных результатов и формулирование рекомендаций по выбору фреймворка в зависимости от структуры приложения и требований к производительности.

В рамках исследования проводились временные замеры и тесты на устройствах с одинаковыми аппаратными характеристиками. Для объективности результатов использовался следующий набор условий:

1. Холодная сборка - первая сборка проекта после полной очистки всех временных файлов (команда `./gradlew clean assembleDebug`).
2. Горячая сборка - повторная сборка проекта без изменения исходного кода (команда `./gradlew assembleDebug`).
3. Runtime-производительность - измерение времени запуска приложения (cold start) при одинаковом наборе экранов и действий (команда `adb shell am start-activity`) [1, 6].

Все замеры производились для каждого фреймворка (Dagger, Koin, Hilt) - как для моно-, так и для многомодульного Android-приложения. Проект приложения среднего размера (5 экранов).

Таблица 1 – Результаты замеров сборки и runtime-производительности приложений

Фреймворк	Тип проекта	Холодная сборка, сек	Горячая сборка, сек	Время запуска (cold start), мс
Dagger	Мономодульный	123	28	448
Dagger	Многомодульный	290	68	560
Koin	Мономодульный	61	18	672
Koin	Многомодульный	145	30	784
Hilt	Мономодульный	92	22	476
Hilt	Многомодульный	199	46	588

В рамках исследования были получены следующие результаты (таблица 1):

– Влияние на сборку:

Koin показывает лучшие результаты и для горячей, и для холодной сборки, что обусловлено отсутствием кодогенерации на этапе сборки проекта. Dagger же продемонстрировал наибольшее время сборки, что связано с работой процессора аннотаций для сканирования кода и генерацией значительного объема кода на этапе компиляции. Hilt, как надстройка над Dagger, занял промежуточное положение.

– Влияние на runtime-производительность (время запуска):

Dagger показал наилучшие результаты благодаря статической кодогенерации. Koin продемонстрировал наибольшие задержки из-за того, что Koin разрешает зависимости во время выполнения. Hilt занял промежуточную позицию.

Таким образом, результаты исследования позволяют сформулировать *рекомендации* по выбору фреймворка с точки зрения производительности для проектов среднего размера:

– Dagger - оптимален для сложных приложений, где первостепенна производительность во время выполнения, однако необходимо учитывать увеличение времени сборки.

– Koin - эффективен для небольших простых проектов и прототипов, где важна скорость сборки, а снижение производительности не критично.

– Hilt - оптимален для проектов средней сложности, требующих баланса между скоростью сборки и производительностью во время исполнения. Дополнительно он сильно проще в освоении, чем Dagger.

Выводы данного исследования были успешно *применены* в рамках разработки Android-

приложения, являющегося основой дипломного проекта «Мобильное приложение для систематизации и отслеживания предметов». Учёт влияния фреймворков DI на производительность и скорость сборки позволил выбрать Hilt в качестве оптимального решения.

ЛИТЕРАТУРА

1. Testing App Startup Performance - Medium. [Электронный ресурс] Режим доступа: <https://medium.com/androiddevelopers/testing-app-startup-performance-36169c27ee55> (дата обращения: 01.03.2025).
2. Шумилова Е. А., Свищев А. В. Внедрение зависимостей в Android-разработке. Особенности и сравнение DI-фреймворков платформы Android // Моя профессиональная карьера. - 2023. - Т. 2, № 54. - С. 245-257. [Электронный ресурс] Режим доступа: <https://elibrary.ru/item.asp?id=54961854> (дата обращения: 06.03.2025).
3. Dependency Injection in Android: Dagger vs Hilt. - Medium [Электронный ресурс] Режим доступа: <https://medium.com/mobile-innovation-network/dependency-injection-in-android-dagger-vs-hilt-59547456ba2c> (дата обращения: 2.03.2025).
4. Сравнение Dagger 2, Dagger Hilt и Koin. - Habr. [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/890462/> (дата обращения: 02.03.2025).
5. Чем отличаются Dagger, Hilt и Koin под капотом? - Habr [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/otus/articles/545222/> (дата обращения: 01.03.2025).
6. Learning about Android runtime - Medium [Электронный ресурс] Режим доступа: <https://medium.com/mateedevs/learning-about-android-runtime-56b409aeb9b2>.

УДК 004.453

А.А. Петров (3 курс бакалавриата),
А.В. Щукин, к.т.н., доцент,
В.А. Пархоменко, старший преподаватель

РАЗРАБОТКА ПЛАГИНА СБОРА МЕТРИК ДЛЯ DRONE И СРАВНЕНИЕ ЕГО С АНАЛОГАМИ В ДРУГИХ ИНСТРУМЕНТАХ CI/CD

CI/CD – востребованная сфера разработки. С рынка ушли многие компании, поэтому на первый план выходят open-source решения, одним из которых является Drone [1]. Jenkins считается лидером на рынке, но он довольно сложен для новичков из-за избыточной функциональности и сложной структуры кода. Drone в свою очередь – это легковесное конвейеризованное CI/CD решение с открытым исходным кодом.

Целью работы является разработка плагина сбора метрик при нагрузочном тестировании для Drone и сравнение его с аналогами. Для достижения этой цели необходимо решение следующих задач:

1. Проанализировать аналогичные решения в разных инструментах CI/CD.
2. Реализовать плагин с использованием выбранных инструментов.
3. Провести тестирование плагина, чтобы убедиться в его корректной работе.

Наш плагин выполняет 3 основные задачи:

1. Проводит нагрузочное тестирование приложения.
2. Собирает метрики.
3. Генерирует отчет в удобной форме и отправляет в подписавшиеся сервисы.

Для получения подобного функционала в Drone или Jenkins требуется последовательное выполнение нескольких независимых друг от друга приложений. Для этого нужно запустить как минимум 3 шага в цепочке pipeline, что усложняет понимание процесса и повышает вероятность возникновения неожиданных ошибок.

Разрабатываемый плагин позволяет выполнить все эти действия за один шаг в pipeline, получая ту же функциональность. Аналогичного готового решения для Drone и Jenkins не удалось найти, поэтому я и выбрал разработку данного плагина.

Архитектура плагина представлена на рисунке 1, а схема его работы на рисунке 2.

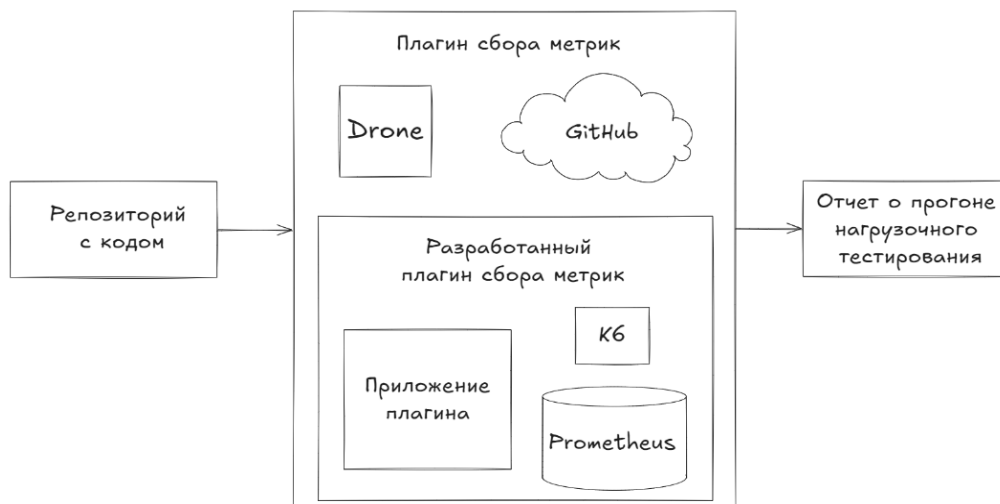


Рисунок 1 – Архитектура плагина

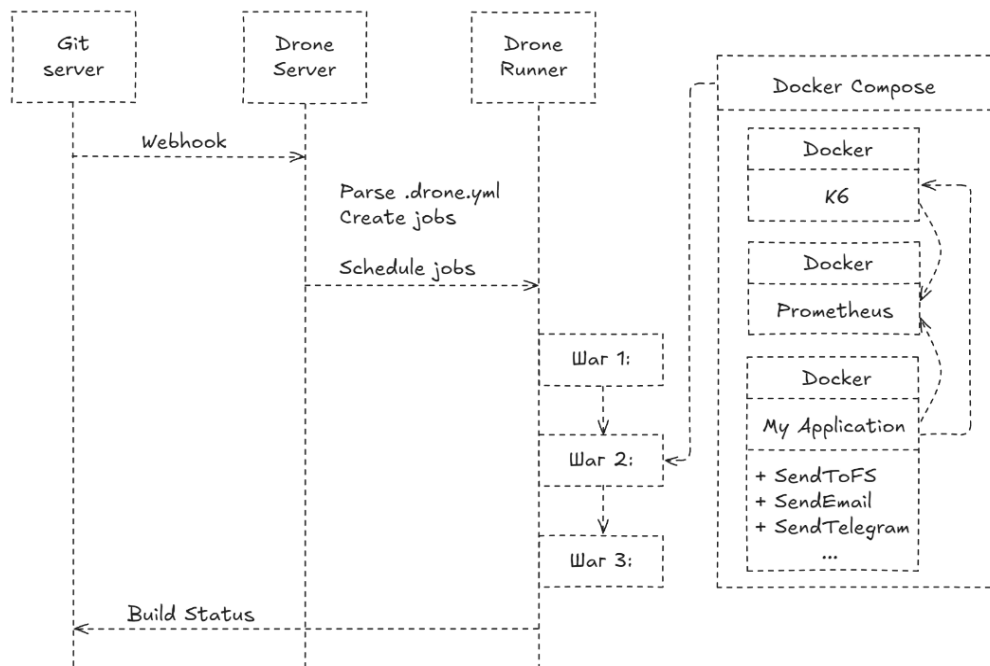


Рисунок 2 – Схема работы плагина

Плагин реализован на языке Java. В качестве фреймворка для нагрузочного тестирования был выбран K6.

Результатом тестирования является корректно работающий плагин. Кроме того, поскольку плагины в Drone CI фактически являются отдельными Docker контейнерами, выполняющими конкретную, конечную задачу, наш плагин можно использовать в любых инструментах CI/CD, поддерживающих технологию контейнеризации.

ЛИТЕРАТУРА

1. Drone CI. Официальная документация [Электронный ресурс]. — URL: <https://docs.drone.io/> (дата обращения: 22.02.2025).
2. Gião H. da, Flores A., Pereira R., Cunha J. Chronicles of CI/CD: A Deep Dive into its Usage Over Time // arXiv. — 2024. — 27 февраля. — arXiv:2402.17588. — URL: <https://arxiv.org/abs/2402.17588> (дата обращения: 25.02.2025).

3. k6. Официальная документация [Электронный ресурс]. — URL: <https://k6.io/docs/> (дата обращения: 26.02.2025).
4. Prometheus. Официальная документация [Электронный ресурс]. — URL: <https://prometheus.io/docs/> (дата обращения: 27.02.2025).
5. Fairbanks J., Tharigonda A., Eisty N. Analyzing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab // arXiv. — 2023. — 29 марта. — arXiv:2303.16393. — URL: <https://arxiv.org/abs/2303.16393> (дата обращения: 1.03.2025).
6. Trubiani C., Bran A., van Hoorn A., Avritzer A., Knoche H. Exploiting Load Testing and Profiling for Performance Antipattern Detection // Information and Software Technology. — 2018. — Т. 95. — С. 329-345. — DOI: 10.1016/j.infsof.2017.11.016. — URL: <https://doi.org/10.1016/j.infsof.2017.11.016> (дата обращения: 4.03.2025).

УДК 681.3.06

М.В. Рафальская (4 курс бакалавриата),
В.А. Семенова-Тян-Шанская, к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РАСЧЕТА ТРУДОЕМКОСТИ ПОСТРОЙКИ СТРОИТЕЛЬСТВА ТРАНСПОРТНЫХ СУДОВ

В условиях развития экономики прогресс судостроительной отрасли зависит от комплексного использования всех источников повышения эффективности производства. Одним из таких источников является обеспечение качества подготовки судостроительного производства, когда грамотно оцениваются планируемые трудовые и материальные затраты на постройку судов с учетом мирового опыта. При этом вопросы обеспечения постройки судов непосредственно влияют на качество принимаемых проектных решений [1].

В настоящей работе описывается приложение для расчета трудоемкости строительства транспортного судна. Разработка выполнена на языке Python с использованием фреймворка PySide для создания удобного графического интерфейса [2]. Основные функциональные возможности приложения включают:

- Выбор методики расчета трудоемкости (через водоизмещение или нагрузки масс судна);
- Ввод параметров судна (тип, водоизмещение, нагрузки масс, главные размерения, год сдачи);
- Расчет трудоемкости на основании имеющихся нормативов;
- Генерацию отчетов по результатам вычислений.

Проектная трудоемкость серийно освоенного судна определяется по формуле:

$$T_0 = \sum_{i=1}^m T_i,$$

где:

m – количество видов работ по судну, трудоемкость которых определена нормативами;

i – порядковый номер вида работ, $i=1, \dots, m$;

T_i – трудоемкость i -го вида работ, чел. ч.

В рамках расчета трудоемкости строительства судна также производится пересчет проектной трудоемкости T_0 с учетом поправочных коэффициентов, определяемыми нормативами в зависимости от года сдачи судна (K_T), количества судов одного проекта в год (K_N) и количества судов серии, предшествующих рассчитываемому (K_i). Например, величина проектной трудоемкости с учетом поправочного коэффициента K_T определяется по формуле [3]:

$$T_{KT} = T_0 * K_T.$$

Для проверки корректности работы программы проведена верификация и валидация данных, в рамках которых результаты расчетов в приложении сравнивались с ранее рассчитанными данными по построенным судам. Данные представлены производственными отчетами предприятий-заказчиков в виде файлов в формате CSV по каждому типу судна. Анализ показал высокую точность разработанного алгоритма.

Разработанное решение представлено в двух версиях: десктопное приложение для Windows и мобильное приложение для операционной системы «Аврора».

На рисунке 1 представлена схема алгоритма расчета трудоемкости.

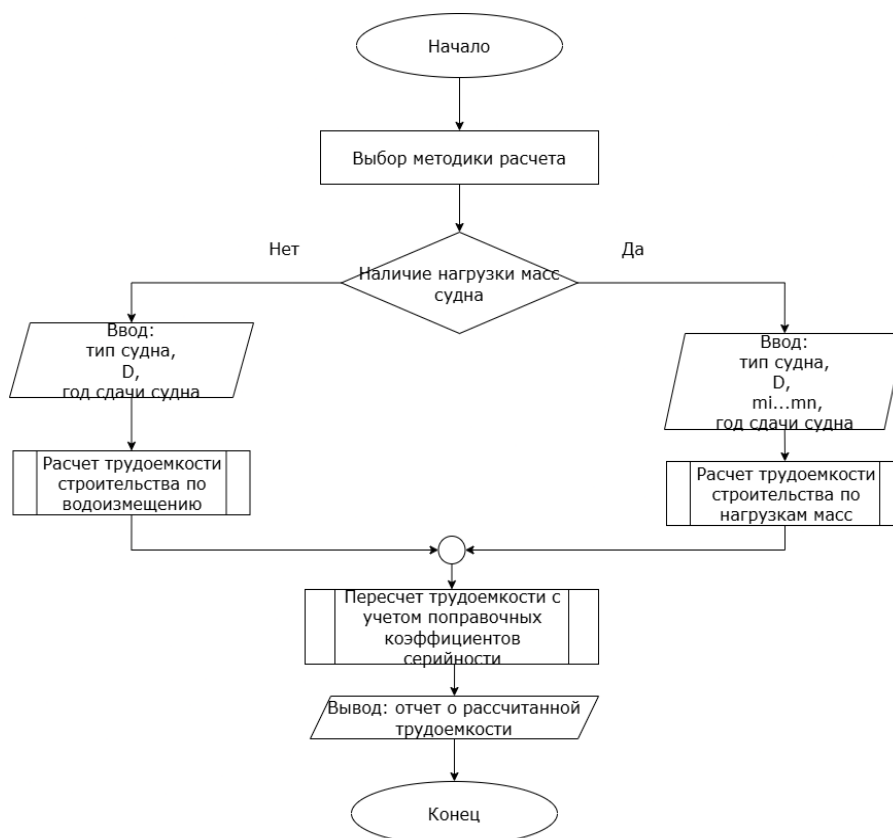


Рисунок 1 – Блок-схема алгоритма расчета трудоемкости

Применение приложения позволит снизить вероятность ошибок при расчетах, повысить точность планирования работ и автоматизировать процесс управления ресурсами при строительстве судов на предприятиях.

ЛИТЕРАТУРА

1. Нормативы трудоемкости строительства судов 2025. Москва, МОРКНИГА, 2025.
2. Любанович Б. Простой Python. Питер, 2021.
3. Руденко А.Н. Нормативы как важнейший элемент управления трудоемкостью строительства и ремонта судов. Труды Крыловского научного центра. Специальный выпуск 2, 2021.

УДК 004.41

В.А. Ромашов (1 курс магистратуры),
И.В. Никифоров, к.т.н., доцент,
О.А. Юсупова, ст. преподаватель

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА AI-АГЕНТА ДЛЯ АВТОМАТИЗАЦИИ РАЗРАБОТКИ ПРОГРАММНОГО КОДА

К 2030 году глобальный рынок искусственного интеллекта (ИИ) будет насчитывать 1.8 триллиона долларов [1]. Около 90% [2] руководителей компаний считают, что искусственный интеллект поможет увеличить прибыль. Решения на базе ИИ созданы для достижения бизнес задач в различных областях, а AI-агенты повышают на 55% эффективность и уменьшают до 35% расходов компаний, использующих их [3].

AI агенты – это искусственные сущности, которые могут автономно воспринимать окружающую среду и влиять на нее для достижения определенных целей. Данные цели могут задаваться человеком, но благодаря искусственному интеллекту сам агент может принимать решения о том, какие решения будут эффективными для достижения поставленных задач.

Целью работы является повышение производительности работы инженеров программистов за счет внедрения AI агентов в интегрированные среды разработки (IDE), которые автоматизируют процесс разработки кода приложений. Их отличительной особенностью является использование больших языковых моделей (Large Language Models, LLM).

Разрабатываемый агент должен стать одним из главных инструментов инженера при разработке и написании кода проекта.

Рассмотрим предлагаемую архитектуру AI агента [4], представленную на рисунке 1. Ниже приводится описание основных компонентов агента, включающего большую языковую модель.

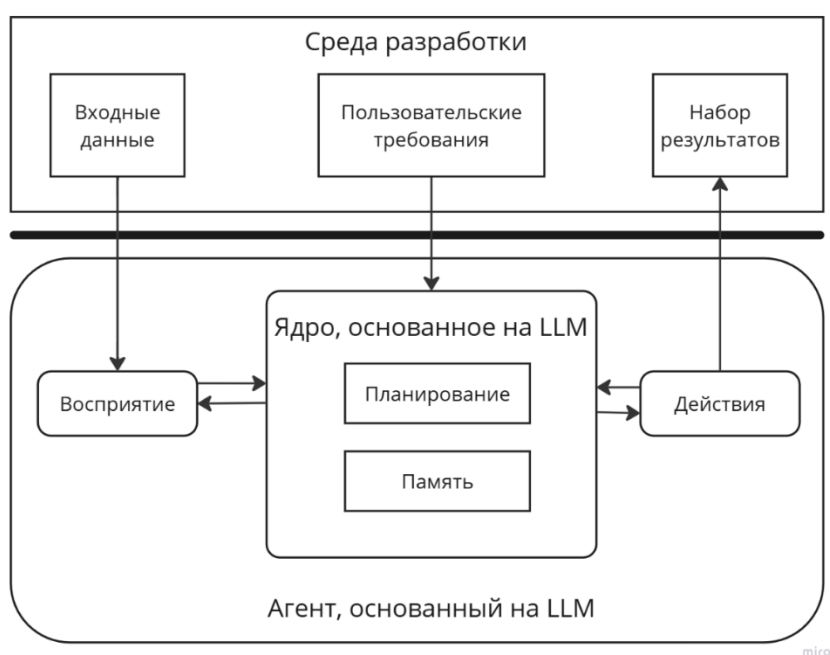


Рисунок 1 – Архитектура базового AI агента

Восприятие (сенсоры) – это интерфейсы, через которые агент получает данные из окружающей среды. В частности, агенты могут воспринимать различные данные, например: текстовые запросы, файлы, сигналы от других программ.

Действие (исполнительные механизмы) – это компоненты, которые позволяют агенту воздействовать на внешнюю среду. Одним из важных действующих механизмов является контроль и использование сторонних программ, которые могут расширить присущие большим языковым моделям возможности, предоставляя доступ к большому количеству внешних ресурсов и расширения пространства действий.

Память: AI агенты могут иметь краткосрочную и долгосрочную память для хранения информации о текущем состоянии, истории действий и результатах. Этот компонент важен, так как позволяет в рамках нашей задачи хранить информацию о запросах, действиях, результатах. Опираясь на них, агент сможет учиться и выдавать качественные ответы.

Планирование (модуль принятия решений) – это компонент, который декомпозирует сложные задачи, планирует их выполнение в контексте с глобальной целью, хранит алгоритмы и predetermined правила для принятия решений.

Таким образом, в работе была представлена архитектура AI агента на основе больших языковых моделей для упрощения и ускорения процесса работы инженера программиста. В дальнейшем планируется реализация прототипа программного решения, подтверждающий

концепцию и демонстрирующий повышение эффективности в рамках экспериментальных исследований на примере автоматизации генерации тестовых сценариев [5] для проекта инженерного портала [6].

ЛИТЕРАТУРА

1. Artificial intelligence (AI) worldwide - statistics & facts // Statista URL: <https://www.statista.com/outlook/tmo/artificial-intelligence/worldwide> (дата обращения: 15.03.2025).
2. Superagency in the workplace: Empowering people to unlock AI's full potential // McKinsey Digital URL: <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/superagency-in-the-workplace-empowering-people-to-unlock-ais-full-potential-at-work> (дата обращения: 15.03.2025).
3. Salesforce Stock Revival Hinges on Success of AI Agents // BNN Bloomberg URL: <https://www.bnnbloomberg.ca/business/company-news/2025/02/25/salesforce-stock-revival-hinges-on-success-of-ai-agents/> (дата обращения: 15.03.2025).
4. Large Language Model-Based Agents for Software Engineering: A Survey // ResearchGate URL: https://www.researchgate.net/publication/383792179_Large_Language_Model-Based_Agents_for_Software_Engineering_A_Survey (дата обращения: 15.03.2025).
5. Никифоров, И. В. Статический метод отладки тестовых сценариев, сгенерированных с использованием эвристик / И. В. Никифоров, А. В. Петров, В. П. Котляров // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. – 2012. – № 4(152). – С. 114-119. – EDN PCHCXX.
6. Ромашов, В. А. Проектирование и реализация пользовательского интерфейса для инженерного портала / В. А. Ромашов, Л. И. Ключкова, 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. 292-294. – EDN TVLHTV.

УДК 004.45

В.В. Радько (4 курс бакалавриата),
К.А. Туральчук, к.т.н., доцент

РАЗРАБОТКА ПЛАГИНА ANDROID GRADLE ДЛЯ ГЕНЕРАЦИИ ДЕМО-ПРИЛОЖЕНИЯ ДИЗАЙН-СИСТЕМЫ ПРОЕКТА

Создание качественного пользовательского интерфейса является одной из важнейших задач при разработке мобильных приложений. Для достижения единообразия и повышения эффективности работы команды разработчиков часто используют дизайн-системы – набор правил, принципов и инструментов, которые обеспечивают единый подход к разработке продуктов или услуг. Однако, демонстрация возможностей такой системы требует наличия специального демо-приложения, которое показывает применение всех компонентов в реальных сценариях [1].

Традиционно создание подобного демо-приложения занимает значительное количество времени и ресурсов, особенно когда речь идет о больших проектах с множеством различных элементов. Чтобы упростить эту задачу и ускорить процесс разработки, можно воспользоваться специализированными инструментами автоматизации.

Одним из таких инструментов является Android Gradle Plugin, предназначенный для автоматизированной и настраиваемой сборки Android-приложений. Этот плагин предоставляет возможность создания собственных задач, которые будут встраиваться в цикл сборки приложения [2]. С помощью данного механизма может быть реализована генерация демо-приложения дизайн-системы.

Таким образом, целью данной работы является разработка Android Gradle плагина, полностью автоматизирующего генерацию демо-приложения дизайн-системы.

Для достижения этой цели необходимо решение следующих задач:

1. Обзор существующих решений для генерации демо-приложений дизайн-систем для Android-проектов.
2. Выделение критериев для нового инструмента по итогу сравнительного анализа найденных подходов.
3. Подбор технологий для реализации решения.
4. Предложение реализации нового подхода с помощью Android Gradle Plugin и Kotlin Symbol Processing.
5. Реализация данного инструмента.
6. Анализ снижения трудозатрат на создание и поддержку демо-приложения с помощью предложенного решения.

По итогу обзора существующих подходов основными критериями для нового инструмента были выбраны:

1. Полная автоматизация создания и поддержки демо-приложения.
2. Удобная настройка генерации с использованием аннотаций и Kotlin DSL.
3. Возможность изменения настроек компонентов дизайн-системы в демо-приложении.

Первым этапом реализации плагина является анализ кода модуля дизайн-системы. Для его осуществления наиболее подходящей технологией был выбран Kotlin Symbol Processing [3], который позволяет анализировать символы внутри модуля. С помощью этого инструмента формируется отчет о компонентах модуля дизайн-системы, который далее будет использоваться для генерации демо-приложения. Отчет представляется в формате JSON. Процессор позволяет находить компоненты, написанные как с использованием View, так и с использованием Jetpack Compose.

Следующим этапом является разработка пользовательского интерфейса итогового демо-приложения. Чтобы осуществить данный этап, необходимо реализовать библиотеку [4] с основной бизнес-логикой демо-приложения. Сгенерированное приложение делегирует всю основную логику отрисовки View- и Compose-компонентов, а также логику изменения настроек компонентов данной библиотеке.

Последним этапом при реализации инструмента является создание основного Android Gradle плагина, который будет выполнять задачу итоговой генерации демо-приложения. Этот плагин запускает задачу анализа кода с помощью модуля Kotlin Symbol Processing, а затем на основе полученного JSON-отчета о компонентах модуля дизайн-системы генерирует код демо-приложения с требуемыми для генерации компонентами.

Модульная архитектура плагина и генерируемого демо-приложения представлена на рисунке 1.

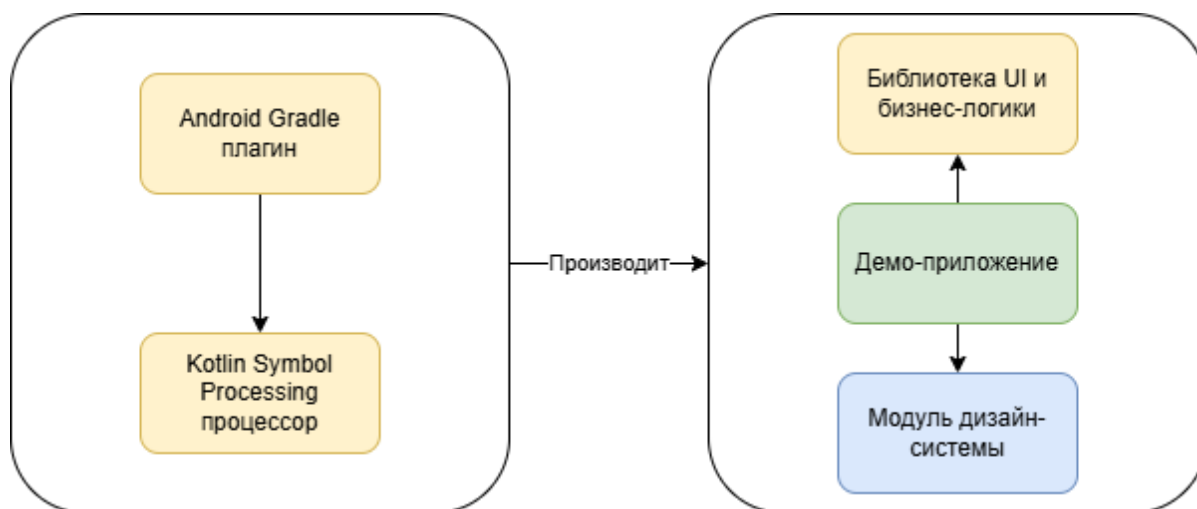


Рисунок 1 – Модульная архитектура плагина и демо-приложения

Итоговый плагин был реализован на языке программирования Kotlin [5]. Данный плагин успешно генерирует демо-приложение для заданного модуля дизайн-системы. Это позволило сократить время создания демо-приложения с приблизительно двух дней до двух часов.

ЛИТЕРАТУРА

1. Что такое дизайн-система, зачем она нужна и какими отечественными решениями вдохновиться. [Электронный ресурс] Режим доступа: https://habr.com/ru/companies/yandex_praktikum/articles/875034/
2. Write Gradle plugins. [Электронный ресурс] Режим доступа: <https://developer.android.com/build/extend-agp>
3. Kotlin Symbol Processing API. [Электронный ресурс] Режим доступа: <https://kotlinlang.org/docs/ksp-overview.html>
4. Create an Android library. [Электронный ресурс] Режим доступа: <https://developer.android.com/studio/projects/android-library>
5. Kotlin Programming Language. [Электронный ресурс] Режим доступа: <https://kotlinlang.org/>

УДК 004.42

А.А. Рыкова (4 курс бакалавриата),
Р.А. Золотарев, Г.В. Смородников (2 курс магистратуры),
А.В. Самочадин., к.т.н., доцент

РАЗРАБОТКА СРЕДСТВ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ДЛЯ ОБЛАЧНЫХ ХРАНИЛИЩ

Облачные хранилища обеспечивают гибкий доступ к вычислительным и хранимым ресурсам, но их эффективное распределение остается сложной задачей из-за изменяющихся нагрузок и разнообразных запросов. Непредсказуемость входных заявок на распределение новых ресурсов и необходимость стабильной работы облачного хранилища требуют динамичных подходов. Без адаптивных механизмов может снизиться эффективность системы и возникнуть перегрузки. Важно разработать методику для эффективного распределения ресурсов с учетом состояния системы и прогнозируемых потребностей.

Эффективное управление ресурсами в облачных хранилищах требует анализа поступающих заявок и выбора оптимального метода их распределения. При низкой нагрузке традиционные алгоритмы работают стабильно, но с увеличением загруженности системы их эффективность снижается. В таких условиях ключевую роль играет прогнозирование: предсказательная модель [1] оценивает загрузку на ближайший час, предоставляя данные для динамической адаптации алгоритмов распределения ресурсов. Как показано в работе [2], использование предсказательных моделей позволяет улучшить балансировку нагрузки и сократить время отклика системы.

Интеграция прогнозных моделей на основе LSTM помогает не только предвидеть изменение нагрузки, но и гибко корректировать стратегии распределения ресурсов [3]. Такой подход позволяет системе адаптироваться в реальном времени: при низкой нагрузке используются более простые методы, а при росте нагрузки активируются прогнозные алгоритмы. Это сохраняет баланс между вычислительной эффективностью и сложностью расчетов, повышая общую устойчивость системы и эффективность использования ресурсов [4].

Целью работы является разработка системы динамического распределения ресурсов, которая анализирует статистику заявок и загруженность системы, выбирая наиболее эффективные алгоритмы. В условиях низкой нагрузки используются традиционные методы, а при высокой – учитываются результаты модели прогнозирования нагрузки, что потенциально позволяет предотвращать перегрузки.

Для достижения цели необходимо: разработать стенд для обработки заявок и запуска алгоритмов; реализовать традиционные методы распределения; создать алгоритм скоринга с учетом прогнозируемой нагрузки; собирать и анализировать статистику заявок; разработать API для управления распределением; визуализировать работу алгоритмов и оценку качества распределения, произвести качественную оценку работы алгоритмов распределения на основе введенных метрик.

Стенд представляет собой клиент-серверное веб-приложение, реализованное с применением микросервисной архитектурой. Серверная часть моделирует работу облачной инфраструктуры, обрабатывает входные данные и взаимодействует с сервисами прогнозирования и распределения ресурсов. Клиентская часть предоставляет интерфейс для настройки параметров и анализа результатов. сценариям тестирования. На рисунке 1 представлена высокоуровневая схема работы системы прогнозирования и распределения ресурсов.

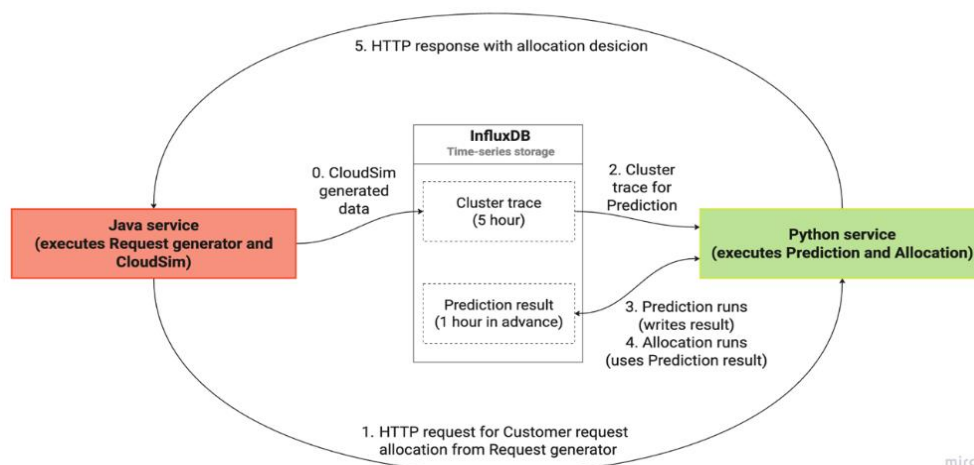


Рисунок 1 – Высокоуровневая схема системы

Java-сервис реализован на Spring Boot WebFlux [4] и отвечает за генерацию заявок и моделирование работы облачной инфраструктуры. Python-сервис разработан на FastAPI [5] и включает модули прогнозирования нагрузки и распределения ресурсов.

Модуль прогнозирования оценивает нагрузку на систему на ближайший час, используя временные ряды. Модуль распределения анализирует входные данные, текущее состояние системы и выбирает подходящий алгоритм. При высокой загрузке и неоднородности поступающих заявок используется алгоритм скоринга, который присваивает каждому серверу очки на основе его характеристик, текущего состояния и прогноза нагрузки. В результате выбирается сервер с наибольшим количеством очков.

Микросервис Executor выполняет финальное распределение виртуальных машин, размещая их на физических серверах в соответствии с решением модуля распределения. Он получает данные из InfluxDB, рассчитывает скоринговый балл с учетом метрик системы и передает решение о размещении ресурсов.

Вся информация о состоянии системы и поступающих заявках сохраняется в InfluxDB, что обеспечивает эффективную работу с временными рядами [6].

ЛИТЕРАТУРА

1. Muhammad F., Adnan A., Muhammad S., Naeem A. "Resource Allocation Techniques in Cloud Computing: A Review and Future Directions", Elektronika ir Elektrotechnika, 2020.
2. Zhou, L., et al. "Dynamic Resource Allocation with Predictive MinMin Algorithm," Future Generation Computer Systems, 2019.
3. Wang, J., Li, X., Zhang, Y. "Enhancing Cloud Computing Efficiency Through Optimized Resource Allocation Techniques for Load Balancing Using LSTM Machine Learning Algorithm," Journal of Cloud Computing, 2022.

4. WebFlux Documentation. [Электронный ресурс] Режим доступа: <https://docs.spring.io/spring-framework/reference/web/webflux.html>
5. FastAPI Documentation. [Электронный ресурс] Режим доступа: <https://fastapi.tiangolo.com>
6. InfluxDB: Time-Series Database for Cloud Monitoring, InfluxData Documentation. [Электронный ресурс] Режим доступа: <https://docs.influxdata.com/influxdb/v2>

УДК 004.6

Д.В. Сергеев (4 курс бакалавриата),
К.А. Каширин (2 курс магистратуры),
А.В. Самочадин, к.т.н., доцент

СРЕДСТВА ГЕНЕРАЦИИ СИНТЕТИЧЕСКИХ ДАННЫХ ДЛЯ ФОРМИРОВАНИЯ ДАТАСЕТА ДЛЯ РАБОТЫ С ОБЛАЧНЫМИ ВЫЧИСЛЕНИЯМИ

С развитием облачных вычислений и внедрением алгоритмов машинного обучения (МО) для прогнозирования нагрузки (Prediction) и распределения ресурсов (Allocation) возникает проблема недостаточности исторических данных. Их объем или разнообразие часто ограничены, а доступ затруднен из-за конфиденциальности данных или их стоимости. Это критично для алгоритмов прогнозирования и распределения, где неточности в предсказаниях и управлении ресурсами приводят к финансовым потерям. Результаты исследований доказывают, что дефицит исторических данных серьезно снижает эффективность алгоритмов машинного обучения [1], поэтому важно дополнять их синтетическими данными, близкими к реальным ситуациям.

Генерация синтетических данных представляет собой сложную задачу, требующую учета структуры реальных данных, их статистических свойств (например, распределений) и особенностей применения в облачных средах. Разработанный инструмент должен обеспечивать создание данных, пригодных для моделирования нагрузок, анализа производительности и обучения алгоритмов в условиях облачных вычислений. Такие данные должны воспроизводить ключевые сценарии работы облачных кластеров, включая периоды пиковых нагрузок, сезонные колебания.

Целью данной работы является разработка средств генерации синтетических данных, предназначенных для формирования датасета, который будет использоваться в задачах облачных вычислительных систем. Эти средства должны обеспечивать создание реалистичных данных для последующей обработки и анализа в облачной среде.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Исследование возможностей генерации синтетических данных с использованием современных инструментов, таких как библиотеки для статистического моделирования (например, NumPy [2], SciPy [3]), фреймворки генеративных моделей (например, GANs [4] или VAE [5]).
2. Разработка архитектуры средств генерации синтетических данных.
3. Реализация алгоритмов моделирования данных с учетом характеристик облачных вычислений (например, распределений [6] нагрузки)
4. Интеграция сгенерированных данных в облачную инфраструктуру для тестирования.
5. Валидация качества и применимости синтетических данных.

Предполагается, что исходный датасет состоит из списка параметров, таких как использование CPU и памяти, временной метки и других параметров, связанных с пропускной способности сети. Используя разработанный инструмент, проводится анализ, позволяющий определить закон распределения параметров (например, нормальное, экспоненциальное), выявить зависимости между ними (например, корреляция между запросами и использованием ресурсов), а также рассчитать другие статистические характеристики, такие как среднее значение, дисперсия и аномалии.

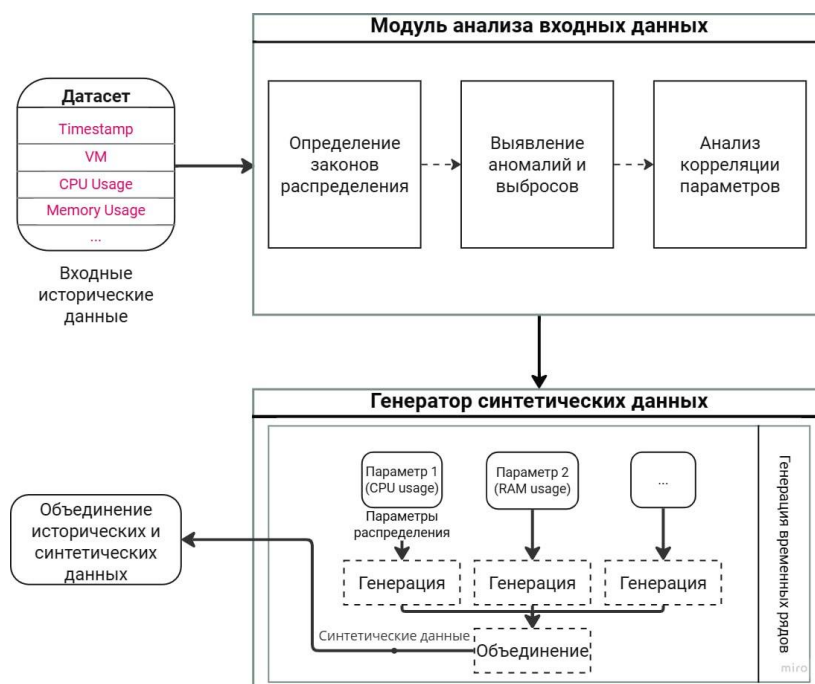


Рисунок 1 – Архитектура системы

Архитектура проекта представлена на рисунке 1, она включает 4 основных блока: модуль источника данных (Датасет), модуль анализа входных данных, который определит к каким распределениям относятся данные, модуль генерации синтетических данных, который по параметрам, полученным от предыдущего модуля, будет генерировать данные о CPU, RAM и другие, модуль отвечающий за объединение исходных исторических данных с сгенерированными синтетическими данными.

ЛИТЕРАТУРА

1. Machine learning in it: opportunities and challenges [Электронный ресурс] Режим доступа <https://it-vacancies.ru/blog/machine-learning-in-it-opportunities-and-challenges/>
2. NumPy. [Электронный ресурс] Режим доступа: <https://numpy.org/doc/>
3. SciPy. [Электронный ресурс] Режим доступа: <https://docs.scipy.org/doc/scipy/>
4. GANs. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Генеративно-состязательная_сеть
5. VAE. [Электронный ресурс] Режим доступа: <https://pyro.ai/examples/vae.html>
6. Вероятностные распределения. [Электронный ресурс] Режим доступа: <https://education.yandex.ru/handbook/ml/article/veroyatnostnye-raspredeleniya>

УДК 004.8

А.А. Сафронов (4 курс бакалавриата),
С.Э. Сараджишвили, к.т.н., доцент

ОЦЕНКА ПРИМЕНИМОСТИ ГИБРИДНОГО ПОДХОДА К АНАЛИЗУ ЛОГОВ: СПРАВНЕНИЕ SUPERVISED, SEMI-SUPERVISED И UNSUPERVISED МЕТОДОВ

Введение. Анализ лог-файлов является важным фактором в обеспечении отказоустойчивости распределенных систем. Для обнаружения аномалий применяют глубокие нейросетевые модели, однако их производительность во многом определяется наличием размеченных данных. Supervised-методы дают высокую точность, но требуют предварительной разметки, semi-supervised частично уменьшают эту зависимость, а unsupervised-алгоритмы подходят для выявления новых типов аномалий. Настоящее исследование рассматривает гибридный подход, сочетающий данные стратегии, с

фокусом на анализе HDFS-логов [1].

Основная часть. В ходе экспериментов применены следующие архитектуры: supervised (CNN, LogRobust, NeuralLog), semi-supervised (PLELog) и unsupervised (DeepLog, LogAnomaly, LogBERT). Изучено, как на точность влияет предобработка логов, выбор схем разбиения (fixed, sliding, session windows) и комбинирование моделей. Для оценки качества использованы показатели Precision, Recall и F1-Score, а также рассмотрены вычислительная сложность и устойчивость к шумовым данным [1].

Выводы. Применение гибридного метода показывает потенциальное улучшение точности и адаптивности лог-аналитики. Полученные результаты указывают на наиболее эффективные комбинации supervised, semi-supervised и unsupervised-подходов в различных условиях. В дальнейшем перспективными задачами станут разработка адаптивных гибридных стратегий, включение Explainable AI (XAI) и масштабное тестирование в условиях высоких нагрузок [2].

ЛИТЕРАТУРА

1. Le V.-H., Zhang H. Log-based Anomaly Detection with Deep Learning: How Far Are We? // 2022 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). – 2022. – Pp. 1–12. – DOI: 10.1145/3510003.3510155.
2. Luz H., Peace P., Luz A., Joseph S. Explainable AI in CI/CD Deployment Orchestration // ResearchGate. – 2024. – December. – URL: <https://www.researchgate.net/publication/387556307>.

УДК 004.514

М.А. Сидоркина (3 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОЦЕНКИ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА НА ОСНОВЕ АНАЛИЗА СОВМЕСТНЫХ ДВИЖЕНИЙ ГЛАЗ И МЫШИ

Современные веб-приложения предъявляют высокие требования к качеству пользовательского опыта. Традиционные методы тестирования интерфейсов, основанные только на анализе кликов или перемещений мыши, не всегда позволяют объективно оценить восприятие информации пользователем [3]. Технологии трекинга взгляда, позволяющие анализировать визуальное внимание пользователя, открывают новые возможности для исследований в сфере UX [2, 3]. Однако их применение в веб-тестировании зачастую ограничено высокой стоимостью и сложностью интеграции аппаратных решений [5], что делает их малодоступными для массового использования.

Актуальность проблемы обусловлена необходимостью создания инструмента, способного объединить данные трекинга взгляда и движений курсора для комплексного анализа пользовательского опыта. Такой подход позволяет не только оценить, какие элементы интерфейса привлекают внимание пользователя, но и выявить расхождения между визуальным восприятием и явными действиями, например кликами, что может свидетельствовать о когнитивном диссонансе или проблемах в дизайне интерфейса [1, 4].

Целью данной работы является разработка инструмента для анализа поведения пользователя и тестирования веб-приложений на основе синхронного трекинга взгляда и мыши. Разработанный прототип позволит выявлять проблемные участки интерфейса, вызывающие отклонения в поведении пользователя, например, свидетельствующие о когнитивном диссонансе, как описано в [2].

Для достижения поставленной цели необходимо решение следующих *задач*:

1. Исследование современных методов трекинга взгляда и движений курсора в контексте веб-взаимодействия.
2. Определение критериев оценки синхронности и расхождений движений глаз и мыши, а также разработка методологии их трактования.

3. Создание алгоритмов анализа, структурирования и визуализации данных о движениях взгляда и курсора.
4. Разработка прототипа приложения для сбора, обработки и анализа данных о движениях глаз и мыши.
5. Проведение тестирования разработанного прототипа и оценка его эффективности в условиях реального использования.

В рамках работы проведён обзор существующих аппаратных и программных решений для трекинга взгляда. Аппаратные методы, например, Tobii Eye Tracker, обеспечивают высокую точность фиксации микродвижений глаз с погрешностью менее 0.5° , однако требуют специализированного оборудования и дорогостоящей интеграции [5]. Программные решения, такие как WebGazer.js, реализуют трекинг взгляда посредством стандартных веб-камер с погрешностью порядка 100–200 пикселей [6], что делает их доступными для массового тестирования веб-приложений. В ходе проведения сравнительного анализа в качестве основного инструмента трекинга взгляда был выбран WebGazer.js.

Архитектура приложения для тестирования веб-приложений с помощью трекинга взгляда и мыши представлена на рисунке 1. Модуль сбора данных использует библиотеку WebGazer.js для трекинга взгляда через веб-камеру и браузерные API для фиксации событий мыши. Эти данные синхронизируются по временным меткам с помощью механизма EventEmitter. Сервис хранения организует данные в сессии. API Gateway предоставляет унифицированный интерфейс для работы с сервисами трекинга и хранения.

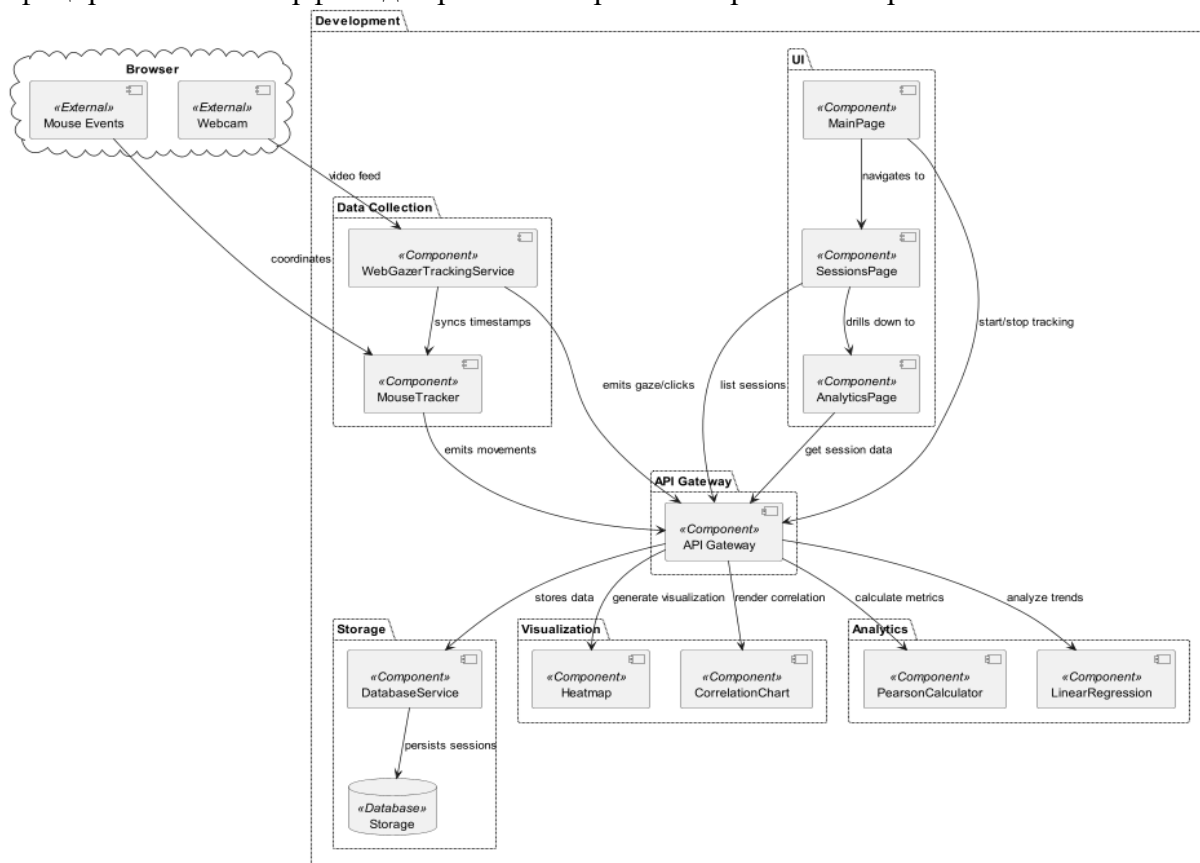


Рисунок 1 – Архитектура приложения

В приложении реализован аналитический модуль, который преобразует результаты сессий трекинга в количественные метрики для оценки пользовательского опыта. Кроме того, предусмотрена графическая визуализация для наглядного представления собранных данных.

Подход реализован на языке TypeScript, что обеспечивает строгую типизацию и улучшенную поддержку модульности кода, повышая надёжность и удобство сопровождения проекта. Для разработки пользовательского интерфейса использовалась библиотека React.

React позволяет создавать динамичные и отзывчивые компоненты, а также интегрировать инструменты визуализации данных, такие как тепловые карты и графики корреляции.

ЛИТЕРАТУРА

1. Chen X., Epps J., Ruiz N. Eye activity as a measure of human mental effort in HCI // Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI '11). – New York: ACM, 2011. – P. 315–318. – DOI 10.1145/1943403.1943454.
2. Kurzhals K., Hlawatsch M., Seeger C., Weiskopf D. Gaze Stripes: Image-Based Visualization of Eye Tracking Data // IEEE Transactions on Visualization and Computer Graphics. – 2016. – Vol. 22. – No 1. – P. 1005–1014. – DOI 10.1109/TVCG.2015.2468091.
3. Bergstrom J. R., Schall A. Eye Tracking in User Experience Design. – Waltham: Elsevier, 2014. – 368 pp.
4. Steil J., Huang M. X., Bulling A. Fixation detection for head-mounted eye tracking based on visual similarity of gaze targets // Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18). – New York: ACM, 2018. – P. 1–9. – DOI 10.1145/3204493.3204538.
5. Tobii Eye Tracker 5. [Электронный ресурс] Режим доступа: <https://gaming.tobii.com/product/eye-tracker-5/> (дата обр. 24.02.2025).
6. WebGazer. [Электронный ресурс] Режим доступа: <https://github.com/brownhci/WebGazer> (дата обр. 24.02.2025).

УДК 004.8

М.В. Стопченко (3 курс бакалавриата),
В.Э. Шмаков, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ УМНОГО ДОМА С ГОЛОСОЫМ УПРАВЛЕНИЕМ

Современные технологии постоянно развиваются, и с каждым годом мы видим все больше и больше умных устройств, которые облегчают нашу жизнь и делают ее более комфортной. Одной из наиболее популярных областей развития подобных технологий является «умный дом».

«Умный дом» — это система автоматизации жилого пространства, которая позволяет управлять устройствами, освещением, климатическими системами, и многими другими аспектами жизни через единую систему управления.

Одним из наиболее востребованных способов автоматизации является голосовое управление. Оно позволяет контролировать устройства и системы с помощью голосовых команд, что делает управление простым и удобным [1].

Разработка полноценной системы «умного дома» с интеграцией голосового управления имеет большой потенциал для улучшения жизни людей. Такие системы позволяют автоматизировать многие повседневные задачи, такие как управление освещением, отоплением, кондиционированием воздуха, безопасностью дома и т.д. Это особенно важно для людей с ограниченными возможностями, которым может быть сложно управлять устройствами вручную [2].

Интеграция голосового управления в систему «умного дома» также позволяет создавать новые возможности для оптимизации жилищных условий. Например, система может автоматически регулировать уровень освещения и температуру в зависимости от времени суток и погодных условий. Это не только улучшает комфорт жизни, но и позволяет снизить расходы на энергию.

Кроме того, система «умного дома» может быть полезной не только для жилых помещений, но и для коммерческих объектов, таких как офисы, магазины, рестораны и т.д. Умная система может помочь улучшить работу и повысить эффективность бизнес-процессов, а также снизить расходы на энергию и обслуживание зданий.

Выбор технологий и средств разработки для системы с интеграцией голосового управления был произведен с помощью анализа существующих решений в области «умных домов» и голосового управления. Были рассмотрены существующие решения в области голосового управления, такие как Amazon Alexa [3], Google Assistant и Apple Siri [4], Яндекс Алиса [5]. Это позволяет создать эффективную и надежную систему, которая соответствует современным трендам и потребностям пользователей.

Для достижения цели по созданию системы «умный дом» необходимо решение следующих задач:

1. Обзор существующих средств реализации системы «умный дом».
2. Проведение сравнительного анализа этих средств.
3. Разработка функциональности системы.
4. Реализация и тестирование системы.

Выполнение перечисленных задач на первом этапе предоставляет систему «умного дома» для удобства использования которой, на втором этапе, необходимо внедрить голосовое управление.

Для достижения цели по внедрению голосового управления в систему «умный дом» необходимо решение следующих задач:

1. Обзор существующих средств для интеграции голосового управления.
2. Проведение сравнительного анализа этих средств.
3. Выбор наиболее подходящего средства управления с помощью голосовых команд.
4. Внедрение этого средства в систему «умного дома» для использования голосового управления.

Таким образом, разработка системы «умного дома» с интеграцией голосового управления является актуальной и перспективной задачей, которая может улучшить качество жизни людей и создать новые возможности для автоматизации и оптимизации жилищных и коммерческих условий. Разработка такой системы требует тщательного выбора технологий и средств разработки, чтобы обеспечить высокую эффективность и надежность работы.

ЛИТЕРАТУРА

1. «Умный дом»: 5 технологий будущего, URL: <http://www.lookatme.ru/mag/live/future-research/194385-smart-home>
2. Что такое «Умный дом»? // URL: <https://www.intelvision.ru/blog/what-issmarthome>
3. «Amazon Alexa». Amazon, 2024. Amazon Alexa, URL: <https://www.amazon.com/alexa>
4. «Google Assistant». Google, 2024. Google Assistant, URL: <https://assistant.google.com/>
5. «Яндекс Алиса». Яндекс, 2024. Яндекс Алиса, URL: <https://alice.yandex.ru>

УДК 004.453

К.В. Сыпачёв (2 курс магистратуры),
В.А. Пархоменко, ст. преподаватель

ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ ВОЗМОЖНОСТЕЙ LLM ДЛЯ ПОИСКА СЕМАНТИЧЕСКИХ ОШИБОК

С развитием больших языковых моделей (далее – LLM, Large Language Models) появилась возможность исследовать их потенциал для тестирования программного кода и поиска как функциональных, так и семантических ошибок.

Цель данной работы — исследовать возможности YandexGPT, GigaChat и O1 для задач тестирования кода, выделить и оценить их сильные и слабые стороны и определить, могут ли эти модели использоваться как самостоятельные инструменты для тестирования и выявления семантических ошибок или же они эффективны только как дополнение к существующим методам. Для достижения цели решены следующие задачи:

1. Изучить возможности LLM в задачах тестирования программного обеспечения.

2. Провести тестирование с помощью моделей YandexGPT, GigaChat и O1 на наборах данных с семантическими ошибками.

3. Сравнить результаты тестирования по ключевым параметрам, а именно точность обнаружения ошибок, скорость обработки.

4. Выявить сильные и слабые стороны использования LLM в задачах тестирования кода.

LLM сравниваются в таблице 1 по критериям: разработчик; год выпуска модели; количество токенов в бесплатном режиме; количество токенов в платном режиме; цена доплаты за 1000 токенов после превышения бесплатного лимита; время отклика, т.е. время, которое требуется LLM на то, чтобы отреагировать на введенный пользователем запрос; дата окончания знаний модели, т.е. время, после которой LLM перестает изучать новую информацию.

Таблица 1 – Описательное сравнение LLM, участвующих в эксперименте

Критерий	GigaChat MAX	YandexGPT	O1
Разработчик	Сбер	Яндекс	OpenAI (запрещена в РФ)
Год выпуска модели	2023	2024	2024
Кол-во токенов (free)	50 000	8 000	-
Кол-во токенов (paid)	1 000 000	32 000	50 запр. в неделю
Доплата за 1000 ток.	1,95 руб с НДС	1,2 руб с НДС	
Время отклика	~4-5с/	Зависит от режима	Увеличенное
Дата окончания обучения	-	-	Октябрь 2023

Для исследования выбраны три программы. *Первая* выбрана из [1] и представляет является Java-кодом с использованием обработки исключений и цикла для работы с массивом целых чисел. Ошибку в строке 14. *Вторая* «хрестоматийная» программа DeleteTag Андреаса Зеллера [2] должна удалять HTML-теги из входящей строки, оставляя лишь текст без тега. Ошибка заключается в логическом условии на строках 12–14. *Третья* программа AccountSetupBasics.java [3] обширна и содержит ошибку в строке 277. Так, в части кода, где проверяется пароль используется переменная «UserName», однако там должна стоять переменная «password». Результаты экспериментов приведены в таблице 2, где «Р» имеет значения 1, 2 или 3, соответствующие программам.

Таблица 2 – Описательное сравнение LLM, участвующих в эксперименте поиска ошибок

Р	Модель	Входных ток.	Выходных ток.	Время отклика (с)	Ошибок найдено
1	GigaChat MAX	229	908	4	2
1	YandexGPT 4	229	988	3	3
1	o1	229	2057	10	2
2	GigaChat MAX	194	170	2	0
2	YandexGPT 4	194	475	5	0
2	o1	194	576	10	1
3	GigaChat MAX	6056	0	0	0
3	YandexGPT 4	6056	280	3	0
3	o1	6056	2206	10	0
3	GigaChat MAX	204	653	3	1
3	YandexGPT 4	204	280	3	0
3	o1	204	204	2	1

Результаты экспериментов показали, что каждая из моделей имеет хотя бы одно преимущество перед другими. YandexGPT 4 имеет самую низкую доплату за 1000 токенов. Количество токенов в GigaChat в платной версии достигает 1000000. А o1 каждый раз перед ответом дольше обдумывает свое решение. LLM могут находить семантические ошибки в небольших программах. Однако возникают трудности, когда код становится большим.

ЛИТЕРАТУРА

1. Leveraging Stack Traces for Spectrum-based Fault Localization in the Absence of Failing Tests. — 2024. — ([Accessed 26-12-2024]). <https://zenodo.org/records/11062413>.

2. Zeller, Andreas. Why programs fail: a guide to systematic debugging. Morgan Kaufmann, 2009.
3. AccountSetupBasics.java. — ([Accessed 27-12-2024]).
<https://android.googlesource.com/platform/packages/apps/Email/+e3def6c/src/com/android/email/activity/setup/AccountSetupBasics.java>.

УДК 004.4

Е.Е. Тимошин (4 курс бакалавриата),
Н.В. Воинов, к.т.н., доцент

СРАВНЕНИЕ ПАТТЕРНОВ ОТКАЗОУСТОЙЧИВОСТИ НА ПРИМЕРЕ РАЗРАБОТКИ МИКРОСЕРВИСНЫХ ПРИЛОЖЕНИЙ

Современные информационные системы всё чаще строятся на основе микросервисной архитектуры, которая предоставляет ряд преимуществ, таких как модульность, гибкость и масштабируемость [1-3]. Однако при переходе от монолитных приложений к микросервисам возникают новые вызовы, особенно в области обеспечения отказоустойчивости. С увеличением количества сервисов растёт и сложность их взаимодействия, что делает систему уязвимой к каскадным сбоям, перегрузкам и временным сбоям в сетевом соединении.

Одной из ключевых проблем является необходимость обеспечения надёжности при коммуникации между сервисами [4]. Часто сбой в одном из сервисов могут привести к цепной реакции, затронув другие компоненты системы и в конечном итоге привести к деградации всей системы. Существующие решения, такие как простые механизмы повторных попыток (retry) или настройка таймаутов, не всегда достаточны для надёжной защиты. Более того, они могут привести к избыточной нагрузке на систему, если не сопровождаются дополнительными мерами контроля [5].

Целью данной работы является анализ и применение паттернов отказоустойчивости Circuit Breaker, Bulkhead и Timeout & Retry для повышения устойчивости микросервисной архитектуры на платформе Java. Эти паттерны позволяют справляться с различными типами сбоев, предотвращать перегрузки и ограничивать влияние отказов отдельных компонентов на всю систему.

Для достижения поставленной цели необходимо решить следующие задачи:

- Исследовать существующие подходы к обеспечению отказоустойчивости в микросервисных архитектурах.
- Проанализировать особенности применения паттернов Circuit Breaker, Bulkhead и Timeout & Retry в контексте платформы Java.
- Разработать и реализовать прототип системы с использованием данных паттернов.
- Оценить эффективность предложенного решения на основе проведения нагрузочных тестов.

В ходе работы будут применены следующие технологии:

- Язык программирования Java для реализации серверной части приложения [6].
- Фреймворк Spring Boot для создания и конфигурации микросервисов.
- Библиотека Resilience4j для внедрения паттернов отказоустойчивости [7].
- СУБД PostgreSQL для управления базами данных.
- Платформа Docker для контейнеризации сервисов.
- Фреймворк JUnit и библиотека Testcontainers - для проведения интеграционных и нагрузочных тестов [8].

Ожидается, что результаты работы позволят продемонстрировать, как применение данных паттернов способствует повышению отказоустойчивости, уменьшению времени простоя системы и предотвращению каскадных отказов. В качестве практической реализации будут использоваться инструменты Resilience4j и Spring Cloud, которые предоставляют широкие возможности для внедрения данных решений в современные микросервисные приложения.

ЛИТЕРАТУРА

1. Barabanov, A. V. Method for collecting architecture security-specific information for microservice-based systems / A. V. Barabanov // *Secure Information Technologies*, 06–07 апреля 2021 года, 2021. – P. 27-31. – EDN CZJFGD.
2. Селезнев, В. А. Разработка микросервисов взаимодействия с пользователями в рамках биллинговой системы / В. А. Селезнев, Н. В. Воинов // *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. 202-204. – EDN DBYGFH.
3. Дроздов, С. В. Разработка мобильного приложения учета личных финансов с применением микросервисной архитектуры / С. В. Дроздов, Н. В. Воинов // *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. 31-33. – EDN PVADBM.
4. Скуратов, Н. Н. Архитектура системы для автоматизированного обнаружения уязвимостей микросервисных архитектур / Н. Н. Скуратов, Д. С. Левшун // *Региональная информатика и информационная безопасность : Сборник трудов Санкт-Петербургской международной конференции и Санкт-Петербургской межрегиональной конференции*, Санкт-Петербург, 23–25 октября 2024 года. – Санкт-Петербург: Санкт-Петербургское Общество информатики, вычислительной техники, систем связи и управления, 2024. – С. 133-135. – EDN COHIAW.
5. Punithavathy, E. Auto retry circuit breaker for enhanced performance in microservice applications / E. Punithavathy, N. Priya // *International Journal of Electrical and Computer Engineering*. – 2024. – Vol. 14, No. 2. – P. 2274. – DOI 10.11591/ijece.v14i2.pp2274-2281. – EDN LIHUL.
6. Квитка, С. А. Разработка программного обеспечения серверной среды разработки на языке Java / С. А. Квитка // *Проблемы управления в социально-экономических и технических системах : Материалы XIX Международной научно-практической конференции*, Саратов, 13–14 апреля 2023 года. – Саратов: ИЦ "Наука", 2023. – С. 211-214. – EDN ZSENSA.
7. A declarative approach and benchmark tool for controlled evaluation of microservice resiliency patterns / C. M. Aderaldo, T. M. Costa, D. M. Vasconcelos [et al.] // *Software - Practice and Experience*. – 2024. – DOI 10.1002/spe.3368. – EDN GVEHJW.
8. Shukla, S. Streamlining Integration Testing with Test Containers: Addressing Limitations and Best Practices for Implementation / S. Shukla // *International Journal of Latest Engineering and Management Research (IJLEMR)*. – 2023. – Vol. 9, No. 3. – P. 19-26. – DOI 10.56581/ijlemr.8.3.19-26. – EDN SJCSFA.

УДК 004.93

И.С. Тампио (4 курс бакалавриата),
И.А. Селин, к. т. н., доцент

СИСТЕМА МОНИТОРИНГА БЕЗОПАСНОСТИ РАБОЧЕГО ПРОЦЕССА С ПРИМЕНЕНИЕМ ТЕХНОЛОГИЙ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Работа на производстве или в промышленности неразрывно связана с риском для жизни. По оперативным данным Роструда за 2022 год было зарегистрировано 5563 несчастных случаев на производствах по всей стране [1]. Современные стандарты охраны труда и промышленной безопасности требуют строгого соблюдения правил использования средств индивидуальной защиты (далее – СИЗ) на рабочих местах. Несмотря на наличие нормативных требований, контроль соблюдения правил ношения СИЗ зачастую осуществляется вручную, что приводит к высоким затратам времени и ресурсов, а также повышает вероятность человеческих ошибок и халатности. Сложность организации эффективного мониторинга

усиливается в условиях крупных предприятий и объектов с большой численностью работников и высокой интенсивностью рабочих процессов.

Один из способов автоматизировать процесс контроля ношения СИЗ – использовать технологии компьютерного зрения в совокупности с системой видеонаблюдения [2,3]. Внедрение такого подхода позволяет не только осуществлять полный контроль правильности применения СИЗ, но и снизить количество нарушений [4].

В открытом доступе имеются реализации моделей компьютерного зрения для детекции ношения СИЗ, однако отсутствуют реализации систем «от начала до конца», в которых бы был реализован процесс получения видеопотока с нескольких камер и отправка сообщений о нарушениях СИЗ в отдельную систему мониторинга в условиях промышленной эксплуатации [6,7]. Помимо детекции СИЗ, такая система должна справляться со следующими проблемами: предотвращать дополнительные срабатывания при определении нарушений СИЗ на одном и том же человеке, эффективно предсказывать наличие СИЗ для нескольких видеопотоков одновременно.

Таким образом, *целью* данной работы является разработка прототипа с открытым исходным кодом системы мониторинга ношения СИЗ «от начала до конца» за счёт обучения и оптимизации модели распознавания наличия СИЗ и дальнейшей «упаковки» этой модели в среду промышленной эксплуатации.

Для достижения этой цели необходимо решение следующих *задач*:

1. Обзор существующих подходов реализации автоматического контроля СИЗ с использованием технологий компьютерного зрения.
2. Обзор существующих подходов реализации систем анализа видеопотока.
3. Проведение сравнительного анализа найденных подходов.
4. Предложение архитектуры прототипа системы анализа видеопотока для контроля ношения СИЗ в промышленной среде.
5. Реализация данного подхода.
6. Демонстрация работоспособности прототипа.

Примером архитектуры системы видеоаналитики в промышленной среде может послужить следующая схема:

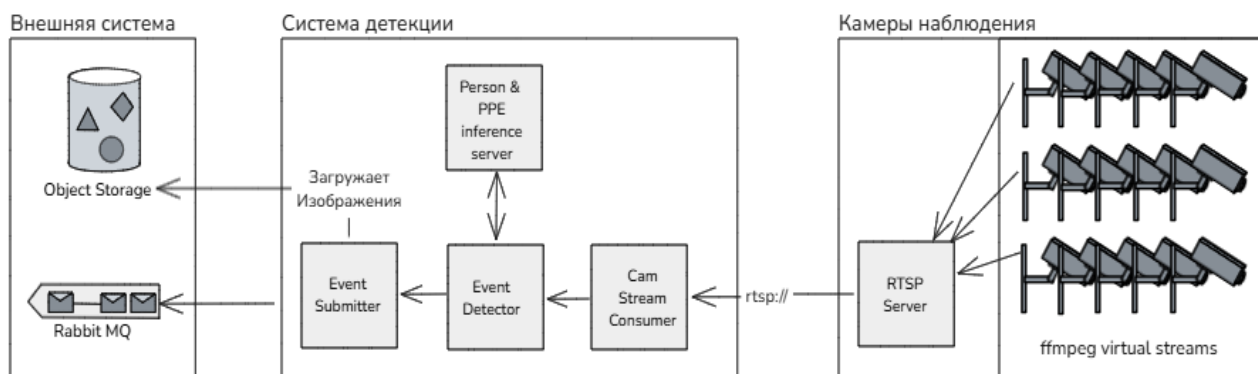


Рисунок 1 – Пример архитектуры системы в промышленной эксплуатации

Данная архитектура состоит из трёх основных компонентов – камер наблюдений, чей поток доступен по протоколу RTSP, система детекции, которая отсылает сообщения о нарушениях СИЗ в брокер сообщений внешней системы и файлы изображений нарушений СИЗ в объектное хранилище внешней системы.

Система детекции состоит в свою очередь из нескольких компонентов. Cam Stream Consumer – отвечает за получение видеопотоков по протоколу RTSP. Event Detector осуществляет логику обнаружения событий на основе предсказаний. Person & PPE inference server осуществляет предсказание событий на видеопотоке. Event Submitter осуществляет отправку событий нарушений СИЗ во внешнюю систему.

Подход был реализован в программном средстве на языке Python – основном языке,

использовавшегося для разработки проектов машинного обучения в 2024 году [8]. Сервис системы детекции представляет собой Docker-контейнер с приложением, которое получает кадры с камер, предварительно указанных в настройках, и отправляющее события в брокер сообщений RabbitMQ и отсылающее изображения с нарушениями в объектное хранилище.

Алгоритм обнаружения нарушений СИЗ основан на последовательном применении двух моделей машинного обучения: детекции людей и классификации наличия СИЗ. Для детекции использовалась архитектура Mask-RCNN, реализованная в библиотеке Detectron2, для классификации использовалась архитектура EfficientNet, представленная в библиотеке PyTorch. Для трекинга людей использовался алгоритм SORT [9].

ЛИТЕРАТУРА

1. Информация о работе технической инспекции труда профсоюзов в 2022 году. [Электронный ресурс] Режим доступа: <https://fnpr.ru/documents/dokumenty-federatsii/informatsiya-o-rabote-tekhnicheskoy-inspektсии-truda-profsoyuzov-v-2022-godu.html>
2. Godoy D. A. [и др.]. Detection of Personal Protection Elements in a Recycling Plant Using Convolutional Neural Networks Communications in Computer and Information Science / под ред. S. Nesmachnow, L. Hernández Callejo, Cham: Springer Nature Switzerland, 2024. С. 203–215.
3. Wanlin E., Yang Z., Yu J. Detection and Recognition of Personal Protection Equipment Wearing Based on an Improved YOLOv5 Algorithm Nanjing, China: IEEE, 2023. С. 229–232.
4. Проект: «Контроль ношения СИЗ». [Электронный ресурс] Режим доступа: <https://digitalmining.ru/tpost/7480ubuyr1-kontrol-nosheniya-siz>
5. Fang Q. [и др.]. Detecting non-hardhat-use by a deep learning method from far-field surveillance videos // Automation in Construction. 2018. (85). С. 1–9.
6. Barlybayev A. [и др.]. Personal protective equipment detection using YOLOv8 architecture on object detection benchmark datasets: a comparative study // Cogent Engineering. 2024. № 1 (11).
7. ppe-detection – GitHub Topics. [Электронный ресурс] Режим доступа: <https://github.com/topics/pppe-detection>
8. Developer Nation Community. [Электронный ресурс] Режим доступа: <https://www.developernation.net/developer-reports/dn26/>
9. Bewley A. [и др.]. Simple online and realtime tracking Phoenix, AZ, USA: IEEE, 2016. С. 3464–3468.

УДК 004.912

А.А. Ткаченко (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент

ПОДХОД КОНСОЛИДАЦИИ ДАННЫХ ДЛЯ АВТОМАТИЗАЦИИ АНАЛИЗА И УПРАВЛЕНИЯ ИТ-ЛАНДШАФТОМ КОМПАНИИ

В каждой компании существует сложная ИТ инфраструктура, которая требует эффективного управления ИТ ресурсами, критичными для обеспечения бесперебойной работы бизнеса [1]. При этом современные компании сталкиваются с растущей сложностью ИТ-инфраструктуры, включающей сотни конфигурационных единиц (КЕ), распределенных между системами мониторинга, управления активами и сервисами. Ручной сбор и анализ данных приводит к высоким трудозатратам (до 40% времени ИТ-специалистов), рискам ошибок из-за человеческого фактора, а также задержкам в принятии решений.

Целью работы является снижение трудоемкости процесса и повышение точности формирования запросов на изменения ИТ-инфраструктуры [2, 3] за счет подхода интеграции данных о конфигурационных единицах из различных информационных систем (управления активами CMDB, мониторинга SolarWinds, сетевой инвентаризации NetBox и безопасности MaxPatrol), реализованного в программном средстве в контуре компании ООО Газпромнефть-ИТО. На рисунке 1 обозначены этапы работ, где применим программный продукт.

Преимуществом кастомного решения стали низкие эксплуатационные затраты, высокая адаптивность к изменениям в источниках данных и соответствие требованиям безопасности компании. Ключевым преимуществом кастомного инструмента стала его способность адаптироваться к специфике legacy-систем, а также выполнять функции верификатора предоставляемой информации, которые отображены на рисунке 2 в соответствующей нотации.

Разработанное решение базируется на трехслойной архитектуре. Бэкенд реализован на PHP с использованием фреймворка Laravel, выбранного за поддержку паттерна MVC. Интеграция с внешними системами, такими как SolarWinds и NetBox, выполнена через REST API и GraphQL, что позволило унифицировать взаимодействие с разнородными источниками данных. Для работы с иными источниками написаны скрипты с пакетной обработкой. База данных MS SQL Server обеспечивала совместимость с корпоративным решением. Фронтенд реализован классическими инструментами с динамической подгрузкой посредством AJAX. Тестирование покрывает юнит и E2E сценарии через PHPUnit и Selenium с показателем выше 70%. Для автоматизации развертывания применен Jenkins, сокративший время вывода обновлений. Статические анализаторы кода и документация по ГОСТ гарантировали соответствие регуляторным требованиям.

В статье предложен подход к автоматизации управления сложным ИТ-ландшафтом компании, основанный на консолидации данных из разнородных систем. Он демонстрирует эффективность в управлении сложными ИТ-ландшафтами, сочетая гибкость кастомной разработки с требованиями компании. Разработанное программное решение, реализованное с использованием кастомного инструментария, демонстрирует снижение трудозатрат на 20% согласно полученным. Решение валидировано в условиях реальной инфраструктуры компании ООО «Газпромнефть-ИТО», что подтверждает его эффективность в условиях ограничений legacy-систем. Дальнейшие исследования будут направлены на интеграцию методов машинного обучения для прогнозирования рисков изменений и расширение поддержки стандартов.

ЛИТЕРАТУРА

1. Варламов, Д. А. Подход для повышения стабильности ИТ-инфраструктуры на основе информации о ее сбоях / Д. А. Варламов, И. В. Никифоров, С. М. Устинов // 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. 189-191. – EDN AQFWZG.
2. Ивлев, В. А. Актуальность автоматизированной настройки инфраструктуры ит-проекта / В. А. Ивлев, 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. 79-80. – EDN QQAVXT.
3. Ткаченко, А. А. Подход повышения эффективности процесса обработки запросов на инфраструктурные изменения в информационно-технологической среде организации / А. А. Ткаченко, С. М. Устинов, 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. 222-224. – EDN IGTRX.
4. "Configuration Management Database (CMDB) Principles and Practices" - TechNet, 2021
5. "Data Integration Patterns and Practices" - Kimball, 2020
6. Defining Requirements for An Incident Management System: A Case Study - Marko Jäntti; FOURTH INTERNATIONAL CONFERENCE ON SYSTEMS, 2009.

ПРИМЕНЕНИЕ МЕТОДА К-СРЕДНИХ ДЛЯ АНАЛИЗА ЭКСПЛУАТАЦИОННЫХ ХАРАКТЕРИСТИК СУДНА

Метод К-средних является одним из ключевых инструментов в области анализа данных и широко применяется в различных инженерных и научных исследованиях. Данный метод позволяет эффективно разделять данные на кластеры, что помогает аналитикам выявлять закономерности и принимать обоснованные решения. В отличие от иерархических методов, метод К-средних относится к неиерархическим и отличается высокой скоростью работы, что делает его особенно полезным при обработке больших объемов данных [1].

В данной работе рассматривается применение метода К-средних для анализа эксплуатационных характеристик судна. Эксплуатационные характеристики, такие как скорость судна, дедвейт, и прочие играют важную роль в обеспечении эффективности морских перевозок. Для анализа были выбраны следующие параметры: 1) дедвейт, 2) скорость судна, 3) осадка, 4) количество палуб.

Данные для анализа были собраны из внешних источников, включая специализированные веб-сайты. Для автоматизации процесса сбора данных был разработан парсер на языке Python, который извлекал информацию и сохранял ее в базе данных PostgreSQL для дальнейшей обработки. Схема получения и обработки данных представлена на рисунке 1.



Рисунок 1 – Схема получения данных

После сбора данных и их предварительной обработки, в соответствии с собственным алгоритмом, был применен метод К-средних с использованием библиотеки Scikit-learn [2]. Основные характеристики метода, такие как вычислительная сложность, представлены в таблице 1.

Таблица 1 – Характеристики метода к-средних

Алгоритм кластеризации	Вычислительная сложность
Иерархический	$O(n^2)$, где n – число точек
К-средних	$O(n * k * l)$, где n – число точек, k – число кластеров, l – число итераций

Для реализации метода К-средних была разработана программа, использующая языки программирования Python, а также библиотека React и Electron Framework для реализации пользовательского интерфейса [3]. Программа состоит из Backend части, которая выполняет вычисления, и FrontEnd части, представляющей собой настольный интерфейс для визуализации результатов.

После выполнения кластеризации на серверной части формируется набор данных в формате JSON, который передается в пользовательское приложение для отображения. Для визуализации результатов использовались библиотеки Chart.js и/или Recharts. Результат представлен на рисунке 2.

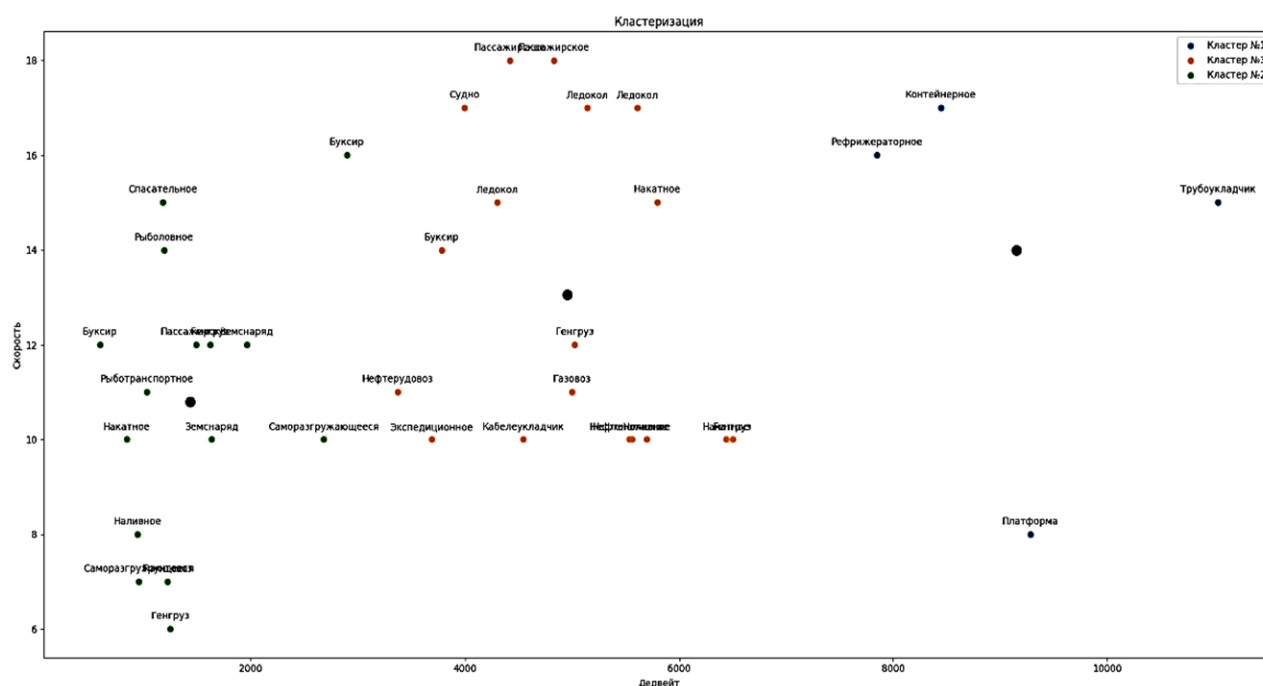


Рисунок 2 – Результаты кластеризации методом К-средних

На основе полученных результатов аналитик может принимать решения, направленные на оптимизацию эксплуатационных характеристик судна. Например, можно выявить закономерности, связанные с скоростью из-за осадки, и предложить меры по её увеличению. Подобные действия позволяют повысить общую эффективность эксплуатации судна и снизить затраты на его обслуживание.

Таким образом, метод К-средних является мощным инструментом для анализа эксплуатационных характеристик судна, позволяющим выявлять ключевые факторы, влияющие на эффективность морских перевозок, и принимать обоснованные решения для их оптимизации.

ЛИТЕРАТУРА

1. Фролов, В.В. Метод расчета числа кластеров для алгоритма k-means. Экономика. Информатика. (том 47, номер 1) [Текст]/ В.В. Фролов, С.Е. Слипченко, О.Ю. Приходько – Белгород: БГНИУ, 2020.
2. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем, 2-е издание [Книга]. – Москва: Диалектика-Вильямс, 2020.
3. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными [Книга]. – Москва: Вильямс, 2017.

BACKEND И FRONTEND СТЕНДА ДЛЯ ИССЛЕДОВАНИЯ АЛГОРИТМОВ ПРОГНОЗИРОВАНИЯ И РАСПРЕДЕЛЕНИЯ РЕСУРСОВ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

Эффективность облачных вычислений зависит от алгоритмов прогнозирования нагрузки и распределения ресурсов. Различие задач, архитектур и входных данных облачного кластера определяет выбор оптимальных алгоритмов. Важно подобрать связку алгоритмов, обеспечивающую точность предсказаний, скорость выполнения, адаптивность к нагрузкам и устойчивость к изменениям [3]. Однако тестирование на реальных облачных вычислениях требует значительных вычислительных ресурсов, времени и финансовых затрат, к тому же потребляет потенциальные вычислительные ресурсы облака [1]. В связи с этим предлагается использовать программный стенд, который позволит подключать разные алгоритмы и проводить исследования на имитационной модели с гибкими настройками архитектуры облака, используемых алгоритмов и собираемых метрик [2].

Целью данной работы является разработка backend и frontend программных модулей стенда для исследования алгоритмов прогнозирования и распределения ресурсов в облачных вычислениях.

Для достижения цели необходимо решить следующие задачи: Разработка API для связи frontend и backend, включая передачу данных и обработку запросов; Реализация модуля обработки данных для взаимодействия с модулем симуляции и генерации синтетических данных; Внедрение механизма сбора и хранения результатов и метрик в ELK-стек (Elasticsearch, Logstash, Kibana) для последующего анализа эффективности алгоритмов; Организация хранения временных рядов в InfluxDB с возможностью выгрузки и анализа исторической информации; Интеграция backend с сервисами предсказания нагрузки и распределения ресурсов; Разработка UI/UX пользовательского интерфейса, включая прототипирование и реализацию интуитивно понятного дизайна; Создание интерфейсов для конфигурации моделирования, управления симуляцией и визуализации результатов.

Архитектура проекта представлена на рисунке 1.

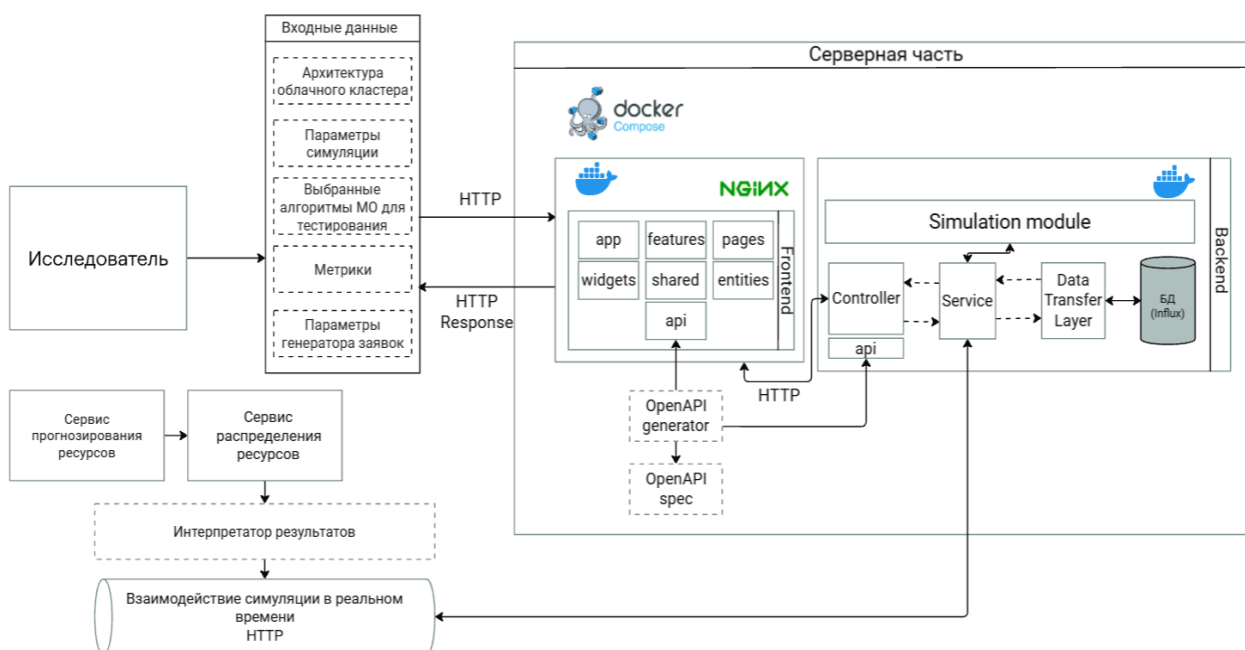


Рисунок 1 – Архитектура стенда

Стенд реализован в виде клиент-серверного веб-приложения, представляющего собой сервис, который интегрируется в микросервисную архитектуру, взаимодействуя с сервисами прогнозирования и распределения ресурсов. Серверная часть содержит модуль, который моделирует работу облачной инфраструктуры, и обрабатывает данные, а также модуль генерации данных и интеграции с сервисами, выполняющими алгоритмы машинного обучения. Клиентская часть обеспечивает доступ к результатам моделирования и настройкам симуляции. Подход с четким разделением слоев позволяет легко адаптировать систему к различным сценариям тестирования.

Backend реализован на основе реактивного фреймворка Spring Boot WebFlux с использованием многоуровневой архитектуры для разделения приложения на несколько слоев, каждый из которых ответственен за свою задачу [4]. Серверная часть предоставляет REST API для управления процессами симуляции, взаимодействия с ее компонентами и подключением алгоритмов машинного обучения к симуляционной облачной инфраструктуре в реальном времени. API спецификация описана при помощи OpenAPI Specification, сигнатуры методов контроллеров на backend и клиентские интерфейсы на frontend генерируются по ней с использованием Swagger OpenAPI Generator. Сбор, хранение и поиск логов для проведения аналитики по собранным результатам и метрикам обеспечиваются ELK-стеком (Elasticsearch, Logstash, Kibana). Для модульного и интеграционного тестирования используются JUnit, Reactor-test, Hamcrest и Mockito.

Frontend разработан на React с использованием Typescript и соблюдением методологии FSD (Feature-Sliced Design). Визуальное оформление выполнено на Bootstrap 5, а дизайн макетов создан в Figma. Интерфейс позволяет пользователю запускать, управлять симуляциями и отслеживать их состояние [5].

В ходе разработки системы реализованы ключевые эндпоинты: получение статуса, списка активных симуляций, поиск по ID, запуск статической симуляции, а также получение архитектуры симуляции с параметрами компонентов и их загруженностью. Разработан интерфейс для визуализации данных и гибкой настройки. Также выполнено развертывание SPA-приложения frontend совместно с backend с использованием Docker-compose.

Предстоит реализация endpoint для динамической симуляции, настройка ELK-стека. В интерфейсе необходимо добавить элементы управления для запуска динамической симуляции и работы с алгоритмами машинного обучения.

ЛИТЕРАТУРА

1. Cloud Computing Value Chains: Research from the Operations Management Perspective [Электронный ресурс] / S. Chen [и др.] // Manufacturing & Service Operations Management. —2023. —Т. 25, вып. 4. —С. 1338-1356. —Режим доступа: <https://doi.org/10.1287/msom.2022.1178>
2. Sahar Ahmadi Sakha, Vasilios Andrikopoulos (2024). “Architecting for sustainability of and in the cloud: A systematic literature review”.
3. Procaccianti G., Bevin S., Lago P (2013). “Energy efficiency in cloud software architectures”, 291-299.
4. Sabina Necula (2024). “Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications” Micro-frontend architecture with react: a comprehensive guide
5. Feature-sliced design as universal architecture for frontend projects / M.I. Skomorokhov // Международный студенческий научный вестник (International Student Scientific Herald). —2023. — Вып. 6. —С. 1-1.

УДК 004.421

А.В. Багрова, А.Л. Мальцев, К.Д. Романчук (2 курс бакалавриата),
В.В.Амосов, к.т.н., доцент

ПРОГРАММНАЯ И АППАРАТНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ДЕЙКСТРЫ

В связи с появлением сложных программируемых логических интегральных схем (ПЛИС) CPLD и систем на кристалле SoC, содержащих на одном микрочипе центральный процессор, интерфейсы памяти и программируемую часть, одним из способов аппаратного ускорения выполнения процессов является создание вычислительной логики для конкретного алгоритма с использованием ПЛИС. По этой причине в настоящее время актуальной является задача аппаратной и программной реализации алгоритмов обработки информации на современной элементной базе.

Целью работы является разработка программно-аппаратной реализации алгоритма Дейкстры, активно использующегося в системах маршрутизации, планирования и анализа сетей и решающего проблему кратчайших путей из одного источника, то есть находящего кратчайшие пути от заданной вершины до всех остальных в графе без ребер отрицательного веса.

В ходе проекта разработан комплексный программный продукт, состоящий из следующих элементов:

1. Программная реализация алгоритма Дейкстры на языке программирования C++;
2. Аппаратная реализация алгоритма Дейкстры на языке программирования VHDL;
3. Конфигурирование Register-Transfer Level (RTL) и Technology Map схем алгоритма;
4. Реализация алгоритма Дейкстры в командах 32-разрядного RISC-процессора с Гарвардской архитектурой – DP32;
5. Выполнение ко-симуляции на поведенческой и структурной модели DP32.

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.

Учитывая возможности программируемых логических интегральных схем последних поколений, разработанная система представляет собой комплексное решение, которое можно применять в широком спектре промышленных задач, связанных с поиском оптимальных путей при обработке графов.

ЛИТЕРАТУРА

1. E. W. Dijkstra, “A note on two problems in connexion with graphs”, Numerical Mathematics, vol. 1, no. 1, pp. 269–271, 1959.
2. Kenneth L. Short, “VHDL for Engineers”, International Journal of Computer Science, vol. 12, no. 3, pp. 195-209, 2014.

3. В.В.Амосов, “Схемотехника и средства проектирования цифровых устройств” – СПб.: БХВ-Петербург, 2007 – 560 с.
4. Вахитов Р.М. Язык VHDL. Практическое применение. — М.: ДМК Пресс, 2020
5. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. — MIT Press, 2009.

УДК 004.414.53

М.Д. Бородулин (4 курс бакалавриата),
А.В. Петров, ст. преподаватель

ТЕСТИРОВАНИЕ НА ОСНОВЕ СВОЙСТВ (PROPERTY-BASED TESTING): ИСКУССТВО ПРОВЕРКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЧЕРЕЗ УНИВЕРСАЛЬНЫЕ СВОЙСТВА

Современные стандарты в разработке программного продукта требуют не только написание функционального кода, но и обеспечения его надежности и устойчивости. Одним из более эффективных методов, гарантирующих достижение этой цели, является тестирование на основе свойств (Property-Based Testing) [1]. Данный метод позволяет разработчикам проверять не только конкретные сценарии использования, но и общие свойства, которые должны соблюдаться в рамках программного продукта.

Тестирование на основе свойств является методом автотестирования, при котором входные данные генерируются на основании заданных свойств и правил, реализованных в программном коде с целью обеспечения согласованности последующих бизнес-операций [2]. Ключевой особенностью данного вида тестирования является автоматическое генерирование разнообразных наборов входных данных, вместо создания множества однотипных тестов с фиксированными данными [3]. Данный подход к тестированию программного продукта значительно увеличивает шанс выявления ошибок, которые могут быть пропущены при традиционном тестировании. Причиной этого является более обширный спектр ситуаций и условий, который охватывает тестирование на основе свойств [4]. Увеличение шанса обнаружения ошибок особенно важно в сложных многоуровневых приложениях, где взаимодействие между компонентами может привести к непредсказуемой ситуации.

Таким образом, целью данной работы является исследование и анализ методологии тестирования на основе свойств (Property-Based Testing) как эффективного инструмента для повышения надежности и устойчивости программных продуктов, а также разработка рекомендаций по его внедрению в процессы разработки и тестирования программного обеспечения для повышения качества и уменьшения рисков возникновения ошибок.

Для достижения этой цели необходимо решение следующих задач:

1. Анализ существующих подходов и методов тестирования на основе свойств.
2. Создание рекомендаций по интеграции данного метода в существующие процессы разработки и тестирования.

Одним из основных компонентов разрабатываемого подхода к тестированию на основе свойств является использование формальных методов валидации, которые позволяют обеспечить соответствие программного обеспечения заданным требованиям. В ходе проведения сравнительного анализа в качестве инструмента для реализации тестирования на основе свойств был выбран QuickCheck [5]. Данная библиотека создана для автоматической генерации тестов для языка программирования Haskell [6].

Для эффективной интеграции Property-Based Testing в существующие процессы разработки и тестирования, необходимо определить критические участки, в которые можно будет внедрить автоматическую генерацию тестов. Это повысит вероятность обнаружения некорректного поведения программного продукта и улучшит его качество

Такой подход был реализован в Spotify [7]. Компания Spotify Technology S. A. при разработке своего приложения активно интегрировала тестирование на основе свойств для проверки корректной работы плейлистов. Это помогло компании эффективно и самое главное качественно протестировать свой продукт.

Подводя итог, тестирование на основе свойств является мощным инструментом, способным значительно повысить надежность и устойчивость программного продукта. Интегрирование этого метода в процессы разработки и тестирования позволяет не только минимизировать риски возникновения ошибок, но и улучшить общее качество продукта.

ЛИТЕРАТУРА

1. Property-Based Testing. [Электронный ресурс] Режим доступа: https://ru.hexlet.io/courses/js-advanced-testing/lessons/property-based-testing/theory_unit
2. Dale Miller, Alberto Momigliano “Property-Based Testing by Elaborating Proof Outlines” – 2024 – P. 1 – 2.
3. Особенность тестирования. [Электронный ресурс] Режим доступа: <https://www.baeldung.com/java-jqwik-property-based-testing>
4. Причины эффективности. [Электронный ресурс] Режим доступа: <https://quality-lab.ru/blog/testirovanie-na-osnove-svoystv-podxod-uskoryayushhij-reliz/>
5. Koen Claessen, John Hughes “QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs” – 2000 – P. 10 –12
6. Jan van Eijck “Haskell for Knowledge Representation” – 2012 – P. 4-8
7. Spotify. [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/Spotify>

УДК 004.932.2

Д.В. Волкова (3 курс бакалавриата),
П.А. Шагалова, к.т.н., доцент

ИССЛЕДОВАНИЕ ИНСТРУМЕНТОВ ОПТИМИЗАЦИИ КОДА В ЗАДАЧЕ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

На данный момент применение алгоритмов компьютерного зрения для обработки цифровых изображений играет ведущую роль в повышении качества результатов, полученных в ходе выполнения человеком исследования графических объектов в разных сферах деятельности. Во многих прикладных задачах требуется возможность работы программы в режиме реального времени или имеются ограничения по вычислительной мощности устройства, на котором происходит обработка изображений. В данной связи особенно актуальными являются анализ программных решений, с целью выявления негативных факторов, приводящих к снижению эффективности и быстродействия, а также оптимизация и повышение их производительности.

В данной работе рассмотрены подходы к анализу эффективности алгоритмов компьютерного зрения, а также методы оптимизации и повышения их производительности. Для проведения исследования выполнена программная реализация ряда алгоритмов компьютерного зрения, в том числе алгоритмы поиска ключевых точек на изображениях, сегментации, идентификации простейших фигур. Далее описаны инструменты и подходы к решению задачи на примере одного из алгоритмов – преобразования Хафа[4,5] для поиска параболических кривых.

Алгоритм Хафа использует массив, называемый аккумулятором, для определения присутствия кривой. Анализ строится на вероятностном преобразовании, то есть рассматриваются только контрольные точки изображения. Согласно алгоритму, каждый пиксель в пространстве изображения соответствует параболе в пространстве Хафа и наоборот. Для выполнения преобразования необходимы координаты вершины параболы, а также минимально и максимально возможное расстояние между вершиной и фокусом параболы. Положительный возвращаемый результат – полярный угол обнаруженной параболы и расстояние между вершиной и фокусом кривой. Отрицательный – отсутствие параболы на изображении с текущими входными параметрами. Преобразование Хафа применимо к бинарным изображениям, в которых пиксели могут принимать только два значения, из

которых «единичное» обозначает контрольную точку[2]. Применение алгоритма Хафа позволит повысить математическую точность найденных параболических кривых.

При разработке были описаны следующие основные методы:

- Обработка изображения, выполняющая преобразование Хафа;
- Нахождение индексов ключевых точек изображения, поддающиеся обработке;
- Нахождение локального максимума в массиве аккумулятора.

При разработке использовались: интегрированная среда Visual Studio, компилятор gcc, профилировщик gprof. Выбор инструментария при реализации играет немаловажную роль, поскольку оптимизация проекта требует знания очень широкого диапазона технологий, от компиляторов и операционных систем до логического проектирования[1]. Профилировщик gprof кратко предоставляет всю необходимую информацию о времени работы программы и отдельных программных компонент. Профилирование с использованием данного инструмента выполняется на достаточном уровне качества, применительно к сложным по структуре программам.

Реализация подразумевает обработку большого числа двумерных и одномерных массивов, циклов, а также многократный вызов функций, который может делать недоступными регистры для выполнения прочих операций. Вследствие чего при составлении ручной оптимизации[3] был выбран курс на понижение силы операций и исключение итеративных переменных, распределение значений в регистрах. Также в программе проделана работа по замене частей обработки массивов на методы модуля «vector.h», что в том числе положительно повлияло на производительность.

Итогом разработки является программная реализация метода Хафа для поиска параболических кривых на бинарном изображении, при которой главная функция обработки занимает не более 5% от общего времени исполнения программы. На данный момент проводится повышение качества кода для более точного отображения найденных кривых на входном графическом ресурсе.

ЛИТЕРАТУРА

1. Хеннесси Джон Л., Паттерсон Дэвид А. Компьютерная архитектура. Количественный подход. Издание 5-е. Москва: ТЕХНОСФЕРА, 2016. – 936 с., ISBN 978-5-94836-413-1
2. Рейнхард Клетте. Компьютерное зрение. Теория и алгоритмы / пер. с англ. А. А. Слинкин. – М.: ДМК Пресс, 2019. – 506 с.: ил. ISBN 978-5-97060-702-2
3. Командные опции GNU CC [Электронный ресурс] Режим доступа: <https://www.opennet.ru/docs/RUS/gcc/gcc1-2.html>
4. Преобразование Хафа [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/otus/articles/745946/>
5. OpenCV шаг за шагом. Преобразование Хафа [Электронный ресурс] Режим доступа: <https://robocraft.ru/computervision/502>

УДК 519.81, 004.65

Е.А. Волосов (4 курс бакалавриата),
С.А. Нестеров, к.т.н, доцент

ПРИМЕНЕНИЕ МЕТОДА ПЛАКЕТТА-БЕРМАНА ДЛЯ АНАЛИЗА ИТ-СИСТЕМ НА БАЗЕ POSTGRESQL

Современные ИТ-системы, такие как базы данных, облачные хранилища и распределенные системы, требуют тщательной настройки конфигурационных параметров для достижения максимальной производительности. В условиях роста объемов данных и увеличения сложности ИТ-инфраструктуры, ручная настройка становится неэффективной и затратной. Одним из эффективных методов анализа и оптимизации таких систем является метод Плакетта-Бермана (Р&В-дизайн), который позволяет систематически варьировать параметры и оценивать их влияние на производительность.

Метод Р&В-дизайна, описанный в работах по планированию экспериментов [1], основан на варьировании параметров конфигурации в пределах допустимых значений. Для каждого параметра задаются два уровня: «+1» (выше базового) и «-1» (ниже базового). На основе этого строится матрица экспериментов, которая позволяет минимизировать количество тестов при сохранении точности анализа. Ключевым этапом анализа является расчёт коэффициента вариации (CV), который показывает чувствительность системы к изменению параметров. CV вычисляется по формуле:

$$CV = \frac{\mu}{\sigma} \quad (1)$$

где μ – среднее время выполнения экспериментов, σ – стандартное отклонение.

Интерпретация коэффициента вариации (CV) заключается в следующем. Если $CV < 0.1$, параметр имеет низкую чувствительность и почти не влияет на производительность. Значение CV от 0.1 до 0.2 указывает на умеренное влияние, а $CV > 0.2$ — на высокую чувствительность и значительное влияние на производительность.

Как отмечается в работе [2], правильная настройка параметров может значительно улучшить производительность системы, особенно при работе с большими объемами данных. В ходе анализа производительности систем, работающих на базе PostgreSQL, были выявлены ключевые конфигурационные параметры, оказывающие наибольшее влияние на эффективность работы:

1. shared_buffers – объем памяти для хранения страниц базы данных [3];
2. work_mem – память для обработки временных данных [3];
3. checkpoint_timeout – регулирует частоту контрольных точек [4];
4. fsync – обеспечение согласованности данных на диске [4];
5. effective_io_concurrency – регулирует уровень параллелизма ввода-вывода [5];
6. max_parallel_workers – максимальное количество параллельных процессов [5].

Для анализа выбраны диапазоны значений параметров, представленные в таблице 1.

Таблица 1 – Выбор диапазонов параметров

Параметр	-1(ограничение)	Нормальное значение	+1 (разгон)
shared_buffers	512MB	2GB	8GB
work_mem	16MB	64MB	256MB
checkpoint_timeout	300 сек	900 сек	1800 сек
Fsync	off	on	On
effective_io_concurrency	1	2	8
max_parallel_workers	2	4	8

Для оценки влияния параметров была сформирована экспериментальная матрица, представленная в таблице 2.

Таблица 2 – Пример экспериментальной матрицы

Exp	shared_buffers	work_mem	checkpoint_timeout	sync	effective_io_concurrency	Max_parallel_workers
1	8GB	16MB	300 сек	on	1	8
2	512MB	256MB	1800 сек	off	8	2
3	8GB	256MB	300 сек	on	8	2
4	512MB	16MB	1800 сек	off	1	8
5	8GB	16MB	1800 сек	on	1	2
6	512MB	256MB	300 сек	off	8	8
7	8GB	256MB	1800 сек	off	1	2
8	512MB	16MB	300 сек	on	8	8

Для анализа были выбраны следующие тестовые сценарии:

1. Q1 - Подсчёт количества строк в большой таблице;
2. Q2 - Запрос с сортировкой и фильтрацией;
3. Q3 - Агрегация и группировка данных.

Результаты выполнения экспериментов представлены в таблице 3.

Таблица 3 – Время выполнения экспериментов

Exp	Q1 Time (ms)	Q2 Time (ms)	Q3 Time (ms)
1	1200	450	350
2	2000	600	500
3	1100	400	320
4	2200	700	550
5	1500	500	400
6	1300	480	370
7	1700	520	420
8	2500	800	600

Результаты показали, что запрос Q1 ($CV = 0.28$) наиболее чувствителен к `shared_buffers` (память) и `effective_io_concurrency` (ввод-вывод). Запрос Q2 ($CV = 0.24$) зависел от `work_mem` (оперативная память) и `checkpoint_timeout`, влияющих на сортировку и фильтрацию. Запрос Q3 ($CV = 0.21$) был чувствителен к `max_parallel_workers` (многопоточность) и `effective_io_concurrency` (ввод-вывод), что важно для агрегации данных.

Методология Р&В-дизайна позволяет эффективно анализировать влияние параметров конфигурации на производительность ИТ-систем. Использование данного подхода помогает выявить ключевые параметры, требующие оптимизации, и снизить затраты на ручную настройку систем.

ЛИТЕРАТУРА

1. Montgomery, D. C. Design and Analysis of Experiments (9th ed.). – Hoboken, NJ: John Wiley & Sons, Inc., 2017 – ISBN: 978-1-119-11347-8.
2. Кузнецов С.Д. Основы баз данных. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2019. – 512 с. – ISBN 978-5-9556-0123-4.
3. Домбровская Г., Новиков Б., Бейликова А. Оптимизация запросов в PostgreSQL / пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2022. – 278 с.: ил. – ISBN 978-5-97060-963-7
4. PostgreSQL Documentation: Write-Ahead Log. [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/docs/current/runtime-config-wal.html>
5. PostgreSQL Documentation: Parallel Query. [Электронный ресурс]. Режим доступа: <https://www.postgresql.org/docs/current/runtime-config-resource.html#GUC-EFFECTIVE-IO-CONCURRENCY>

УДК 004.514

И.П. Ерошин (2 курс магистратуры),
А.В. Щукин, к.т.н., доцент,
В.А. Пархоменко, ст. преподаватель

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ ОЦЕНКИ УДОВЛЕТВОРЁННОСТИ ПОЛЬЗОВАТЕЛЬСКИМ ИНТЕРФЕЙСОМ С УЧЁТОМ ИНДИВИДУАЛЬНОГО ЭФФЕКТА ВОЗДЕЙСТВИЯ

Актуальность работы обусловлена ростом значимости персонализации пользовательского опыта и необходимости объективной оценки интерфейсов программных приложений с учётом индивидуальных различий пользователей [1][2].

Целью работы является экспериментальное исследование алгоритмов компьютерного зрения (далее – CV) для оценки удовлетворённости пользовательским интерфейсом с выявлением индивидуального эффекта воздействия (далее – ITE). Для достижения поставленной цели необходимо решить следующие задачи:

1. Обзор и сравнительный анализ существующих методов оценки удовлетворённости пользовательским интерфейсом.

2. Разработка методики экспериментального исследования, основанной на алгоритмах CV и расчёте ITE.
3. Реализация экспериментальной платформы с использованием современных инструментов CV (WebGazer.js [3], Face-api.js [4]).
4. Проведение экспериментов по сбору и анализу данных пользователей.
5. Анализ эффективности использования алгоритмов компьютерного зрения в браузерах Firefox и Chromium.

Экспериментальная платформа реализована с использованием web-технологий и алгоритмов компьютерного зрения, в частности, библиотек WebGazer.js и Face-api.js с применением языков программирования JavaScript [5] и Python [6]. Данные библиотеки позволяют отслеживать взгляд пользователя и распознавать его эмоциональные реакции в реальном времени. В ходе экспериментов получены количественные метрики взаимодействия пользователей с интерфейсами, которые использованы для расчета индивидуального эффекта воздействия.

Архитектура системы экспериментального исследования алгоритмов компьютерного зрения для оценки удовлетворённости пользовательским интерфейсом с учётом индивидуального эффекта воздействия представлена на рисунке 1. Система отражает взаимодействие бизнес-процессов, приложений и технологий для оценки удовлетворённости интерфейсом на основе индивидуального эффекта воздействия с применением компьютерного зрения.

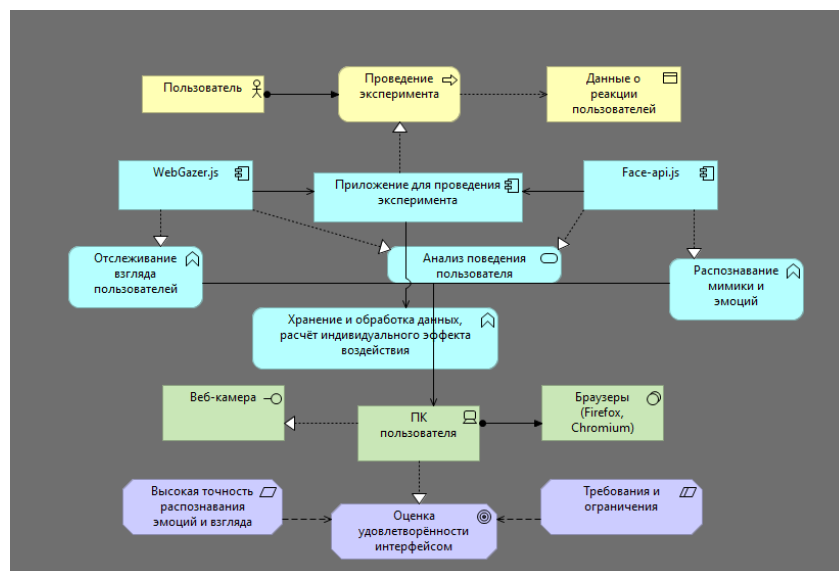


Рисунок 1 – Архитектура системы экспериментального исследования алгоритмов компьютерного зрения для оценки удовлетворённости пользовательским интерфейсом с учётом индивидуального эффекта воздействия (нотация ArchiMate)

Результатами экспериментов подтверждена эффективность алгоритмов компьютерного зрения для объективной оценки удовлетворённости пользователей с учётом ITE. Совместное использование обеих библиотек позволило добиться высокой точности и всесторонней оценки пользовательского опыта за счёт комбинирования данных о взгляде и эмоциональных реакциях пользователей.

ЛИТЕРАТУРА

1. Chatti M. A., Guesmi M., Vorgerd L., Ngo T., Joarder S., Ain Q. U., Muslim A. Is More Always Better? The Effects of Personal Characteristics and Level of Detail on the Perception of Explanations in a Recommender System // Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization. – 2022. – P. 254–264.
2. Carter B. T., Luke S. G. Best Practices in Eye Tracking Research // International Journal of Psychophysiology. – 2020. – P. 49–62.

3. Basurah M., Swastika W., Kelana O. H. Implementation of Face Recognition and Liveness Detection System Using TensorFlow.js // Jurnal Informatika Polinema. – 2023. – P. 509-516.
4. Papoutsaki A., Sangkloy P., Laskey J., Daskalova N., Huang J., Hays J. WebGazer: Scalable Webcam Eye Tracking Using User Interactions // Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI). – 2016. – P. 3839–3845.
5. Python Documentation [Электронный ресурс] Режим доступа: <https://www.python.org/doc/>
6. JavaScript reference [Электронный ресурс] Режим доступа: <https://devdocs.io/javascript/>

УДК 004.822, 681.51

П.Д. Демчук (4 курс бакалавриата),
И.В. Шошмина, к.т.н., доцент

МОДЕЛИРОВАНИЕ ВЫЧИСЛИТЕЛЬНОГО КОНВЕЙЕРА GPU С ПОИСКОМ ОПТИМАЛЬНЫХ ПАРАМЕТРОВ С ПОМОЩЬЮ MODEL CHECKING

Современные графические процессоры (GPU) [1] с множеством параллельных вычислительных модулей делают доступным решение высокопроизводительных задач, требующих параллельных вычислений. Оптимальная по потреблению ресурсов работа программы, исполняемой на GPU, зависит от множества параметров, подбор которых затруднен в силу их многочисленности и взаимозависимости.

В этой работе мы развиваем подход, сформированный в работе [2], где подбор параметров выполняется с помощью метода проверки моделей. В этом подходе программа моделируется в верификаторе, а поиск параметров осуществляется как поиск контрпримера свойства недостижимости оптимального состояния, задаваемого линейной темпоральной логикой. Это свойство называется свойством сверхурочности. В процессе подбора параметров большое влияние оказывает то, как задания распределяются на вычислительные элементы GPU, насколько полно использованы вычислительные ресурсы, как они синхронизируются между собой при выполнении прикладной задачи. Все эти детали описываются в вычислительном конвейере (pipeline) GPU. Однако, в работе [2] вычислительный конвейер (pipeline) операций GPU не был промоделирован.

Таким образом, целью данной работы является моделирование вычислительного конвейера операций GPU, позволяющего подбирать оптимальные параметры методом проверки модели.

Для достижения этой цели были поставлены следующие задачи:

1. Разработать модель для вычислительного конвейера программы, выполняемой на GPU.
2. Разработать Promela модель исполнения программы, написанной на OpenCL.
3. Найти минимальное время исполнения программы через проверку свойства сверхурочности.
4. Получить значения оптимальных параметров из результатов верификации.

Для построения модели вычислительного конвейера GPU мы использовали данные из открытого проекта GPGU Sim [3]. Нами были выделены основные элементы GPU, входящие в работу вычислительного конвейера. Опишем их. Абстрактный GPU может состоять из нескольких устройств (device). Устройства объединяют в себе потоковые мультипроцессоры, которые в свою очередь содержат элементы обработки.

По стандарту OpenCL программа для GPU состоит из хост-части, исполняемой на обобщенном вычислителе (хосте), и ядра, экземпляры которого исполняются на специализированных узлах (элементах обработки). Рабочий элемент – исполняемый экземпляр параллельной программы или один поток. Потоки программно делятся на рабочие группы, которые связываются с конкретным мультипроцессором. Рабочие группы делятся на варпы [4]. Элементы варпа исполняются на одном устройстве.

В качестве инструмента для верификации мы использовали верификатор SPIN [5] и его

язык описания взаимодействующих процессов Promela [6].

Для каждого элемента в вычислительном конвейере GPU мы выделили отдельные процессы. Процесс потокового мультипроцессора unit исполняет инструкции OpenCL kernel. В единицу времени на одном юните исполняется одна инструкция для одного варпа. Процесс планировщика варпов warp_scheduler делит рабочую группу на варпы, и в зависимости от готовности варпа назначает его на выполнение на мультипроцессор по алгоритму раунд-робин. Процесс устройства device назначает на исполнение рабочие группы. Процесс хост-программы host запускает процесс device и ожидает, когда все OpenCL инструкции будут исполнены. Основной процесс main определяет значения параметров настройки, количество необходимых для обработки всего массива данных узлов и запускает host. Взаимодействие процессов происходит через каналы (рисунок 1).

Эксперименты с моделью проводились на примере задачи суммирования четных чисел, для программы, написанной на OpenCL [7]. В данной работе производится настройка таких параметров, как количество рабочих групп и их размер, размер массива данных, обрабатываемого одним потоком. Для определения оптимальных параметров проверялось свойство сверхурочности – всегда, если программа завершилась, то за время большее, чем заданное T. Контрпример этого свойства указывает на то, что программа может завершиться за меньшее время. Последовательность повторных проверок с уменьшающимся на каждом шаге значением T позволяет найти контрпример с минимально возможным временем выполнения программы, из которого извлекаются искомые параметры настройки.

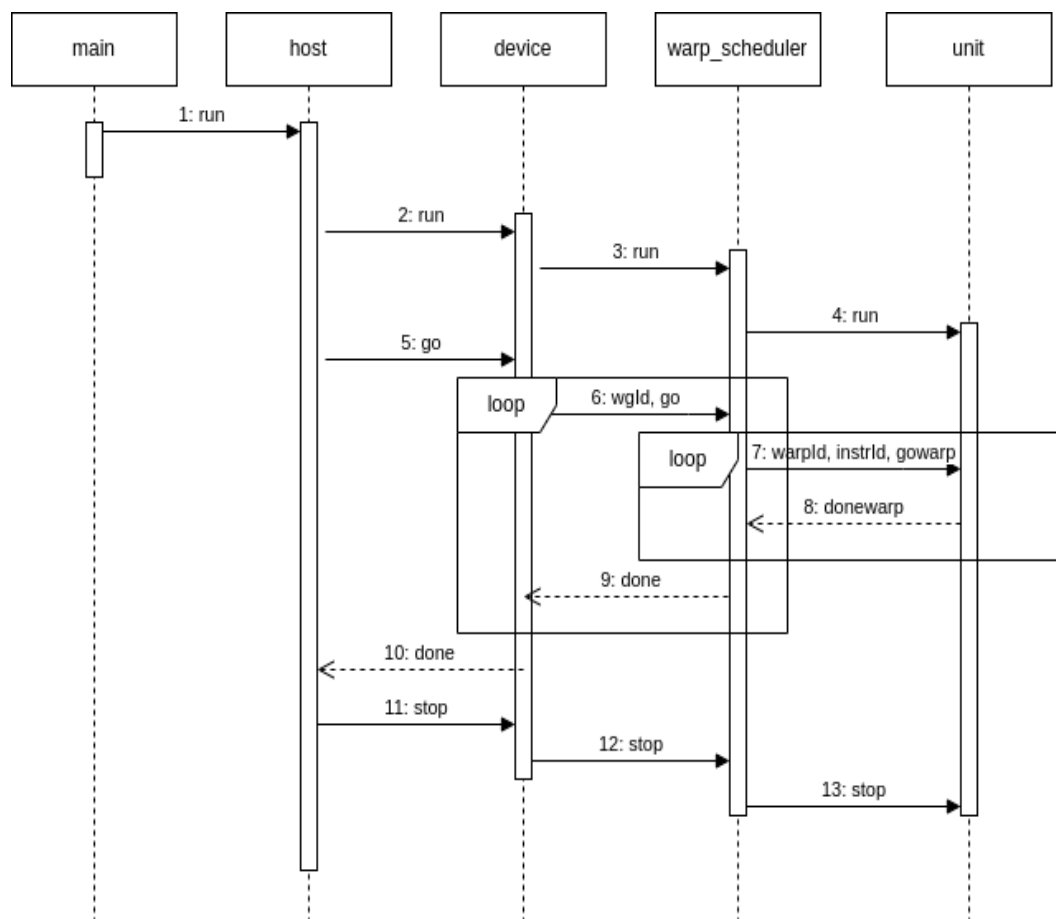


Рисунок 1 – Диаграмма последовательности взаимодействий в Promela модели

ЛИТЕРАТУРА

1. Хеннеси Дж.Л., Паттерсон Д.А. «Компьютерная архитектура. Количественный подход» Изд-е 5. М.: Техносфера, 2016. – 936 с.
2. N. O. Garanina, S. P. Gorlatch, “Autotuning parallel programs by model checking”, Модел. и анализ информ. систем, 28:4 (2021), 338–355

3. GPGU Sim, manual. Revision 1.2 (GPGPU-Sim 3.1.1). Ed.: Tor M. Aamodt, Wilson W.L. Fung, Tayler H. Hetherington, http://gpgpu-sim.org/manual/index.php/Main_Page#Fetch_and_Decode_Software_Model
4. Dao, Thanh & Lee, Jaejin. (2017). An Auto-tuner for OpenCL Work-group Size on GPUs. IEEE Transactions on Parallel and Distributed Systems. PP. 1-1. 10.1109/TPDS.2017.2755657.
5. Документация Spin [Электронный ресурс] Режим доступа: <https://spinroot.com/spin/Man/Spin.html>
6. Документация Promela [Электронный ресурс] Режим доступа: <https://spinroot.com/spin/Man/promela.html>
7. Контекст VeHa [Inline (Иннополис) с 18.10 по 21.10 / Online] <https://sites.google.com/view/veha2024>

УДК 004.021

Д.О. Королев (1 курс аспирантуры),
И.В. Никифоров, к.т.н., доцент,
О.А. Юсупова, ст. преподаватель

МЕТОДЫ ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ ДЛЯ ИЗУЧЕНИЯ И ПРЕДОТВРАЩЕНИЯ СБОЕВ И АНОМАЛИЙ В РАБОТЕ ОБЛАЧНЫХ ИНФРАСТРУКТУР

Облачные инфраструктуры играют ключевую роль в современной отрасли IT технологий. Они используются для хранения и обработки больших объемов данных, что особенно востребовано в аналитике, машинном обучении и искусственном интеллекте [1]. В разработке программного обеспечения облачные платформы позволяют командам работать удаленно, тестировать и развертывать приложения в виртуальной среде. В бизнесе облачные сервисы применяются для управления корпоративными ресурсами [2], CRM-системами и автоматизации бизнес-процессов. Также облачные технологии активно используются в образовании, медицине и телекоммуникациях, обеспечивая доступ к данным и сервисам из любой точки мира. Благодаря своей универсальности облачные инфраструктуры стали неотъемлемой частью цифровой трансформации в большинстве отраслей [3].

Облачные инфраструктуры обладают преимуществами, перечисленные ниже.

1. Обеспечивают гибкость и масштабируемость, позволяя компаниям адаптироваться к изменяющимся требованиям рынка без значительных затрат ресурсов.
2. Безопасность данных в облачных платформах часто оказывается выше за счет постоянных обновлений и специализированных команд, работающих над защитой информации.
3. Способствуют снижению затрат на обслуживание и управление IT-ресурсами, позволяя сосредоточиться на инновациях и развитии бизнеса.

Таким образом, облачные инфраструктуры становятся необходимым инструментом для успешного ведения бизнеса в динамичной цифровой среде.

Однако, облачные инфраструктуры содержат множество сложных компонентов, таких как вычислительные серверы, маршрутизаторы, коммутаторы, брандмауэры, системы хранения и другие компоненты, выполняющие роль вычислительных и сетевых ресурсов [4]. Большое количество компонентов увеличивает возможность возникновения сбоев и аномалий, среди которых можно выделить: аппаратные сбои, программные сбои, сбои из-за отказа сторонних сервисов, сбои на уровне потребителей (клиентов), плановые или аварийные обслуживания. Эти сбои могут вызывать отказ в работе некоторых функций облачных сервисов или полную потерю их работоспособности, что может повлечь за собой значительные финансовые потери и снижение доверия пользователей [5].

Однако, изучение сбоев в работе облачных инфраструктур непосредственно в промышленном окружении сопряжено с рядом трудностей [6]:

– ограниченные возможности тестирования: тестирование промышленных систем в реальной среде часто ограничено из-за риска нарушить их работу и высокой цены ошибки;

- сложность распределенных систем: промышленные системы часто состоят из множества распределенных компонентов, взаимодействующих друг с другом;
- работа систем в реальном времени и высокие нагрузки: промышленные системы часто работают в режиме реального времени или с высокими нагрузками, что увеличивает риск возникновения критических ошибок;
- ограниченный доступ к инфраструктуре: доступ к промышленным системам может быть ограничен из-за строгих требований безопасности, лицензирования или удаленности объектов.

Поэтому актуальной является задача применения методов дискретно-событийного моделирования для изучения и предотвращения сбоев и аномалий в работе облачных инфраструктур.

В данном исследовании для определения эффективности методов дискретно-событийного моделирования [7] будет применяться метрика доступности [8]. Она определяет процент максимально возможного количества критических сбоев в информационных системах и рассчитывается по следующей формуле:

$$AP = \frac{AST - DT}{AST} * 100, \quad (1)$$

где AP – процент доступности (Availability Percentage);

AST – согласованное время работы (Agreed service time);

DT – время простоя (Downtime).

Так, целью исследования является снижение частоты сбоев и аномалий в работе облачных инфраструктур на основе методов дискретно-событийного моделирования, реализованных в программной системе с использованием параллельной симуляции моделей облачной инфраструктуры.

Задачи, необходимые для достижения цели, перечислены ниже.

1. Провести исследование предметной области, сбор информации об аномалиях и сбоях в работе облачных инфраструктур и их классификация.
2. Отражение аномалий в имитационной модели для моделирования.
3. Провести сравнительный анализ существующих подходов моделирования.
4. Предложить методы предотвращения аномалий на основе дискретно-событийного моделирования.
5. Реализовать предложенное решение в программном средстве с применением параллельной симуляции моделей облачной инфраструктуры.
6. Продемонстрировать эффективность разработанного решения в ходе экспериментальных исследований на основе параллельной симуляции моделей облачной инфраструктуры.

ЛИТЕРАТУРА

1. Заборовский, В. С. Интеллектуальная система управления ресурсами суперкомпьютерного центра коллективного пользования / В. С. Заборовский, С. Ф. Сергеев, Л. В. Уткин // Интеллектуальные технологии в эргономике и когнитивных науках : Сборник материалов всероссийской научно-практической онлайн-конференции с международным участием, Брянск, 04–06 июня 2024 года. – Брянск: Брянский государственный технический университет, 2024. – С. 170-173. – EDN VWMFYJ.
2. Исследование подходов к созданию высокопроизводительного масштабируемого облачного сервиса для дедупликации данных в хранилище / Н. В. Ермаков, Т. С. Таленфельд, И. В. Никифоров, П. Б. Егоров // Информатика и кибернетика (ComCon-2016) : сборник докладов студенческой научной конференции Института компьютерных наук и технологий, Санкт-Петербург, 04–09 апреля 2016 года / Министерство образования и науки РФ; Санкт-Петербургский политехнический университет Петра Великого. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2016. – С. 144-147. – EDN WIBBWZ.

3. Fady A. M. Ibrahim. Trusted Cloud Computing Architectures for infrastructure as a service: Survey and systematic literature review / Fady A. M. Ibrahim, Elsayed E. Hemayed // Computers & Security. – DOI 10.1016/j.cose.2018.12.014. – 2019. – №82. – P 196-226.
4. Sunilkumar S. Manvi. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey / Sunilkumar S. Manvi, Gopal Krishna Shyam // Journal of Network and Computer Applications. – DOI 10.1016/j.jnca.2024.110291. – 2013. – №41. – P 424-440.
5. Замятина О. М. Моделирование систем: Учебное пособие / Замятина О. М. – Томск: Изд-во ТПУ, 2009. – 204 с.
6. Alexandre Magno Castañon Guimarães. Discrete-event simulation software selection for manufacturing based on the maturity model / Alexandre Magno Castañon Guimarães, José Eugenio Leal, Paulo Mendes // Computers in Industry. – DOI 10.1016/j.compind.2018.09.005. – 2018. – №103. – P 14-27.
7. Применение методов машинного обучения для построения моделей поведения многоагентных систем и их анализа / Д. А. Вихляев, А. М. Сабуткевич, И. В. Никифоров, А. В. Самочадин // 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. 122-124. – EDN UOKUXP.
8. A study on the prediction of service reliability of wireless telecommunication system via distribution regression / Yan-Fu Li, Wei Zhao, Chen Zhang [и др.] // Reliability Engineering & System Safety. – DOI 10.1016/j.ress.2024.110291. – 2024. – №250.

УДК 004.421

А.В. Климков (4 курс бакалавриата),
А.С. Герасимов, к.ф.-м.н., доцент

ГЕНЕРАЦИЯ УЧЕБНЫХ ЗАДАНИЙ НА ТОПОЛОГИЧЕСКУЮ СОРТИРОВКУ

Тестирование знаний студентов позволяет объективно оценить уровень усвоения материала и способность применять знания на практике. Важно, чтобы задания в тестах были уникальными и стимулировали самостоятельное мышление, поскольку доступ к готовым решениям может снижать эффективность обучения. В связи с этим возникает задача автоматизации генерации учебных заданий, что позволит сократить время преподавателей на подготовку заданий и обеспечить студентов разнообразными заданиями. В частности, для курса «Алгоритмы и анализ сложности» [1] требуется автоматически генерировать задания по топологической сортировке (линеаризации) ориентированных графов (орграфов). *Линеаризацией* орграфа называется такое упорядочение всех его вершин, что каждое его ребро идёт из меньшей вершины в большую (относительно этого порядка).

Целью данной работы является создание программы, которая генерирует учебные задания на топологическую сортировку и обоснования ответов на них. Генерируемые учебные задания должны соответствовать следующим шаблонам:

Шаблон №1. Дан ориентированный граф $G = (V, E)$, где
 $V = \{1, 2, \dots, \text{ЧислоВершин}\}$ и $E = \text{МножествоРёбер}$.
 Сколько всего линеаризаций у графа G ?
 (1) $N-1$ или более; (2) $N-2$; (3) $N-3$; ...; (4) $N-2$; (5) $N-1$; (6) 0

Шаблон №2. Дан ориентированный граф $G = (V, E)$, где
 $V = \{1, 2, \dots, \text{ЧислоВершин}\}$ и $E = \text{МножествоРёбер}$.
 Какие последовательности вершин являются линеаризациями графа G ?
 (1) 1-я перестановка множества V ; ...; (5) $(N-1)$ перестановка множества V ;
 (6) ни один из ответов (1)–(5) не подходит

Для достижения цели работы мы решаем следующие задачи:

1. Провести анализ предметной области:

- осуществить обзор базовых сведений об орграфах, ациклических орграфах и линейаризации орграфов;
- изучить существующие подходы к генерации учебных заданий;
- осуществить обзор существующих программных решений, близких по назначению к разрабатываемому.

2. Разработать общую схему генерации учебного задания, включающую в себя описание параметров программы-генератора.

3. Детализировать на уровне псевдокода следующие основные алгоритмы:

- алгоритм генерации ориентированных графов;
- алгоритм визуализации сгенерированных орграфов;
- алгоритм формирования обоснований ответов на учебные задания.

4. Реализовать программу-генератор учебных заданий на языке Java.

5. Задokumentировать программу-генератор заданий.

Было найдено множество онлайн-ресурсов, частично решающих поставленные задачи.

Например, веб-сервисы [2] и [3] позволяют линейаризовать орграф и визуализировать работу алгоритма топологической сортировки, однако их функционал ограничен. Первый из них случайно генерирует лишь множество рёбер; второй не генерирует орграф, а лишь позволяет задать его через матрицу смежности; оба сервиса не строят все линейаризации заданного орграфа.

Алгоритмы RD-Gen [4] генерируют случайные орграфы для задач планирования, но имеют ряд недостатков. Они не формируют координаты вершин генерируемого графа, что затрудняет его визуализацию. Кроме того, создаваемые графы нередко имеют простую структуру и потому не представляют интереса для линейаризации.

Нам не удалось найти существующие программные решения, которые генерируют учебные задания по линейаризации орграфов и предоставляют обоснования ответов.

Основным компонентом программы-генератора учебных заданий является алгоритм генерации орграфов, который также генерирует координаты их вершин. В этом алгоритме орграф вида $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$ мы называем *цепочкой вершин* $v_1, \dots, v_n$. Упомянутый алгоритм разработан нами, и в общих чертах, без учета генерации координат, он выглядит следующим образом:

1. Генерируется начальная цепочка вершин длины, выбираемой случайно.

2. Пока количество вершин, добавленных в строящийся орграф, меньше желаемого:

2.1. Выбираем случайно две разные вершины графа.

2.2. Соединяем их новой цепочкой, длина которой выбирается случайным образом.

Обоснование ответа на задание по шаблону №1 представляется в виде таблицы с пошаговым построением всех линейаризаций орграфа. Обоснование для заданий по шаблону №2 таково: неправильный ответ объясняется указанием ребра, не идущего от вершины к одной из последующих вершин; правильный ответ иллюстрируется изображением орграфа, где вершины расположены в порядке линейаризации, данной в этом ответе.

На текущий момент разработаны все основные алгоритмы, которые составят нашу программу-генератор. Остается их доработка и реализация на языке Java [5], а также разработка интерфейса командной строки и графического интерфейса с использованием Swing API [6].

ЛИТЕРАТУРА

1. Герасимов А.С. Материалы учебного курса «Алгоритмы и анализ сложности», 2024.
2. Branch and Bound - Topological Sort [Электронный ресурс] Режим доступа: <https://algorithm-visualizer.org/branch-and-bound/topological-sort>.

3. Topological Sort (DFS) Visualization [Электронный ресурс] Режим доступа: <https://www.cs.usfca.edu/galles/visualization/TopoSortDFS.html>.
4. Atsushi Y., Takuya A. RD-Gen: Random DAG Generator Considering Multirate Applications for Reproducible Scheduling Evaluation // 2023 IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC). – 2023. — P. 21– 31. – DOI 10.1109/ISORC58943.2023.00015.
5. Java Platform Standard Edition 8 Documentation [Электронный ресурс] Режим доступа: <https://docs.oracle.com/javase/8/docs/>
6. Java Swing API Documentation. [Электронный ресурс] Режим доступа: <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

УДК 004.021

Н.В. Кулаков (4 курс бакалавриата)

АЛГОРИТМИЧЕСКОЕ ИССЛЕДОВАНИЕ СТРАТЕГИЙ ИНВАЛИДАЦИИ КЕША ПРИ ВЗАИМОДЕЙСТВИИ С ВЫСОКОЛАТЕНТНЫМ И НЕПРОЗРАЧНЫМ ИСТОЧНИКОМ ДАННЫХ

В современных системах обработки данных одним из важнейших факторов эффективности является обеспечение быстрого отклика при сохранении корректности и актуальности возвращаемой информации. В условиях, когда внешний источник не предоставляет механизмов прямой или немедленной информации о моменте изменения данных, проблема поддержания согласованности кешированного контента существенно усложняется [1].

Инвалидация кеша является базовым механизмом управления жизненным циклом кешированных объектов. Её суть состоит в том, чтобы вовремя распознать, что данные в кеше стали устаревшими, и инициировать их обновление или удаление, поддерживая целостность и актуальность системы. Без грамотной инвалидации кеша увеличивается риск накопления некорректной информации, что может привести к ошибочным результатам обработки и снижению доверия к сервису [2].

Однако разработка алгоритмов инвалидации усложняется, если источник данных:

- Обладает высокой латентностью – любые дополнительные обращения к нему негативно влияют на общую производительность системы.
- Непрозрачен с точки зрения внутренних процессов – отсутствует возможность получать сигналы, которые указывали бы на факт изменения или устаревания содержимого.
- Потенциально динамичен, то есть частота и объём обновлений данных могут варьироваться во времени.

Цель настоящего исследования – алгоритмически обосновать и разработать подходы к реактивной (адаптивной) инвалидации кеша, учитывающие реальные и потенциальные задержки доступа к внешнему источнику, а также неопределённость моментов изменения данных. Для этого необходимо:

1. Проанализировать классические стратегии кеширования (фиксированные правила TTL, эвристические модели) и выявить их ограничения в условиях высокой задержки и непрозрачности источника.
2. Предложить методики, позволяющие динамически рассчитывать параметры кеширования: например, период обновления, вероятность предиктивной выборки или предел допустимого устаревания [3].
3. Разработать гибридные стратегии, сочетающие элементы предиктивного анализа (на основе исторических данных о запросах и косвенных признаках изменения) с механизмами реактивной инвалидации (использующими статистику актуальных обращений).

4. Сформировать имитационную модель, основанную на положениях теории массового обслуживания (ТМО), для количественной оценки времени отклика, вероятности просрочки данных и нагрузки на внешний источник [4].

В данном контексте теория массового обслуживания (ТМО) выступает эффективным инструментом для формализации процессов прихода, обслуживания и истечения срока актуальности кешированных данных [4]. Модель ТМО предоставляет математический аппарат для описания случайных потоков запросов, моделирования временных задержек на каждом этапе (обращение к кешу, взаимодействие с внешним источником) и анализа вероятностных характеристик систем с большим количеством параллельных запросов [4]. За счёт использования понятий λ - (интенсивность поступления) и μ -параметров (интенсивность обслуживания) можно вывести основные метрики работы кеш-сервиса: ожидаемое время отклика, длину очереди и вероятность возникновения ситуаций, когда данные оказываются устаревшими [5].

Применение ТМО позволяет динамически варьировать стратегию инвалидации кеша, рассматривая особенности каждого конкретного подхода. Например, если интенсивность потока запросов растёт, а частота обновлений источника остаётся низкой, система может реже запрашивать новые данные, полагаясь на более долгий срок хранения кеш-объектов. Напротив, при увеличении вероятности изменений во внешнем источнике алгоритм способен автоматически сокращать время хранения данных, даже ценой более частых обращений к источнику. Тем самым достигается баланс между риском возвращения устаревших сведений и избыточными запросами, поддерживая адаптивность и надёжность работы во всём диапазоне реальных нагрузок [6].

Существенная часть исследования посвящена адаптивности: в отличие от статических схем (где параметры инвалидации не меняются), предлагается реагировать на изменение внешних факторов – например, на возрастание интенсивности запросов, на обнаружение косвенных сигналов о возможном обновлении или на колебания пропускной способности внешнего сервиса. Подобная гибкая система должна уметь пересматривать частоту проверки кеша, корректировать интервалы обновления или автоматически переходить к более агрессивным стратегиям инвалидации при резком всплеске вероятности устаревания.

Результатом работы станет комплекс математических моделей и практических рекомендаций, демонстрирующих, как переход от жёстко заданных параметров к реактивно-адаптивным алгоритмам позволяет повысить общую производительность системы (уменьшить латентность для конечных пользователей) и минимизировать риски некорректной кешированной информации. Полученные выводы имеют потенциальное применение в целом ряде задач: от распределённых веб-приложений до аналитических платформ, интегрирующих данные из медленных или малоинформативных источников.

ЛИТЕРАТУРА

1. Tanenbaum A. S., Steen M. Van. *Distributed Systems: Principles and Paradigms*. – Prentice-Hall, 2017.
2. Jacobson V., McCanne S. *Adaptive Replacement Cache Algorithms*. // IEEE/ACM Transactions on Networking, 2018.
3. Vvedenskaya N. D., Dobrushin R. L., Karpelevich F. I. Queueing System with Selection of the Shortest Queue: An Asymptotic Approach // Problems of Information Transmission, 1996.
4. Гнеденко Б. В., Коваленко И. Н. *Введение в теорию массового обслуживания*. – М.: Наука, 1987.
5. Кузнецов В. П., Морозов Е. Ю. *Вероятностные модели и методы анализа систем массового обслуживания*. — СПб.: Изд-во Политехнического университета, 2014.
6. Панкрашкин В. Я. *Математические модели систем управления: теория и практика*. — СПб.: СПбПУ Петра Великого, 2016.

МЕТОД МОДЕЛИ-ОРИЕНТИРОВАННОЙ РАЗРАБОТКИ НА ОСНОВЕ ДИСКРЕТНО-СОБЫТИЙНОГО МОДЕЛИРОВАНИЯ

В связи с быстрым развитием и повсеместным внедрением информационных технологий, зачастую реализуемых в виде программных систем и комплексов, повышается сложность и трудоемкость процесса разработки [1]. Одним из подходов к повышению эффективности проектирования и разработки программных систем является метод модели-ориентированной разработки [2], основывающийся на дискретно-событийном моделировании [3].

Целью данной работы является создание программной системы, реализующей метод модели-ориентированной разработки с использованием парадигмы дискретно-событийного моделирования. Ключевым преимуществом системы является возможность легкого добавления новых компонентов модели, что позволит использовать ее для решения широкого спектра прикладных задач, к примеру, моделирование и оптимизация коммуникаций в рамках компьютерных сетей и телекоммуникационных систем, управление облачными инфраструктурами и аппаратными-ресурсами.

Метод модели-ориентированной разработки – это подход к разработке программного обеспечения, где основным артефактом разработки является модель, а не код. Ее преимуществом является то, что такой подход применим во всех областях инженерии [2]. Модели-ориентированная разработка может быть реализована на основе различных формальных нотаций, к примеру, с использованием Unified Modeling Language (UML) для информационного моделирования и Business Process Model and Notation (BPMN) для моделирования процессов, а также с помощью дискретно-событийного моделирования. Дискретно-событийное моделирование – это метод имитационного моделирования, где состояние системы меняется только в дискретные моменты времени, называемые событиями [4]. Дискретно-событийное моделирование может быть применено для решения большинства инженерных задач за счет поддержки различных уровней абстракции. В качестве примера программной системы модели-ориентированной разработки можно привести Collimator [5], который позволяет описывать логику поведения разрабатываемой системы при помощи графической нотации, однако, он обладает недостатками, главный из которых – расширяемость для различных предметных областей в силу отсутствия открытого программного кода и интерфейсов.

На концептуальном уровне для реализации дискретно-событийного моделирования можно выделить событие, обработчик, алгоритм симуляции и список запланированных событий. Событие выступает в роли неизменяемого объекта данных, характеризующегося набором атрибутов. В отличие от него, обработчик представляет собой сущность, инкапсулирующую логику обработки конкретного типа события. Обработчики определяют динамику системы, реагируя на события, изменяя состояние и планируя новые события. Необходимость введения понятия обработчика в отдельную сущность обуславливается желанием обеспечить расширяемость системы, упрощая добавления новых типов событий и изменение логики обработки существующих.

Список запланированных событий представляет собой упорядоченный по времени наступления список событий. События попадают туда тремя способами:

- генерируются другими событиями в ходе симуляции;
- добавляются с помощью функций распределения (для моделирования случайных процессов);
- задаются пользователем до начала моделирования.

Алгоритм симуляции выступает центральным управляющим элементом. На каждой итерации алгоритм извлекает событие из списка, передает управление соответствующему обработчику, который выполняет логику обработки, включая возможное порождение новых событий. В случае порождения новых событий, последние добавляются в список запланированных событий. Симуляция завершается, когда список запланированных событий оказывается пустым или достигается заданное время окончания.

Ключевым преимуществом данной реализации является поддержка расширения за счет механизма наследования, позволяющего создавать новые типы событий и обработчиков, сохраняя совместимость с ядром алгоритма симуляции.

Программная реализация выполнена на языке программирования Python и включает в себя несколько ключевых классов. Для представления событий используется класс Event, который хранит основную информацию: уникальный идентификатор события, время его выполнения и ссылку на обработчик (класс Handler). Класс Event поддерживает сравнение по времени, что необходимо для упорядочивания событий в списке будущих событий. Класс Handler предоставляет базовую структуру для обработки событий, с абстрактным методом do, предназначенным для реализации логики обработки конкретных типов событий.

Для реализации списка будущих событий (FEL) используется класс FutureEventList, основанный на очереди с приоритетом и структуре данных "куча" (heap) для эффективного хранения и извлечения событий по времени. Класс Clock представляет собой простую реализацию модельного времени, предоставляя методы для его установки и получения. Для сбора и хранения данных симуляции реализован класс Stats, который записывает информацию о каждом событии в формате JSON, предназначенном для последующего анализа результатов моделирования. Класс Engine является ядром библиотеки, управляющим временем (класс Clock), списком будущих событий (класс FutureEventList) и сбором статистики (класс Stats). При запуске симуляции Engine циклически извлекает события из FEL, обновляет модельное время и передает управление обработчику события, после чего информация о событии записывается в Stats.

В ходе выполнения данной работы была успешно создана программная система, в основе которой лежит модели-ориентированный метод на основе дискретно-событийное моделирование. Реализованная архитектура системы обеспечивает легкость добавления новых типов событий и обработчиков. Таким образом, созданная система обладает необходимой гибкостью и расширяемостью для адаптации к широкому спектру прикладных задач в области информационных технологий.

ЛИТЕРАТУРА

1. Nguyen-Duc, Anh. (2017). The impact of software complexity on cost and quality - A comparative analysis between Open source and proprietary software. 10.48550/arXiv.1712.00675.
2. А. В. Борщёв. От системной динамики и традиционного ИМ – к практическим агентным моделям: причины, технология, инструменты. — 2004.
3. G. Wagner, M. Seck and F. McKenzie, "Process modeling for simulation: Observations and open issues," 2016 Winter Simulation Conference (WSC), Washington, DC, USA, 2016, pp. 1072-1083, doi: 10.1109/WSC.2016.7822166.
4. Discrete-Event System Simulation. / Banks, Jerry; Carson, John S.; Nelson, Barry L. et al. 4 ed. Prentice-Hall, 2005. 608 p.
5. G. Wagner, "An abstract state machine semantics for discrete event simulation," 2017 Winter Simulation Conference (WSC), Las Vegas, NV, USA, 2017, pp. 762-773, doi: 10.1109/WSC.2017.8247830.
6. Collimator [Электронный ресурс] // Collimator - URL: <https://www.collimator.ai/index.html> (дата обращения 15.03.2025)

ИНТЕГРАЦИЯ DEVOPS И ТЕСТИРОВАНИЯ: КАК ОБЕСПЕЧИТЬ КАЧЕСТВО ПО НА
КАЖДОМ ЭТАПЕ

В современных условиях для обеспечения конкурентоспособности и устойчивости в динамично изменяющейся цифровой среде критически важно оперативно и качественно поставлять программные решения. Одной из методологий, способствующих достижению этих целей, является концепция DevOps [1], которая предусматривает интеграцию усилий команд разработки и эксплуатации с целью оптимизации рабочих процессов и сокращения циклов поставки. Однако ускорение процессов поставки создает новые трудности в поддержании высокого качества программного обеспечения.

В традиционных моделях разработки тестирование зачастую выполнялось как отдельная фаза под конец цикла разработки, что увеличивало риск выявления критических ошибок на поздних этапах. Напротив, применение методологии DevOps предполагает интеграцию испытаний на каждом этапе жизненного цикла продукта [2]. Такой подход реализуется посредством использования автоматизированного тестирования, а также механизмов непрерывной интеграции (CI) и непрерывного развертывания (CD), что позволяет для каждого изменения в кодовой базе проводить всестороннюю проверку до выпуска в эксплуатацию [3].

«Целое всегда есть нечто большее, чем простая сумма его частей», - гласит онтологический принцип холизма [5]. Это значит, что свойства и поведение системы не могут быть полностью поняты, если рассматривать ее только как отдельные компоненты. В контексте DevOps и тестирования эта идея акцентирует внимание на важности всесторонней синергии между разработкой и тестированием, что позволяет обеспечить максимальную эффективность и качество конечного программного продукта. Так, целью данной работы является анализ синергетического эффекта от объединения DevOps и тестирования, превосходящего просто суммарное влияние отдельных компонентов. В рамках исследования ставится задача изучения влияния данной интеграции на повышение качества программного обеспечения, ускорение процессов разработки и оптимизацию общих бизнес-процессов, а также демонстрация практических результатов на примере реализованного проекта.

Бесспорно, эффективное тестирование в DevOps требует стратегического подхода, обеспечивающего соответствие принципам непрерывной интеграции и развертывания (CI/CD). Одной из ключевых стратегий является переход тестирования на ранние стадии жизненного цикла разработки («тестирование со сдвигом влево»), что позволяет своевременно идентифицировать и устранять дефекты. В DevOps это означает, что тестирование инициируется одновременно с процессом написания кода, позволяя минимизировать издержки, связанные с последующим исправлением ошибок. Также важную роль играет автоматизация тестирования, которая способствует повышению скорости, последовательности и надёжности выполнения тестовых сценариев без непосредственного участия человека. Сочетание раннего тестирования и автоматизации способствует более рациональному использованию ресурсов, снижению затрат на устранение недостатков и ускорению вывода продукта на рынок.

Важным шагом в оптимизации процессов является выбор соответствующих инструментов DevOps и тестирования. Системы контроля версий, такие как Git, инструменты CI / CD, такие как Jenkins, решения мониторинга и логирования, такие как Prometheus, Grafana и контейнеризация, такая как Docker, позволяют автоматизировать максимальное количество повторяющихся процессов. Это позволит поднять общую эффективность, а также снизит вероятность ошибок, связанных с человеческим фактором. Чтобы подробнее остановиться на системном подходе, нам нужно посмотреть на реальную модель интеграции и тестирования DevOps.

После изучения успешных практик внедрения DevOps в крупные корпорации, такие как Amazon, Capital One, Netflix, Spotify, Etsy, я стала применять данную методику для моего рабочего проекта. Старые методы разработки в проекте: отдельные ветки кода для разработчиков и развертывание вручную программного продукта. Это приводило к ошибкам, несоответствию моделям данных и задержкам в разработке. Однако DevOps и тестирование создали культуру совместной ответственности. В ежедневных статусах обсуждались насущные задачи и возникающие проблемы. GitLab CI [6] гарантировал выполнение тестов на момент merge request. Кроме того, система непрерывной доставки (CD) разворачивала приложение автоматически на тестовых стендах, система мониторинга (Prometheus, Grafana) [7] помогла разработчикам быстро реагировать на найденные дефекты. Итак, сочетание DevOps и тестирования указывает на оптимальность холистического подхода, что делает процедуру разработки эффективной.

Наконец, DevOps и тестирование еще сильнее объединяют разработчиков, тестировщиков и других членов команды, создавая систему взаимной ответственности за качество ПО на каждом этапе.

ЛИТЕРАТУРА

1. Maximilien De Bayser, Leonardo Guerreiro Azevedo, Renato Cerqueira “ResearchOps: The case for DevOps in scientific applications” – 2015.
2. Sujeewan Vijayakumaran “DevOps Frameworks, Techniques, and Tools” – 2025.
3. Saeed Parsa “Software Testing Automation Testability Evaluation, Refactoring, Test Data Generation and Fault Localization” – 2023.
4. Gene Kim, Jez Humble, Jez Humble, John Willis, Gene Kim, Patrick Debois “The DevOps Handbook How to Create World-Class Agility, Reliability, & Security in Technology Organizations” – 2021.
5. Holism. [Электронный ресурс] Режим доступа: <https://en.wikipedia.org/wiki/Holism>
6. GitLab CI: Feature Overview, Tutorial and Best Practices. [Электронный ресурс] Режим доступа: <https://codefresh.io/learn/gitlab-ci/>
7. Введение в мониторинг серверов с помощью Prometheus и Grafana. [Электронный ресурс] Режим доступа: <https://habr.com/ru/articles/652185/>

УДК 004.453

Н.Н. Мехоношин, А.У. Миниахметов (3 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ И РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ИНСТРУМЕНТА ПОИСКА СЕМАНТИЧЕСКИХ ОШИБОК AVICENNA

Отладка программных систем является неотъемлемой частью разработки ПО, которая позволяет выявлять и устранять ошибки программного кода, повышая надежность итогового продукта [4]. Существует множество различных инструментов, которые предназначены для поиска семантических ошибок в программном коде [3]. Одним из таких инструментов является Avicenna [1] – инструмент отладки, предназначенный для автоматического определения причин и условий сбоев в работе программы. Однако, Avicenna разработана на Python и поддерживает отладку и вызов кода только на «родном» языке, что ограничивает потенциал и масштаб использования инструмента в условиях современного многообразия языков программирования.

Цель работы – изучить и расширить возможности Avicenna на Java и C++. Для достижения этих целей необходимо решить следующие задачи:

1. Обзор существующих инструментов поиска семантических ошибок.
2. Проведение сравнительного анализа найденных вариантов.
3. Расширение Avicenna для отладки программ на Java и C++.

4. Демонстрация возможности отладки программ на Java и C++ с использованием Avicenna.

Архитектура решения изображена на рисунке 1. Она включает в себя следующие компоненты: интеграционный слой, реализованный на базе связки Java/Python, который обеспечивает механизм прямого вызова скомпилированных функций Java посредством Python-интерфейса, разработанного с использованием JPyре [2], а также интеграционный слой на базе связки C++/Python, реализующий механизм прямого вызова скомпилированных функций C++ посредством Python-интерфейса, разработанного с применением библиотеки pybind11 [6]. Модуль верификации ошибок (оракул) обнаруживает ошибки путем анализа кодов завершения программы и обработки исключений.

Подход реализован следующим образом. Для функций, написанных на Java, исходный код компилируется в байт-код, обеспечивающий совместимость со средой выполнения Python посредством виртуальной машины Java, запускаемой через JPyре. Для функций на C++ исходный код компилируется в машинный код, а для обеспечения совместимости с Python применяется интеграционный слой на базе pybind11. В обоих случаях использование интеграционных механизмов приводит к дополнительным временным затратам, связанным с преобразованием данных и переключением контекста, однако такой подход оказывается существенно быстрее, чем, например, взаимодействие через сокеты (py4j [5]). Инструмент Avicenna генерирует тестовые входные данные с использованием эволюционных алгоритмов, при этом каждый сгенерированный тестовый вход последовательно передается в скомпилированную функцию через соответствующий Python-интерфейс.

На этапе верификации оракул выполняет анализ результатов исполнения, включая проверку корректности выходных данных и фиксацию аномальных состояний.

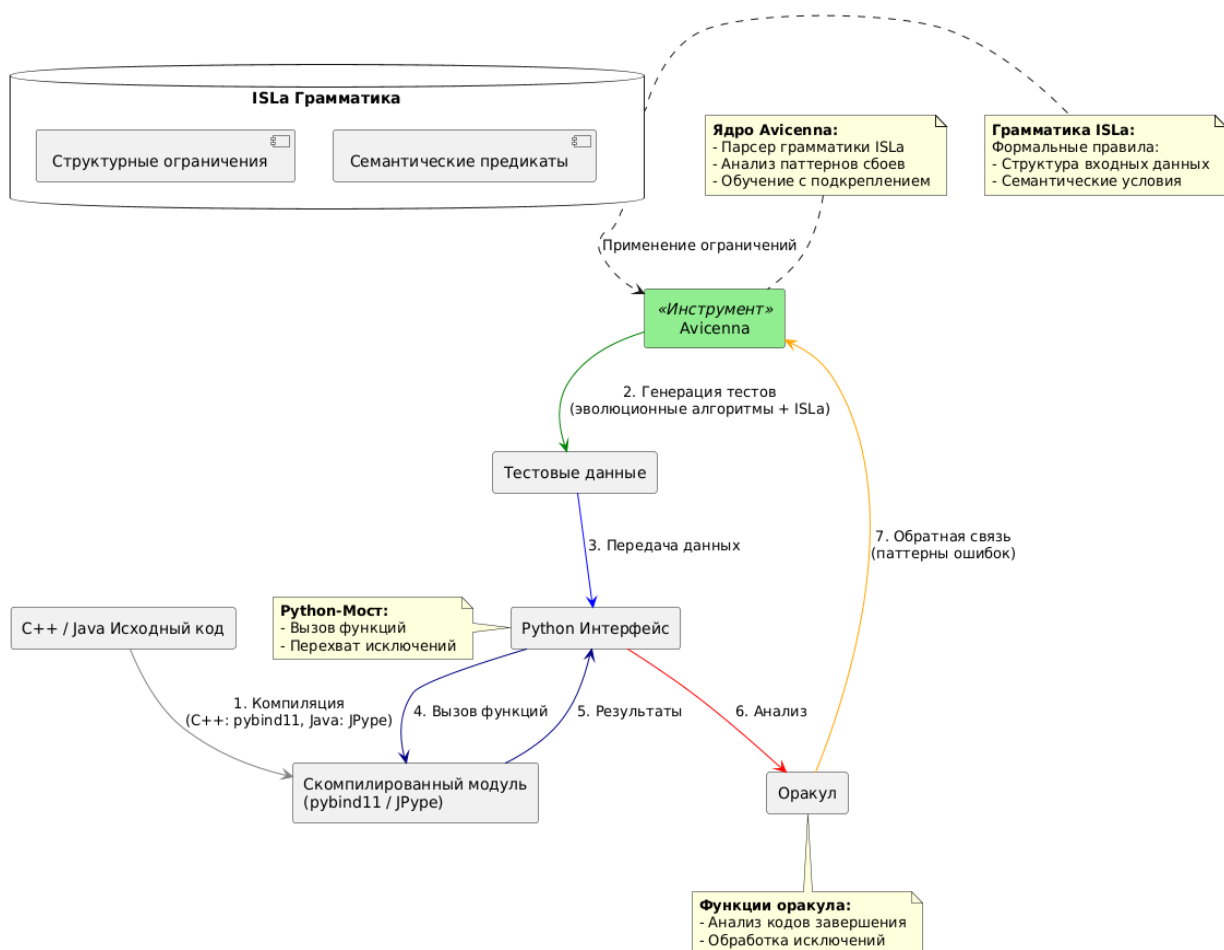


Рисунок 1 – Архитектура решения

В результате исследования разработано расширение Avicenna на Java и C++. Новая возможность протестирована, получены результаты предварительных экспериментов с отладкой программ на языках Java и C++.

ЛИТЕРАТУРА

1. Avicenna. [Электронный ресурс] Режим доступа: <https://github.com/martineberlein/avicenna> (дата обращения: 19.02.2025).
2. JPyre. [Электронный ресурс] Режим доступа: <https://jpyre.readthedocs.io/en/latest/> (дата обращения: 14.03.2025).
3. Eberlein, M., Smytzek, M., Steinhofel, D., Grunske, L., Zeller, A Semantic Debugging // Software Engineering 2024 (SE 2024). – Bonn: Gesellschaft für Informatik e.V., 2024. – С. 135–136. – Режим доступа: <https://dl.gi.de/server/api/core/bitstreams/d9a84e1c-d381-4c4b-9862-5bf73e2be251/content> (дата обращения: 23.02.2025).
4. Zeller, A. Why Programs Fail: A Guide to Systematic Debugging. – Berlin, Heidelberg: Springer, 2002. – 356 с. (дата обращения: 24.02.2025).
5. About Py4j. [Электронный ресурс] Режим доступа: <https://www.py4j.org/about.html#alternatives-to-py4j> (дата обращения: 14.03.2025).
6. Pybind11. [Электронный ресурс] Режим доступа: <https://pybind11.readthedocs.io/en/stable/basics.html> (дата обращения: 14.03.2025).

УДК 004.543

Д.П. Московских (3 курс бакалавриата),
О.Г. Мелентьев, д.т.н., профессор

РАЗРАБОТКА БЛОКА УПРАВЛЕНИЯ FFT НА ОСНОВЕ КОНЕЧНЫХ АВТОМАТОВ

Современные системы цифровой обработки сигналов широко применяют алгоритмы FFT для преобразования данных из временной области в частотную [1,2]. Для эффективного взаимодействия аппаратных компонентов (ядра FFT, twiddle ROM, внешней памяти) необходим специализированный блок управления, способный обеспечить корректную и синхронизированную настройку всех параметров [3]. В данной работе описывается разработка такого блока на основе конечного автомата, в котором последовательность состояний соответствует этапам инициализации, конфигурации, запуска и завершения работы FFT-модуля [4]. Блок управления реализован в модуле `fft_ctrl` и представляет собой конечный автомат, состоящий из ряда дискретных состояний. Ключевые этапы работы автомата включают:

1. Ожидание сигнала запуска (состояние `FFT_CTRL_STATE_INIT`).
2. Последовательная запись параметров: задание размера области памяти, определение адресов исходных и результирующих данных, а также загрузка коэффициентов для вычислений [5].
3. Запись управляющего слова, иницирующего вычисления.
4. Отслеживание статуса вычислительного ядра.
5. Возврат системы в исходное состояние с выдачей сигнала о завершении операции.

Применение макроса `REG_WRITE_OP` для записи в регистры обеспечивает компактное и единообразное описание операций, что снижает вероятность ошибок [4]. Модуль FFT (`main_fft`) интегрирует блок управления, вычислительное ядро FFT, блок twiddle ROM и внешнюю память (RAM) в единую систему. Чёткое разделение функциональных блоков и строгая регламентация интерфейсов обеспечивают масштабируемость и модульность системы [3].

Для повышения надежности и эффективности реализации алгоритма были разработаны дополнительные меры: проведён анализ временных задержек на каждом этапе работы

конечного автомата, применены методы синхронизации на уровне тактовой частоты, а также реализована проверка корректности передачи управляющих сигналов. Дополнительно выполнена оценка вычислительной сложности, что позволило оптимизировать использование аппаратных ресурсов при реализации системы. Приведённая формула FFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi kn}{N}}, \quad k = 0,1,\dots,N-1$$

отражает базовую математическую модель, лежащую в основе вычислительного ядра. Анализ показал, что параллельная обработка и распределённое вычисление могут существенно повысить производительность системы при работе с большими объёмами данных.

FFT_CTRL_STATE_INIT	Состояние инициализации. Ожидание сигнала запуска.
FFT_CTRL_STATE_WRITE_SIZE	Запись размера области памяти для FFT.
FFT_CTRL_STATE_WRITE_SRC	Задание адреса исходных данных.
FFT_CTRL_STATE_WRITE_DEST	Задание адреса для результирующих данных.
FFT_CTRL_STATE_WRITE_FACTOR[1-n]	Последовательная загрузка коэффициентов FFT.
FFT_CTRL_STATE_WRITE_CONTROL	Запись управляющего слова, инициирующего запуск FFT.
FFT_CTRL_STATE_READ_STATUS	Мониторинг статуса вычислительного ядра.
FFT_CTRL_STATE_FINISH	Завершающее состояние, сигнал о завершении работы FFT.
REG_WRITE_OP	Макрос для выполнения операции записи в регистр, обеспечивающий единообразие операций.

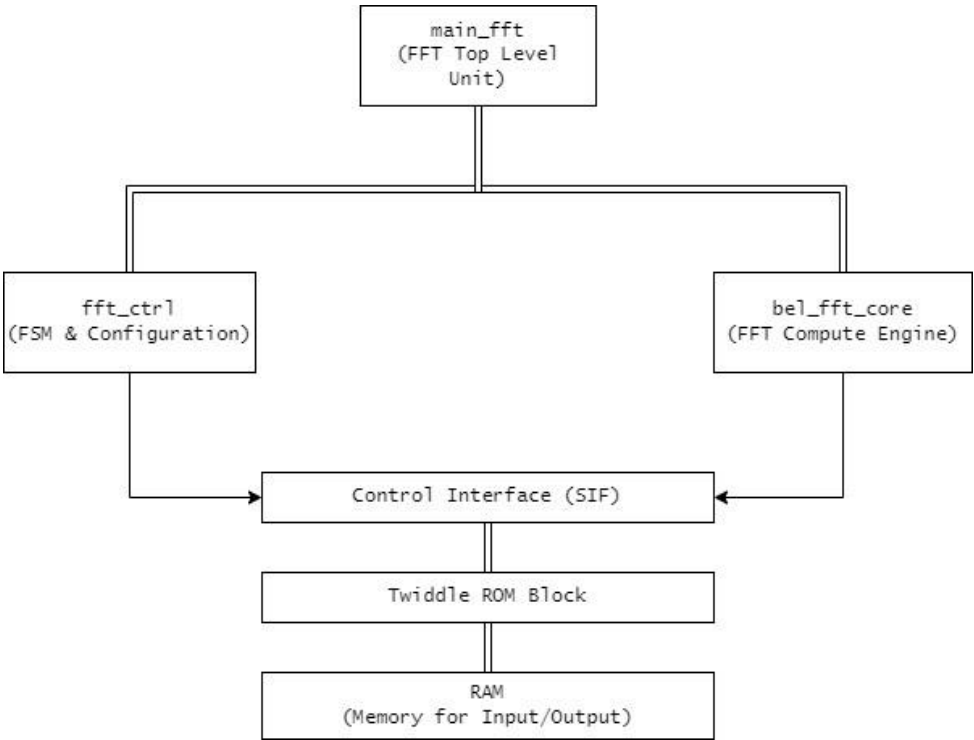


Рисунок1–Структурная схема системы

Была реализована система управления FFT, разработанная на языке SystemVerilog, которая обеспечивает последовательную и синхронизированную работу вычислительного ядра FFT, загрузку коэффициентов и контроль завершения работы модуля. Реализация включала оптимизацию временных задержек, проверку целостности управляющих сигналов и

эффективное использование аппаратных ресурсов, что позволяет системе работать с высокими скоростями и обеспечивает отказоустойчивость при обработке больших объёмов данных.

ЛИТЕРАТУРА

1. Oppenheim, A.V., Schaffer, R.W. *Discrete-Time Signal Processing*. 3rd ed. – Upper Saddle River: Prentice Hall, 2009.
2. Lyons, R.G. *Understanding Digital Signal Processing*. 3rd ed. – Upper Saddle River: Prentice Hall, 2011.
3. Proakis, J.G., Manolakis, D.G. *Digital Signal Processing: Principles, Algorithms, and Applications*. – Upper Saddle River: Prentice Hall, 2006.
4. Smith, S.W. *The Scientist and Engineer's Guide to Digital Signal Processing*. – California: California Technical Publishing, 1997.
5. Smith, S.W. *The Scientist and Engineer's Guide to Digital Signal Processing*. – California: California Technical Publishing, 1997.

УДК 004.94

И.А. Норицин (3 курс бакалавриата),
К.И. Брагин, ст. преподаватель

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ АЛГОРИТМОВ ПЛАНИРОВАНИЯ РЕСУРСОВ МОБИЛЬНОЙ СЕТИ ПРИ ПЕРЕДВИЖЕНИИ АБОНЕНТОВ

Современные мобильные сети сталкиваются с постоянно растущими требованиями к пропускной способности, качеству обслуживания (англ. QoS, Quality of Service) и стабильности соединения. Одним из аспектов, влияющих на работу сети, являются динамические изменения нагрузки на базовые станции, которые возникают в результате перемещения абонентов. Эти изменения могут привести к перегрузке отдельных элементов сети и, как следствие, к временному снижению качества связи. Чтобы обеспечить стабильный уровень обслуживания абонентов, необходимы эффективные механизмы управления ресурсами.

Для рационального управления ресурсами сети используются различные алгоритмы планирования на уровне доступа к среде (англ. MAC, Medium Access Control) [1], которые отвечают за распределение частотно-временных ресурсов и адаптируют схемы модуляции для абонентских устройств в зависимости от изменяющегося состояния канала и объёма поступающего трафика.

Имитационное моделирование – это один из инструментов для анализа и оценки эффективности таких алгоритмов. С его помощью можно воссоздать сложные сценарии, связанные с передвижением пользователей, изменением сетевых параметров и моделировать работу различных механизмов управления ресурсами. Это особенно важно для сетей 5G New Radio и будущего поколения 6G [2], где адаптивное распределение ресурсов играет ключевую роль в обеспечении высокой производительности.

Таким образом, основная цель этой работы – моделирование работы планировщика радиоресурсов сетей мобильной связи с учётом перемещения пользователей. Создание данной модели поможет выявить оптимальные стратегии управления радиоресурсами в условиях высокой мобильности абонентов, а также станет основой для изучения этой темы студентами.

Требуется разработать модель, имитирующую процессы в мобильной сети с наибольшей степенью соответствия реальным условиям. На начальных этапах планируется исследовать такие алгоритмы планирования, как Round Robin, Proportional Fair и Best CQI.

Модель MAC-планировщика была реализована при помощи языка программирования Python [3]. Были использованы такие библиотеки как NumPy, matplotlib, а также math.

Планировщик осуществляет распределение ограниченного количества ресурсных блоков в нисходящем канале (англ. downlink) [4] среди некоторого количества заранее заданных пользователей.

На вход планировщика подаются данные о пользовательских устройствах. Каждое устройство имеет ряд параметров, а именно: уникальный идентификатор, количество данных для передачи (в килобайтах), начальные координаты X и Y, скорость передвижения (в метрах в секунду), тип передвижения (пешеход, велосипед, автомобиль или поезд), а также начальный угол передвижения.

Перед запуском модели необходимо указать два обязательных параметра: ширину полосы (в мегагерцах) и время симуляции (в секундах). После того как симуляция работы планировщика была запущена, пользователи, исходя из их параметров передвижения, начинают перемещаться в пространстве в течение заданного времени, тем самым имитируя мобильность абонентов мобильной сети. В зависимости от расстояния до базовой станции, каждое пользовательское устройство получает значение сигнал/шум (SNR), которое потом преобразовывается устройством в индикатор качества канала (CQI).

Для расчёта значения сигнал/шум в данной работе используется такая модель распространения сигнала как Окумура-Хата и её модификация COST231 [5]. Уровень потерь радиосигнала в зависимости от удалённости пользователя рассчитывается по формуле (1):

$$PL(d) = 46.3 + 33.9 \cdot \log_{10}(f) - 13.82 \cdot \log_{10}(hBS) - a + s \cdot \log_{10}(d) + Lclutter \quad (1)$$

где f – несущая частота сигнала, МГц; d – расстояние между приёмником и передатчиком, км; hBS – высота подвеса антенны базовой станции, м;

Параметр a зависит от высоты антенны мобильного устройства hms , несущей частоты f и от типа местности. Так как модель планировщика имитирует работу при плотной городской застройке, то параметр a определяется по формуле (2):

$$a = 3.2 \cdot (\log_{10}(11.75 \cdot hms))^2 - 4.97 \quad (2)$$

Составляющая s зависит от высоты базовой станции hBS , от несущей частоты f и от расстояния между абонентом и базовой станцией d и определяется как (3):

$$\begin{cases} 44.9 - 6.55 \cdot \log_{10}(f), & \text{для } d \geq 1 \text{ км} \\ (47.88 + 13.9 \cdot \log_{10}(f) - 13.9 \cdot \log_{10}(hBS)) \cdot \frac{1}{\log_{10}(50)}, & \text{для } d < 1 \text{ км} \end{cases} \quad (3)$$

Последняя составляющая в выражении (1) – это константа $Lclutter$, значение которой зависит от типа местности. Учитывая, что модель функционирует в условиях плотной городской застройки, значение константы будет составлять 3.

Результатом работы модели планировщика является заполненная ресурсная сетка между множеством пользователей. Пример такой сетки, заполненной в ходе работы алгоритма Proportional Fair, можно увидеть на рисунке 1. Она показывает распределение ресурсных блоков между абонентами в течение 20 мс при ширине полосы 3 МГц. Один блок может занимать только один определённый пользователь.

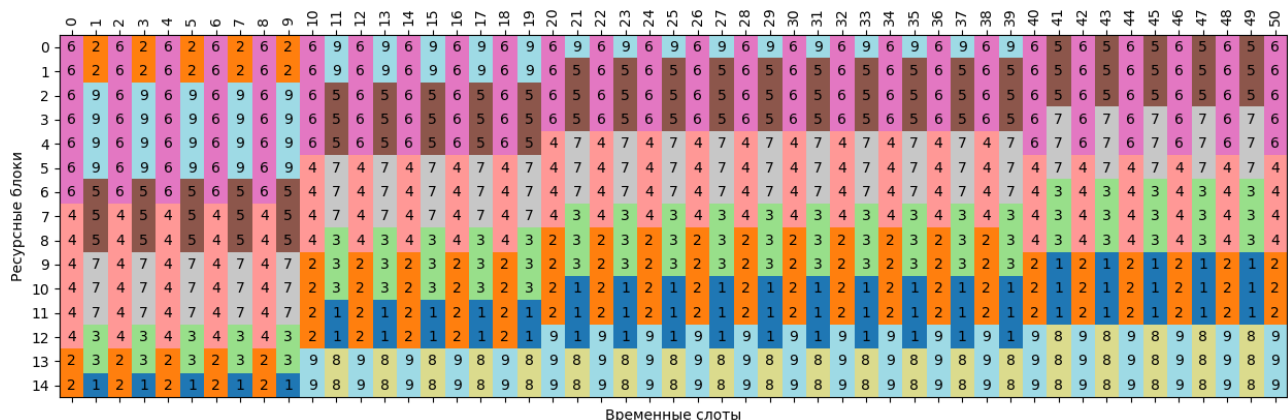


Рисунок 1 – Планирование ресурсов с помощью алгоритма Proportional Fair

ЛИТЕРАТУРА

1. LTE MAC Scheduler [Электронный ресурс]. – Режим доступа: <https://techlteworld.com/lte-mac-scheduler/>
2. Официальный портал МСЭ [Электронный ресурс] // ИМТ-2030 Пресс-релиз. – Режим доступа: <https://www.itu.int/en/mediacentre/Pages/PR-2023-12-01-IMT-2030-for-6G-mobile-technologies.aspx>
3. Python Programming Language [Электронный ресурс]. – Режим доступа: <https://www.python.org/>
4. Описание физического уровня LTE [Электронный ресурс]. – Режим доступа: https://anisimoff.org/lte/phy_description.html
5. Дроздова В. Г. Основы мобильных сетей LTE: учебно-методическое пособие. – Новосибирск: СибГУТИ, 2017. – 43 с.

УДК 004.65

Е.А. Палий (4 курс бакалавриата),
С.А. Нестеров, к.т.н., доцент

СОВМЕСТНОЕ ИСПОЛЬЗОВАНИЕ РЕЛЯЦИОННЫХ И NOSQL СУБД В СОВРЕМЕННЫХ ПРОГРАММНЫХ ПРОДУКТАХ НА ПРИМЕРЕ POSTGRESQL И REDIS

В условиях стремительного роста объёмов данных и разнообразия их структур современные программные системы требуют эффективных решений для управления и обработки данных. Традиционные реляционные системы управления базами данных (СУБД) обеспечивают надёжность и согласованность данных, однако могут испытывать трудности при работе с неструктурированными данными и масштабируемостью. В то же время NoSQL СУБД представляют собой совокупность различных типов хранилищ данных, включая ключ-значение, документоориентированные, колоночные и графовые. Они предлагают гибкость и высокую производительность, но часто уступают в обеспечении транзакционной целостности. В связи с этим всё большую популярность приобретает гибридный подход, сочетающий преимущества различных типов СУБД [1].

Реляционные СУБД управляют данными, структурированными в виде таблиц с чётко определёнными схемами, что обеспечивает целостность и согласованность информации. Однако жёсткость схемы и ограниченная масштабируемость могут стать препятствием при обработке больших объёмов неструктурированных данных [2]. NoSQL СУБД, включая документоориентированные, графовые и ключ-значение, предлагают более гибкие модели данных и горизонтальную масштабируемость, что делает их привлекательными для современных приложений, работающих с большими данными. Например, Redis – это СУБД в памяти с открытым исходным кодом, которая хранит данные в виде пар «ключ-значение» и обеспечивает высокую производительность [6].

Приложения с гибридным подходом к хранению данных, т. е. использующие как реляционные, так и NoSQL СУБД, позволяют объединить сильные стороны обеих технологий. Например, PostgreSQL – мощная объектно-реляционная СУБД с открытым исходным кодом, обеспечивающая надёжность и поддержку ACID-транзакций [3]. Redis – высокопроизводительная in-memory СУБД типа ключ-значение, часто используемая для кеширования и быстрого доступа к данным [4]. Совместное использование PostgreSQL и Redis позволяет разработчикам создавать масштабируемые и эффективные приложения, удовлетворяющие разнообразным требованиям к обработке данных. Такой подход обеспечивает гибкость и производительность, необходимые для современных программных продуктов.

В современных веб-приложениях производительность системы во многом зависит от скорости выполнения запросов к базе данных. Например, в системе управления пользователями аккаунты и их данные хранятся в PostgreSQL, а приложение часто запрашивает информацию о конкретных пользователях. Каждый раз выполнение SQL-запроса с JOIN требует времени, так как данные извлекаются с диска и проходят обработку на

уровне СУБД. При высокой нагрузке (например, при тысячах обращений в секунду) повторное выполнение таких запросов может замедлять работу системы.

Чтобы решить эту проблему, часто запрашиваемые данные можно кешировать в Redis. Сначала информация извлекается из PostgreSQL с помощью SQL-запроса, затем результат сохраняется в Redis в виде структуры "ключ-значение". При повторных обращениях приложение уже запрашивает данные из Redis, что значительно ускоряет их получение. Такой подход особенно эффективен для редко изменяемых данных, так как позволяет сократить нагрузку на основную СУБД.

Для подтверждения преимуществ гибридного подхода были проведены замеры времени выполнения запросов в PostgreSQL и Redis. В данном тесте измеряется разница во времени выполнения одного и того же запроса при разных условиях. Первый раз данные извлекаются из PostgreSQL с JOIN-операцией между таблицами ACCOUNT (100 000 записей) и CLS (500 000 записей). Затем полученный результат сохраняется в Redis, и последующие обращения выполняются уже через Redis. Это моделирует реальное использование кеша в приложении. Результаты представлены в таблице 1.

Таблица 1 – Сравнение времени выполнения запросов в PostgreSQL и Redis

Тип базы данных	SQL-запрос / Команда	Результат выполнения:	Время выполнения
PostgreSQL	EXPLAIN ANALYZE SELECT * FROM ACCOUNT JOIN CLS ON CLS.CREATOR_ID = ACCOUNT.ID;	Execution Time: 21.0 ms	21.0 ms
Redis	redis-cli GET "account:creator:464"	"value of account 464" (0.8 ms)	0.8 ms

Как видно из результатов, PostgreSQL обрабатывает и выполняет сложный SQL-запрос с объединением таблиц, обеспечивая целостность данных, но требуя значительно больше вычислительных ресурсов и времени на выполнение операции. В данном случае поиск, сопровождаемый соединением таблиц с JOIN на таблицах с 100 000 и 500 000 записями, занял 21,0 мс. В то же время Redis, работая как кеш, извлекает предварительно сохраненные данные всего за 0,8 мс. Это объясняется тем, что данные хранятся в оперативной памяти, а не извлекаются с диска в реляционной СУБД. Это снижает нагрузку на PostgreSQL при повторяющихся запросах. Однако Redis не поддерживает транзакции и сложные аналитические запросы, но ускоряет чтение неизменяемых данных. В данном эксперименте Redis выполняет роль кеша: часто запрашиваемые данные из PostgreSQL предварительно сохраняются в Redis, что позволяет сократить время отклика и уменьшить нагрузку на реляционную СУБД. Это особенно полезно для ускорения чтения неизменяемых или редко обновляемых данных.

Гибридный подход предоставляет возможность разрабатывать более гибкие и производительные программные продукты, способные эффективно обрабатывать различные типы данных. Комбинирование реляционных и NoSQL СУБД, таких как PostgreSQL и Redis, позволяет достичь оптимального баланса между надёжностью, производительностью и масштабируемостью, что особенно важно в условиях постоянно растущих объёмов информации. Кроме того, использование гибридных систем способствует более эффективному управлению ресурсами и повышению общей эффективности приложений.

ЛИТЕРАТУРА

1. Галигузова Е. В., Илларионова Ю. Е. Сравнение реляционных и нереляционных СУБД // Символ науки. – 2023. – №1–2. – С. 14-17.
2. Сравнительный анализ реляционных и нереляционных систем управления базами данных: особенности, преимущества и недостатки. [Электронный ресурс] Режим доступа: [https://nauchforum.ru/archive/MNF_interdisciplinarity/22\(254\).pdf#page=47](https://nauchforum.ru/archive/MNF_interdisciplinarity/22(254).pdf#page=47)

3. Новиков Б. А., Левин М. Ю. Сравнительный анализ производительности SQL и NoSQL СУБД // Компьютерные инструменты в образовании. – 2017. – №4. – С. 48-63.
4. Мартишин М.В. Базы данных: практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем. – Москва: ДМК Пресс, 2016. – 320 с.
5. Доуни А.Б. Алгоритмы и структуры данных: извлечение информации на языке Java. – Санкт-Петербург: Питер, 2013. – 288 с.
6. Redis и MongoDB: разница между базами данных NoSQL. [Электронный ресурс] Режим доступа: <https://aws.amazon.com/ru/compare/the-difference-between-redis-and-mongodb/>

УДК 004.432.4

А.В. Савченко (2 курс магистратуры),
О.Н. Петров, к.т.н., доцент

РАЗРАБОТКА ПЛАГИНА ДЛЯ НАПИСАНИЯ И ВЫПОЛНЕНИЯ ЛОГИЧЕСКИХ ПРОГРАММ

Экспертные системы имеют большое распространение в решении задач, связанных с накопленным опытом, правилами, которыми может руководствоваться эксперт при принятии решений. Такими задачами, например, являются задачи бизнес-планирования и обучения. [1]

В отличие от человека, системы позволяют предоставлять решение без присутствия эксперта, а также формировать свой «опыт» на основании знаний нескольких экспертов и делать прогнозы. [2]

Основой экспертной системы является база знаний, хранящая известные факты, благодаря которым, получая на вход запрос, система может предложить вывод. Реализация базы знаний часто основывается на языках декларативного, логического программирования. Примечательным плюсом логического языка является возможность не перебирать все возможные варианты, а выполнять возврат к подцели, когда вариант не удовлетворил условиям. [3] Наиболее известным языком логического программирования является Prolog, на основе которого в рамках данной работы формируется база знания для решения поставленной задачи логической программы.

Для решения задачи разработки плагина выбран движок Unreal Engine, который отмечается доступностью и универсальностью как для разработки, так и для пользования. [4] Целью работы является реализация возможности создания и исполнения логических программ внутри Unreal Engine посредством инструментальных средств на Blueprints, таким образом создавая базу знаний для экспертной системы.

В процессе реализации плагина требуется выделить особенности логического языка, которые необходимы для выполнения логической программы, и представить их в структурах и функциях скриптовой системы Unreal Engine Blueprints. Составляющими плагина, отвечающими за работу логической программы, являются:

1) классы, представляющие разные типы переменных входных и выходных данных, которые не определяются в процессе написания логической программы;

2) структуры, реализующие хранение предикатов – логических зависимостей от нескольких аргументов [5], выполняющие хранения истинных фактов, а также правил получения результата;

3) функции, организующие выполнение правил получения результата, в порядке, аналогичном тому, который реализует логический язык Prolog.

Важной особенностью логического языка, в том числе языка Prolog, является то, что результатом работы программы всегда является булево значение (Истина/Ложь), а входные данные представлены в виде полностью истинных утверждений. В дополнение к этой функциональности плагин позволяет получать в качестве результата частичную истинность утверждений.

Представление утверждений посредством нечеткой логики, то есть в виде числа от 0 до 1, указывающего, насколько утверждение истинно, позволяет работать в условиях нехватки информации, быстрее подстраиваться под новую информацию, а также получать более оптимизированный результат. [6]

С целью получить возможность оптимизировать получаемый результат, в реализации плагина создания логической программы выполнено представление базы данных и правил получения результата в виде утверждений со степенями истинности. Нечеткость позволяет формализовать частичное выполнение требований, условий, что является актуальным при решении практических задач, для которых может отсутствовать полностью удовлетворяющее всем требованиям решение. В качестве примера такой задачи рассматривается задача составления расписания с учётом предпочтений.

ЛИТЕРАТУРА

1. Kireev, T. D. Expert systems in knowledge-intensive fields of knowledge / T. D. Kireev, I. V. Markevich, P. Yu. Vaitekunaite // Молодые ученые в решении актуальных проблем науки : сборник материалов Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых (с Международным участием), Красноярск, 21–22 апреля 2022 года. – Красноярск: Федеральное государственное бюджетное образовательное учреждение высшего образования "Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева", 2022. – Р. 1015-1017. – EDN FQPOOH.
2. Boyko A. A. et al. The dynamic simulation model of calculating equipment purchase with the bond loan //Journal of Physics: Conference Series. - IOP Publishing, 2019. - . 1399. - . 3. - . 033120.
3. Патрикеева, Л. В. Логические языки как первые языки обучения программированию / Л. В. Патрикеева // Актуальные научные исследования в современном мире. – 2021. – № 12-12(80). – С. 224-230. – EDN JIYCFD.
4. Unreal Engine // Unreal Engine [Электронный ресурс]. Режим доступа: <https://www.unrealengine.com/en-US/eula/unreal>
5. Гутман, Г. Н. Язык программирования SWI-Prolog / Г. Н. Гутман, М. Н. Саушкин. – Самара : Самарский государственный технический университет, 2020. – 110 с. – EDN SWFTZL.
6. Кравец, Е. В. Анализ понятия "нечеткая логика", методы и области применения нечеткой логики / Е. В. Кравец, О. С. Солодова // "Цифра" - реальность, меняющая мир: готовность российской экономики к новым правилам игры : Материалы Национальной научно-практической конференции, Москва, 23 апреля 2019 года. Том 13. – Москва: Федеральное государственное унитарное предприятие "Всероссийский научно-исследовательский институт "Центр", 2019. – С. 110-112. – EDN YXOTFD.

УДК 004.432

А.Н. Слугин (4 курс бакалавриата),
С.Ю. Сартасов, ст. преподаватель

МЕТОДЫ РАБОТЫ С УТЕЧКАМИ ПАМЯТИ В JVM

При выполнении программ, написанных на языках, которые исполняются при помощи JVM, одной из распространённых ошибок является OutOfMemoryError [1], которая указывает на то, что во время выполнения программы происходят утечки памяти.

Для решения проблемы утечки памяти разрабатывалось множество инструментов, которые позволяют обнаруживать и анализировать утечки памяти, также были предложены способы борьбы с утечками памяти во время исполнения. Однако данные инструменты не были внедрены в версии JVM, которые используются в индустрии, так как все они требуют модификации различных частей JVM и при этом предоставляют только определённую вероятность определения утечки памяти, а также влекут за собой дополнительное потребление ресурсов. Поэтому самым популярным методом нахождения утечек памяти, применяемым в индустрии, является анализ снимка состояния кучи. Но данный способ не может предоставить

всю необходимую информация о причине возникновения утечки памяти для её эффективного устранения.

Целью работы является исследование целесообразности применения способов обнаружения и предотвращения утечек памяти в программах, исполняющихся при помощи виртуальной машины Java.

Для выполнения цели были поставлены следующие задачи:

1. Обзор методов, которые позволяют обнаруживать и предотвращать утечку памяти.
2. Определить методологию и сравнить рассмотренные методы.
3. Составить требования к методу для предотвращения утечек памяти.

Большинство существующих инструментов для обнаружения утечек памяти используют следующие методы [2]: анализ состояния программы, визуализация состояния кучи, статический анализ исходного кода, обнаружение неиспользуемых объектов в куче, анализ роста используемой памяти. Первые три способа не предназначены для борьбы с утечками памяти во время выполнения программы, а только помогают проанализировать поведение программы во время выполнения.

Были рассмотрены инструменты, основанные на обнаружении неиспользуемых объектов, существование которых может означать утечку памяти.

Bell — Bit Encoding Leak Location [3] — вероятностный подход, который позволяет определить место выделения каждого объекта в исходном коде с высокой точностью, имея достаточное количество объектов с закодированной информацией и известный ограниченный набор возможных мест выделения новых объектов.

Sleigh [3] — инструмент для обнаружения утечек памяти, который определяет неиспользуемые объекты и использует Bell для получения информации о месте выделения и последнего использования объекта.

LeakSurvivor [4] — инструмент для предотвращения утечек памяти во время выполнения программы для языков со сборщиком мусора. Идея работы заключается в периодическом перемещении объектов, которые потенциально являются утечкой памяти, из виртуальной памяти на диск.

Melt [5] — инструмент для борьбы с утечками памяти, которые гарантирует, что дополнительное потребление виртуальной памяти и увеличение времени выполнения будут пропорциональными размеру используемой виртуальной памяти.

Также были рассмотрены инструменты, основанные на анализе роста различных параметров во время работы приложения.

Panacea [6] — фреймворк для разработки ПО с возможностью самовосстановления.

Cork [7] — инструмент для выявления утечек памяти во время выполнения программы для языков со сборщиком мусора, основанный на анализе роста используемой памяти.

Таблица 1 – Сравнение инструментов для борьбы с утечками памяти

Инструмент	Определяет утёкшие объекты	Предотв ращает сбой программы	Требуе т изменения сборщика мусора	Требуе т изменения компилятора	Требуе т изменения исходного кода	Издержки по времени	Издержки по памяти
Sleigh	Да	Нет	Да	Нет	Нет	29%	0
LeakSurvivor	Да	Да	Да	Нет	Нет	23,7%	0-9 Мб
Melt	Да	Да	Да	Да	Нет	6%	-
Panacea	Нет	Да	Нет	Нет	Да	2,5%	-
Cork	Да	Нет	Да	Нет	Нет	2,3%	1%

В таблице 1 проведено сравнение инструментов для обнаружения и предотвращения утечек памяти. У всех инструментов дополнительное потребление виртуальной памяти является незначительным или вовсе отсутствует. Отсутствие дополнительного расхода памяти не свидетельствует о её нулевом использовании, а указывает на то, что эти инструменты эффективно уменьшают общий объем используемой памяти, справляясь с утечками объектов.

Рассмотренные инструменты не содержат механизма для информирования об обнаруженных утечках памяти. Поэтому для более эффективного исправления ошибок в исходном коде в реализации метода необходимо добавить возможность информирования, например с помощью логирования.

При использовании методов, которые перемещают объекты на диск, выполнение программы прекращалось из-за переполнения диска. Для предотвращения этого необходим механизм удаления объектов с диска.

Проанализировав способы работы с утечками памяти, были сформулированы задачи, которые должен выполнять метод для предотвращения утечек памяти:

1. обнаружение объектов, которые являются утечкой памяти;
2. информирование о возможной утечке;
3. перемещение обнаруженных объектов на диск с возможностью их восстановления;
4. удаление объектов с диска при переполнении диска.

ЛИТЕРАТУРА

1. Oumous Mouad. The Most Common Java Runtime Errors [Электронный ресурс]. – Электрон. текст. дан. – Режим доступа: <https://medium.com/thefreshwrites/the-most-common-java-runtime-errors-b6bc49a7716e>
2. Sor Srirama. Memory leak detection in Java: Taxonomy and classification of approaches // The Journal of Systems and Software. – 2014. – Vol. 96. – с. 139–151.
3. Bond McKinley. Bell: Bit-Encoding Online Memory Leak Detection // ACM SIGARCH Computer Architecture News. – 2006. – Vol. 34, № 5. – с. 61–72.
4. Tang Gao Qin. LeakSurvivor: towards safely tolerating memory leaks for garbage-collected languages // USENIX 2008 Annual Technical Conference. – 2008. – с. 307–320.
5. Bond McKinley. Tolerating Memory Leaks // OOPSLA'08: Proceedings of the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications. – 2008. – с. 109–126.
6. Breitgand Goldstein. PANACEA – Towards a Self-healing Development Framework // 10th IFIP/IEEE International Symposium on Integrated Network Management. – 2007.
7. Jump McKinley. Detecting memory leaks in managed languages with Cork // Software – Practice and Experience. – 2009. – Vol. 40, № 1. – с. 1–22.

УДК 004.415.2

А.К. Сопрачев (2 курс магистратуры),
П. Д. Дробинцев., к.т.н., доцент

ПРЕИМУЩЕСТВА ПРИМЕНЕНИЯ SERVERLESS-ПОДХОДА ПРИ РАЗРАБОТКЕ МИКРОСЕРВИСНЫХ ПРИЛОЖЕНИЙ

В современной практике разработки backend-приложений общепринятым подходом к решению задач географического и вычислительного (горизонтального) масштабирования является микросервисная архитектура. Для оркестрации и управления микросервисами практически во всех случаях используют Kubernetes. В данной работе рассматривается альтернативный подход к оркестрации микросервисов — Serverless, а также оцениваются его преимущества и недостатки.

В основе Serverless подхода лежит концепция, при которой разработчику не нужно управлять инфраструктурой на уровне серверов. Вместо этого используются облачные сервисы, предоставляющие в аренду вычислительные ресурсы (объем оперативной памяти, процессорное время, объем хранилища и количество запросов) по модели оплаты по фактическому использованию (pay-as-you-go).

В отличие от традиционного подхода Kubernetes, где необходимо оплачивать выделенные мощности кластера независимо от фактической нагрузки, основным

преимуществом Serverless является возможность окращение расходов за счёт оплаты фактически используемых мощностей без потери автоматического масштабирования. Таким образом необходимо оплачивать только те серверные ресурсы, которые действительно были использованы приложением, что эффективно при переменной нагрузки. Кроме того, этот подход "масштабируется в ноль", что означает полное отсутствие потребления ресурсов при отсутствии нагрузки.

Другим значимым преимуществом является отсутствие необходимости в администрировании и обслуживании серверов, что позволяет разработчикам концентрироваться на создании и улучшении функционала.

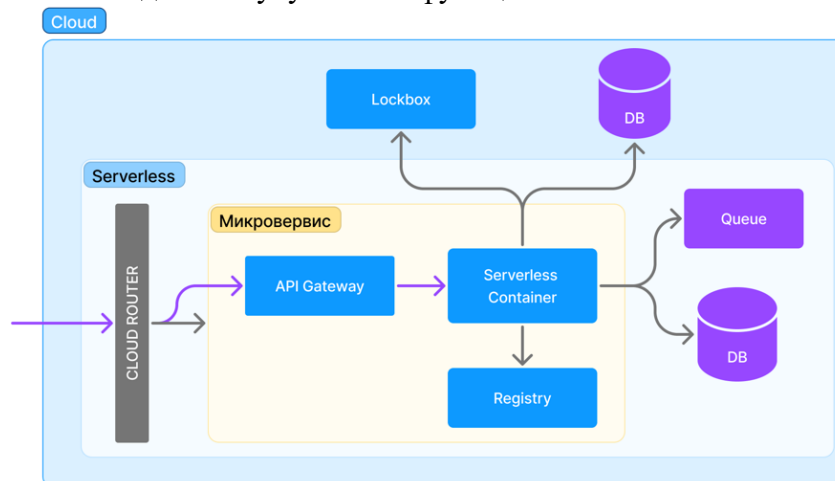


Рисунок 1 – Взаимодействие Serverless сервисов

Реализуется этот подход с помощью использования Serverless сервисов, общие взаимосвязи которых представлены на рисунке 1. Основными сервисами являются:

- API Gateway – внешний интерфейс приложения, описанный по спецификации OpenAPI и являющийся точкой входа.

- Serverless Containers – запускает Docker контейнеры в бессерверном режиме. На каждый запрос создаётся новый экземпляр контейнера, что обеспечивает горизонтальное масштабирование. Тарификация за время использования процессора и оперативной памяти, с квантизацией по 10мс.

- Serverless Database – бессерверные, обычно документоориентированные базы данных с оплатой за использование дискового пространства и количество обращений (например, DynamoDB).

- Object Storage – бессерверное хранилище объектов по протоколу S3, тарификация происходит по объёму данных и числу обращений к ним.

При этом стоит отметить, что Serverless-подход требует строгой архитектурной дисциплины, в частности минимизации зависимостей и соблюдения принципов чистой архитектуры для обеспечения лёгкого масштабирования и независимости компонентов. Также проблемой может стать ограничение облачных провайдеров, которые реализуют не полный перечень сервисов. Однако, применение Serverless подхода не исключает использование классических (не Serverless) сервисов.

Для преодоления сложности ручного конфигурирования облачной инфраструктуры широко применяется подход Infrastructure as Code с использованием Terraform. Этот инструмент позволяет декларативно описывать инфраструктуру в виде конфигурационных файлов, которые можно хранить вместе с кодом микросервисов в системе контроля версий. Совмещение Terraform с Serverless-архитектурой позволяет одной командой разворачивать и обновлять не только production- и dev-окружения, но и создавать новую копию backend на каждый PullRequest, что крайне полезно для проведения End-to-End тестирования.

Таким образом, применение Serverless-технологий в микросервисной архитектуре позволяет эффективно использовать ресурсы, значительно снизить затраты на инфраструктуру и упростить её администрирование.

ЛИТЕРАТУРА

1. Трушин Д. С., Калашников А. О. Применение Terraform для автоматизации управления облачной инфраструктурой // Вестник МГТУ им. Н. Э. Баумана. Серия: Информатика, вычислительная техника и управление. – 2021. – № 4 (142). – С. 65–81.
2. Brisals, S. Serverless Development on AWS: Building Enterprise-Scale Serverless Solutions. O'Reilly Media. – 2020.
3. Jonas E., Schleier-Smith J., Sreekanti V. Cloud Programming Simplified: A Berkeley View on Serverless Computing // arXiv preprint arXiv:1902.03383. – 2019.
4. Kelly, D., Glavin, F. G., & Barrett, E. Denial of wallet – Defining a looming threat to serverless computing. Journal of Information Security and Applications, 58. – 2021
5. Yandex Cloud Serverless. [Электронный ресурс] Режим доступа: <https://yandex.cloud/ru/solutions/serverless>
6. Terraform by HashiCorp. [Электронный ресурс] Режим доступа: <https://www.terraform.io>

УДК 004.453

А.К. Селиверстова (3 курс бакалавриата),
В.А. Пархоменко, ст. преподаватель

ИССЛЕДОВАНИЕ И ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ СРЕДСТВ ТЕСТИРОВАНИЯ ИНФРАСТРУКТУРЫ С ПОМОЩЬЮ INSPEC, TERRATEST, TEST KITCHEN

Современные инструменты управления инфраструктурой нуждаются в автоматизации, обеспечивающей воспроизводимость развертывания. Важной концепцией DevOps является использование подхода “инфраструктуры как кода” (далее – IaC), который позволяет управлять ресурсами с помощью программного кода [2]. Одним из таких инструментов является Terraform, позволяющий с помощью единого языка конфигурации работать с различными инфраструктурными ресурсам [2]. Продукт выбран среди аналогов благодаря его широким возможностям автоматизации, кроссплатформенности и декларативному подходу к управлению инфраструктурой, что обеспечивает предсказуемость и воспроизводимость развертывания [4].

Однако с ростом сложности инфраструктуры возрастает потребность в инструментах, способных автоматизировать тестирование развернутых инфраструктурных компонентов, включая их корректность, безопасность и соответствие заданным параметрам. Эффективное тестирование инфраструктуры позволяет выявлять ошибки и обеспечивать соответствие ресурсов заданным требованиям. Для этого применяются различные инструменты. В данной работе будут исследованы такие средства тестирования инфраструктуры, как InSpec, Terratest, Test Kitchen [1, 5]. Приведенные инструменты тестирования инфраструктуры преследуют общую цель — автоматизированную проверку развернутых ресурсов на корректность, соответствие требованиям и работоспособность. InSpec, Terratest и Test Kitchen пересекаются в своих возможностях, но акцентируются на разных аспектах тестирования. InSpec проверяет безопасность и соответствие конфигурации установленным стандартам, Terratest позволяет проводить инфраструктурные тесты, верифицируя развернутые ресурсы, а Test Kitchen фокусируется на интеграционном тестировании.

Целью работы является анализ и сравнение инструментов для автоматизированного тестирования инфраструктуры, выявление их преимуществ и недостатков на примере Terraform, InSpec, Terratest, Test Kitchen [1, 6]. Для достижения этой цели необходимо решение следующих задач:

1. Изучить принципы IaC и автоматизированного тестирования в DevOps.

2. Рассмотреть Terraform, InSpec и аналоги (Terratest, Test Kitchen).
3. Создать инфраструктуру с помощью Terraform.
4. Провести экспериментальное тестирование инфраструктуры с использованием указанных инструментов.
5. Сравнить подходы, выявить их сильные и слабые стороны.

С целью детального понимания применяемых инструментов была проведена их сравнительная характеристика. В таблице 1 представлено сравнение InSpec, Terratest, Test Kitchen по ключевым параметрам. Представленные инструменты обладают рядом схожих характеристик: open-source code, типом инструментов – библиотека (фреймворк), отсутствием API на языках, отличных от языка разработки [3].

Таблица 1 – Описательное сравнение инструментов тестирования инфраструктуры

Инструмент	Лицензия	Язык реализации	Основная область тестирования
InSpec	Пробная (ограничение на 10 хостов)	Ruby	Безопасность, конфигурация
Terratest	Бесплатная	Go	Инфраструктурное тестирование
Test Kitchen	Бесплатная	Ruby	Интеграционное тестирование

Из таблицы 1 следует, что инструменты имеют разное основное предназначение. InSpec в первую очередь ориентирован на тестирование безопасности и конфигурации, тогда как Terratest и Test Kitchen используются для тестирования инфраструктуры и интеграционного тестирования соответственно. Тем не менее, InSpec может быть использован и для проверки инфраструктуры, например, при валидации соответствия развернутых ресурсов заданным требованиям. В связи с этим, в данной работе рассматриваются сценарии, в которых эти инструменты могут быть использованы для тестирования инфраструктуры, что делает их сопоставимыми.

Эксперимент проводился на инфраструктурных ресурсах, созданных с помощью Terraform, с целью сравнения быстродействия инструментов тестирования и оценки затраченных ресурсов (оперативной памяти). Для создания контейнера, обрабатывающего HTTP-запросы, использовался провайдер Docker и образ nginx. Контейнер был настроен для прослушивания входящих запросов через порт 8000 на хостовой машине.

Для тестирования использовались инструменты InSpec, Terratest и Test Kitchen. Анализ производительности и потребляемых ресурсов проводился с помощью инструментов мониторинга. Полученные данные позволили выявить закономерности в работе инструментов и определить оптимальные сферы их применения.

ЛИТЕРАТУРА

1. tecRacer. Testing Terraform with InSpec – Part 1 [Электронный ресурс]. – URL: <https://www.tecracer.com/blog/2021/10/testing-terraform-with-inspec-part-1.html> (дата обращения: 19.02.2025).
2. HashiCorp. Terraform Documentation [Электронный ресурс]. – URL: <https://developer.hashicorp.com/terraform/docs> (дата обращения: 19.02.2025).
3. Chef Software. InSpec Documentation [Электронный ресурс]. – URL: <https://docs.chef.io/inspec/> (дата обращения: 19.02.2025).
4. Dinu F., Ninawe S.. 11 популярных альтернатив Terraform, которые стоит знать в 2025 году [Электронный ресурс] // Spacelift Blog. URL: <https://spacelift.io/blog/terraform-alternatives> (дата обращения: 21.02.2025).
5. Testing Terraform with Kitchen and Inspec [Электронный ресурс] // Rearc Blog. URL: <https://www.rearc.io/blog/testing-terraform-with-kitchen-and-inspec> (дата обращения: 22.02.2025).
6. Ninawe S.. Что такое Terratest и как им пользоваться [Электронный ресурс] // Spacelift Blog. URL: <https://spacelift.io/blog/what-is-terratest> (дата обращения: 25.02.2025).

УДК 004.42

А.А. Андрианов (2 курс магистратуры),
И.В. Никифоров, к.т.н., доцент

МЕТОД ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ХРАНИЛИЩАМИ ПОЛЬЗОВАТЕЛЬСКИХ ЗАПИСЕЙ В LINUX ДИСТРИБУТИВАХ

Системы идентификации и управления пользовательскими данными (Identity and Access Management, IAM) сегодня активно используются в сферах хранения данных, облачных вычислений, разработки программного обеспечения, кибербезопасности и многих других [1]. Наиболее востребованными на сегодняшний день системами на рынке IAM являются Microsoft Entra ID, Microsoft Active Directory, Keycloak, AWS IAM и Google Cloud IAM [1]. Для идентификации, аутентификации и авторизации пользователей, хранящихся в IAM системах, должна быть организована совместимость с целевой системой. Поэтому актуальной является задача изучения подходов интеграции внешних и локальных хранилищ о пользовательских записях в Linux дистрибутивах и их совершенствование.

Для работы с дистрибутивами на основе Linux базы пользователей должны иметь совместимость с Name Service Switch (NSS) – стандартным набором модулей и UNIX/glibc библиотек для работы с записями пользователей и групп [2]. Подключение модулей к NSS напрямую позволяет производить аутентификацию и разграничивать доступ к данным на основе разрешений файлов и информации из пользовательской записи.

Альтернативой подключения IAM к NSS может стать использование протокола Varlink, позволяющее предоставлять стандартизированное API для модификации записей, расширять возвращаемые поля, добавлять пагинацию и фильтрацию запросов [3]. Использование Systemd Userdbd в свою очередь позволяет организовывать поддержку совместимости с NSS путем реализации согласованного Varlink API интерфейса [4].

Таким образом целью работы является повышение эффективности работы с пользовательскими записями за счет нового подхода организации подключения хранилищ на основе встраиваемых сервисов в дистрибутив Linux.

Для достижения поставленной цели необходимо решение задач, перечисленных ниже.

1. Выявить общие требования к методу на основе сравнительного анализа пользователей системы и существующих IAM систем.
2. Изучить и сравнить подходы для получения информации о пользователях.
3. Спроектировать архитектуру с использованием полученного метода.
4. Реализовать метод в виде набора встраиваемых сервисов.
5. Оценить эффективность метода в ходе проведения экспериментальных исследований.

По результатам анализа наиболее популярных IAM систем можно прийти к выводу, что решение должны быть модульным, поддерживать возможность ускорения запросов путем

фильтрации и пагинации. Анализ пользователей системы показал, что решение должно поддерживать возможность модификации записей, управление записями через стандартные механизмы Linux и известные сетевые протоколы (gRPC, REST API и другие), а также должно поддерживать возможность введения собственной схемы представления данных.

Функциональная схема организации управления хранилищами на основе протокола Varlink приведена на рисунке 1. Ключевой особенностью является возможность модульного подключения различных источников с использованием сервисов-провайдеров, а также возможность прямых запросов к хранилищу по собственным API. Для обеспечения совместимости используется система Systemd Userdbd, подключаемая к NSS в виде готового модуля [4-5].

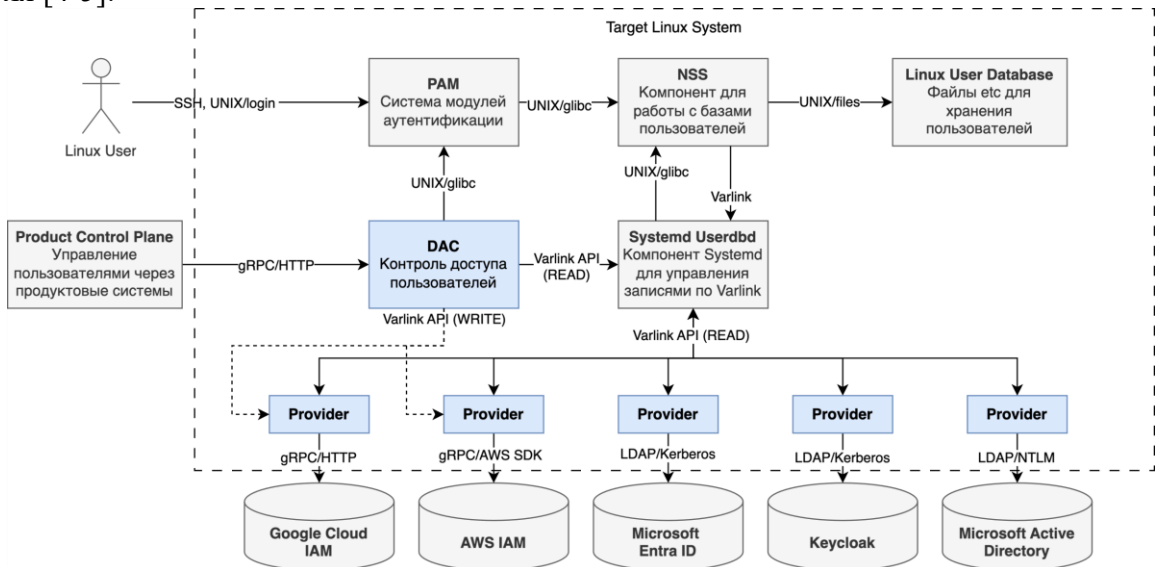


Рисунок 1 – Функциональная схема организации управления хранилищами пользовательских записей

Программные средства были разработаны на языке Golang с использованием пакета для работы с Varlink API. Для обеспечения обратной совместимости с Linux системой были использованы сервисы-провайдеры, предоставляемые утилитой Systemd Userdbd [4], позволяющие производить параллельный опрос баз пользователей по согласованному интерфейсу с дальнейшим преобразованием данных в формат, используемый NSS. Модульное тестирование полученного подхода проводилось с использованием пакета Testify, интеграционное - с использованием пакета Testcontainers.

По результатам экспериментов, представленным на рисунке 2, было выявлено, что разработанный подход с использованием прямого обращения к сервисам-провайдерам данных в 4 раза эффективнее использования UNIX/glibc вызовов и в 2 раза эффективнее использования сервисов-провайдеров, предоставляемых Systemd Userdbd.

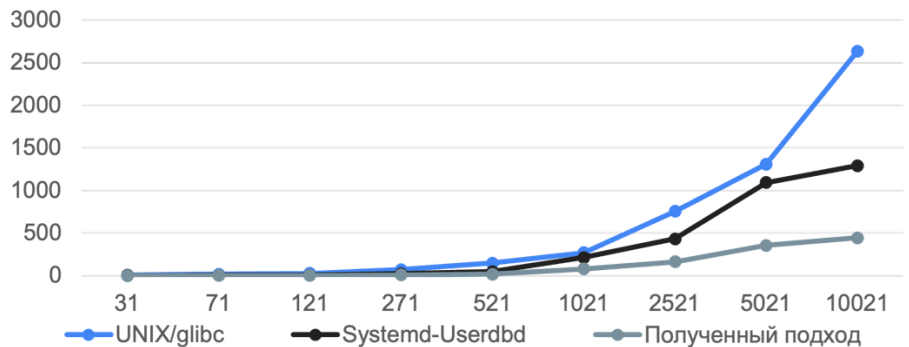


Рисунок 2 – Сравнение времени выполнения запроса (мс) от количества пользовательских записей

Разработанные программные средства могут быть использованы для параллельного подключения IAM систем, разграничения доступа и идентификации пользователей, ускорения запросов и повышения безопасности системы.

ЛИТЕРАТУРА

1. Identity And Access Management // 6sense URL: <https://6sense.com/tech/identity-and-access-management> (дата обращения: 12.03.2025).
2. Зылева П.С., Пестов И.Е., Тремель И.С., Юрова У.С. Интеграция клиентской машины на операционной системе Linux в домен Active Directory // АКТУАЛЬНЫЕ ПРОБЛЕМЫ ИНФОТЕЛЕКОММУНИКАЦИЙ В НАУКЕ И ОБРАЗОВАНИИ (АПИНО 2023) Сборник научных статей. XII Международная научно-техническая и научно-методическая конференция. В 4 т. Том 1. - СПб.: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2023. - С. 559-564.
3. Interface Definition // Varlink URL: <https://varlink.org/Interface-Definition> (дата обращения: 12.03.2025).
4. User/Group Record Lookup API via Varlink // Systemd URL: https://systemd.io/USER_GROUP_API (дата обращения: 12.03.2025).
5. nss-systemd // Freedesktop URL: <https://www.freedesktop.org/software/systemd/man/latest/nss-systemd.html> (дата обращения: 13.03.2025).

УДК 004.451.51

М. Гавриленко (2 курс бакалавриата),
А.И. Васенина, инженер-исследователь

РЕАЛИЗАЦИЯ ЛОГ-СТРУКТУРИРОВАННОГО БЛОЧНОГО УСТРОЙСТВА В ЯДРЕ LINUX

Современные системы хранения данных предъявляют высокие требования к производительности операций ввода-вывода на накопителях. Одной из ключевых проблем является снижение производительности при выполнении большого числа случайных операций записи, что приводит к увеличению времени доступа к данным.

Для решения этой проблемы предлагается применение лог-структурированного подхода, который преобразует случайные записи в последовательные, в следствие чего – повышает производительность. Лог-структурированный подход известен давно, однако до сих пор остаются нерешёнными вопросы, касающиеся его оптимальной реализации и места интеграции в систему. Для определения наиболее эффективного способа его внедрения необходимы дальнейшие исследования.

Проект реализован в виде модуля ядра Linux, предоставляющего интерфейс виртуального блочного устройства с лог-структурированной адресацией. Входящие запросы записи поступают на виртуальное блочное устройство и перенаправляются на конечное блочное устройство, последовательно заполняя накопитель. Чтение осуществляется через таблицу блоков, которая заполняется при операции записи. Таблица хранит данные о соотношении логических и физических адресов запросов. При реализации лог-структурированного подхода важно учитывать такие требования, как высокая скорость записи и чтения данных и минимизация времени доступа к данным.

Выбор структуры данных для хранения таблицы влияет на скорость поиска и обновления записей, что критично с точки зрения производительности. На текущем этапе работы была реализована базовая версия устройства, использующая синхронные реализации структур данных. В качестве структур данных были выбраны B+ дерево, хеш-таблица, список с пропусками и красно-черное дерево, обладающие оптимальными асимптотическими оценками сложности операций. Для большей гибкости виртуального блочного устройства была реализована обработка блоков разного размера и возможность добавления множественных пар “виртуальное блочное устройство — конечное блочное устройство”.

В рамках данной работы было разработано базовое виртуальное блочное устройство с лог-структурированной адресацией. Был внедрен принцип лог-структурированности на основе различных структур данных и добавлен единый интерфейс взаимодействия со структурами данных. В качестве проверки было проведено тестирование целостности реализованного виртуального блочного устройства.

С реализованной функциональностью можно ознакомиться на GitHub¹. В дальнейшем планируется:

1. Реализовать “lock-free” версии структур данных.
2. Протестировать блочное устройство на сервере.
3. Провести анализ производительности различных структур данных.

Работа выполнена в Лаборатории технологий программирования инфраструктурных решений, СПбГУ.

ЛИТЕРАТУРА

1. Curless B. CSE 326: Data Structures B-Trees and B+ Trees. (URL: <https://courses.cs.washington.edu/courses/cse326/08sp/lectures/11-b-trees.pdf>) (дата обращения: 26 декабря 2024 г.).
2. De Jonge W., Kaashoek M. F., Hsieh W. C. The logical disk: A new approach to improving file systems // Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles. — 1993.
3. Hu Y., Yang Q. DCD—disk caching disk: A new approach for boosting I/O performance // ACM SIGARCH Computer Architecture News. — 1996. — Т. 24, № 2. — С. 169–178.
4. Mohamed S. El Sayed. Analysis and Optimization of Storage IO in Distributed and Massive Parallel High Performance Systems. 2011. (URL: https://www2.informatik.uni-stuttgart.de/bibliothek/ftp/medoc.ustuttgart_fi/MSTR-3196/MSTR-3196.pdf) (дата обращения: 26 декабря 2024 г.).
5. Ousterhout J., Douglass F. Beating the I/O bottleneck: A case for log-structured file systems // ACM SIGOPS Operating Systems Review. — 1989. — Т. 23, № 1. — С. 11–28.
6. Xie Z., Cai Q., Jagadish H. V. и др. Parallelizing skip lists for in-memory multi-core database systems // 2017 IEEE 33rd International Conference on Data Engineering (ICDE). — IEEE, 2017. — С. 119–122.
7. Rosenblum M., Ousterhout J. K. The design and implementation of a log-structured file system // ACM Transactions on Computer Systems (TOCS). — 1992. — Т. 10, № 1. — С. 26–52.

УДК 004.4

Б.А. Денисенко (2 курс магистратуры),
О.А. Юсупова, ст. преподаватель,
И.В. Никифоров, к.т.н., доцент

РЕАКТИВНАЯ СИСТЕМА НА ОСНОВЕ NETLINK НОТИФИКАЦИЙ: GRPC-ДОСТАВКА И HOTPLUG ДЛЯ УПРАВЛЕНИЯ СЕТЕВОЙ КОНФИГУРАЦИЕЙ

Оперативное реагирование на изменения состояния сети является одним из ключевых требований для современных систем управления сетевыми конфигурациями [1]. Традиционные подходы, такие как периодический опрос (поллинг) [2], не всегда способны обеспечить надежное и ресурсоэффективное реагирование, особенно в условиях высокой нагрузки или при работе с большим количеством устройств. Реактивные системы [3], напротив, позволяют мгновенно реагировать на события, минимизируя задержки и использование ресурсов. Однако при использовании нотификаций, особенно в распределённых системах с доставкой через gRPC, важно учитывать возможные потери уведомлений. Это требует разработки механизмов, обеспечивающих надежность и отказоустойчивость [4] при передаче и обработке событий.

¹ <https://github.com/qrutyy/lb-bdd>

Целью данной работы является создание эффективного и надежного механизма для мониторинга сетевых событий и динамического управления сетевыми конфигурациями.

Для отслеживания состояния сети традиционно используются два подхода: поллинг и нотификации. Поллинг предполагает периодический опрос состояния системы, что может приводить к избыточному использованию процессорного времени и задержкам в обнаружении изменений. В отличие от этого, нотификации позволяют системе мгновенно реагировать на события, такие как изменение сетевых интерфейсов или появление новых устройств, без необходимости постоянного опроса. В данной работе был выбран подход с использованием нотификаций на основе Netlink, так как он обеспечивает минимальные задержки и эффективное использование ресурсов. Netlink, как механизм ядра Linux, предоставляет возможность получать уведомления о сетевых событиях в реальном времени, что делает его идеальным выбором для построения высокопроизводительных систем управления сетевыми конфигурациями.

Архитектура системы представлена на рисунке 1. Основная идея системы заключается в том, чтобы минимизировать использование ресурсов за счёт использования единого провайдера нотификаций на основе Netlink с одним сокетом. Это позволяет избежать создания множества провайдеров для каждого клиента или типа уведомлений.

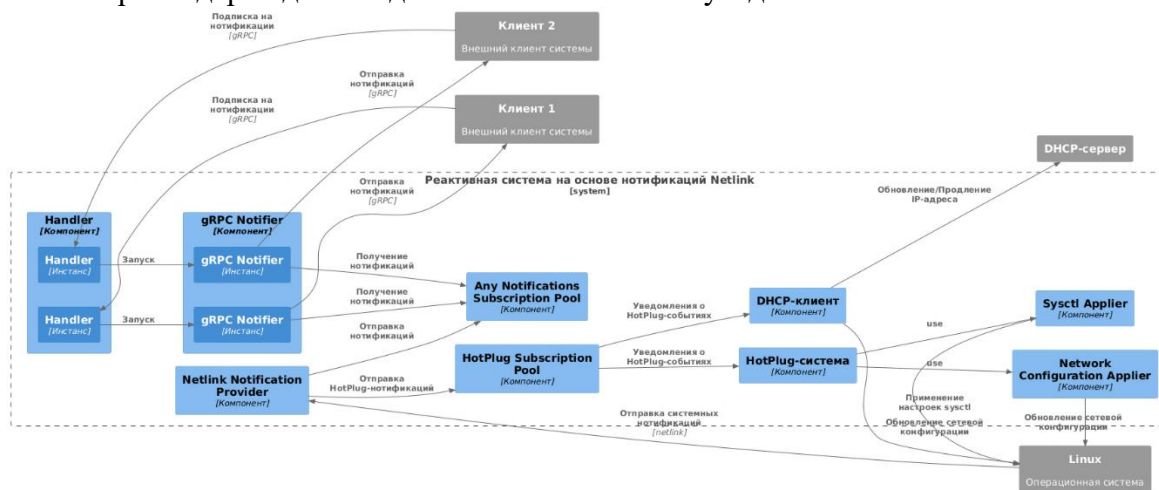


Рисунок 1 – Архитектура реактивной системы на основе Netlink-нотификаций

Для того чтобы разные клиенты, подписанные на один и тот же тип нотификаций, не создавали новых провайдеров, была разработана сущность Subscription Pool. Этот компонент ретранслирует нотификации всем клиентам, подписанным на соответствующий тип событий. Таким образом, нотификации доставляются всем заинтересованным сторонам без дублирования ресурсов.

Внутренняя HotPlug-система также использует Netlink-провайдер нотификаций. Когда HotPlug-система получает уведомление о появлении нового устройства, она запускает процесс применения персистентной сетевой конфигурации и всех настроек sysctl, которые были заданы пользователем. В случае переполнения Netlink-сокета HotPlug-система переподписывается на провайдер нотификаций заново. После переподписки система повторно применяет персистентную конфигурацию и настройки sysctl, так как в процессе переподписки могла быть потеряна нотификация о событии.

DHCP-клиент также использует нотификации от Netlink-провайдера. При получении уведомления о появлении нового устройства DHCP-клиент перезапускается (если должен был быть запущен на появившемся устройстве), при необходимости обращается к DHCP-серверу и назначает IP-адрес на новое устройство.

В результате работы была предложена и программно реализована архитектура реактивной системы на основе Netlink-нотификаций. Она гарантирует применение сетевой конфигурации даже в случае потери уведомления благодаря избыточности действий [5] системы HotPlug, которая повторно применяет конфигурации при переподписке.

Достоинством системы является использование единого провайдера уведомлений, что исключает необходимость создания дополнительных сокетов [6] и минимизирует нагрузку на ресурсы. Разработанная архитектура легко расширяема, что позволяет добавлять новые виды уведомлений без значительных изменений в существующей реализации.

ЛИТЕРАТУРА

1. Денисенко, Б. А. Реализация декларативного подхода к управлению сетевой конфигурацией / Б. А. Денисенко, М. В. Тянутов, И. В. Никифоров // Неделя науки ИКНХ :Материалы докладов научно-практической конференции, Санкт-Петербург, 15–17 апреля 2024 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. – С. 17-19. – EDN ONQRRN.
2. Levy H., Sidi M. Polling systems: applications, modeling, and optimization //IEEE Transactions on communications. – 1990. – Т. 38. – №. 10. – С. 1750-1760.
3. Bainomugisha E. et al. A survey on reactive programming //ACM Computing Surveys (CSUR). – 2013. – Т. 45. – №. 4. – С. 1-34.
4. Дутова, А. А. Методика проектирования отказоустойчивых хранилищ данных в высоконагруженных системах / А. А. Дутова, И. В. Никифоров, Л. П. Котлярова // 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. 238-239. – EDN BGMLSZ.
5. Липаев В. В. Надежность программного обеспечения (обзор концепций) //Автоматика и телемеханика. – 1986. – №. 10. – С. 5-31.
6. Rosen R. Netlink sockets //Linux Kernel Networking: Implementation and Theory. – Berkeley, CA : Apress, 2013. – С. 13-35.

УДК 004.627

А.С. Дмитриевцев (2 курс бакалавриата),
В.И. Гориховский, к.ф.-м.н., доцент

ПОВЫШЕНИЕ КАЧЕСТВА ДЕДУПЛИКАЦИИ С ПОМОЩЬЮ FREQUENCY BASED CHUNKING

Дедупликацией данных называется метод, направленный на устранение повторяющихся копий данных на одном или группе носителей с целью снижения накладных расходов на хранение и передачу информации. Используется в системах хранения резервных копий, таких как TATLIN.BACKUP от компании YADRO. Различают два вида дедупликации: файловую и блочную (чанковую). Идея чанковой дедупликации заключается в разбиении данных на фрагменты и сохранении лишь уникальных экземпляров, заменяя дубликаты ссылками на оригинальный блок.

Content-Defined Chunking (CDC) – подход к разделению на блоки для первичной дедупликации, в котором используется контекст для определения границ будущих чанков. Примерами CDC являются RabinCDC, FastCDC, Leap-based CDC, SuperCDC[1]. Среди этих методов нет такого, который бы стабильно давал лучшее сжатие. Существуют два подхода улучшения дедупликации: Similarity Based Chunking и Frequency Based Chunking[2]. Основная идея FBC – дедупликация чанков из CDC-разбиения. Подход использует частотный анализ блоков меньшего размера (подчанков), на основе которого происходит дробление чанков. Работы по исследованию методов улучшения дедупликации проводились только для RabinCDC, также не анализировалось взаимодействие алгоритмов с файловой системой. В

представленной работе рассматривается FBC (исследования SBC ведутся¹ другим участником проекта).

ChunkFS[3] – модельная файловая система с встроенной поддержкой методов CDC, созданная для комплексного анализа алгоритмов дедупликации, поэтому она была выбрана для интеграции FBC и анализа эффективности его применения к разным алгоритмам дедупликации. Поскольку ChunkFS написана на Rust, для реализации метода был выбран этот язык.

Целью работы является разработка и комплексный анализ эффективности FBC совместно с разными алгоритмами дедупликации. Для её выполнения были поставлены следующие задачи: обзор реализаций алгоритмов FBC с открытым кодом, предпочтительно на Rust; реализация метода FBC и его интеграция в ChunkFS; эмпирическая оценка качества дедупликации.

В открытом доступе нет реализаций FBC на Rust, поэтому рассматривались статьи с описанием моделей и реализации на других языках программирования. Существует реализация Frequency Based Chunking на Python². Эта реализация не является содержательной в контексте дедупликации, так как работает с текстовыми данными и используется для сбора особенностей в NLP-пайплайне. Таким образом, необходима разработка собственной реализации на Rust.

В первую очередь была разработана базовая реализация, использующая для частотного анализа полный перебор всех возможных подчанков. Однако данный подход оказался недостаточно эффективным для проведения экспериментов на серьезных объемах данных, поэтому было принято решение внедрить несколько эвристических улучшений. В частности, при составлении словаря рассматриваются только блоки фиксированного размера (64 и 128 байт)

Также была внедрена методика разрешения коллизий, основанная на рассмотрении исключительно непересекающихся подчанков. Для эффективной проверки отсутствия элемента в словаре используется фильтр Блума. Такая реализация показала достаточную производительность и была интегрирована³ в ChunkFS.

Для сравнения качества дедупликации FBC совместно с разными алгоритмами CDC были проведены эксперименты для вычисления коэффициента дедупликации: отношения объема исходного датасета к дедуплицированному. Были выбраны датасеты как с большим количеством дубликатов – две версии ядра Linux (3.4.6 и 3.4.7), так и с малым – Enron Emails. Эксперименты проводились для алгоритмов SuperCDC, RabinCDC и FastCDC.

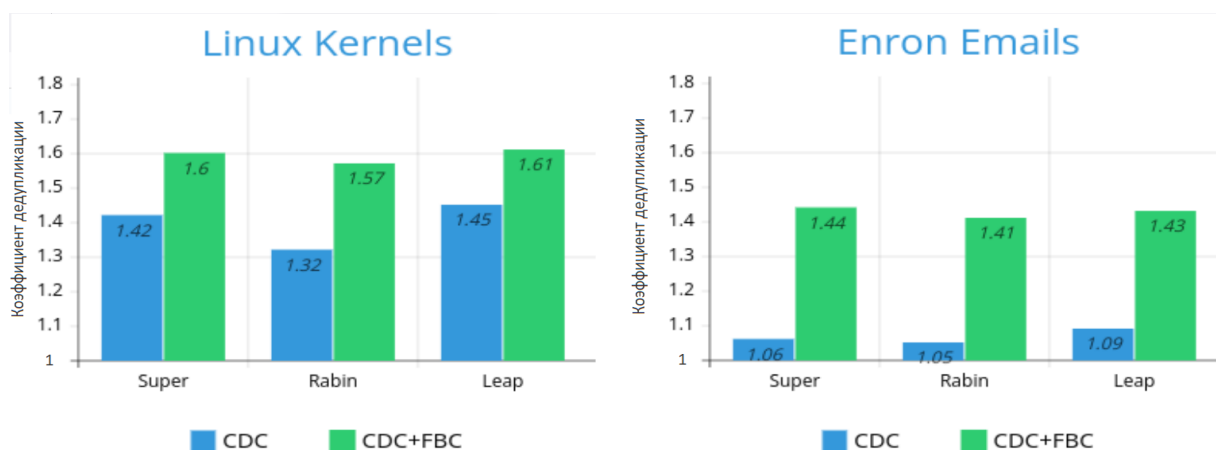


Рисунок 1 – коэффициент дедупликации для различных видов CDC и применения FBC к ним для Linux Kernels (слева) и для Enron Emails (справа).

¹ https://github.com/maxscherbakov/sbc_algorithm

² <https://github.com/sergicastellasape/semantic-compression>

³ https://github.com/admitrievtsev/fbc_scrubber

Результаты экспериментов представлены на рисунке 1. На данных с высокой энтропией FBC позволяет добиться значительно более сильного сжатия. Это обусловлено тем, что в Enron Emails мало дубликатов размером более 4 Кб. При этом выбор базового CDC-алгоритма практически не влияет на качество дедупликации. На датасете Linux Kernels FBC увеличивает коэффициент дедупликации на 10-13% после SuperCDC и LeapCDC, и на 18-19% после RabinCDC. Несмотря на различный прирост коэффициента дедупликации на датасете с низкой энтропией, итоговый коэффициент дедупликации оказывается близким, то есть выбор исходного CDC алгоритма слабо влияет на конечный результат.

Таким образом можно сделать вывод, что FBC не рекомендуется использовать в файловых системах, в связи с существенным снижением скорости чтения/записи. Однако в системах хранения резервных копий допускается применение наиболее производительного алгоритма чанкирования совместно с FBC для значительного уменьшения объема хранимых данных.

ЛИТЕРАТУРА

1. Marcel Gregoriadis et al, "A Thorough Investigation of Content-Defined Chunking Algorithms for Data Deduplication", 28 сентября 2024 г.
2. Lu Guanlin Jin Yu Du David. Frequency Based Chunking for Data De-Duplication, 2010.
3. Пилецкий О. А., Гориховский В. И. "Разработка инструмента для сравнения Content Defined Chunking на Rust" // Современные технологии в теории и практике программирования: сборник материалов конференции, Санкт-Петербург, 24 апреля 2024 года / Санкт-Петербургский политехнический университет Петра Великого. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2024. – С. 253-254.

УДК 004.453

Г. Дюльдин (4 курс бакалавриата),
И.Ф. Райкевич (3 курс бакалавриата),
М.С. Соломонов, ассистент

АРХИТЕКТУРА CSI-ДРАЙВЕРА ДЛЯ KUBERNETES С ПОДДЕРЖКОЙ СПЕЦИФИКАЦИИ SWORDFISH

За последние годы контейнеризация и оркестрация приложений в Kubernetes получили широкое распространение [1]. Это требует надежных и гибких решений для управления хранилищами данных. Одной из ключевых технологий в этой области является CSI-драйвер, который обеспечивает связь между Kubernetes и системами хранения.

В рамках данного проекта был разработан CSI-драйвер [2], реализующий спецификацию Swordfish [3,4] – стандартизированный протокол управления корпоративными системами хранения, основанный на Redfish [5]. Такой подход позволяет динамически провиженить тома, а также интегрировать расширенные возможности мониторинга и управления хранилищем. Это особенно важно для масштабируемых корпоративных систем.

Современные системы хранения данных характеризуются высокой разнородностью: различные производители используют собственные API и протоколы [6], и это затрудняет их интеграцию в единое управляемое решение. До появления универсальных интерфейсов, разработчики были вынуждены создавать индивидуальные решения для каждого типа хранилища и это, в свою очередь, повышает вероятность ошибок и усложняет сопровождение. Помимо этого, традиционные методы работы с томами не удовлетворяют требованиям масштабируемости и автоматизации, предъявляемым к современным облачным и распределенным системам. В связи с этим возникла необходимость разработки CSI-драйвера, который объединял бы возможности стандартного интерфейса Kubernetes с расширенными функциями управления, определенными спецификацией Swordfish.

На рисунке 1 представлена архитектура CSI-драйвера для хранилищ, совместимых с Swordfish API. Главными компонентами являются Identity-сервис, Node-сервис и Controller-сервис, а также интеграция со спецификацией Swordfish, позволяющая обращаться к Swordfish API [7] для создания, изменения и получения информации о томах.

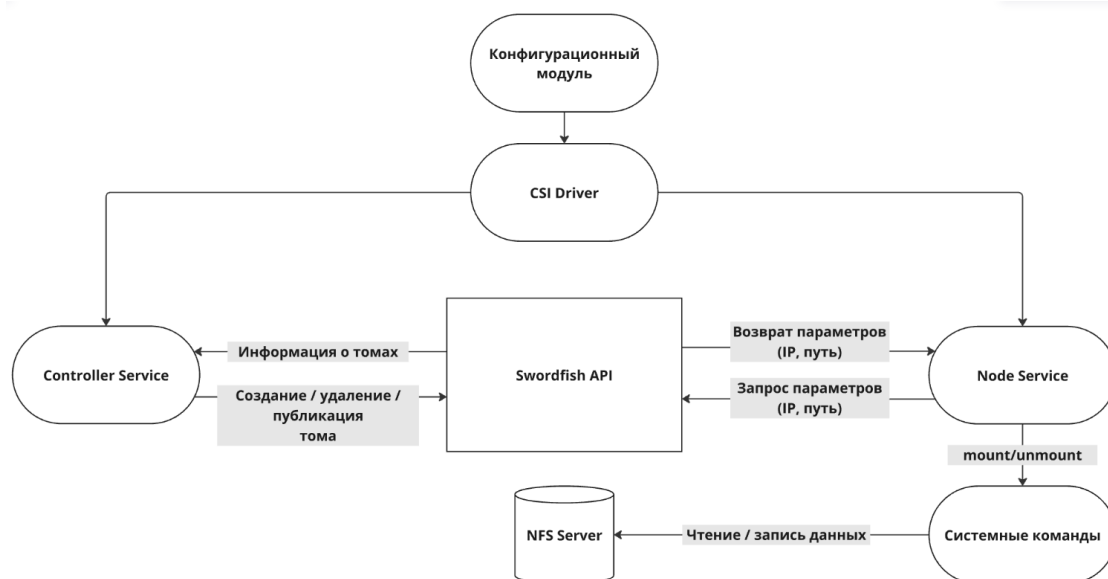


Рисунок 1 – Предложенная архитектура CSI драйвера

Для обеспечения быстрой и надежной связи между компонентами, и для взаимодействия с инструментами тестирования используется gRPC. Тестирование проводится за счет Unit тестов и csi-sanity [8], подтверждающая корректность реализации базовых gRPC методов и соответствие спецификации CSI. Каждый модуль отвечает за отдельные задачи:

- *Конфигурационный модуль*. Драйвер использует конфигурационный файл в формате YAML, содержащий параметры для настройки сетевого взаимодействия, адреса Swordfish API, а также параметры логирования.

- *Identity-сервис* отвечает за предоставление информации о плагине: его имени, версии и доступных возможностях. Этот модуль является первым звеном в цепочке взаимодействия с Kubernetes.

- *Node-сервис* реализует функции монтирования и демонтажа томов на узлах Kubernetes. Он взаимодействует с операционной системой для выполнения команд монтирования, а также обращается к Swordfish API для получения актуальных данных о состоянии томов, файловых систем и IP адрес NFS сервера.

- *Controller-сервис* обеспечивает управление жизненным циклом томов на уровне кластера. Контроллерный сервис отвечает за операции создания, публикации и удаления томов. При создании тома драйвер обращается к Swordfish API для инициирования провиженинга, после чего данные о созданном томе сохраняются в локальном кеше. ControllerPublishVolume обеспечивает привязку том к конкретному узлу. Аналогично, метод ControllerUnpublishVolume - отмена привязки.

- *Интеграция со спецификацией Swordfish*. Основное преимущество использования Swordfish заключается в стандартизации управления хранилищем. Драйвер, посредством вызовов к Swordfish API, получает данные о параметрах тома, конфигурации NFS-сервера и состоянии систем хранения. Интеграция с расширенными функциями управления через спецификацию Swordfish позволяет драйверу работать с томами на глубоком уровне: изменять параметры, осуществлять мониторинг состояния и оптимизировать распределение ресурсов. Это, в свою очередь, повышает эффективность использования системы хранения, снижает трудоемкость операционных процессов и позволяет оперативно реагировать на изменения в инфраструктуре.

В ходе разработки были реализованы основные компоненты CSI-драйвера, подтвержденные инструментом тестирования CSI-sanity. Основные операции, такие как GetPluginInfo, Probe, NodePublishVolume и NodeUnpublishVolume, работают корректно и соответствуют спецификации CSI. При публикации тома драйвер проверяет наличие уже созданного ресурса, привязывает его к указанному узлу и, в случае необходимости, сообщает о нестыковках (например, если том уже опубликован на другом узле). Интеграция расширенных функций управления через спецификацию Swordfish позволяет минимизировать ручные операции и снижает риск ошибок при работе с хранилищем. Модульный характер драйвера обеспечивает возможность дальнейшего расширения функционала без кардинальных изменений в существующем коде.

Таким образом, предложенная архитектура CSI-драйвера, объединяющая стандартные возможности Kubernetes с расширенными функциями, предлагаемыми спецификацией Swordfish, демонстрирует высокую эффективность и гибкость при управлении системами хранения данных. Внедрение конфигурационного модуля, Identity-, Node- и Controller-сервисов позволило создать масштабируемое и расширяемое решение, способное автоматизировать процессы провиженинга, монтирования и управления томами. Реализованный контроллерный слой обеспечивает динамическое создание и публикацию томов и открывает возможности для дальнейшей интеграции с корпоративными системами хранения. На данный момент реализована архитектура, поддерживающая протокол NFS, в дальнейшем планируется расширить набор поддерживаемых протоколов.

ЛИТЕРАТУРА

1. Shemyakinskaya, A. S. Hard drives monitoring automation approach for Kubernetes container orchestration system / A. S. Shemyakinskaya, I. V. Nikiforov // Proceedings of the Institute for System Programming of the RAS. – 2020. – Vol. 32, No. 2. – P. 99-106. – DOI 10.15514/ISPRAS-2020-32(2)-8. – EDN ABZBLX.
2. CSI драйвер для Kubernetes. Электронный ресурс [Режим доступа]: <https://kubernetes-csi.github.io/docs/introduction.html> (Дата обращения: 13.03.2025)
3. Спецификация Swordfish. Электронный ресурс [Режим доступа]: https://www.snia.org/sites/default/files/technical-work/swordfish/release/v1.2.5a/html/Specification/Swordfish_v1.2.5a_Specification.html (Дата обращения: 13.06.2025)
4. Разработка Ansible-модулей по спецификации SWORDFISH для управления конфигурацией СХД / С. Ф. Солодовников, Г. А. Худяков, И. В. Никифоров, О. В. Александрова // 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. 294-297. – EDN LHUAKG.
5. Спецификация Redfish. Электронный ресурс [Режим доступа]: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.20.0.html (Дата обращения: 13.03.2025).
6. Различные CSI драйверы. Электронный ресурс [Режим доступа]: <https://kubernetes-csi.github.io/docs/drivers.html> (Дата обращения 13.03.2025)
7. Юлдашев, В. М. особенности и проблемы mock-сервера Swordfish API Emulator / В. М. Юлдашев, И. В. Никифоров, О. В. Александрова // 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. 298-301. – EDN XSKJFS.
8. Документация CSI Sanity теста. Электронный ресурс [Режим доступа]: <https://github.com/kubernetes-csi/csi-test/blob/master/pkg/sanity/README.md> (Дата обращения 13.03.2025)

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ АСИНХРОННОГО РАНТАЙМА ТОКИО

TATLIN.BACKUP¹ — проект группы компаний YADRO², посвященный созданию системы резервного копирования с использованием алгоритмов дедупликации написанный на языке Rust. Реализация асинхронного рантайма этого языка (tokio) ограничивает пропускную способность системы. Представленная работа посвящена изучению и развитию tokio для решения этой проблемы.

Язык Rust предоставляет `async / await` [1] интерфейс для обработки асинхронных событий, не фиксируя реализацию, позволяя пользователю выбирать асинхронный рантайм. tokio³ — проект, предоставляющий две имплементации асинхронного рантайма (многопоточный и однопоточный), частично реализует стандартную библиотеку языка с асинхронным интерфейсом, примитивы синхронизации, коллекции, таймеры, профилировщики — одним словом целую экосистему для написания асинхронных программ.

В TATLIN.BACKUP использующим многопоточный рантайм из tokio был замечен недостаток текущей реализации: общая очередь асинхронных событий, разделяемая всеми потоками tokio, защищена [2] мьютексом. По предположению инженеров из YADRO этот мьютекс ограничивает расширяемость их системы, поскольку является бутылочным горлышком.

Целью данной работы является проверка гипотезы о бутылочном горлышке и поиску исправления этой проблемы. Для её достижения поставлены следующие задачи:

1. Разработка исследовательского стенда, эмулирующего TATLIN.BACKUP.
2. Проведение анализа собираемых стендом метрик и подтверждение ограничения на производительность.
3. Реализация прототипа обновления tokio и анализ его качества.

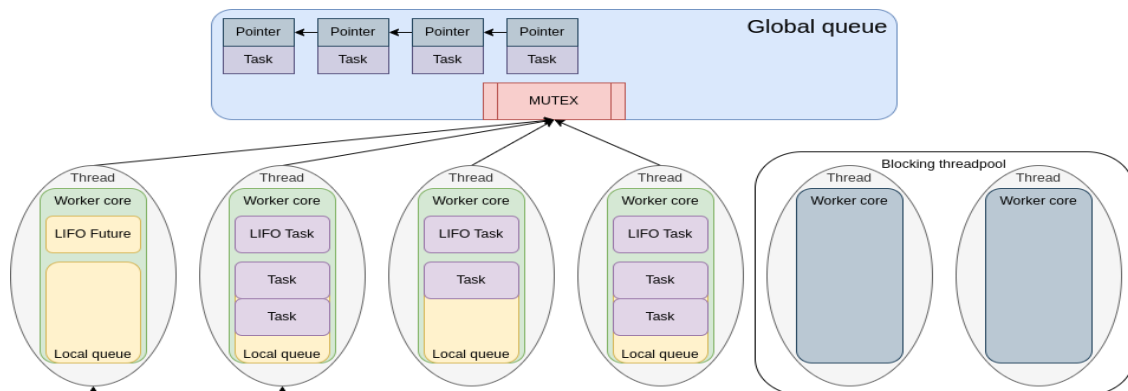


Рисунок 1 – Архитектура tokio

На основе архитектуры представленной на рисунке 1 был выполнен поиск сценариев для измерения производительности. С деталями реализации полученных сценариев можно ознакомиться в репозитории⁴.

Было выявлено увеличение пропускной способности при использовании нескольких экземпляров рантайма, представленное на рисунок 2. Из графика заметным становится увеличение пропускной способности системы, при равномерном распределении работы между большим количеством экземпляров рантаймов. Силами команды TATLIN.BACKUP, как закрытого проекта, были проведены исследования, подтвердившие гипотезу. Однако, такой

¹ <https://yadro.com/ru/tatlin/backup/> Сайт проекта TATLIN.BACKUP.

² <https://yadro.com/> Сайт компании YADRO.

³ <https://tokio.rs/> Сайт проекта tokio.

⁴ <https://github.com/IgorErin/tokiobench> Репозиторий исследовательского стенда.

подход усложняет поддержку и не дает никаких гарантий надежности: известны случаи критических ошибок при использовании нескольких рантаймов в продуктивном коде нарушающих стабильность системы.

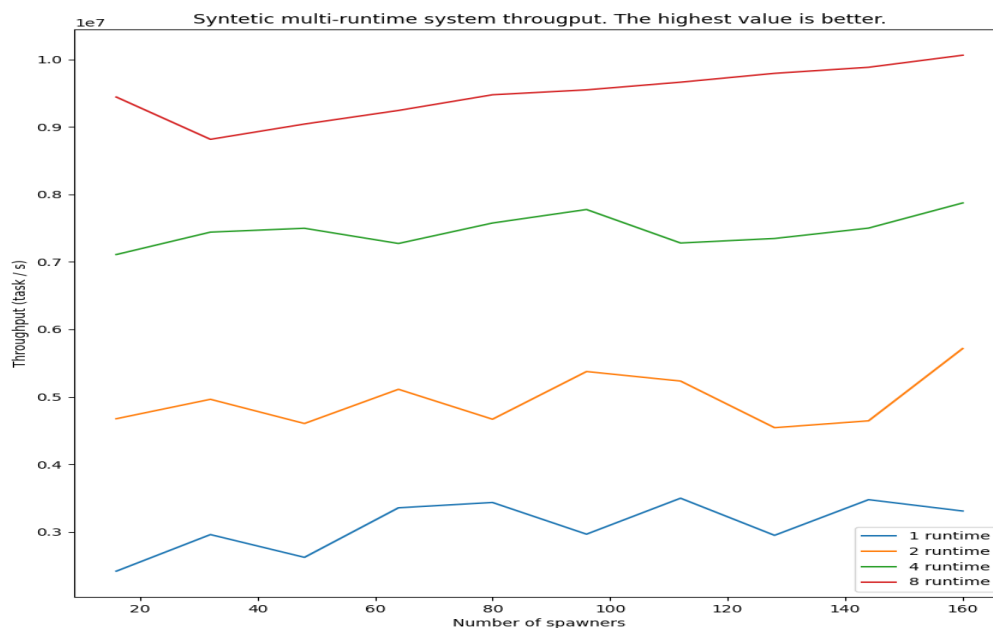


Рисунок 2 – Пропускная способность системы из нескольких рантаймов

Для получения стабильного решения была предпринята попытка¹ выделить несколько рантаймов в одном инстансе: с использованием scheduler'a, сохраняющего семантику обработки задач. Такое решение сохраняет гарантии стабильности и предлагает пользователю интерфейс нескольких рантаймов. Однако, прирост производительности оказывается не столь значительным, из чего можно сделать вывод: помимо глобальной очереди в рантайме может присутствовать другие компоненты, ограничивающие пропускную способность. В будущем предполагается дальнейшая оптимизация tokio.

ЛИТЕРАТУРА

1. Дизайн tokio [Электронный ресурс] Режим доступа: <https://tokio.rs/blog/2019-10-scheduler>
2. Sym Don, Petricek Tomas, Lomov Dmitry. The F# Asynchronous Programming Model. — Vol. 6539. — 2011. — 01. — P. 175–189.24.

УДК 004.4`2

Э.Э. Истаев, Д.А. Фоломкин (4 курс бакалавриата),
О.В. Александрова, ст. преподаватель

РАЗРАБОТКА ДИНАМИЧЕСКОЙ РОЛЕВОЙ МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ ДЛЯ СЕРВЕРА-ЭМУЛЯТОРА SWORDFISH

Современные системы хранения данных (СХД) требуют унифицированных подходов к управлению ресурсами и обеспечению безопасности. Стандарт Swordfish [1], разработанный организацией SNIA, предоставляет интерфейс REST API для взаимодействия с различными СХД. В рамках проекта лаборатории СПбПУ «Технологии программирования Yadro-Политех» разрабатывается сервер-эмулятор Swordfish, предназначенный для тестирования

¹ <https://github.com/IgorErin/tokio/pull/3> Изменения рантайма.

Ansible-модулей [2]. Одной из ключевых задач является реализация ролевой модели управления доступом (RBAC) [3], которая обеспечивает безопасное взаимодействие пользователей с ресурсами эмулятора.

С увеличением сложности архитектуры СХД возрастает необходимость в эффективных механизмах управления доступом. Управление доступом на основе ролей является стандартным подходом в современных системах для минимизации рисков несанкционированного доступа и повышения эффективности администрирования. В руководстве Swordfish User Guide [4] определены три основные роли: DevOps, StorageAdmin и CloudAdmin, каждая из которых имеет уникальный набор привилегий, а также интегрирует роли Administrator и ReadOnly из Redfish [5]. Реализация RBAC в сервере-эмуляторе позволит не только соответствовать спецификации [6], но и обеспечить гибкость при тестировании различных сценариев взаимодействия с СХД, что особенно важно для автоматизации процессов управления. На данный момент в эмуляторе реализована лишь статическая версия ролевой модели [7], где пользователи и роли жестко заданы в коде. Это ограничивает возможности тестирования и не соответствует требованиям спецификации. Кроме того, отсутствует поддержка динамического управления пользователями и ролями через API, что усложняет адаптацию эмулятора под различные сценарии использования. Например, добавление новых пользователей или изменение их прав требует ручного внесения изменений в исходный код, что непрактично и затрудняет масштабирование системы.

Цель работы заключается в снижении трудоемкости управления доступом к ресурсам сервера-эмулятора Swordfish за счет разработки динамической ролевой модели, позволяющей автоматизировать процессы назначения прав доступа и управления ими. Для достижения этой цели предполагается внедрение механизма проверки прав доступа на уровне middleware и сервисов, а также использование базы данных для хранения информации о ролях и привилегиях, что обеспечит соответствие требованиям спецификации.

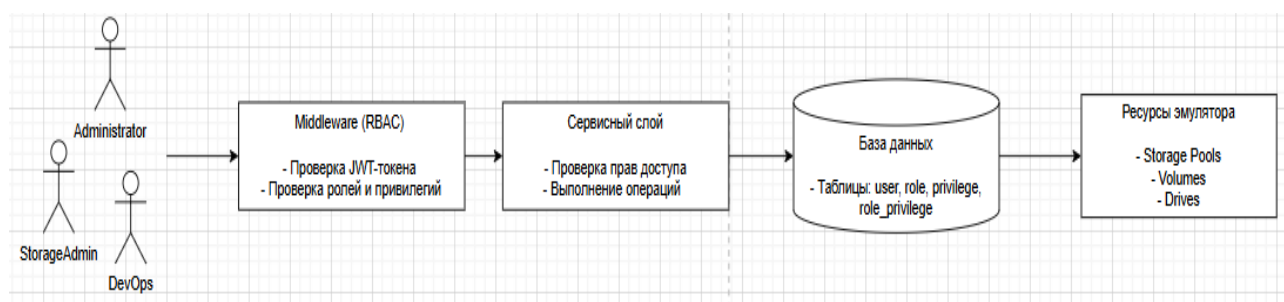


Рисунок 1 – Архитектура ролевой модели управления доступом

На рисунке 1 показана архитектура ролевой модели управления доступом, включающая роли, middleware, базу данных и ресурсы эмулятора.

Для реализации динамической ролевой модели предлагается выполнить следующие шаги:

- загрузка ролей и пользователей из ресурсов эмулятора с использованием базы данных PostgreSQL. Информация о ролях, пользователях и их привилегиях будет храниться в таблицах user, role, privilege и role_privilege;
- интеграция middleware для проверки прав доступа, которая будет анализировать JWT-токен пользователя и его привилегии перед выполнением операций;
- реализация проверки прав доступа на уровне сервисов для каждой операции, чтобы гарантировать, что пользователь имеет необходимые права для её выполнения.

На текущий момент реализована статическая версия ролевой модели. Разработаны 5 основных ролей: Administrator, DevOps, StorageAdmin, CloudAdmin и ReadOnly. Проведено тестирование статической версии ролевой модели на 10 сценариях взаимодействия с эмулятором. В результате тестирования удалось сократить время настройки прав доступа на 40%.



Рисунок 2 – Демонстрация статической реализации ролевой модели

На рисунке 2 результат одного из сценариев, где 1 – адрес ресурса, доступ к которому хочет получить пользователь, 2 – логин и пароль пользователя, 3 – ответ сервера-эмулятора на тестовый запрос. Запрос на получение доступа к ресурсам не прошел по причине отсутствия необходимых прав у роли пользователя. Получается, у данного пользователя нет доступа к ресурсам, и система распределения ролей работает.

Разработка динамической ролевой модели управления доступом является важным шагом в совершенствовании сервера-эмулятора Swordfish и будущих СХД, где эта модель будет применяться [8]. Она позволит упростить управление пользователями и ролями, обеспечить соответствие спецификации Swordfish и повысить безопасность взаимодействия с ресурсами эмулятора. В дальнейшем планируется интеграция новых специфических ролей для решения задач в рамках определенной СХД, а также усиление безопасности за счет внедрения двухфакторной аутентификации или использования YubiKey.

ЛИТЕРАТУРА

1. Документация Swordfish (2023). Электронный ресурс [Режим доступа]: <https://www.snia.org/forums/smi/swordfish> (Дата обращения: 23.02.2025).
2. Разработка Ansible-модулей по спецификации SWORDFISH для управления конфигурацией СХД (2024) / С. Ф. Солодовников, Г. А. Худяков, И. В. Никифоров, О. В. Александрова // 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. 294-297. – EDN LHUAKG.
3. Реализация модуля управления доступом на основе ролей для сервер-эмулятора Swordfish (2024) / 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.
4. Swordfish User Guide (2023). [Электронный ресурс] Режим доступа: https://www.snia.org/sites/default/files/technical-work/swordfish/release/v1.2.5a/html/UserGuide/Swordfish_v1.2.5a_UserGuide.html (Дата обращения: 23.02.2025).
5. Документация Redfish (2014). [Электронный ресурс] Режим доступа: <https://www.dmtf.org/standards/redfish>
6. Спецификация Swordfish (2023). [Электронный ресурс] Режим доступа: https://www.snia.org/sites/default/files/technical-work/swordfish/release/v1.2.5a/html/Specification/Swordfish_v1.2.5a_Specification.html

7. Статическая реализация ролевой модели (2023). [Электронный ресурс] Режим доступа: <https://github.com/ValeryYldashev/Swordfish-API-Emulator-fork/pull/9/files>
8. Особенности и проблемы mock-сервера Swordfish API Emulator (2024). [Электронный ресурс] Режим доступа: <https://habr.com/ru/companies/yadro/articles/795211>

УДК-004.4'22

Э.М. Клеблеева (3 курс бакалавриата),
Д.М. Вайндрах (2 курс бакалавриата),
И.А. Селин, к.т.н., доцент

АНАЛИЗ АРХИТЕКТУРНОЙ ДОКУМЕНТАЦИИ СРАВНЕНИЕ PLANTUML-ДИАГРАММ С ИСХОДНЫМИ ФАЙЛАМИ НА ЯЗЫКЕ GOLANG

В современной разработке с увеличением сложности программных систем всё больше внимания уделяется проблемам строгого описания структуры приложений с целью последующей верификации и валидации в виде формальных моделей [1-3]. В частности, важную роль играет визуализация архитектуры системы с помощью диаграмм, таких как PlantUML [4]. Один из аспектов проблемы – сопоставление составленной архитектуры посредством диаграмм с полученным исходным кодом проекта. С течением времени, исходный код и содержимое диаграммы могут расходиться, что будет приводить к ошибкам в проектировании и поддержке системы. Возникает необходимость в автоматизированном инструменте, который сможет сравнивать PlantUML-диаграммы с исходным кодом и выявлять несоответствия.

Цель работы – разработать приложение, которое автоматически анализирует соответствие элементов классовых PlantUML-диаграмм (классов, интерфейсов) и типов данных в исходном коде на языке Golang, а также генерирует отчет о найденных несоответствиях. Язык Golang выбран в связи с запросом компании YADRO, как активно использующийся в организации и глобально в сфере разработки.

Для решения рассматриваемой проблемы к реализации предложено приложение со следующей структурой:

1. Сервис анализа исходного кода на Golang – реализован на языке Golang, анализирует структуры данных, интерфейсы и другие типы в исходных файлах. Результат работы сервиса – текстовый файл с информацией о структурах данных в формате JSON.

2. Сервис сопоставления структур данных – реализован на Python, принимает на вход классовую PlantUML-диаграмму и результат работы первого сервиса. Программа сравнивает элементы диаграммы с типами данных в исходном коде и генерирует отчет о несоответствиях.

В ходе разработки первой части приложения была написана программа для сбора, анализа и структурирования характеристик классов и интерфейсов Golang-кода. Данный инструмент создавался в связи с отсутствием готовых решений, удовлетворяющих требованиям проекта, поэтому было принято решение реализовать его самостоятельно.

Для этой задачи была выбрана стандартная библиотека языка Golang – ast [5] (Abstract Syntax Tree), которая позволяет представлять любую часть кода в виде узлов синтаксического дерева. Использование ast дает возможность анализировать структуру кода на уровне исходных данных, извлекать информацию о классах, интерфейсах, их методах и полях, а затем представлять эти данные в упорядоченном виде.

Работа сервиса состоит из двух ключевых этапов. Первый этап – это анализ проекта или отдельных файлов, написанных на языке Golang. На этом этапе происходит разбор исходного кода и построение его абстрактного синтаксического дерева. Второй этап – это преобразование собранных данных в удобный и читаемый формат, пригодный для дальнейшей обработки. В качестве формата хранения результатов был выбран JSON, так как

он удобен для интеграции с другими инструментами и легко читается как программами, так и людьми.

Работа второго сервиса включает в себя обработку PlantUml-диаграмм и сравнение их с разобранным кодом Golang. Обработка диаграммы состоит из двух этапов: полный разбор исходного .puml файла на составные элементы (пакеты, модули, классы, интерфейсы, их поля и методы, декоративные элементы, описание отношений элементов, комментарии и заметки), затем форматирование и выделение только необходимой для анализа информации (информация о классах и интерфейсах).

В ходе разработки второй части приложения была решена смежная задача написания грамматики для анализа и разбора классовых PlantUml-диаграмм, поскольку готовых инструментов для этой цели, удовлетворяющей используемому стеку технологий, найдено не было. Для реализации был использован синтаксический анализатор библиотеки Lark [6] языка программирования Python. Для него написана грамматика в нотации EBNF [7], покрывающая все основные элементы корректно написанной классовой PlantUml диаграммы.

После обработки Golang-кода и PlantUml-диаграммы наступает этап сравнения полученных обработанных данных. Поскольку ранее информация из PlantUml была отфильтрована и обработана в соответствии с предоставляемым JSON файлом Golang-кода, то сравнение происходит по принципу поиска прямого соответствия или несоответствия элементов в двух структурах. Для автоматизации этого процесса используется библиотека deepDiff [8] языка программирования Python.

По результатам сравнения формируется два отчёта: для машинной обработки в человекочитаемом виде. Отчет содержит такую информацию как: список случаев, когда в PlantUML-диаграмме присутствует элемент (класс, интерфейс и т.д.), но в исходном коде соответствующий тип отсутствует; нормированная статистика для PlantUML-диаграммы по покрытию ее элементов в исходном коде.

Демонстрационная версия приложения была протестирована на нескольких проектах с использованием PlantUML-диаграмм и исходного кода на Golang. В результате тестирования были получены следующие результаты:

1. Сервис по разбору PlantUml-диаграммы правильно идентифицирует классы и интерфейсы на корректно написанных диаграммах.

2. Сопоставление элементов с соответствующими типами в исходном коде происходит корректно.

3. Отчеты генерируются в машиночитаемом виде, что позволяет легко интегрировать их в процесс разработки.

4. Приложение поддерживает конфигурацию через YAML-файлы, что упрощает настройку для различных проектов.

ЛИТЕРАТУРА

1. Волович, М. Е. Верификация UML-моделей программных систем / М. Е. Волович, О. А. Дерюгина // Cloud of Science. – 2015. – Т. 2, № 1. – С. 138-146. – EDN TNDJFP.
2. 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.
3. 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.
4. Класс синтаксиса. Схема и особенности [Электронный ресурс] // Lark Documentation. – URL: <https://lark-parser.readthedocs.io/en/latest/parsers.html> (дата обращения: 28.02.2025).
5. Abstract Syntax Tree documentation [Электронный ресурс] // ast documentation. – URL: <https://pkg.go.dev/go/ast> (дата обращения: 04.03.2025).
6. Parsers — Lark documentation [Электронный ресурс] // Lark documentation. – URL: Parsers — Lark documentation (дата обращения: 04.03.2025).

7. ISO/IEC 14977:1996(E). Information technology — Syntactic metalanguage — Extended BNF. Geneva: International Organization for Standardization, 1996.
8. DeepDiff OLD 4.0.7 documentation [Электронный ресурс] // DeepDiff Documentation. – URL: <https://deepdiff.readthedocs.io/en/latest/> (дата обращения: 04.03.2025).

УДК 004.932.2

Р.С. Кузьмин (2 курс бакалавриата),
П.А. Шагалова, к.т.н., доцент

РАЗРАБОТКА И ОПТИМИЗАЦИЯ АЛГОРИТМА ПОИСКА УГЛОВЫХ ТОЧЕК НА ИЗОБРАЖЕНИИ

Алгоритмы компьютерного зрения открывают большие возможности для инноваций в различных сферах человеческой деятельности, таких как здравоохранение, безопасность, промышленность, сельское хозяйство, транспортная инфраструктура.

Существуют библиотеки компьютерного зрения, которые предоставляют инструменты для реализации сложных алгоритмов обработки и анализа изображений [4]. При этом зачастую требуется оптимизация и повышение скорости алгоритмов обработки изображений, например, при использовании их в системах реального времени.

В данной работе выполнена реализация алгоритма поиска угловых точек, основанного на локальных и глобальных свойствах кривизны [1] с последующей оптимизацией для повышения производительности. Рассматриваемый алгоритм анализирует границы изображения, для каждой границы строит соответствующую ей функцию кривизны с последующим поиском и анализом точек экстремума. Реализация выполнена на языке C++ с использованием библиотеки OpenCV [2], оптимизация включает ускорение вычисления функции кривизны и поиска экстремумов.

Для повышения производительности неоптимизированного кода были применены алгоритмические и компиляторные оптимизации. Для выполнения компиляторных оптимизаций использованы флаги -O1, -O2, -O3 и -Ofast, которые показали рост скорости на 5-10%.

Кроме компиляторных оптимизаций выполнен анализ кода с использованием средств профилирования и сделан ряд алгоритмических оптимизаций [3]. Описание найденных проблем и способы их решения представлены ниже.

1. В процессе анализа кода было выявлено, что ядро Гаусса вычисляется заново для каждого вызова функции детекции углов, что приводит к избыточным вычислениям. Данная проблема была устранена путем внедрения кэширования ядра, которое в оптимизированной версии кода пересчитывается только при изменении параметра sig (ширина ядра). Это позволило сократить время обработки кривых на 10-15%.
2. Реализованная в алгоритме последовательная обработка границ, приводила к линейному росту времени выполнения с увеличением количества границ. Для ускорения была внедрена параллельная обработка с использованием OpenMP, что позволило динамически распределять задачи между потоками и полностью задействовать ресурсы CPU. Это особенно важно для изображений с большим количеством границ.
3. Ручная реализация свертки и вычисления производных через циклы создавала высокие накладные расходы и неэффективно использовала SIMD-инструкции процессора. Замена этих операций на векторные функции OpenCV (например, filter2D и Sobel) позволила значительно ускорить вычисления [2].

4. Поиск экстремумов и фильтрация углов, выполняемые через сложные циклы с множеством условий, были оптимизированы за счет упрощения логики поиска экстремумов (через отслеживание состояния минимума/максимума) и предварительного выделения памяти для результатов. Это уменьшило количество ветвлений и операций с памятью, что особенно важно для границ с большим количеством точек.

В результате оптимизации общее время выполнения алгоритма сократилось, что сделало его более эффективным для обработки изображений с большим количеством границ. Кроме того, внедрение параллельной обработки и использование векторных операций позволило полностью задействовать ресурсы CPU, обеспечивая высокую производительность даже на сложных данных.

ЛИТЕРАТУРА

1. He, X. C., Yung, N. H. C. Corner detector based on global and local curvature properties // Optical Engineering. – 2008. – Vol. 47, № 5. – P. 057008.
2. Bradski, G., Kaehler, A. Learning OpenCV: Computer Vision with the OpenCV Library. – O'Reilly Media, 2008. – 580 p.
3. Knuth, D. E. The Art of Computer Programming, Volume 1: Fundamental Algorithms. – Addison-Wesley, 1997. – 672 p.
4. Forsyth, D. A., Ponce, J. Computer Vision: A Modern Approach. – Pearson, 2011. – 792 p

УДК 004.415.25

Е.П. Лебедева (4 курс бакалавриата),
И.В. Никифоров, к.т.н, доцент

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ РЕПУТАЦИЕЙ УЗЛОВ ОБЪЕКТНОГО ХРАНИЛИЩА FROSTFS

Система хранения данных (СХД) - это комплекс аппаратных и программных средств, который предназначен для хранения и оперативной обработки информации, как правило, большого объема. СХД различаются по способу доступа к данным: файловый, блочный, объектный. Большой пласт хранимой в мире информации занимает медиа - неструктурированные данные большого объема [1]. Для хранения такого типа данных больше всего подходят объектные хранилища, и задача разработки и улучшения СХД с объектным типом доступа является крайне актуальной.

Современные СХД, как правило, представляют собой распределённые системы, работающие под высокой нагрузкой. В распределённых системах крайне необходимо обеспечивать безопасность и надёжность взаимодействия участников. Для этого вводятся различные системы управления репутацией.

Для реализации модуля управления репутацией было выбрано объектное хранилище с открытым исходным кодом FrostFS [2] - децентрализованное распределённое объектное хранилище на базе блокчейна NEO N3 Blockchain, разработкой и поддержкой которого занимается команда из компании YADRO.

Одним из самых известных, успешных и надёжных алгоритмов оценки репутации узлов в распределённых системах является EigenTrust [3]. Этот алгоритм, предложенный в 2003 году, является мощным инструментом благодаря транзитивным и безопасным стратегиям вычисления глобального доверия. Но процесс расчёта коэффициентов доверия создаёт высокую нагрузку на цепочку блоков, что негативно сказывается на производительности системы.

Таким образом, целью данной работы является сокращение количества транзакций, необходимых для фиксации качества взаимодействия узлов в сети, за счет внедрения нового

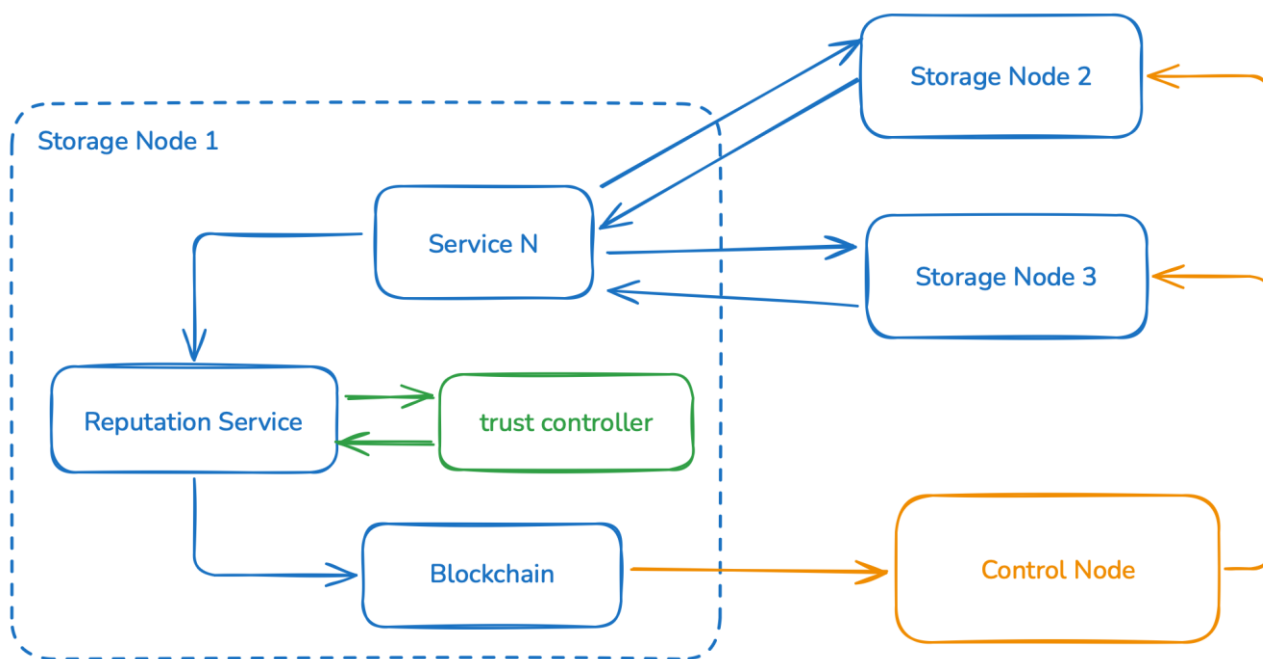


Рисунок 1 – Архитектура репутационной системы

модуля управления репутацией узлов через P2P сеть и сетевые протоколы прикладного уровня.

Для достижения данной цели, необходимо решить следующие задачи:

1. Провести исследование существующих алгоритмов оценки репутации узлов в распределённых системах.
3. Предложить модуль управления репутацией узлов для CXД FrostFS.
4. Реализовать предложенный модуль в CXД FrostFS.
5. Продемонстрировать, что количество транзакций для расчёта репутации узла действительно снизилось.

Для достижения поставленной цели были разработаны расширения для FrostFS API и SDK, был реализован сервис управления репутацией, представляющий собой сервер и контроллер доверия, отвечающий не только за вычисление значений доверия, но и за агрегацию локальных значений.

Архитектура решения представлена на рисунке 1. Репутационный сервис получает информацию о взаимодействии узла хранения с другими узлами, после чего на основании полученных метрик контроллер производит расчёт коэффициентов доверия, агрегирует их и сохраняет их локально. В конце эпохи (искусственной единицы времени системы) каждый узел объявляет своё локальное доверие, создавая транзакцию в блокчейн. "Управляющий" узел (им может стать любой из узлов сети, см. документацию [2]) вычитывает информацию о локальном доверии каждого узла и вычисляет глобальное доверие для текущей эпохи, после чего объявляет его другим участникам сети.

Сервис управления репутацией узлов распределённой децентрализованной системы хранения данных FrostFS был реализован на языке Go [4] - нативном для FrostFS. Для сбора метрик использовалась библиотека Prometheus [5], а для визуализации, мониторинга и анализа собранных данных использовалась открытая платформа Grafana [6].

ЛИТЕРАТУРА

1. Анализ данных трендов YouTube / В. К. Сычев, Ю. В. Ленкова, Е. А. Цветкова, 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. 131-132. – EDN GGZRXB.

2. True Cloud Lab FrostFS. [Электронный ресурс] Режим доступа: <https://git.frostfs.info/TrueCloudLab>
3. Sepandar D. Kamvar, Mario T. Schlosser, Hector Garcia-Molina The EigenTrust Algorithm for Reputation Management in P2P Networks // WWW '03: Proceedings of the 12th international conference on World Wide Web. – 2003. – P. 640-651. – DOI 10.1145/775152.775242.
4. Golang programming language. [Электронный ресурс] Режим доступа: <https://go.dev/>
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.

УДК-004.4

М.Р. Лонишин (3 курс бакалавриата),
Н.Р. Богданов (4 курс бакалавриата),
И.А. Селин, к.т.н., доцент

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ СРАВНЕНИЯ КЛАССОВЫХ ДИАГРАММ PLANTUML И PROTOBUF ФАЙЛОВ

Одной из распространённых проблем в разработке является несоответствие между кодом и его документацией [1]. В условиях активной разработки структура программы может изменяться, но не всегда эти изменения оперативно вносятся в сопроводительную документацию. Это может привести к серьезным проблемам при долгосрочном развитии проекта.

Чтобы минимизировать эти риски, необходимо автоматизировать процесс сравнения структуры данных. Компанией YADRO была поставлена задача по сравнению структур, описанных в Protocol Buffers (protobuf) [2] и в классовых диаграммах PlantUML [3]. Такая автоматизация позволит быстро выявлять различия, обеспечивая согласованность между документацией и реализацией. Программное решение должно уметь находить любые различия между ними: несовпадающие типы данных, пропущенные поля, лишние элементы. Результатом работы программы является отчёт в формате JSON (JavaScript Object Notation) [4], а также, при желании, формирование отчёта в читабельном для человека виде.

Для сравнения protobuf-файлов и классовых диаграмм PlantUML был выбран подход из 2 этапов. На первом этапе происходит обработка и преобразование входных файлов в единое промежуточное представление (структуру JSON). На втором этапе осуществляется сравнение двух JSON структур с выявлением различий и формирование отчета о несоответствиях.

Данная программа была реализована с помощью языка Python [5]. Обработка protobuf-файлов выполняется в несколько этапов. Сначала каждый .proto файл компилируется в бинарный формат .pb с использованием инструмента protoc. В результате получаются файлы, содержащие сериализованное представление структуры данных, так называемый FileDescriptorSet. Далее, с помощью библиотеки google.protobuf, эти FileDescriptorSet преобразуются в единый, удобный для дальнейшей работы формат JSON. Этот процесс повторяется для всех исходных protobuf-файлов, после чего полученный результат для каждого файла объединится в одну структуру.

Обработка PlantUML-файлов выполняется в несколько этапов. Спецификой классовой диаграммы является то, что описание структур и их расположений повторяет синтаксис, используемый в языках программирования. Чтобы наиболее эффективно подготовить

подобную структуру использовалась идея обратной польской записи [6] для преобразования сложных выражений со скобками в более простую и вычислимую форму. Это помогло перенести вложенное описание содержимого описанного в диаграмме проекта в пригодную для программ структуру. Диаграмма обрабатывается следующим образом:

1. Каждая строка обрабатывается с помощью регулярных выражений [7].
2. Через цепочку проверок определяем в какой части структуры файла сейчас находится программа.
3. Если присутствуют ключевые слова `folder`, `package` то их имена добавляются в очередь, а если ключевые слова `class`, `interface`, `enum`, `annotation`, `abstract class`, то создаётся запись о структуре со всей информацией о ней и её положении.
4. После обработки файла формируется JSON структура, которая будет использоваться для сравнения с обработчика `protobuf`.

Полученные из обработчиков `protobuf` и `PlantUML` JSON передаются в библиотеку `DeepDiff` [8] для детального сравнения. `DeepDiff` анализирует структуры данных, выявляя добавленные, удаленные и измененные элементы на всех уровнях вложенности. Помимо констатации самих различий, вычисляется метрика покрытия, которая показывает, какой процент элементов из одного представления (например, `protobuf`) присутствует в другом (`PlantUML`). Результаты сравнения, включая подробный отчет о различиях и данные о покрытии, сохраняются в JSON-файл для дальнейшего анализа и использования.

В ходе работы была разработана программа, предназначенная для автоматизированного сравнения структуры данных, представленных в файлах `Protocol Buffers (protobuf)` и диаграммах классов `PlantUML`. Она позволяет выявлять различия между этими описаниями, включая несовпадения типов данных, отсутствие или избыточность полей и элементов.

Полученный результат в виде отчета JSON, а также возможность формирования читабельного отчета, делает процесс анализа данных удобным и наглядным. Использование данной программы способствует обнаружению расхождений, что, в свою очередь, повышает согласованность архитектуры системы и упрощает поддержку системы.

ЛИТЕРАТУРА

1. 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.
2. Language Guide (proto 3) [Электронный ресурс] // Protocol Buffers Documentation. - URL: <https://protobuf.dev/programming-guides/proto3/> (дата обращения 08.03.2025).
3. Построение диаграмм UML с использованием PlantUML - Текст: электронный // plantuml.com: [сайт]. - URL: https://pdf.plantuml.net/PlantUML_Language_Reference_Guide_ru.pdf
4. Crockford, D.; Morningstar, C. Standard ECMA-404 The JSON Data Interchange Syntax; ECMA International: Geneva, Switzerland, 2017.
5. Ramalho, L. (2015). *Fluent Python: Clear, Concise, and Effective Programming*. O'Reilly Media. ISBN: 9781491946251
6. Т. Пратт, М. Зелковиц. Языки программирования: разработка и реализация = Terrence W. Pratt, Marvin V. Zelkowitz. *Programming Languages: Design and Implementation*. — 4-е издание. — Питер, 2002. — 688 с. — (Классика Computer Science).
7. Майкл Фицджеральд. Регулярные выражения. Основы. — М.: «Вильямс», 2015. — 144 с
8. DeepDiff 8.3.0 documentation [Электронный ресурс] // DeepDiff. - URL: <https://zepworks.com/deepdiff/current/> (дата обращения 08.03.2025)

СОЗДАНИЕ СнК НА БАЗЕ ПРОЦЕССОРА SCR1 НА ПЛАТЕ TANG PRIMER 20K

FPGA — это программируемая логическая интегральная схема. Схемы этого типа могут быть многократно запрограммированы после изготовления, поэтому сфера их использования широка. Логика работы платы описывается на семействе языков HDL, наиболее распространенными являются VHDL, Verilog и System Verilog. FPGA используют на этапе pre-silicon проверки процессоров [3], в качестве вспомогательной периферии какой-либо техники, например для работы компьютерного зрения в автомобиле [2], а также в образовательных целях [1]. С платами такого типа имеет смысл использовать процессоры с архитектурой RISC-V, поскольку эта архитектура достаточно гибкая и может быть использована как в устройствах где важна компактность, так и там, где необходима большая вычислительная мощность.

На базе RISC-V строятся различные ядра и процессоры. Существуют открытые решения, например, ядро SCR1. Оно написано на языке System Verilog, и ранее было запущено разработчиками на платах Digilent Arty (Xilinx), Digilent Nexys 4 DDR (Xilinx), Arria V GX Starter (Intel), Terasic DE10-Lite (Intel). В последнее время на рынке FPGA приобретает популярность новый производитель Gowin, некоторые платы этого производителя можно купить по доступным ценам, это является одной из причин возросшей популярности. В связи с этим возникла задача проверить возможность запуска ядра SCR1 на плате с использованием чипа от этого производителя.

Помимо самого ядра, авторами изделия также был разработан загрузчик SC-BL. В первую очередь он необходим для обеспечения возможности загрузки произвольной программы в память системы на кристалле (СнК), с целью её дальнейшего исполнения. В ней размещаются все компоненты, которые в совокупности реализуют функциональность какого-либо устройства.

Таким образом, целями данной работы является запуск ядра SCR1 на плате Tang Primer 20K, и запуск загрузчика на получившейся СнК. Для достижения первой поставленной задачи необходимо определиться с тем, какие устройства периферии необходимо добавить в СнК, и как ядро будет соединено с этими устройствами.

Для соединения ядра с устройствами на СнК была выбрана шина АНВ. Оно уже поддерживает соединение по этой шине, АНВ разделяет фазу адреса и фазу данных, и позволяет в конкретный момент времени передавать данные только от одного устройства периферии. Выбор устройства обеспечивается с помощью АНВ мультиплексора, который необходимо описать самостоятельно, и добавить на СнК.

Для обмена информацией с внешними устройствами, в СнК также необходимо добавить асинхронный приемопередатчик. В предыдущих реализациях использовался UART 16550, но для текущей СнК нельзя переиспользовать реализацию этого устройства из предыдущих проектов, поскольку сейчас используется другая шина. По этой причине использовалась другая открытая реализация АНВ UART 16550 [4], которая размещена на платформе GitHub. Для размещения загрузчика на СнК также было необходимо добавить Dual Port ROM память, которая могла бы работать с шиной инструкций, и шиной данных, что было сделано.

Также в схему SCR1 была добавлена реализация TCM памяти для текущей платы. В силу особенностей B-SRAM памяти на плате, переиспользование имеющихся реализаций TCM для других плат не представлялось возможным. Поддержка режима Dual Port ограничена, и в текущей реализации из памяти можно читать только машинное слово длиной 32 бита.

После добавления всех устройств периферии, СнК выглядит так, как показано на рисунке 1.

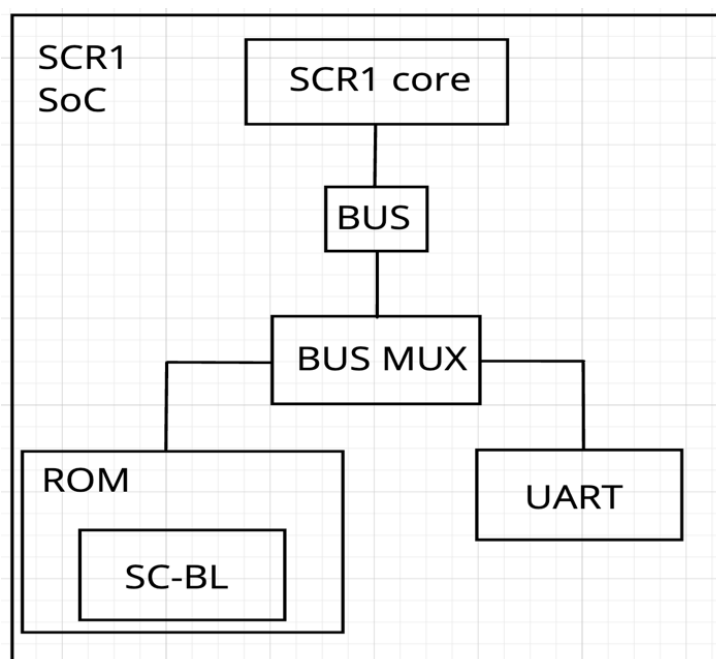


Рисунок 1 – СнК SCR1

После того, как СнК была реализована, появилась необходимость модифицировать программу загрузчика. Т.к. текущая реализация СнК использует отличную от других проектов схему расположения загрузчика в памяти – было необходимо полностью переписать linker script SC-BL. Также была модифицирована программа загрузчика, добавлена условная компиляция для платы Tang Primer 20K, убраны блоки работы с семисегментным индикатором т.к. на плате он отсутствует.

Итогом работы стал запуск ядра SCR1 с загрузчиком SC-BL на плате Tang Primer 20K. С исходным кодом работы вы можете ознакомиться в репозиториях [5], [6], [7].

ЛИТЕРАТУРА

1. Andreas Koch, Ulrich Golze. FPGA Applications in Education and Research [Электронный ресурс]. Режим доступа: https://www.esa.informatik.tudarmstadt.de/assets/publications/materials/1993/1993_EUROCHIP93.pdf. [Дата обращения: 16.08.2024]
2. David Castells-Rufas, Vinh Ngo, Juan Borrego-Carazo, Marc Codina, Carles Sanchez, Debora Gil, Jord Carrabina. A Survey of FPGA-Based Vision Systems for Autonomous Cars [Электронный ресурс]. Режим доступа: <https://ieeexplore.ieee.org/document/9991145>. [Дата обращения: 24.08.2024]
3. Farooq, U., Mehrez, H. Umer Farooq Habib Mehrez. Pre-Silicon Verification Using Multi-FPGA Platforms: A Review [Электронный ресурс]. <https://link.springer.com/article/10.1007/s10836-021-05929-1>. [Дата обращения: 28.08.2024]
4. Github-репозиторий [Электронный ресурс]. Режим доступа: https://github.com/zheInio/ahb_lite_uart16550. [Дата обращения: 04.02.2025]
5. Github-репозиторий [Электронный ресурс]. Режим доступа: <https://github.com/SurfaceYellowDuck/sc-bl>. [Дата обращения: 04.02.2025]
6. Github-репозиторий [Электронный ресурс]. Режим доступа: <https://github.com/SurfaceYellowDuck/fpga-sdk-prj>. [Дата обращения: 04.02.2025]
7. Github-репозиторий [Электронный ресурс]. Режим доступа: <https://github.com/SurfaceYellowDuck/scr1>. [Дата обращения: 04.02.2025]

АНАЛИЗ АРХИТЕКТУРНОЙ ДОКУМЕНТАЦИИ СРАВНЕНИЕ С4-ДИАГРАММ С КАТАЛОГАМИ ПРОЕКТА

В условиях постоянного усложнения программных систем, документирование архитектуры становится неотъемлемой частью эффективного процесса разработки и последующего сопровождения программного обеспечения [1]. В частности, модель С4 [2-4], представляющая собой структурированный подход к визуализации архитектуры, выделяет четыре уровня детализации системы: контекст, контейнеры, компоненты и код. Однако со временем архитектурная документация, представленная в виде С4-диаграмм, может не соответствовать реальной структуре каталогов и файлов проекта. Это несоответствие может привести к трудностям при сопровождении системы, поскольку архитектурные решения, зафиксированные в документации, становятся устаревшими и неактуальными относительно фактической реализации проекта. Для минимизации таких рисков необходимо наличие инструмента, который позволяет автоматически проверять соответствие между С4-диаграммами и реальной структурой проекта, а также выявлять расхождения, которые могут затруднить дальнейшее развитие и поддержку программного обеспечения.

Целью данного исследования является разработка программного приложения, которое будет автоматически анализировать соответствие между элементами контейнерных и компонентных С4-диаграмм и фактической структурой каталогов проекта. Приложение также должно генерировать отчеты о выявленных несоответствиях, что способствует поддержанию актуальности архитектурной документации и помогает разработчикам выявлять и устранять расхождения между проектной документацией и реальной реализацией.

Для достижения поставленной цели было разработано приложение, состоящее из нескольких модулей, каждый из которых решает конкретную задачу в рамках анализа и сравнения данных. Общая схема обработки данных представлена на рисунке 1. Рассмотрим каждый модуль более подробно:

1. Модуль анализа структуры каталогов

Этот модуль реализован на языке Python с использованием стандартной библиотеки `os`. Задача модуля — рекурсивно обходить каталоги проекта и строить графовую структуру, в которой вершины будут представлять папки, а рёбра — связи между ними. Данный граф отражает иерархию каталогов проекта, которая будет использована для дальнейшего анализа.

2. Модуль анализа С4-диаграмм

Модуль также реализован на языке Python, но для обработки XML-файлов [5] используется библиотека `lxml` [6]. Модуль выполняет анализ С4-диаграмм, извлекая информацию о контейнерах и их взаимосвязях. На основе этих данных строится дерево и граф, которые будут использованы для дальнейшего сравнения с графом структуры каталогов. Модуль обеспечивает точное извлечение всех необходимых данных из XML-структуры, что позволяет строить корректные модели архитектуры на основе диаграмм.

3. Модуль сравнения структур

Модуль сравнения структур реализует алгоритм для сопоставления двух графов: одного, полученного на основе анализа структуры каталогов, и другого, извлечённого из С4-диаграмм. Основная задача этого модуля — выявить несоответствия между архитектурной документацией и фактической структурой проекта. Сравнение осуществляется по вершинам графов, и модуль ищет как отсутствующие, так и избыточные элементы. Результатом работы этого модуля является нормированная статистика, которая отражает степень соответствия между архитектурной моделью и реальной реализацией проекта.

4. Модуль генерации отчётов

Генерация отчётов осуществляется с использованием данных, полученных на этапе сравнения. Отчёты могут быть сформированы в машиночитаемом формате JSON, что позволяет легко интегрировать их с другими инструментами разработки. Помимо этого, отчёты содержат визуальные представления графов, сгенерированные с помощью библиотеки matplotlib, что позволяет наглядно представить выявленные несоответствия. В отчётах отражаются следующие данные: информация о выявленных расхождениях, статистика покрытия элементов C4-диаграммы в структуре проекта, а также результаты анализа каталогов и компонентов диаграмм. Такой формат отчетов обеспечивает удобочитаемость и помогает разработчикам быстро анализировать результаты.

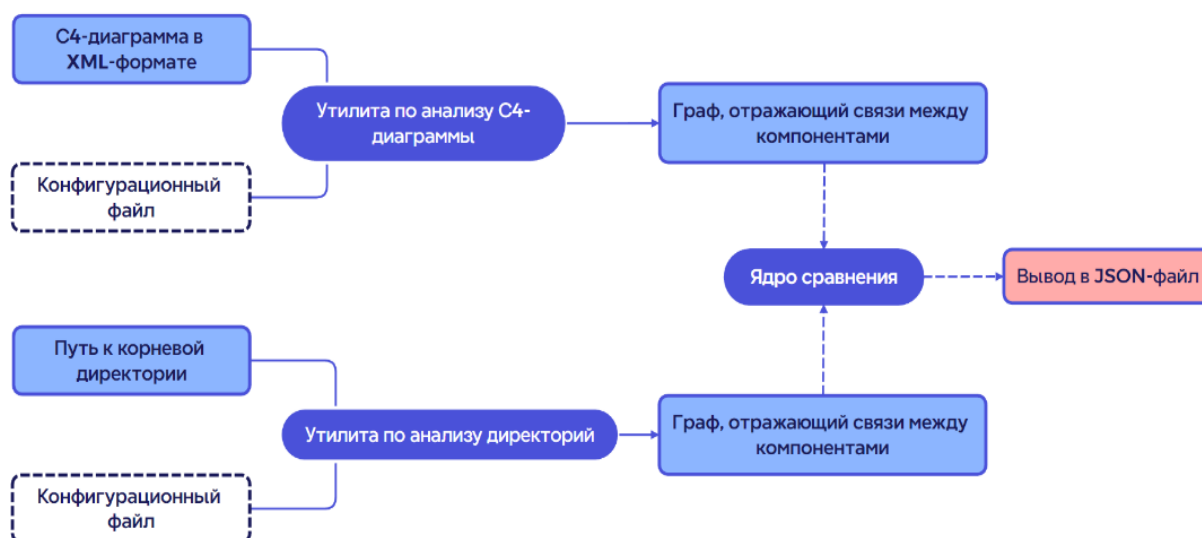


Рисунок 1 – Предлагаемая архитектура сервиса

Для анализа структуры каталогов был выбран язык Python, что обусловлено его высокой производительностью и возможностями для работы с файловой системой, а также техническим заданием. Использование стандартного модуля `os` позволяет эффективно обходить директории и собирать информацию о папках и файлах. Преимущества Python в контексте работы с файловыми системами и его простота позволяют ускорить процесс разработки и тестирования данного модуля.

Для обработки C4-диаграмм был использован парсер XML с библиотекой `lxml.etree`, что обеспечило высокую гибкость и возможность извлечения данных о контейнерах и компонентах без необходимости вручную обрабатывать каждый элемент диаграммы. Для выявления расхождений между структурой каталогов проекта и C4-диаграммой программа осуществляет сопоставление графов структуры каталогов и C4-диаграмм по вершинам, выявляя отсутствующие или избыточные элементы. Результатом является нормированная статистика, отражающая степень соответствия между архитектурной документацией и фактической реализацией проекта.

На основе результатов сравнения формируются отчёты в машиночитаемом формате JSON и человекочитаемом виде с помощью визуализации графов. Отчёты содержат информацию о выявленных несоответствиях, статистику покрытия элементов C4-диаграммы в структуре проекта и наоборот, а также результаты обработки каталогов и компонентов диаграммы. Форматирование отчётов обеспечивает их удобочитаемость и возможность интеграции с другими инструментами разработки.

В ходе разработки приложения был проведен ряд тестов на реальных проектах разного масштаба, содержащих как C4-диаграммы, так и соответствующие структуры каталогов. Тестирование подтвердило, что разработанный инструмент позволяет точно и эффективно выявлять несоответствия между архитектурной документацией и реальной реализацией

проекта. Также было установлено, что отчёты, генерируемые приложением, являются информативными и удобными для анализа, что делает инструмент полезным для поддержания актуальности архитектурной документации и улучшения качества разработки программного обеспечения.

ЛИТЕРАТУРА

1. 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.
2. С. Браун. Модель C4 для визуализации архитектуры программного обеспечения [Электронный ресурс] // C4model.com. – URL: <https://c4model.com/>.
3. Нотация моделирования архитектуры C4 — примеры диаграмм [Электронный ресурс] // Habr. – URL: <https://habr.com/ru/articles/778726/>
4. Описание архитектуры системы с помощью C4 model [Электронный ресурс] // Bool.dev. – URL: <https://bool.dev/blog/detail/c4-model-architecture>.
5. Blyth, A. Security analysis of XML, XML usage and XML parsing / A. Blyth, D. Cunliffe, Ia. Sutherland // Computers & Security. – 2003. – Vol. 22, No. 6. – P. 494. – EDN DZFITH.
6. Lxml — XML и HTML с Python [Электронный ресурс] // Lxml.de. – URL: <https://lxml.de/>.
7. Дрянкова, Д. А. Визуализация данных с помощью библиотек Pandas и Matplotlib для языка программирования Python / Д. А. Дрянкова // Дневник науки. – 2023. – № 6(78). – DOI 10.51691/2541-8327_2023_6_10. – EDN TNKANC.

УДК 621.396

И.А. Попович, В.А. Пушница (4 курс бакалавриата),
А.А. Калачиков, к.т.н., доцент

ЭКСПЕРИМЕНТАЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ СИНХРОНИЗАЦИИ OFDM ДЛЯ БЮДЖЕТНЫХ SDR

Ортогональное частотное разделение каналов (OFDM) широко применяется в беспроводных системах (802.11n, LTE, 5G) связи благодаря устойчивости к многолучевому распространению. Однако OFDM чувствителен к ошибкам синхронизации по времени и частоте. В данной работе реализована OFDM-система на SDR, разработаны алгоритмы синхронизации и проведена их экспериментальная оценка на платформе ADALM-Pluto.

Методика OFDM передаёт данные на ортогональных поднесущих, для чего используется IFFT и циклический префикс. Несовпадение частоты передатчика и приёмника вызывает смещение несущей частоты (CFO), требующее компенсации. Частотная синхронизация реализуется с использованием опорных сигналов Zadoff-Chu (PSS).

Принцип символьной синхронизации. Символьная синхронизация в OFDM необходима для точного определения границ символов в принятом сигнале. Для этого применяется метод автокорреляции, основанный на повторяющихся структурах сигнала, в частности, циклическом префиксе. Автокорреляция позволяет определить точку начала символа и минимизировать влияние межсимвольной интерференции.

Частотная синхронизация компенсирует рассогласование частоты между передатчиком и приёмником, вызванное нестабильностью генераторов, что можем видеть на рисунке 1 и на рисунке 2. Для этого измеряется фазовый сдвиг между повторяющимися фрагментами сигнала и вычисляется частотное смещение, которое затем компенсируется на уровне приёмника, что видим на рисунке 3 и на рисунке 4.

Передатчик и приёмник представляют собой ноутбуки, работающие на операционной системе Ubuntu 22.04. Оценивание проводится с использованием OFDM-сигнала и параметров SDR, представленных в таблице 1.

Таблица 1 – Системные параметры SDR и OFDM

SDR оборудование	Adalm Pluto
Драйвер программного обеспечения	LibIIO+SoapySDR
Несущая частота	2 ГГц
Количество поднесущих	128
Количество пилотных поднесущих	6
Длина циклического префикса	32
Частота дискретизации	1.92 МГц
Расстояние между поднесущими	15 кГц
Форматы модуляции	BPSK, QPSK, QAM-16, QAM-64

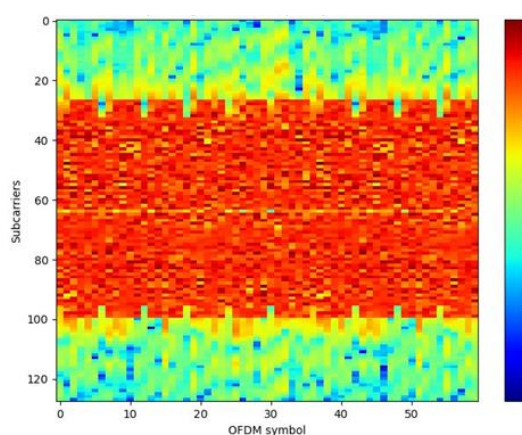


Рисунок 1 – Ресурсная сетка принятого сигнала до частотной коррекции

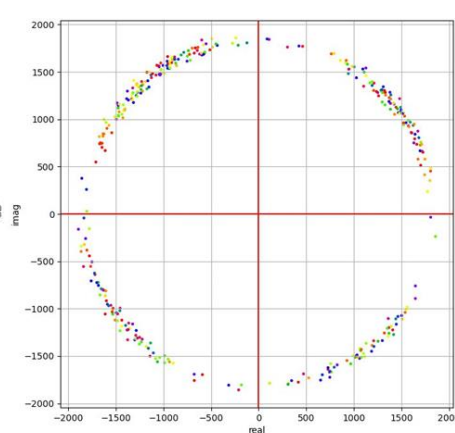


Рисунок 2 – Созвездие сигнала до коррекции частоты

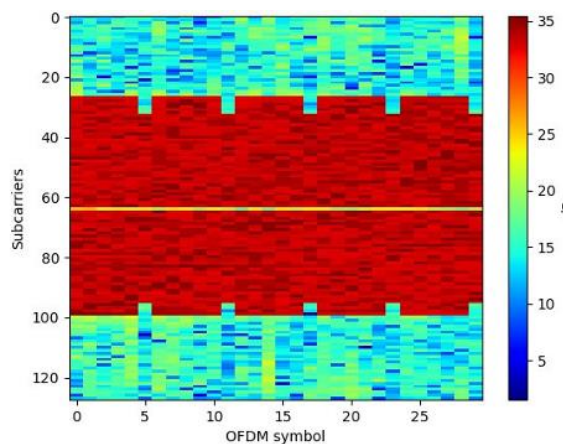


Рисунок 3 – Сетка ресурсов принимаемого сигнала после частотной коррекции

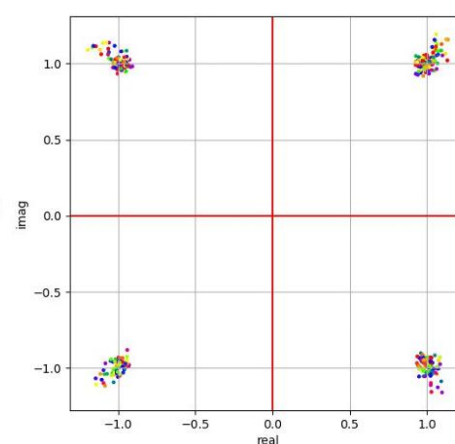


Рисунок 4 – Созвездие сигнала после частотной коррекции

ЛИТЕРАТУРА

1. Ahmadi, Sassan. Lte-advanced: a Practical Systems Approach to Understanding the 3gpp Lte Releases 10 and 11 Radio Access Technologies. Oxford: Elsevier Science, 2014.
2. Travis Collins; Robin Getz; Alexander Wyglinski; Di Pu, Software Defined Radio for Engineers, Artech, 2018.
3. Yong Soo Cho; Jaekwon Kim; Won Young Yang; Chung G. Kang, MIMO-OFDM Wireless Communications with MATLAB, Wiley, 2010

4. M. Morelli, C.-C.J. Kuo, M.-O. Pun, "Synchronization Techniques for Orthogonal Frequency Division Multiple Access (OFDMA): A Tutorial Review". Proc. IEEE, vol. 95, no. 7, pp. 1394-1427.
5. M. M. Wang, A. Agrawal, A. Khandekar, and S. Aedudodla, "Preamble design, system acquisition, and determination in modern OFDMA cellular communications: an overview", IEEE Commun. Mag., vol. 49, n.7, pp. 164-175, July 2011

УДК 004.4

А.А. Пономарев (3 курс бакалавриата),
А.М. Сабуткевич (аспирант),
И.В. Никифоров, к.т.н., доцент

ПРИМЕНЕНИЕ СОБЫТИЙНО-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ (EDA) В СИСТЕМАХ ОАМ ДЛЯ УПРАВЛЕНИЯ СЕТЕВЫМИ УСТРОЙСТВАМИ

Системы управления и автоматизации (ОАМ – Operations, Administration and Maintenance) [1] являются важной частью телекоммуникационной инфраструктуры, обеспечивая мониторинг, управление и настройку сетевых устройств. В традиционных архитектурах такие системы часто используют синхронные запросы и жестко связанные сервисы, что может привести к проблемам с масштабируемостью, гибкостью и устойчивостью при увеличении числа управляемых устройств. В последние годы набирает популярность событийно-ориентированная архитектура (EDA – event-driven architecture) [2], которая позволяет решать эти проблемы за счет асинхронной обработки событий и уменьшения зависимостей между компонентами системы [3].

В работе необходимо рассмотреть, как переход от классической архитектуры ОАМ к событийно-ориентированной архитектуре влияет на управление сетевыми устройствами, с демонстрацией конкретных примеров и анализом преимуществ и недостатков обеих моделей.

Предлагаемая в работе реализация архитектуры компонента базовой станции, отвечающего за функции ОАМ, основываясь на функциональных требованиях, предъявляемых к сетевым элементам [4], представлена на рисунке 1. Функции ОАМ реализуют в виде сервис-ориентированной архитектуры, где отдельные контейнеры имеют строго разграниченные зоны ответственности, что уже дает большой потенциал для масштабирования и является первым шагом для перехода на событийно-ориентированную архитектуру [5].

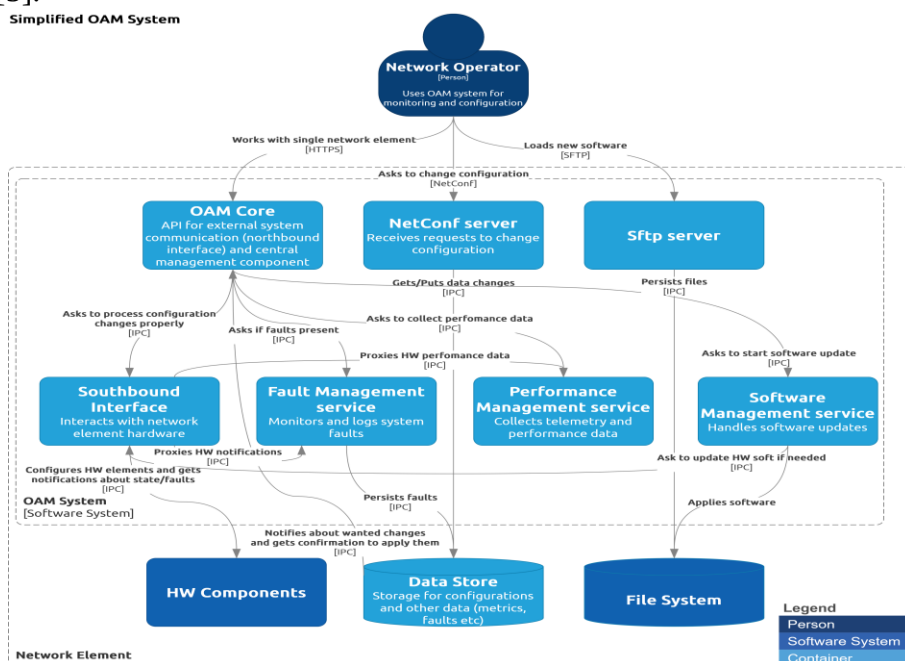


Рисунок 1 – Пример сервис-ориентированной архитектуры ОАМ системы

Специфика управления сетевыми устройствами заключается в требовании к реактивности, так как одну из ключевых ролей играет работоспособность физических компонентов, для чего необходим постоянный мониторинг и определенная гибкость для своевременного реагирования на их сигналы. Одновременно с этим, все процедуры сильно связаны между собой и требуют строгой очередности, декларируя набор зависимостей между ними.

В текущем представлении взаимодействие между компонентами происходит синхронно и приводит к нескольким проблемам: жесткие связи; сбой в одном компоненте может заблокировать всю систему; усложненное масштабирование. Вместе с описанной выше спецификой подобные проблемы лишь усиливаются. Несмотря на очевидные недостатки, такая реализация гораздо проще, легче отлаживается и имеет высокие гарантии доставки за счет непосредственного общения двух участников системы [6].

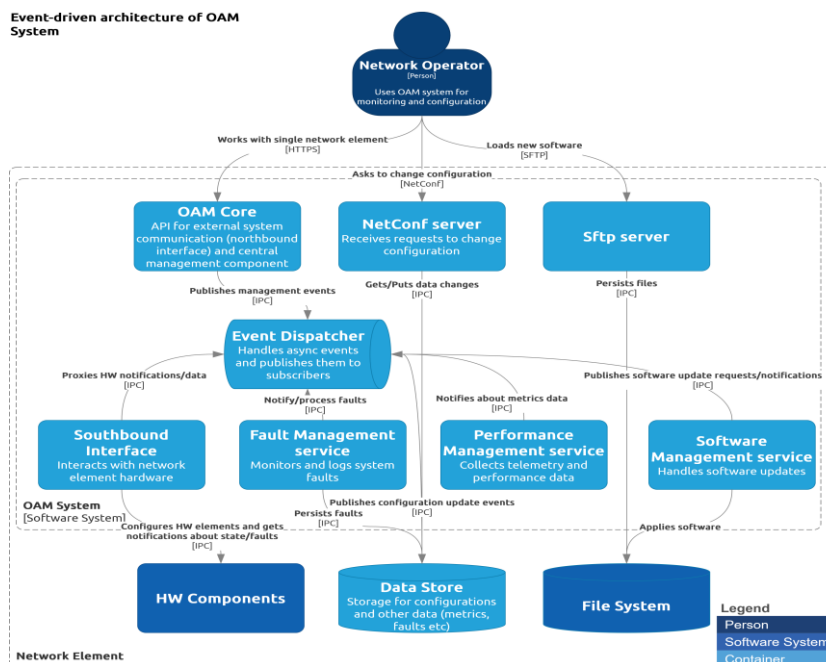


Рисунок 2 – Преобразованный с использованием EDA вариант OAM системы

При переходе на Event-Driven архитектуру, как показано на рисунке 2, центральным в системе становится “Event Dispatcher”, который может быть как просто очередью сообщений, так и оберткой, реализующей некий дополнительный функционал. Этот подход позволяет достичь нескольких преимуществ: сервисы не зависят друг от друга, следовательно, работоспособность системы не зависит от сбоя отдельных компонентов; простое масштабирование и расширение функционала за счет подписки на события; возможность простой интеграции сложной логики зависимостей между событиями, обработки аварий; доступен механизм восстановления состояния системы за счет журнала событий [7]. Однако, такая архитектура приносит возможные проблемы в обеспечении надежности доставки и появлении единой точки отказа.

Внедрение EDA в OAM-системы значительно повышает их эффективность, масштабируемость и гибкость, улучшает устойчивость к отказам, а также может нивелировать свои минусы в комбинации с использованием синхронной модели для критически важных операций (например, экстренной перенастройки). Однако, переход от традиционных архитектурных подходов к событийной модели требует тщательного анализа и адаптации существующих процессов [8] [9].

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: 01.03.2025).

2. 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.
3. 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.
4. 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).
5. 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.
6. Онокой, Л. С. Современные подходы к проектированию архитектуры приложений / Л. С. Онокой, Е. А. Морев // Качество. Инновации. Образование. – 2022. – № 5(181). – С. 90-95. – DOI 10.31145/1999-513x-2022-5-90-95. – EDN LKGDHT.
7. Koldehofe B. et al. Rollback-recovery without checkpoints in distributed event processing systems //Proceedings of the 7th ACM international conference on Distributed event-based systems. – 2013. – С. 27-38.
8. Cabane H., Farias K. On the impact of event-driven architecture on performance: An exploratory study //Future Generation Computer Systems. – 2024. – Т. 153. – С. 52-69.
9. Сысоев, И. М. Создание подхода автоматизации обновления конфигурационных файлов программного обеспечения / И. М. Сысоев, И. В. Никифоров, Е. В. Каплан // Современные технологии в теории и практике программирования : сборник материалов конференции, Санкт-Петербург, 19 апреля 2019 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2019. – С. 126-127. – EDN YARVRM.

УДК 004.4'422

А.С. Росенко (2 курс бакалавриата),
С.А. Молодяков, д.т.н., профессор

ВНЕДРЕНИЕ GOLANG В RISC-V РАЗРАБОТКУ: АНАЛИЗ АКТУАЛЬНЫХ РАЗРАБОТОК И ПЕРСПЕКТИВ

Архитектура RISC-V, отличающаяся открытостью и гибкостью, демонстрирует стремительный рост популярности в различных сегментах — от встраиваемых систем до серверных решений. Компания Syntacore, дочерняя структура YADRO, занимает лидирующую позицию в разработке компиляторов и инструментов для RISC-V, включая такие проекты, как llvm-snipru, преимущественно на языках C и C++. Вместе с тем язык программирования Go (Golang), разработанный Google, благодаря своей простоте, эффективности и встроенной поддержке параллелизма, приобретает все большее значение в современных разработках. Внедрение и поддержка Golang в экосистеме RISC-V становятся актуальными задачами, требующими как оптимизации официального компилятора Go, так и создания собственных решений на базе существующих open-source проектов.

На текущий момент число реализованных решений для RISC-V остается ограниченным, а круг специалистов, работающих с данной архитектурой, — узким. Это подчеркивает необходимость привлечения новых разработчиков в RISC-V сообщество. Интеграция Golang может стать катализатором этого процесса: оптимизировав исполнение Golang на RISC-V системах, можно привлечь специалистов и энтузиастов в сфере разработки и оптимизации контейнеризации и оркестрации приложений, (Docker и Kubernetes – главные инструменты в этой области, написанные на Go). Кроме того, Golang актуален в сфере блокчейн-разработки

(к примеру Geth (go-ethereum) – реализация Ethereum на языке Go), что способно привлечь в RISC-V экосистему специалистов из данной области, расширяя применение RISC-V и стимулируя в том числе выпуск новых физических реализаций – чипов специализированных конфигураций. Таким образом, поддержка Go не только усиливает технические возможности архитектуры, но и способствует росту сообщества разработчиков.

Актуальность внедрения Golang в RISC-V подтверждается современными тенденциями: переход TypeScript на компилятор, написанный на Go, значительно ускорил его работу, а переписывание компилятора Go на самом Go повысило его эффективность. Для компании YADRO, активно использующей Go в проектах для базовых станций, систем хранения и сетевых устройств, оптимизация Go-кода под RISC-V приобретает стратегическое значение в будущем. Это требует целенаправленных усилий по адаптации существующих инструментов и разработке новых решений, совместимых с архитектурой.

Доклад включает в себя практическое исследование и оценку актуальных интеграций Golang в RISC-V архитектуру. Один из тестов предполагает сравнительный анализ производительности и размера бинарных файлов, сгенерированных компиляторами Go (gc), GCC (C) и Clang (C) на эмуляторе QEMU RISC-V. Тест заключался в расчете интеграла методом трапеций. Go значительно превосходит C без оптимизаций, но уступает C с оптимизацией -O3.

Важно отметить, что в стандартной поставке Go начиная с версии 1.14 (2020 год) есть поддержка GOARCH=riscv64, то есть 64-битного RISC-V[1]. На данный момент (Go 1.24) компилятор Go для RISC-V (riscv64) работает стабильно, но уступает по производительности компиляторам для x86_64 и ARM64[2]. Это связано с тем, что оптимизации для RISC-V ещё не полностью развиты – LLVM и GCC оптимизируют код лучше, генерация кода Go для RISC-V сравнительно слабая в аспекте работы с регистрами и инструкциями, и всё ещё отсутствует JIT (just-in-time) компиляция. Тем не менее архитектура поддерживается разработчиками в выходящих версиях компилятора, более того добавляются новые возможности и у сторонних разработчиков есть возможность контрибутировать в проект.

Уместна оптимизация размера бинарных файлов, генерируемых gc (Golang компилятором). Они больше бинарных файлов, генерируемых компиляторами языка C, т. к. компилятор Go добавляет в них собственный runtime, который включает garbage-collector, реализацию горутин, планировщика потоков и безопасной работы с памятью. Таким образом бинарный файл Go обладает большей самодостаточностью, но и его размер больше.

Также необходимо затронуть аспект bare-metal разработки[3], что актуально для программирования микроконтроллеров, оснащенных RISC-V процессорами. В этой сфере существует перспективный проект TinyGo – компилятор Go для микроконтроллеров, создающий компактные бинарные файлы. TinyGo использует LLVM, набор инструментов компиляции, обеспечивающий оптимизации, такие как инлайнинг функций и векторизация. Однако поддержка RISC-V в TinyGo ограничена, а небольшое сообщество и малый список совместимых плат сдерживают его развитие. TinyGo – open-source проект, что позволяет взять его за основу для дальнейших разработок. Перспективными направлениями разработки на основе этого компилятора являются: оптимизация LLVM для лучшей работы с командами RISC-V ISA[4] и параллельная поддержка нового продукта с обновлениями оригинального проекта TinyGo (при выходе новых официальных версий Golang и соответственно новыми версиями данного компилятора).

В рамках доклада проводится анализ текущего состояния и перспектив внедрения Golang в разработку под архитектуру RISC-V, включая направления разработки под ОС (RISC-V Linux дистрибутивы, к примеру Yocto/Poky сборки) и bare-metal программирования. Рассматривается проект TinyGo, использующий LLVM для оптимизации кода, отмечаются ограничения, возможности и перспективы данного и подобных проектов.

Подчёркивается актуальность оптимизации компилятора Go для RISC-V, в том числе улучшение регистровой аллокации, внедрение JIT и сокращение размера исполняемых файлов. Предлагается изучить возможности контрибьютинга в TinyGo и gc (Go) для

расширения поддержки RISC-V. Также обсуждается стратегическая значимость этого направления для компаний YADRO и Syntacore, активно применяющих Go в разработке базовых станций, систем хранения и сетевых устройств.

ЛИТЕРАТУРА

1. Программирование на языке ассемблера RISC-V / пер. с англ. А. В. Логунова; под науч. ред. А. Ю. Романова. – М.: ДМК Пресс, 2025. – 276 с.: ил. – (Книжная полка Истового инженера)
2. Embedded Go. Bare-metal programming RISC-V in Go // [Электронный ресурс]. – 2020. – Режим доступа: https://embeddedgo.github.io/2020/05/31/bare_metal_programming_risc-v_in_go.html.
3. Burela D. Cross-compiling Golang for RISC-V // [Электронный ресурс]. – 2020. – Режим доступа: <https://blog.davidburela.com/2020/11/21/cross-compiling-golang-for-risc-v/>.
4. Mangum D. RISC-V Bytes: Go 1.19 Register Calling // [Электронный ресурс]. – 2022. – Режим доступа: <https://danielmangum.com/posts/risc-v-bytes-go-1-19-register-calling/>.
5. The Go Programming Language. [Электронный ресурс]. – Режим доступа: <https://go.dev/>.

УДК 004.932.2

А.Е. Савкин (2 курс магистратуры),
П.А. Шагалова, к.т.н., доцент

ПРИМЕНЕНИЕ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ В РАЗРАБОТКЕ СИСТЕМ КОМПЬЮТЕРНОГО ЗРЕНИЯ НА БАЗЕ RISC-V

На сегодняшний день разработка решений в области систем искусственного интеллекта и компьютерного зрения, использующих открытую и гибкую процессорную архитектуру RISC-V, является одним из актуальных и перспективных направлений [1,2].

В данной работе рассматриваются вопросы реализации и запуска нейронных сетей на процессорной архитектуре RISC-V. Для проведения исследований выбрана плата Lichee Pi 4A - плата для разработки Linux на основе Risc-V, использующая базовый модуль Lichee Module 4A (его основной чип - TH1520 содержит нейронный процессор 4TOPS @ int8 AI NPU). Апробация выполнена с использованием нейросетевых моделей YOLOv5n, YOLOv11n, обученных для решения задачи анализа изображений микроскопии [3,4].

Рассматриваемый одноплатный компьютер использует фреймворк HNB (Heterogeneous Honey Badger) представляющий собой набор инструментов для развёртывания моделей нейронных сетей, предоставленный компанией T-Head для платформы Xuantie chip. Он включает в себя ряд инструментов, необходимых для развёртывания, таких как оптимизация компиляции, анализ производительности, отладка процессов и моделирование результатов. Поддерживает модели в различных форматах, таких как Caffe, TensorFlow, ONNX и TFLite. Поддерживает такие типы данных, как 8/16-разрядные с фиксированной запятой и 16/32-разрядные с плавающей запятой. Поддерживает симметричное и асимметричное поканальное квантование. Оптимизирует сетевую структуру модели перед развёртыванием. Компилирует и генерирует двоичные файлы, исполняемые на платформе Wujian SoC.

Для запуска модели нейронной сети на одноплатном компьютере были осуществлены следующие шаги:

1. Установка ОС Debian на базе ядра Linux Kernel 5.10.
2. Подготовка окружения (установка HNB на хост-машину, установка SHL (Structure of Heterogeneous Library на одноплатный компьютер).
3. Конвертация обученной модели YOLO в ONNX-формат.
4. Ассиметричная int8 квантизация сконвертированной модели с помощью HNB.
5. Кросс-компиляция модели ONNX в исполняемую программу на CPU/NPU.
6. Запуск модели на целевом устройстве (результаты представлены в таблице 1).

Таблица 1 – Сравнение параметров и показателей эффективности при запуске YOLOv5n, YOLOv11n на одноплатном компьютере Lichee Pi 4A

Модель	Устройство запуска	Precision	Скорость выполнения графа сети, FPS
YOLOv5n	NPU (TH1520)	Int8	186.56
YOLOv5n	CPU (C920)	Fp16	2.59
YOLOv11n	NPU (TH1520)	Int8	5.82
YOLOv11n	CPU (C920)	Fp16	1.01

Как можно видеть из таблицы выше, гетерогенный инференс модели с использованием NPU показывает значительный прирост производительности по сравнению с исполнением только на центральном процессоре. Также можно отметить, что современные архитектуры последнего поколения могут показывать значительное снижение скорости обработки изображений из-за отсутствия нативной поддержки выбранным оборудованием и программным обеспечением и, в связи с этим, требовать дополнительных оптимизаций.

Современные одноплатные компьютеры на базе открытой архитектуры RISC-V отлично подходят для решения самых различных задач компьютерного зрения. Результаты экспериментов подтверждают эффективность аппаратного ускорения нейронных сетей, позволяя достичь скорости обработки изображений до 186 кадров в секунду (без учета времени на пре-процессинг и обработку финального результата), чего с лихвой достаточно для создания энергоэффективных real-time систем. При этом современные архитектуры нейросетей (например, YOLOv11n) требуют дополнительной адаптации под RISC-V, что требует дополнительного времени. Квантование int8 снижает точность незначительно (на 1–2%), но существенно сокращает вычислительные ресурсы.

ЛИТЕРАТУРА

1. Харрис, Сара Л. Харрис Дэвид. Цифровая схемотехника и архитектура компьютера: RISC-V. Ред. А.Ю.Романов. Перев. А.Ю. Романова В.С. Яценкова. ДМС Пресс, 2021. — С. 810.
2. RISC-V [Электронный ресурс] Режим доступа: <https://riscv.org/>
3. Савкин А.Е., Шагалова П.А., Корелин О.Н. Исследование применения технологий компьютерного зрения для анализа биомедицинских изображений // Труды НГТУ им. Р.Е. Алексеева. 2024. № 1 (144). С. 16-23.
4. Савкин А.Е., Захарова А.Ю., Шагалова П.А. Применение нейросетевых технологий для детекции клеток крови на биомедицинских изображениях // XXVII Нижегородская сессия молодых ученых: технические, естественные, математические науки. Материалы сессии. Нижний Новгород, 2022. С. 62-66.

УДК 004.4'2

А.С. Семичев (4 курс бакалавриата),
О.В. Александрова, ст. преподаватель

РАЗРАБОТКА ВЫСОКОПРОИЗВОДИТЕЛЬНОЙ БИБЛИОТЕКИ ДЛЯ АСИНХРОННОГО ВЗАИМОДЕЙСТВИЯ С POSTGRESQL

Современные высоконагруженные приложения требуют эффективных решений для работы с базами данных. Асинхронное взаимодействие с базами данных позволяет сократить задержки и повысить производительность системы за счёт перекрытия операций ввода-вывода и вычислений, выполняемых сервером и клиентом [1]. Однако, существующие библиотеки для работы с PostgreSQL имеют ограничения в области асинхронного взаимодействия [2], что затрудняет их использование в масштабируемых системах на языке программирования C++.

Разработка специализированной библиотеки, интегрированной с фреймворком Seastar, позволит значительно повысить производительность взаимодействия с базой данных.

Целью данного проекта является создание высокопроизводительной библиотеки для асинхронного взаимодействия с PostgreSQL, оптимизированной для многопроцессорных систем и способной обрабатывать большие нагрузки без значительных задержек.

Для достижения указанной цели необходимо решить следующие задачи:

- Реализация асинхронного интерфейса для работы с PostgreSQL.
- Оптимизация обработки SQL-запросов с учетом многопоточности.
- Поддержка ключевых возможностей PostgreSQL, включая подготовленные запросы и транзакции.
- Интеграция с фреймворком Seastar для обеспечения высокой масштабируемости.
- Разработка надежного механизма обработки ошибок и восстановления соединений.
- Разработка тестового набора для проверки производительности и надежности библиотеки.

Архитектура библиотеки будет основана на следующих ключевых компонентах:

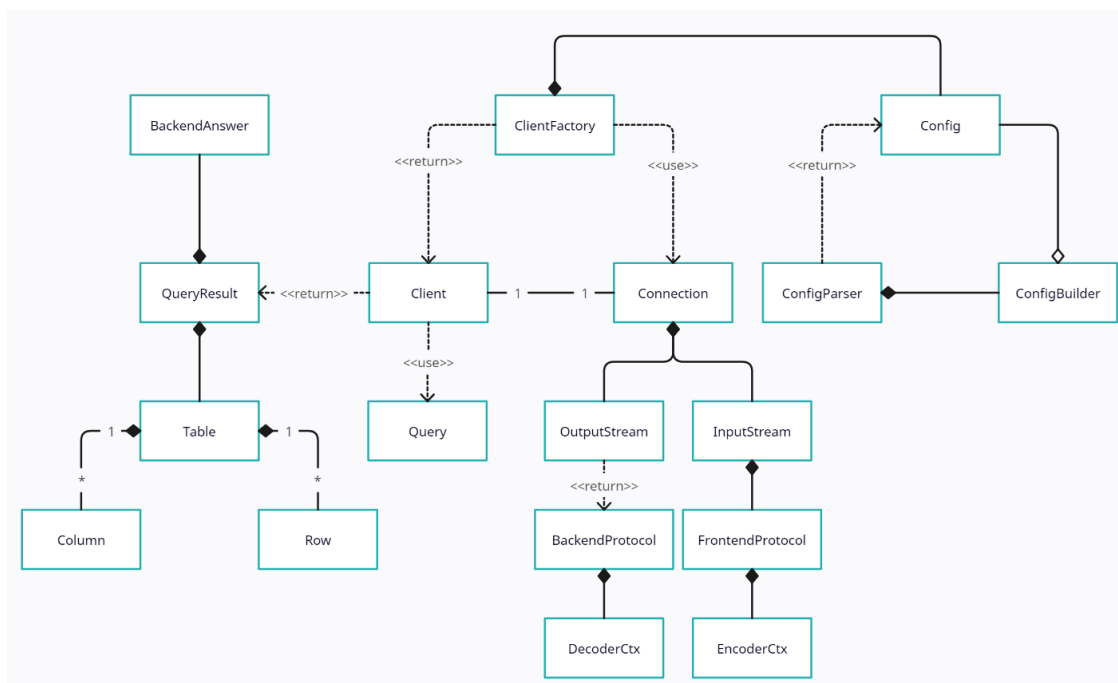


Рисунок 1 – Архитектура библиотеки

– Модуль управления соединением – обеспечивает установку, поддержку и завершение соединений с сервером PostgreSQL.

– Модуль обработки запросов – реализует поддержку протоколов Simple Query и Extended Query, а также управление входящими и исходящими потоками данных.

– Модуль конфигурации – отвечает за обработку параметров подключения и создание соответствующих объектов конфигурации.

– Модуль взаимодействия с протоколами PostgreSQL – реализует кодирование и декодирование сообщений, а также поддержку механизмов аутентификации.

– Модуль обработки результатов – структурирует данные, полученные из базы, и предоставляет удобный интерфейс для работы с ними.

– Интеграционный модуль Seastar – обеспечивает эффективное асинхронное выполнение задач и управление потоками без блокировки.

Разработка библиотеки будет вестись на языке C++ [3] с использованием фреймворка Seastar [4], который предоставляет мощный асинхронный механизм обработки данных. Библиотека будет работать напрямую с бинарным протоколом PostgreSQL [5], минуя использование libpq [6], что позволит минимизировать накладные расходы и повысить скорость взаимодействия. Для тестирования библиотеки будет использоваться фреймворк Catch2 [7], который обеспечит удобство написания и выполнения тестов.

Основные преимущества:

- Высокая производительность – библиотека оптимизирована для работы на многопроцессорных системах.
- Масштабируемость – благодаря шардированию обеспечивается эффективная работа при высокой нагрузке.
- Минимальные задержки – использование Seastar позволяет минимизировать накладные расходы на обработку запросов.
- Гибкость – поддержка ключевых функций PostgreSQL делает библиотеку удобной для широкого круга задач.
- Надежность – механизм автоматического восстановления соединений повышает отказоустойчивость системы.

ЛИТЕРАТУРА

1. Ramachandra, K., Chavan, M., Guravannavar, R., Sudarshan, S. Program Transformations for Asynchronous and Batched Query Submission // Proceedings of the IEEE International Conference on Data Engineering (ICDE). – 2011. – P. 375–386. – DOI: 10.1109/ICDE.2011.5767870.
2. Любимова, Т. В. Асинхронность в программировании / Т. В. Любимова // Университетская наука. – 2019. – № 2(8). – С. 135-138. – EDN BJWENB.
3. ISO/IEC 14882:2024, Programming languages — C++, [Электронный ресурс] Режим доступа: <https://www.iso.org/standard/83626.html>
4. Seastar Documentation. URL, [Электронный ресурс] Режим доступа: <https://docs.seastar.io/master/index.html>
5. PostgreSQL 17.2 Documentation, [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/17/index.html>
6. PostgreSQL 17.2 Documentation: Chapter 32. libpq — C Library, [Электронный ресурс] Режим доступа: <https://www.postgresql.org/docs/17/libpq.html>
7. Catch2 Documentation, [Электронный ресурс] Режим доступа: <https://catch2-temp.readthedocs.io/en/latest>

УДК 004.382.4

Д.В. Симонова (4 курс бакалавриата),
П.А. Шагалова, к.т.н., доцент

ИССЛЕДОВАНИЕ И РЕАЛИЗАЦИЯ ПОДКЛЮЧЕНИЯ УСТРОЙСТВ С ИСПОЛЬЗОВАНИЕМ GPIO ИНТЕРФЕЙСА НА ПЛАТАХ С ПРОЦЕССОРНОЙ АРХИТЕКТУРОЙ RISC-V

Интерфейс ввода-вывода общего назначения (GPIO) – гибкий программируемый интерфейс, используемый для управления устройствами и обеспечения связи между различными аппаратными компонентами, широко применяемый при разработке портативных систем на базе одноплатных компьютеров. В данной работе рассматриваются вопросы использования GPIO интерфейса для разработки и подключения модулей, контролирующих освещение и передвижение предметного столика в автоматизированном портативном устройстве фотофиксации и анализа изображений микроскопии.

Первоначальная реализация системы фотофиксации и анализа изображений микроскопии была выполнена на одноплатном компьютере MangoPi MQ PRO D1[1] однако в процессе разработки, учитывая перспективность и необходимость в высоком быстродействии платформы, было принято решение перейти на одноплатный компьютер Lechee Pi 4A на базе процессорной архитектуры RISC-V. Такой выбор позволяет создать компактное, эффективное и масштабируемое решение, которое в дальнейшем можно будет легко улучшать и оптимизировать [2,3]. Высокая визуальная сложность, присущая изображениям микроскопии, накладывает требования на качество получаемых изображений. В цифровых микроскопах настройка освещения и передвижение предметного столика оказывают непосредственное влияние на качество снимков, что делает разработку соответствующих модулей важной и актуальной задачей.

Разработанная подсистема освещения включает в себя: фотомодуль на базе фоторезистора, набор светодиодов, рассеянных по внутренней части устройства, и резисторы. Определение уровня освещенности при этом выполняется с использованием фотомодуля: если фотомодуль перестает пропускать сигнал – значит освещение недостаточное. Алгоритм настройки освещения следующий: в случае недостаточного освещения сначала с помощью ШИМ-сигнала настраивается яркость светодиода, затем включаются дополнительные светодиоды, после чего они тоже настраиваются и так до тех пор, пока фотомодуль не начнет пропускать сигнал. Вся настройка происходит в автоматическом режиме.

Передвижение предметного столика осуществляется посредством двух сервоприводов SG90. Изначально препарат перемещается до тех пор, пока система не обнаружит его угол. Затем производится фотофиксация, после чего на один из сервоприводов приходит серия кратковременных сигналов, каждый из которых передвигает препарат на расстояние одного кадра микроскопа. Когда достигается противоположная начальному положению сторона исследуемого объекта – в работу включается второй сервопривод и перемещает препарат по другой оси. Процесс повторяется до тех пор, пока не будет достигнут угол препарата. Фотофиксация, передвижение и, при необходимости, склейка целого изображения происходит автоматически.

Система управляется с помощью GPIO[4] интерфейса одноплатного компьютера. Собрана на беспаячных макетных платах на 400 контактов для простоты тестирования и отладки работы прототипа. Программная часть реализована с помощью языка программирования Python и библиотеки pyA20[5], для программного управления GPIO интерфейсом. Тестирование и отладка электронной схемы проводилась в Proteus Design Suite, в котором на базовом уровне было эмулировано поведение одноплатного компьютера.

В результате работы была спроектирована автоматическая система освещения и передвижения предметного столика для электронного микроскопа на базе микрокомпьютера с процессорной архитектурой RISC-V, отличающаяся возможностью легкой модернизации, эффективностью и быстродействием.

ЛИТЕРАТУРА

1. MangoPi [Электронный ресурс] Режим доступа: <https://mangopi.org/>
2. Харрис, Сара Л. Харрис Дэвид. Цифровая схемотехника и архитектура компьютера: RISC-V. Ред. А.Ю.Романов. Перев. А.Ю. Романова В.С. Яценкова. ДМС Пресс, 2021. — С. 810.
3. Шагурин, И.И. Современные микроконтроллеры и микропроцессоры. Горячая линия -Телеком, 2004. — С. 952.
4. The Linux Kernel Archives [Электронный ресурс] Режим доступа: <https://www.kernel.org/doc/html/latest/driver-api/pin-control.html>
5. Orangepi_gpio_pyH3 [Электронный ресурс] Режим доступа: https://github.com/duxingkei33/orangepi_PC_gpio_pyH3

АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ СИСТЕМАМИ ХРАНЕНИЯ ДАННЫХ В KUBERNETES С ИСПОЛЬЗОВАНИЕМ CSI-ДРАЙВЕРА И СПЕЦИФИКАЦИИ SWORDFISH.

В последние годы популярность контейнеризации стремительно растет [1]. В инструменте Kubernetes контейнеры, в отличие от виртуальных машин, являются нестационарными сущностями. Это значит, что данные, хранящиеся внутри контейнера, могут быть утеряны при его перезапуске или удалении. Поэтому для работы с постоянными данными контейнеры должны интегрироваться с внешними системами хранения, которые обеспечат сохранение и доступность данных на протяжении всего жизненного цикла приложения.

Компонентом, обеспечивающим взаимодействие между инструментом Kubernetes и системами хранения данных, является драйвер CSI [2]. Он позволяет инструменту Kubernetes работать с различными типами хранилищ. Однако для работы с корпоративными хранилищами, обладающими сложной архитектурой и высоким уровнем производительности, существующих решений недостаточно. Поэтому появляется потребность в стандартизированных интерфейсах и протоколах для упрощения интеграции, мониторинга и управления хранилищами. Спецификация Swordfish разработана для решения этих задач, обеспечивая унифицированный подход к управлению СХД [3].

Основная задача заключается в автоматизации процессов, связанных с работой с хранилищами в кластере Kubernetes. Внедрение интерфейса CSI в связке со спецификацией Swordfish позволяет минимизировать количество ручных операций, тем самым повысить эффективность управления хранилищами.

Для эффективного управления хранилищами в кластере Kubernetes [4], необходимо учитывать множество аспектов, включая интеграцию с различными СХД, настройку томов, мониторинг их состояния, а также управление данными на протяжении всего жизненного цикла. Без автоматизации данные процессы трудозатратны – это негативно влияет на стабильность и производительность приложений.

В работе предложено моделирование жизненного цикла тома, которое позволяет увидеть, как реализуемое решение автоматизирует управление СХД в кластере Kubernetes. Моделирование состоит из следующих шагов, представленных ниже и на рисунке 1.

1. Создание тома (Provisioning). Когда приложение в кластере Kubernetes запрашивает постоянное хранилище (например, для хранения данных в общих папках), инструмент Kubernetes отправляет запрос через PersistentVolumeClaim (PVC) [5] с необходимыми характеристиками тома. После получения запроса от платформы Kubernetes, драйвер CSI взаимодействует с эмулятором Swordfish API [6, 7] для создания тома. В случае с файловыми системами (например, NFS) эмулятор Swordfish API конфигурирует подключение к серверу NFS, создавая том с требуемыми параметрами.

2. Монтирование тома (Mounting). Система оркестрации контейнеров Kubernetes автоматически монтирует созданный том на соответствующий контейнер или под (Pod). Контейнеры, требующие доступа к постоянным данным, связаны с томом через PersistentVolume (PV). Эмулятор Swordfish API извлекает путь к директории, которая должна быть доступна на сервере NFS. Этот путь добавляется в конфигурацию сервера NFS через etc/exports. Когда инструмент Kubernetes назначает том контейнеру или поду, драйвер CSI выполняет его монтирование, обеспечивая доступ к данным. Для файловых систем, таких как NFS, драйвер CSI учитывает данные, полученные от эмулятора Swordfish API.

3. Удаление тома и очистка ресурсов. Когда том больше не требуется, платформа Kubernetes автоматически удаляет PVC, который инициирует процесс удаления тома. Драйвер CSI взаимодействует с API эмулятора Swordfish для удаления тома. В случае с сетевым файловым хранилищем NFS, система удаляет записи из etc/exports, освобождая ресурсы и очищая связанные метаданные.

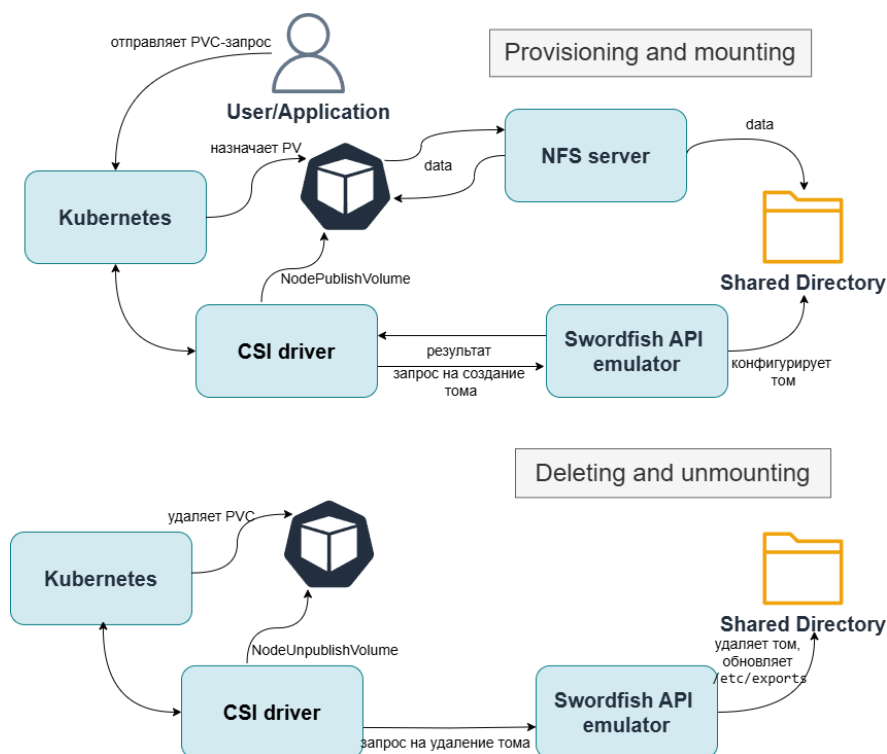


Рисунок 1 – Моделирование жизненного цикла тома

Ранее администраторы вручную создавали, настраивали и монтировали тома хранилищ для контейнеров. Для интеграции между различными типами хранилищ (локальными, облачными или распределенными) требовалась ручная настройка каждого хранилища, а также конфигурация соединений. Благодаря использованию драйвера CSI и спецификации Swordfish эти процессы автоматизируются. Оркестратор контейнеров Kubernetes автоматически запрашивает хранилище через драйвер CSI при создании PVC, сокращая количество операций. Таким образом, до внедрения требовалось выполнить 5 шагов (создание, настройка, экспорт, монтирование, проверка). Теперь же достаточно одного шага – настройки PVC.

ЛИТЕРАТУРА

1. Документация для Kubernetes версии: v1.28 [Электронный ресурс] Режим доступа: <https://v1-28.docs.kubernetes.io/ru/docs/home/>
2. Container Storage Interface (CSI) Specification [Электронный ресурс] Режим доступа: <https://github.com/container-storage-interface/spec?tab=readme-ov-file>
3. SNIA Swordfish v1.2.7. [Электронный ресурс] Режим доступа: <https://www.snia.org/sites/default/files/technical->
4. 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.

5. Konev, I. Algorithm for Containers' Persistent Volumes Auto-scaling in Kubernetes / I. Konev, I. Nikiforov, S. Ustinov // Conference of Open Innovations Association, FRUCT. – 2022. – No. 31. – P. 89-95. – DOI 10.23919/FRUCT54823.2022.9770916. – EDN SJPFON.
6. 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.
7. Юлдашев, В. М. особенности и проблемы mock-сервера Swordfish API Emulator / В. М. Юлдашев, И. В. Никифоров, О. В. Александрова // 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. 298-301. – EDN XSKJFS.

УДК 004.04

С.Ф. Солодовников (1 курс магистратура),
И.В. Никифоров, к.т.н., доцент,
С.М. Устинов, д.ф.-м.н, профессор

ИССЛЕДОВАНИЕ НАГРУЗКИ НА БЛОЧНЫЕ УСТРОЙСТВА ДЛЯ ТЕСТИРОВАНИЯ ЦЕЛОСТНОСТИ СХД

Процесс исполнения большинства тестов для проверки корректности традиционных программных приложений можно рассматривать как процесс валидации корректности работы конечного автомата [1]. В рамках такого подхода фиксируется начальное состояние системы, генерируется конечное множество управляющих сигналов, а затем выполняется проверка соответствия конечного состояния системы ожидаемым результатам.

Однако для высоконагруженных систем, таких как системы хранения данных (СХД), данный метод оказывается неэффективным. Высокая интенсивность входящих запросов и их значительная вариативность не позволяют зафиксировать единственное начальное состояние и предсказать конечное. Дополнительное влияние оказывают случайные факторы, такие как задержки сети при передаче данных, временные задержки дисковых операций и другие системные флуктуации.

По этой причине тестирование и отладка систем хранения данных требуют иного подхода. В данном исследовании используется методика генерации различных нагрузочных профилей с последующей проверкой корректности их обработки системой. Это означает, что в процессе тестирования записывается значительный объем данных (100+% размера блочного устройства), после чего выполняется их чтение и верификация на предмет отсутствия ошибок.

Многообразие нагрузочных профилей и сопутствующих сценариев, включая моделирование отказов различных компонентов системы, определяет полноту тестирования [2]. Однако создание исчерпывающего набора тестовых сценариев, способного гарантировать абсолютную корректность работы системы, представляет собой сложную задачу [3]. Всегда сохраняется вероятность существования редких сочетаний факторов, не учтенных в процессе тестирования, которые могут приводить к сбоям в работе СХД [4].

Целью исследования является анализ характера дисковой нагрузки, создаваемой различными системами управления базами данных (СУБД). Данное исследование направлено на формирование набора профилей для инструмента Flexible I/O tester (fio) [5], отражающих реальные, а не синтетические сценарии, которые могут быть использованы для проверки целостности систем хранения данных. Полученный набор профилей fio поможет сократить ресурсы на проведение acceptance тестирования СХД, потому что тестирование с помощью

инструмента fio имеет поддержку проверки целостности данных, также оно более легковесно, чем запуск реальных СУБД.

Ход исследования представлен ниже.

1. Изучить инструмент fio.
2. Изучить стек инструментов blktrace [6, 7].
3. Собрать трейсы различных сценариев нагрузки нескольких популярных СУБД.
4. Провести анализ данных.
5. Попытаться воспроизвести подобную нагрузку с помощью профилей fio.

Результаты. Для анализа производительности PostgreSQL использовалась утилита pgbench. Тестирование проводилось на тестовой таблице размером 4,5 ГБ, с продолжительностью бенчмарка 10 минут. Испытания выполнялись при различном числе параллельных соединений: 1, 5 и 10. На основе полученных данных был проведен анализ операций чтения и записи с использованием утилиты blkparse. Для каждой категории операций были составлены отдельные таблицы, содержащие ключевые метрики, включая:

- объем записанных данных;
- распределение размеров записанных блоков;
- долю последовательных и случайных операций записи;
- общее соотношение операций чтения и записи.

Для выявления степени последовательности записи был разработан собственный алгоритм, позволяющий анализировать смещение блоков данных.

Бенчмарк СУБД MongoDB проводился с использованием YCSB (Yahoo! Cloud Serving Benchmark) [8] — утилиты с открытым исходным кодом, предназначенной для оценки производительности хранилищ данных. Аналогично для базы данных PostgreSQL были собраны метрики, отражающие характеристики операций чтения и записи. На основе этих данных были построены временные графики, демонстрирующие интенсивность операций в заданных временных интервалах. График количества операций записи/чтения за окно 100 миллисекунд для read-write YCSB профиля представлен на рисунке 1.

Анализ собранных данных позволил составить fio-профиль, который наиболее точно эмулирует нагрузку MongoDB в условиях, описанных выше. График нагрузки созданного профиля представлен на рисунке 2. При этом профиль fio значительно отличается от типичных профилей, применяемых при тестировании систем хранения данных (СХД). В ходе детального исследования трейсов блочного уровня были обнаружены ранее неочевидные закономерности. В частности, выявлено систематическое смещение последовательных записей на 8Кб влево, что требует дальнейшего изучения.

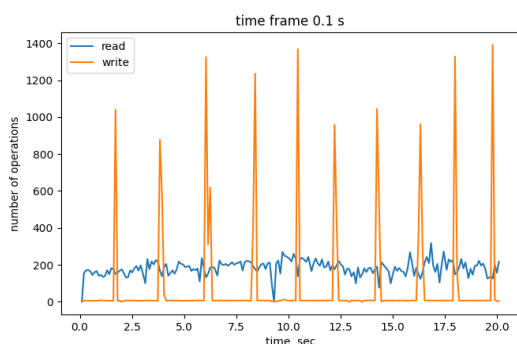


Рисунок 1 – Нагрузка, создаваемая СУБД MongoDB

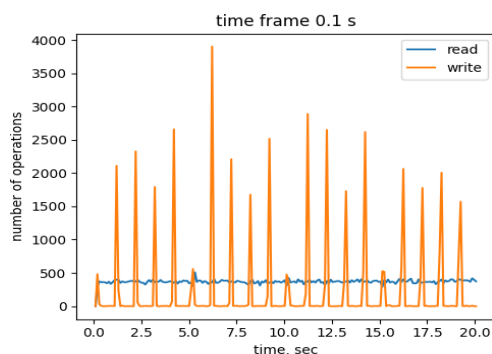


Рисунок 2 – Нагрузка, создаваемая профилем fio

В ходе работы была успешно достигнута цель — разработан fio-профиль, воспроизводящий нагрузку реальной СУБД MongoDB. Полученные результаты подчеркивают специфику работы блочного уровня при обработке данных СУБД, а также демонстрируют

отличия от типичных профилей тестирования СХД. В дальнейшем планируется расширить исследование за счет:

- анализа других систем управления базами данных;
- изучения различных сценариев нагрузки;
- разработки fio-профилей для различных классов СУБД;
- создания программного обеспечения, способного автоматически генерировать fio-профиль на основе нескольких входных параметров.

Настоящее исследование закладывает основу для дальнейшего изучения особенностей работы СУБД на низком уровне и способствует улучшению методов тестирования производительности хранилищ данных.

ЛИТЕРАТУРА

1. Hyoung Seok Hong, Yong Rae Kwon and Sung Deok Cha, "Testing of object-oriented programs based on finite state machines," Proceedings 1995 Asia Pacific Software Engineering Conference, Brisbane, QLD, Australia, 1995, pp. 234-241, doi: 10.1109/APSEC.1995.496972. keywords: {Automata;Flow graphs;Object oriented modeling;Automatic testing;Computer science;IEEE members},
2. Никифоров, И. В. Статический метод отладки тестовых сценариев, сгенерированных с использованием эвристик / И. В. Никифоров, А. В. Петров, В. П. Котляров // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика. Телекоммуникации. Управление. – 2012. – № 4(152). – С. 114-119. – EDN РСНСХЗ.
3. Уравнение с тремя неизвестными: как отлавливать баги в системах хранения данных [Режим доступа]: <https://habr.com/ru/companies/yadro/articles/800237/> (Дата обращения: 03.03.2025)
4. Инкрементальный подход к технологии создания тестов для промышленных 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 ТССАКВ.
5. Polte, Milo, Jiri Simsa, and Garth Gibson. "Enabling enterprise solid state disks performance." (2009).
6. blktrace - generate traces of the i/o traffic on block devices [Режим доступа]: <https://linux.die.net/man/8/blktrace> (Дата обращения: 03.03.2025)
7. blkparse - produce formatted output of event streams of block devices [Режим доступа]: <https://linux.die.net/man/1/blkparse> (Дата обращения: 03.03.2025)
8. Pandey, Rachit. "Performance benchmarking and comparison of cloud-based databases MongoDB (NoSQL) vs MySQL (Relational) using YCSB." *Nat. College Ireland, Dublin, Ireland, Tech. Rep* (2020).

УДК 004.453

Р.М. Суворов (2 курс бакалавриата),
С.В. Григорьев, к.ф.-м.н., доцент,
В.А. Кутуев, инженер-исследователь

ОПТИМИЗАЦИЯ ФУНКЦИИ УМНОЖЕНИЯ МАТРИЦ БИБЛИОТЕКИ SUITESPARSE:GRAPHBLAS С ИСПОЛЬЗОВАНИЕМ ВЕКТОРНОГО РАСШИРЕНИЯ RISC-V

RISC-V [1] – открытая расширяемая ISA (Instruction Set Architecture) с модульной структурой. За 14 лет её существования членами международного сообщества RISC-V стали около 350 организаций. Гибкость, модульность и простота в перспективе могут позволить архитектуре обойти более популярных конкурентов в отдельных областях, например, во встраиваемых системах.

На данный момент серьёзной проблемой является то, что большинство популярных решений (например, некоторые пользовательские библиотеки), используемых на таких архитектурах, как x86_64 и ARM, не адаптированы под RISC-V. Одной из таких библиотек

является SuiteSparse:GraphBLAS [2]. Эта библиотека является реализацией стандарта GraphBLAS [3], описывающего набор базовых операций над матрицами и векторами, активно применяющихся для анализа графов в терминах линейной алгебры [4], высокопроизводительных вычислениях [5] и нейронных сетях [6].

В ходе изучения документации к библиотеке SuiteSparse:GraphBLAS, было обнаружено, что библиотека использует векторные расширения AVX2 и AVX512 архитектуры x86_64 в функции умножения матриц. Данное наблюдение натолкнуло на мысль, что использование векторных инструкций может значительно ускорить выполнение некоторых базовых алгоритмов на архитектуре RISC-V. Цель данной работы – включение поддержки векторного расширения RVV1.0 архитектуры RISC-V в функцию умножения матриц библиотеки SuiteSparse:GraphBLAS.

Для достижения этой цели необходимо решение следующих задач:

1. Разработать механизмы определения архитектуры RISC-V и доступности RVV1.0 в коде библиотеки.
2. Реализовать функцию, использующую RVV1.0.
3. Произвести сравнение ускорения, полученного от использования AVX2 и RVV1.0.

Механизмы определения RISC-V и RVV1.0 были реализованы двумя способами: явно, через передачу флагов в систему сборки CMake и автоматически при помощи библиотеки `cpu_features`. Функция умножения матриц была написана на языке C с использованием `intrinsics` функций RVV1.0. Авторы оригинальной функции, были ограничены фиксированной длиной векторов расширения AVX, возможность использования векторных инструкций RVV1.0 на векторах произвольного размера позволило сократить число строк кода с 1041 до 83. Эксперименты с замерами ускорения, полученного от использования AVX2 и RVV1.0 проводились на наборе матриц из `Suitesparse matrix collection`. Матрицы квадратные и выбирались с целью исследования зависимости ускорения от размера и степени разреженности матриц. Результаты экспериментов представлены в таблице 1.

Таблица 1 – Результаты экспериментов

№	Rows	Nonzeros	AVX2 (ms.)	No AVX2 (ms.)	RVV (ms.)	No RVV (ms.)	AVX speedup (%)	RVV speedup (%)
1	12504	874887	4236.9	5124.9	32008.0	42763.8	17.3	25.2
2	12546	74242	508.8	564.4	2629.7	2814.1	9.8	6.6
3	7500	294986	632.6	864.0	5970.1	8593.8	26.8	30.5
4	7500	28462	90.4	92.3	484.1	555.3	2.0	12.8
5	3948	60882	87.8	117.9	1271.5	1770.8	25.6	28.2
6	4000	8784	17.1	18.2	184.0	203.5	5.9	9.6
7	3079	53843	28.5	41.0	574.2	584.8	30.5	1.8
8	3001	9000	25.2	34.7	279.3	344.9	27.5	19.0
9	2427	28511	31.3	38.5	401.6	490.0	18.8	18.0
10	2508	16727	10.4	12.1	106.5	97.7	14.3	-8.9
11	1544	9408	5.6	9.8	117.9	125.4	42.7	6.0
12	1589	2742	1.5	2.8	31.4	28.5	47.0	-10.0
13	765	24382	2.3	5.5	51.1	65.3	58.8	21.8
14	761	3256	0.6	1.6	10.5	13.1	63.0	19.5

Из таблицы видно, что в подавляющем большинстве случаев использование RVV1.0 дает внушительный прирост производительности. Однако, в некоторых сценариях, при умножении относительно небольших матриц (менее 3000 столбцов или колонок) можно наблюдать

снижение скорости выполнения. В ходе данной работы не удалось установить причины данного поведения.

Был направлен запрос на внесение изменения в официальный репозиторий библиотеки. Работа выполнена в Лаборатории технологий программирования инфраструктурных решений, СПбГУ.

ЛИТЕРАТУРА

1. Официальный сайт RISC-V [Электронный ресурс]. Режим доступа: <https://riscv.org/about/>. [Дата обращения 07.03.2025]
2. Davis Timothy A. Algorithm 1037: Suitesparse: graphblas: Parallel graph algorithms in the language of sparse linear algebra // ACM Transactions on Mathematical Software. — 2023. — Vol. 49, no. 3. — P. 1–30.
3. The GraphBLAS 3.0 Project / Raye Kimmerer, Timothy G Mattson, Scott McMillan et al. // 2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) / IEEE. — 2024. — P. 478–481.
4. LAGraph: A community effort to collect graph algorithms built on top of the GraphBLAS / Tim Mattson, Timothy A Davis, Manoj Kumaret al. // 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) / IEEE. — 2019. — P. 276–284.
5. Davis Timothy A. Graph algorithms via SuiteSparse: GraphBLAS: triangle counting and k-truss // 2018 IEEE High Performance extreme Computing Conference (HPEC) / IEEE. — 2018. — P. 1–6.
6. Davis Timothy A, Aznaveh Mohsen, Kolodziej Scott. Write quick, run fast: Sparse deep neural network in 20 minutes of development time via SuiteSparse: GraphBLAS // 2019 IEEE High Performance Extreme Computing Conference (HPEC) / IEEE. — 2019. — P. 1–6

УДК-004.4

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

СРАВНЕНИЕ ТЕХНИЧЕСКИХ ДОКУМЕНТАЦИЙ СЕТЕВОЙ ИНФРАСТРУКТУРЫ В ФОРМАТАХ PLANTUML И C4

Во время разработки сетевой инфраструктуры Docker-контейнеров команде разработчиков необходимо описывать их желаемую конфигурацию, в которой будут описаны сами контейнеры и их связи. Для этого существует несколько форматов диаграмм, которые могут использоваться командой параллельно. Одними из самых распространенных являются C4-диаграммы [1] и компонентные PlantUML-диаграммы [2]. Однако, синхронизация данных, представленных в диаграммах, может быть затруднена размером конфигурации и её сложностью. Если данные не будут согласованы, это может привести к тому, что часть команды будет использовать устаревшие данные, а другая часть актуальные. В достаточно большом проекте это приведет к крупным проблемам [3].

Для упрощения процесса актуализации можно автоматизировать процесс сравнения диаграмм. Компания YADRO поставила задачу по сравнению C4-диаграмм в формате drawio [4] и компонентных PlantUML-диаграмм. Предполагается полное сравнение связей между одноименными контейнерами в C4-диаграмме и несколькими PlantUML-диаграммами. Решение должно обнаруживать отсутствие косвенной связи между компонентами в PlantUML-диаграммах, если есть такая же прямая связь в C4-диаграмме. По результату сравнения должен быть предоставлен отчет в формате JSON со статистикой покрытия и детализированной информацией о найденных и отсутствующих связях между контейнерами.

Процесс сравнения происходит в два этапа. На первом этапе с помощью парсера из всех диаграммы извлекаются данные для загрузки внутреннее представление — граф. В случае C4-диаграммы её граф строится по одному файлу, а для PlantUML-диаграмм по данным из

нескольких файлов. На втором этапе происходит обход всех ребер C4-графа (что обозначает прямую связь), для каждого ребра ищется косвенная связь на PlantUML-графах.

Реализация программного решения использует язык Python [5]. Для работы с C4-диаграммой используется библиотека xml [6], а для реализации парсера PlantUML библиотека Lark [7]. Для создания графов и их сравнения — библиотека networkx [8].

Для обработки PlantUML диаграммы в библиотеке Lark необходимо определить грамматику PlantUML генерации парсера. Грамматика была написана с нуля по документации доступной на сайте проекта PlantUML [9]. После парсинга данных используется пользовательский класс, унаследованный от класса lark.Visitor, который вызывается на каждом узле синтаксического дерева, содержащего данные о связях между компонентами. Эти данные последовательно загружаются PlantUML-графы.

C4-диаграмма не имеет конкретных стандартов, а поэтому язык разметки диаграммы определяется реализацией. Компания YADRO использует C4-диаграммы формата drawio, который в свою очередь является xml-файлом. Для парсинга диаграммы используется стандартная библиотека xml, через которую селекторами извлекаются теги, содержащие информацию о связях внутри диаграммы. Данные также загружаются в C4-граф.

На втором этапе происходит детальное сравнение диаграмм и формирование отчета по результату. Библиотека network представляет простые и быстрые абстракции для поиска связей между элементами в графе. Последовательных обход пар точек, составляющих ребра C4-графа позволяет определить если такие связи в PlantUML-графах.

По результатам сравнения формируется детализированный отчет со статистикой покрытия. Формат отчета JSON, что позволяет достаточно легко интегрировать программное решение в пайплайн. Также для конфигурации программы доступны как аргументы командной строки, так и конфигурационный файл [10].

По итогу работы было разработано программное решение для сравнения данных в C4-диаграммах и соответствующим им PlantUML-диаграммах. Сравнение предполагает, что оба вида диаграмм согласованы, если для каждой прямой связи в C4-диаграмме существует одна косвенная в одной из PlantUML-диаграмм. Использование программы позволяет поддерживать актуальность обоих типов диаграмм, что исключает возможность использование устаревшего описания конфигурации Docker-контейнеров.

ЛИТЕРАТУРА

1. Container diagram [Электронный ресурс] // The C4 model for visualising software architecture. - URL: <https://c4model.com/diagrams/container> (дата обращения 16.03.2025)
2. Component Diagram [Электронный ресурс] // PlantUML at a Glance. - URL: <https://plantuml.com/component-diagram> (дата обращения 16.03.2025)
3. 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.
4. Create C4 models and diagrams [Электронный ресурс] // Security-first diagramming for teams. - URL: <https://www.drawio.com/blog/c4-modelling> (дата обращения 16.03.2025)
5. Лусиану Рамальо Python: к вершинам мастерства. - 2-е изд. - М.: ДМК Пресс, 2022. - 898 с.
6. XML Processing Modules [Электронный ресурс] // The Python Standard Library. - URL: <https://docs.python.org/3/library/xml.html> (дата обращения 16.03.2025)
7. Grammar Reference [Электронный ресурс] // Welcome to Lark's documentation! - URL: <https://lark-parser.readthedocs.io/en/stable/index.html> (дата обращения 16.03.2025)
8. Reference [Электронный ресурс] // Software for Complex Networks - URL: <https://networkx.org/documentation/stable/reference/index.html> (дата обращения 16.03.2025)
9. PlantUML Language Reference Guide [Электронный ресурс] // PlantUML at a Glance. - URL: <https://plantuml.com/guide> (дата обращения 16.03.2025)
10. TOML v1.0.0 [Электронный ресурс] // A config file format for humans. - URL: <https://toml.io/en/v1.0.0> (дата обращения 16.03.2025)

АВТОМАТИЗАЦИЯ СБОРА ДИАГНОСТИЧЕСКИХ ДАННЫХ С СЕРВЕРНЫХ СИСТЕМ С ПОМОЩЬЮ КОНСОЛЬНОЙ УТИЛИТЫ С ИСПОЛЬЗОВАНИЕМ ДЕКЛАРАТИВНОЙ КОНФИГУРАЦИИ

В современных IT-системах диагностика и сбор данных для анализа являются важными, но трудоемкими процессами. Существующие инструменты часто требуют ручного вмешательства, сложны в настройке или не поддерживают гибкость в выборе данных для сбора [1]. В связи с этим возникает необходимость в создании инструмента, который автоматизирует процесс сбора диагностической информации, обеспечивает гибкость конфигурации и поддерживает параллельный сбор данных для повышения производительности.

Datacollect – это консольная утилита, предназначенная для автоматизированного сбора и упаковки диагностической информации (логи, файлы, результаты выполнения команд и так далее) с систем [2], что делает его незаменимым инструментом для системных администраторов и DevOps-инженеров. Структура организации Datacollect представлена на рисунке 1.

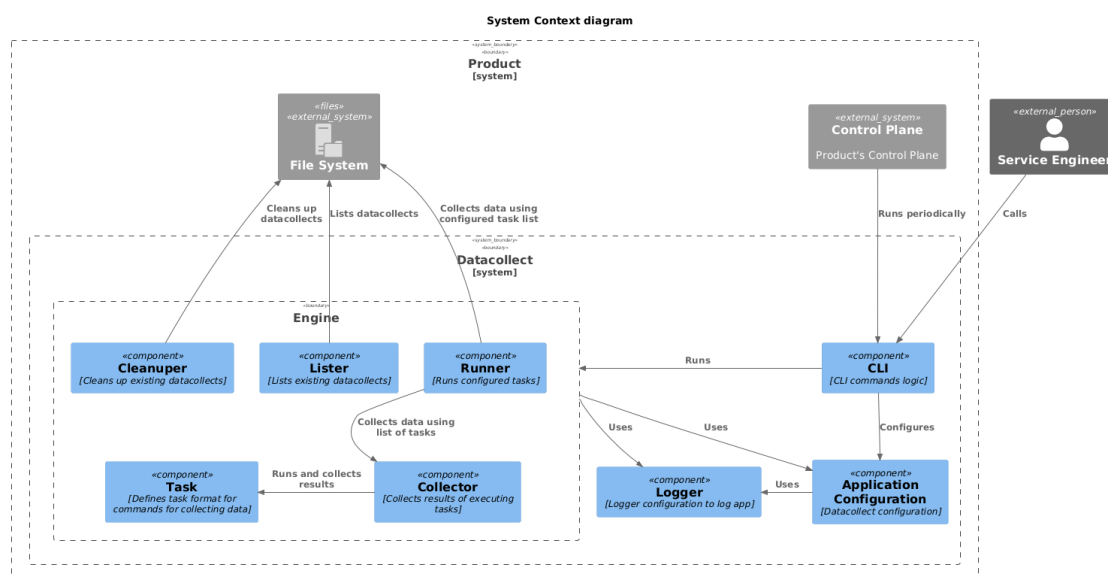


Рисунок 1 – Структура организации Datacollect.

Существует множество готовых решений для сбора и анализа диагностических данных. Например, Splunk Forwarder подходит для централизованного сбора логов, Fluentd и Logstash — для агрегации и обработки данных [3], а Prometheus Node Exporter — для сбора метрик [4]. Однако, как видно из их характеристик, эти инструменты либо требуют сложной настройки, либо ограничены в функциональности. В отличие от них, Datacollect сочетает в себе простоту конфигурации на основе YAML, поддержку параллельного сбора, гибкость профилей и автоматическую архивацию данных, что делает его более универсальным и удобным решением для диагностики систем.

Для демонстрационной версии Datacollect была разработана консольная утилита, предназначенная для работы в системах с операционной системой Linux. Утилита реализована на языке программирования Golang [5], что обеспечивает высокую производительность и простоту в использовании.

Одной из ключевых особенностей утилиты является декларативная конфигурация через YAML-файлы. Это позволяет пользователям настраивать процесс сбора данных, определяя порядок и параметры выполнения задач в простом и понятном формате. Также Datacollect

поддерживает сложные логические конструкции через использование внешних скриптов, что делает инструмент одновременно простым и мощным. Для повышения эффективности работы утилита поддерживает профили сборки данных, что позволяет независимо собирать различные части информации в зависимости от текущих задач. Также одним из наиболее значимых преимуществ Datacollect является конкурентный сбор, что позволяет утилите значительно ускорить процесс сбора данных на системах с большим объемом информации.

Для удобства хранения и передачи данных Datacollect автоматически архивирует собранные данные. Это упрощает процесс передачи данных для дальнейшего анализа. Утилита также предоставляет удобные инструменты для управления ранее собранными наборами данных: удаление старых и просмотр списка ранее собранных архивов. Также утилита поддерживает вывод результатов в формате JSON, что делает ее идеальным инструментом для интеграции в автоматизированные процессы. JSON-формат удобен для автоматического анализа результатов работы, что особенно важно в CI/CD-конвейерах и других автоматизированных системах.

Таким образом, Datacollect сочетает в себе простоту настройки, высокую производительность и гибкость, что делает его мощным инструментом для диагностики и анализа систем. Его уникальные особенности выделяют его среди существующих решений, делая его незаменимым помощником в работе с диагностическими данными.

ЛИТЕРАТУРА

1. Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining (2014). [Электронный ресурс] Режим доступа: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781118834732>
2. Brendan Gregg; Systems Performance (Addison-Wesley Professional Computing Series): Enterprise and the Cloud, 2nd Edition (2020). [Электронный ресурс] Режим доступа: <https://www.brendangregg.com/systems-performance-2nd-edition-book.html>
3. Fluentd. [Электронный ресурс] Режим доступа: <https://www.fluentd.org/architecture>
4. Monitoring Linux host metrics with the Node Exporter. [Электронный ресурс] Режим доступа: <https://prometheus.io/docs/guides/node-exporter/>
5. Golang programming language. [Электронный ресурс] Режим доступа: <https://go.dev/>

УДК 004.9

М.В. Тянутов (2 курс магистратуры),
О.А. Юсупова, ст. преподаватель

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ И УПРАВЛЕНИЯ ШАБЛОНОВ ZABBIX

В современных IT-инфраструктурах непрерывный мониторинг систем и сетевых ресурсов приобретает критическое значение [1]. Даже кратковременные сбои оборудования, такие как отказ серверов или нарушения в работе систем хранения данных (СХД), могут привести к недоступности ключевых сервисов, финансовым потерям и утрате доверия пользователей [2, 3]. В условиях постоянно растущего числа сервисов и их взаимозависимостей критически важно обеспечить оперативное выявление неисправностей и своевременную реакцию на них [4].

Для минимизации рисков, связанных с простоями, активно применяются системы мониторинга. Среди них Zabbix выделяется благодаря своей универсальности, поддержке множества протоколов (SNMP, IPMI, JMX и др.) и возможности сохранения настроек в виде шаблонов [5]. Эти шаблоны содержат предопределенные метрики, триггеры и правила, что позволяет стандартизировать процесс настройки мониторинга для разнообразных элементов инфраструктуры, включая BMC (Baseboard Management Controller) и СХД. Стандартизация обеспечивает согласованность метрик и единообразие триггеров, что особенно важно для повышения надёжности специализированных систем.

Таким образом, создание и поддержка шаблонов Zabbix может стать ключевой частью сервиса, предоставляемого производителями IT-систем. Это упрощает интеграцию мониторинга для конечных пользователей и снижает вероятность ошибок при ручной настройке. Однако ручные операции, такие как миграция шаблонов между версиями (например, с Zabbix 6.0 на 7.0), корректировка параметров и тестирование, требуют значительных временных затрат. Вследствие этого растёт потребность в инструментах, способных автоматизировать подобные процессы, обеспечивая высокую надёжность и повторяемость процедур.

Для решения указанной проблемы предлагается инструмент, основанный на интеграции с Zabbix API и поддержке одновременной работы с несколькими версиями Zabbix, что позволяет централизованно управлять шаблонами и осуществлять их миграцию между различными конфигурациями [6]. Гибкая система конфигурационных файлов позволяет адаптировать логику работы приложения к специфике любого окружения, задавая конкретные версии Zabbix, перечень необходимых шаблонов и хостов, а также набор правил валидации. Механизм параллельного доступа к нескольким серверам Zabbix обеспечивает возможность одновременного мониторинга и тестирования множества узлов, что существенно сокращает общее время выполнения операций и повышает точность диагностики. Применяемое частично автоматизированное тестирование, основанное на сборе и анализе данных о состоянии элементов, позволяет своевременно выявлять потенциальные несоответствия, фиксировать их в отчётах и формировать рекомендации по оптимизации, которые затем учитываются при доработке шаблонов или внесении изменений в конфигурацию.

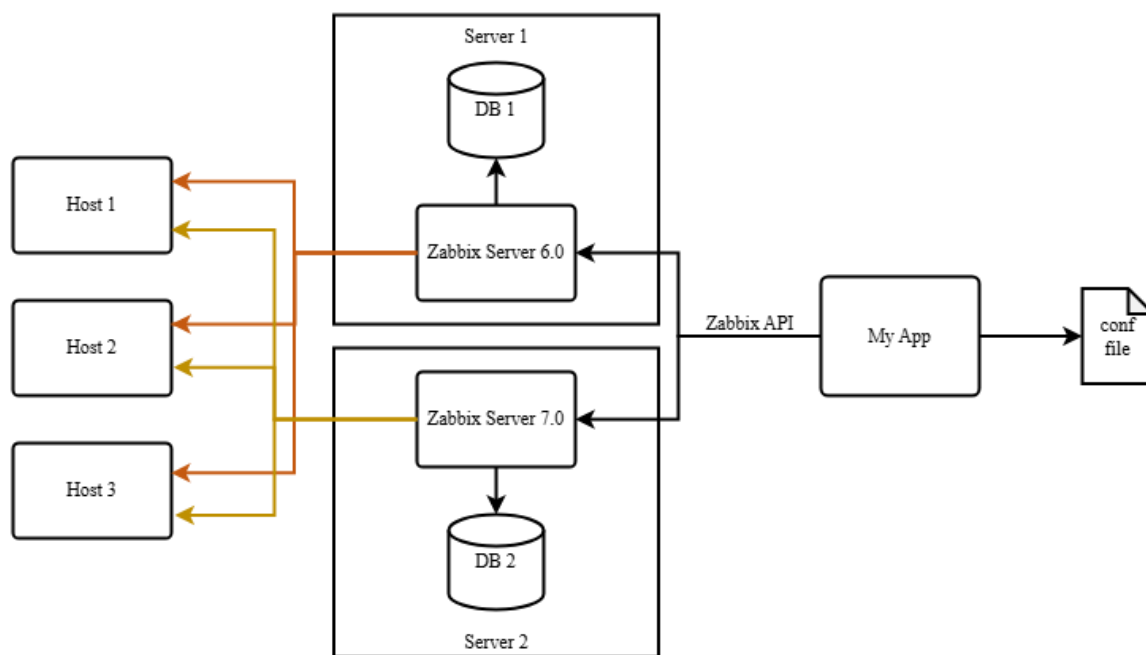


Рисунок 1 – Архитектура предлагаемого решения

Рисунок 1 иллюстрирует архитектуру предлагаемого решения, включающую интеграцию с API для управления шаблонами и хостами, поддержку различных версий Zabbix и возможность параллельного мониторинга нескольких узлов. Конфигурация инструмента задаётся посредством отдельных файлов, что обеспечивает гибкость в определении параметров тестирования и правил валидации, а также упрощает расширение функционала при добавлении новых версий Zabbix или типов проверок [7]. Генерация отчётов, содержащих результаты автоматизированного тестирования, выявленные ошибки и рекомендации по оптимизации, играет важную роль в повышении прозрачности и эффективности процессов. Такой подход способствует упрощению миграции шаблонов между версиями, снижению риска ошибок при обновлениях и предоставляет единый механизм централизованного

управления шаблонами и хостами, что позволяет значительно ускорить процесс поддержки шаблонов мониторинга, а также уменьшить количество потенциальных ошибок.

ЛИТЕРАТУРА

1. Zhang W. H. et al. Value of information analysis in civil and infrastructure engineering: a review //Journal of Infrastructure Preservation and Resilience. – 2021. – Т. 2. – №. 1. – С. 16.
2. Novotný V. et al. Critical infrastructure monitoring system //2021 IEEE 17th International Colloquium on Signal Processing & Its Applications (CSPA). – IEEE, 2021. – С. 165-170.
3. Афонина, А. А. Концепция мониторинга деградации, обновления и модернизации отдельных конфигурационных единиц вычислительного оборудования / А. А. Афонина, А. Е. Кузькин, 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. 155-157. – EDN VVIAEQ.
4. Alcaraz C., Zeadally S. Critical infrastructure protection: Requirements and challenges for the 21st century //International journal of critical infrastructure protection. – 2015. – Т. 8. – С. 53-66.
5. Dalle Vacche A. Mastering Zabbix. – Packt Publishing Ltd, 2015.
6. Вакке А. Zabbix. Практическое руководство. – Litres, 2022.
7. Денисенко, Б. А. Реализация декларативного подхода к управлению сетевой конфигурацией / Б. А. Денисенко, М. В. Тянутов, И. В. Никифоров // Неделя науки ИКНХ : Материалы докладов научно-практической конференции, Санкт-Петербург, 15–17 апреля 2024 года. – Санкт-Петербург: Санкт-Петербургский политехнический университет Петра Великого, 2024. – С. 17-19. – EDN ONQRRN.

УДК 004.4`2

Д. А. Фоломкин (4 курс бакалавриата),
И. В. Никифоров, к.т.н., доцент,
С. М. Устинов, д.ф-м.н, профессор

МЕТОДИКА РАЗВЕРТЫВАНИЯ ОКРУЖЕНИЯ ДЛЯ РАЗРАБОТКИ СИСТЕМ ХРАНЕНИЯ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА ANSIBLE

С развитием цифровых технологий неминуемо растет и то количество данных, которое необходимо хранить. Например, за 2023 год суммарный объем данных увеличился на 120 зеттабайт, а к концу 2025 года это число увеличится на 50% - до 181 зеттабайта [1]. По этой причине для эффективного и безопасного хранения таких массивов данных разрабатываются специализированные устройства - системы хранения данных (сокращенно - СХД). Однако процесс разработки, тестирования и внедрения СХД очень трудоёмок, так как включает в себя следующие компоненты:

- создание специальных версий ядра операционной системы на базе Linux;
- создание собственных ОС, а также ключевых компонентов и утилит СХД;
- тестирование СХД (компонентное, регрессионное, системное и др.).

Специфика задач не позволяет выполнять их на стандартных рабочих станциях из-за высокой цены физического оборудования, а также рисков при работе: например, с ядром ОС. Для решения этой проблемы наиболее эффективный выход – использование средств виртуализации [2]. В качестве инструмента виртуализации особенно выделяется QEMU – универсальный эмулятор и виртуализатор рабочих машин и пользовательских пространств с открытым исходным кодом [3]. Одно из ключевых преимуществ данной утилиты – совместимость с ведущими архитектурами процессоров: arm, ppc64le, RISC-V, x86_64 и другими. Однако мало создать виртуальные машины, так как их необходимо еще и настроить. Причем настройки касаются не только установки базовых утилит, но и конфигурации сети устройства, утилит контейнеризации (Docker), репозиторийных зеркал. К тому же, непосредственно процесс создания виртуальных машин необходимо автоматизировать.

С этой задачей поможет справиться Ansible — инструмент автоматизации с открытым исходным кодом, который автоматизирует подготовку, управление конфигурацией, развертывание приложений и многие другие аспекты, присущие работе с информационными системами [4]. К тому же, для него можно написать свои модули, отвечающие за конфигурацию отдельных систем, специфичных для вашей области работы [5].

Таким образом, целью работы является снижение трудоемкости разработки и тестирования СХД за счет автоматизированной системы по развертыванию персональных виртуальных машин с использованием инструмента управления конфигурациями Ansible.

Для достижения поставленной цели необходимо решение задач, перечисленных ниже.

1. Формализовать бизнес-процессы для построения системы по развертыванию персональных виртуальных машин.

2. Оценить существующие подходы и инструменты развертывания виртуального окружения.

3. Предложить методику автоматизированного развертывания персональных виртуальных машин.

4. Реализовать предложенную методику с помощью инструментов Ansible и интегрировать в инструмент непрерывной интеграции работу с Ansible-репозиторием посредством создания специальной задачи на выполнение развертывания окружения.

5. Оценить эффективность методики в рамках экспериментального исследования, которое продемонстрирует снижение трудоемкости и повышение скорости развертывания окружения разработки.

Отдельно стоит выделить оценку подходов и инструментов развертывания виртуального окружения в таблице 1. Как видно из сравнения, дополнительные опции, требуемые для разработчиков и тестировщиков СХД, отсутствуют на уже имеющихся решениях, таких как Yandex Cloud [6] и VK Cloud [7]. Помимо этого, доверие к данным облачным инструментам не может быть 100%, так как это сторонние утилиты.

Таблица 1 – Сравнение существующих решений по развертыванию виртуальных машин с требуемым.

Название решения/ Критерий	Действия с ВМ	Выбор ресурсов	Выбор ОС	Чекбоксы установки ПО	Сохранение данных домашней партии при пересоздании ВМ	DNS имена	Утилиты настройки
Yandex Cloud	+	+	+	-	-	+	YC CLI, Terraform
VK Cloud	+	+	+	-	-	-	OpenStack CLI
Собственная методика	+	+	+	+	+	+	Ansible, Jenkins

Из таблицы сравнительного анализа следует, что имеет место создание собственной методики для ликвидации этих недостатков и потенциальных проблем с утечкой конфиденциальных данных по разработке СХД.

В результате, с помощью предложенной собственной методики удалось увеличить скорость и снизить трудоемкость развертывания окружения для разработки СХД, реализовав все те специфичные элементы конфигурирования ВМ, которые отсутствуют у обычных провайдеров виртуального окружения. Среднее время конфигурации базового окружения не превышает 5 минут, а расширенного – не более 10 минут. Для сравнения, ручное развертывание базовой конфигурации занимает около 30 минут, а расширенной – 45 минут. О снижении трудоемкости говорит тот факт, что запуск системы по развертыванию окружения

требует выбора около 15 параметров, в то время как ручное развертывание потребует написание нескольких десятков команд, а также двух-трех конфигурационных файлов.

Работа выполнена при поддержке компании YADRO.

ЛИТЕРАТУРА

1. Amount of Data Created Daily [Электронный ресурс]. Режим доступа: <https://explodingtopics.com/blog/data-generated-per-day> (дата обращения: 24.02.2025)
2. Беляев, Р. Н. Обзор современных серверных платформ виртуализации / Р. Н. Беляев, П. В. Волков, М. Р. Хаматдинова // Информационные технологии в современном мире - 2019: материалы XV Всероссийской (с международным участием) студенческой конференции, Екатеринбург, 14 мая 2019 года / под науч. ред. Н. В. Хмельковой. – Екатеринбург: Гуманитарный университет, 2019. – С. 35-40. – EDN RSMPPMR.
3. Куличенко, А. Д. QEMU: её характеристика и сравнение с известными аналогами / А. Д. Куличенко, Я. С. Пискун, И. Д. Розанцев // Аллея науки. – 2022. – Т. 1, № 12(75). – С. 712-719. – EDN VFBZSA.
4. Пономарев, А. А. Методика разработки Ansible-модулей для спецификации Swordfish / А. А. Пономарев, И. В. Никифоров, О. В. Александрова // 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. 290-292. – EDN LLSNQ.
5. Разработка Ansible-модулей по спецификации SWORDFISH для управления конфигурацией СХД / С. Ф. Солодовников, Г. А. Худяков, И. В. Никифоров, О. В. Александрова // 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. 294-297. – EDN LHUAKG.
6. Логвинов, Д. В. Возможности отечественной облачной инфраструктуры на примере сервиса Yandex Cloud / Д. В. Логвинов, С. С. Савкин // Актуальные проблемы науки и образования в условиях современных вызовов: Сборник материалов XIII Международной научно-практической конференции, Москва, 15 августа 2022 года. – Москва: Печатный цех, 2022. – С. 23-25. – EDN DMKKOK.
7. VK Cloud [Электронный ресурс]. Режим доступа: <https://cloud.vk.com/> (дата обращения: 09.03.2025)

УДК 004.453

Г.А. Худяков (4 курс бакалавриата),
А.С. Макейкин (1 курс магистратуры),
М.С. Соломонов, ассистент

АВТОМАТИЗАЦИЯ КОНФИГУРИРОВАНИЯ И УПРАВЛЕНИЯ СИСТЕМАМИ ХРАНЕНИЯ ДАННЫХ ЗА СЧЁТ ANSIBLE-МОДУЛЕЙ ДЛЯ СПЕЦИФИКАЦИИ SWORDFISH.

Современные центры обработки данных (ЦОД) и IT-инфраструктуры сталкиваются с растущими требованиями к масштабируемости, гибкости и эффективности управления системами хранения данных (СХД). Ручное конфигурирование и администрирование таких систем становятся всё более сложными и трудоёмкими, особенно в условиях увеличения объёмов данных и разнообразия используемых СХД.

Для решения задач автоматизации конфигурирования и управления системами хранения данных (СХД), часто применяется инструмент Ansible [1]. Его ключевые преимущества заключаются в простоте использования, декларативном подходе, идемпотентности и безагентности. Для того, чтобы автоматизировать процесс конфигурации, необходимо определить набор Ansible-модулей [2]. Ansible-модули представляют собой небольшие

программы, которые выполняют конкретные задачи на управляемых узлах, например, установка и обновление пакетов.

Большинство популярных на данный момент систем хранения данных предоставляют REST API для взаимодействия с ними [3]. Однако, каждый производитель предоставляет свой REST API, который может отличаться от API другого производителя. Это заставляет потребителей, использующих СХД различных моделей и от разных производителей, одновременно работать с несколькими протоколами взаимодействия, что является трудоёмким как с точки зрения разработки, так и поддержки.

Для решения описанной ранее проблемы некоммерческая организация SNIA (Storage Networking Industry Association), разработала унифицированную спецификацию Swordfish [4,5]. Спецификация Swordfish устанавливает единые требования к REST API, которые предоставляют СХД. При этом производитель сам принимает решение о поддержке Swordfish своим СХД.

На данный момент нет Ansible-модулей, позволяющих взаимодействовать с СХД через Swordfish API, и этот недостаток влечет за собой необходимость каждому производителю СХД самостоятельно реализовывать необходимые команды, что усложняет процесс конфигурации.

Таким образом, целью данной работы является снижение трудоёмкости процесса конфигурирования СХД за счёт разработки Ansible-модулей, которые используются в скриптах Ansible, для управления СХД, удовлетворяющих спецификации Swordfish.

Для реализации Ansible-модулей было необходимо реализовать две сущности: сам модуль и ресурс из спецификации Swordfish, с которым этот модуль будет работать. На основе анализа существующих решений и подходов [6] была разработана архитектура, позволяющая быстро и гибко реализовывать новые Ansible-модули, удовлетворяющие спецификации Swordfish (рисунок 1).

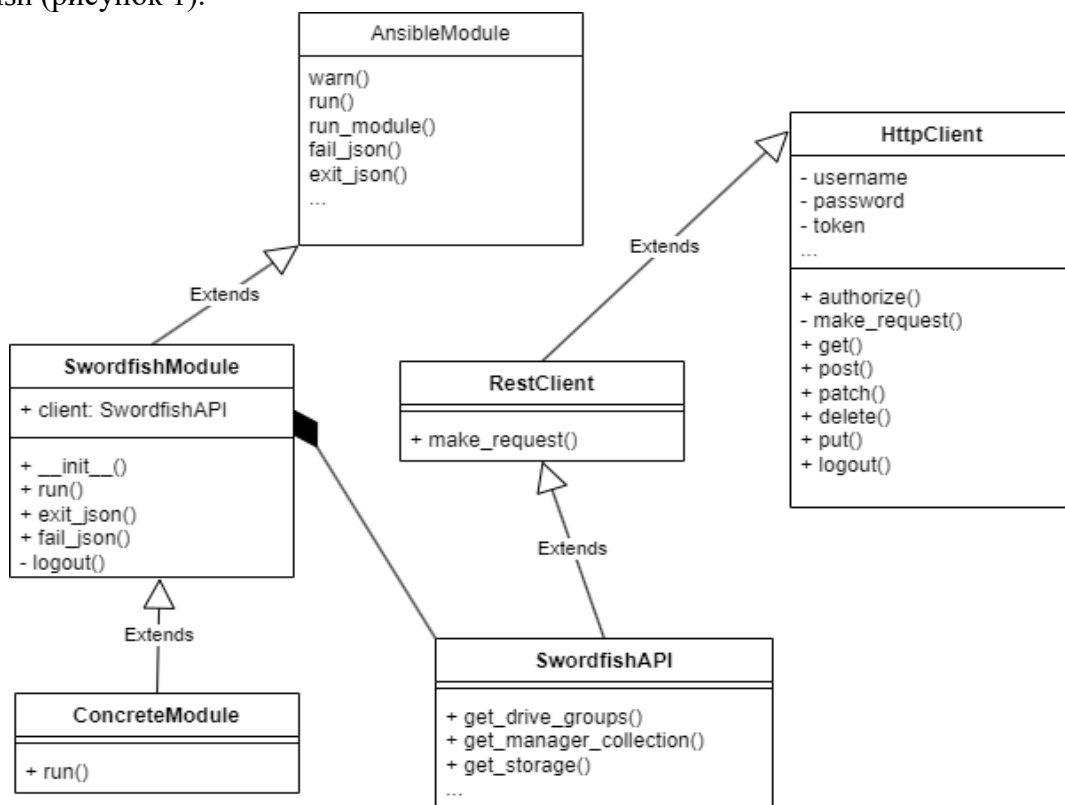


Рисунок 1 – Предложенная архитектура Ansible-модуля

AnsibleModule – базовый класс, отвечающий за взаимодействие с инструментом Ansible. Класс SwordfishModule наследуется от AnsibleModule инкапсулирует логику по подключению к эмулятору Swordfish, которую должен переиспользовать любой создаваемый модуль. Этот класс предоставляет абстрактный метод `run`, который должен быть определён наследником и

отвечает за его своевременный вызов. Основным атрибутом класса SwordfishModule это SwordfishAPI клиент. Клиент предоставляет API для работы со Swordfish эмулятором. Так как эмулятор работает по HTTP и соответствует RESTful архитектуре, для обеспечения соответствующей гибкости SwordfishAPI клиент наследуется от класса RestClient, который отвечает за реализацию RESTful архитектуры, а RestClient в свою очередь наследуется от класса HttpClient, который предназначен для создания и обработки запросов с помощью протокола HTTP.

Также было проведено исследование, в ходе которого участникам предлагалось выполнить следующий сценарий с СХД:

- 1) Создать StoragePool.
- 2) Получить информацию о созданном на 1 шаге StoragePool.
- 3) Создать Volume в созданном на 1 шаге StoragePool.
- 4) Изменить атрибут «Description» в созданном на 3 шаге Volume.
- 5) Получить информацию о созданном на 3 шаге Volume.

В ходе исследования было выявлено, что в среднем применение Ansible-модулей уменьшает время выполнения предложенного сценария на ~50%: 15 минут с применением Postman против 7 минут с применением Ansible-модулей.

Разработанные Ansible-модули для спецификации Swordfish представлены в открытом репозитории [7]. Реализованные модули демонстрируют снижение трудоёмкости процесса конфигурирования систем хранения данных на ~50%.

ЛИТЕРАТУРА

1. Ansible модуль. Электронный ресурс [Режим доступа]: https://docs.ansible.com/ansible/latest/module_plugin_guide/modules_intro.html (Дата обращения: 30.11.2023).
2. Пономарев, А. А. методика разработки Ansible-модулей для спецификации Swordfish / А. А. Пономарев, И. В. Никифоров, О. В. Александрова // 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. 290-292. – EDN LLSNQ.
3. Сысоев, И. М. Создание подхода автоматизации обновления конфигурационных файлов программного обеспечения / И. М. Сысоев, И. В. Никифоров, Е. В. Каплан // Современные технологии в теории и практике программирования : сборник материалов конференции, Санкт-Петербург, 19 апреля 2019 года. – Санкт-Петербург: Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский политехнический университет Петра Великого", 2019. – С. 126-127. – EDN YARVRM.
4. Спецификация Swordfish. Электронный ресурс [Режим доступа]: https://www.snia.org/sites/default/files/technical-work/swordfish/release/v1.2.5a/html/Specification/Swordfish_v1.2.5a_Specification.html (Дата обращения: 23.06.2023).
5. Документация Swordfish. Электронный ресурс [Режим доступа]: <https://www.snia.org/forums/smi/swordfish/> (Дата обращения: 23.06.2023).
6. Подходы к разработке Ansible модулей. Электронный ресурс [Режим доступа]: https://docs.ansible.com/ansible/latest/dev_guide/developing_modules_checklist.html#developing-modules-checklist (Дата обращения: 30.11.2023).
7. Репозиторий с Ansible-модулями для спецификации Swordfish. Электронный ресурс [Режим доступа]: https://gitlab.com/IgorNikiforov/swordfish_ansible_plugin

ОБРАБОТКА ИЗОБРАЖЕНИЙ НА ПЛАТФОРМЕ RISC-V

Современные системы компьютерного зрения активно используют нейронные сети, однако их работа на платформе RISC-V сопряжена с рядом технических сложностей. В отличие от традиционных архитектур x86 и ARM, платформа RISC-V [5] имеет ограниченную поддержку популярных фреймворков машинного обучения, таких как TensorFlow и Pytorch, что затрудняет использование предобученных моделей. В связи с этим для работы с нейросетями приходится использовать ONNX, что накладывает дополнительные ограничения.

Для повышения производительности и ускорения вычислений при работе с ONNX на платформе RISC-V, предлагается интеграция Execution Provider (EP). Execution Provider – это механизм, который позволяет разработчикам специализированных решений для аппаратного ускорения интегрировать ONNX Runtime с их аппаратной платформой. Для создания нового Execution Provider необходимо сначала выбрать уникальное имя для EP и настроить интерфейс ONNX Runtime, что дает возможность выполнять модели ONNX с использованием оптимизированных для конкретной платформы вычислений.

Разрабатываемое приложение предназначено для обработки видеопотока [6]. Оно функционирует следующим образом:

- Видеопоток поступает с камеры и передается на устройство Lichee Pi 4a, работающее на RISC-V.
- Каждый кадр передается в нейронную сеть, которая выполняет детекцию объектов и классификацию.
- Обработанный кадр с наложенной разметкой передается на экран монитора.

Общий процесс обработки изображений представлен на рисунке 1.

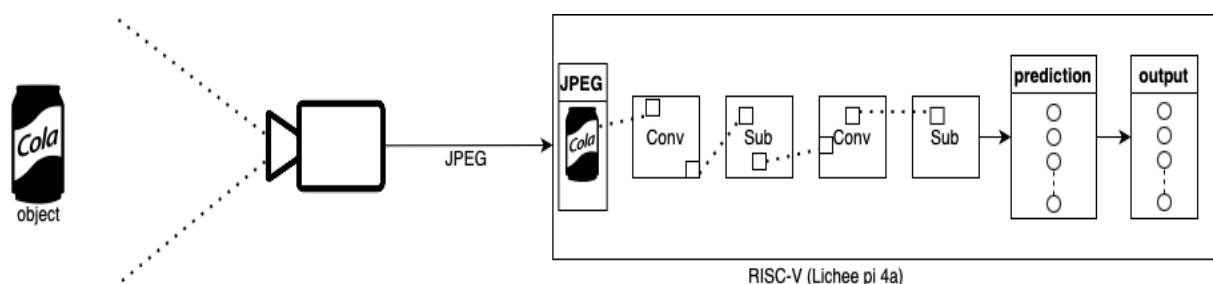


Рисунок 1 – Структура модели обработки изображений CNN

В данной работе используется нейронная сеть YOLOv8s (You Only Look Once, версия 8 small) – одна из современных архитектур для детекции объектов. YOLOv8s [3] отличается высокой скоростью работы и компактностью модели по сравнению с более тяжелыми вариантами. Она основана на сверточных слоях (Conv), механизме Feature Pyramid Networks (FPN) и Cross Stage Partial Networks (CSP), что позволяет эффективно извлекать признаки объектов на изображении [2].

Одной из ключевых проблем является низкая скорость обработки изображений. В текущей реализации приложения, выполняющего обработку изображений с помощью нейронной сети на RISC-V [1], достигает всего 0.5-1 кадр в секунду (FPS), что недостаточно для большинства практических задач компьютерного зрения. Оптимизация вычислений является необходимым шагом для увеличения производительности, с целью достижения 15-20 FPS.

Для решения данной проблемы рассматриваются следующие подходы к оптимизации:

- Векторизация вычислений с использованием RVV-инструкций, доступных на архитектуре RISC-V.
- Распараллеливание операций на уровне многопоточных вычислений.
- Ручная реализация векторизации через использование интринсиков, что позволяет более эффективно использовать ресурсы процессора.

Примером ускорения с помощью RVV является матричное умножение. В обычных вычислениях умножение матриц выполняется с использованием стандартных циклов, что может быть медленным для больших матриц. Векторизация операций с использованием RVV (RISC-V Vector Extension) позволяет одновременно обрабатывать несколько элементов матриц в одном цикле, что значительно ускоряет процесс. Это достигается за счет параллельных вычислений, ускоряя выполнение операции умножения на несколько порядков, что дает повышение производительности.

Целью данной работы является исследование, указанных методов оптимизации обработки изображений на платформе RISC-V [4], а также практическая реализация для увеличения пропускной способности системы.

Ожидается, что предложенные решения позволят не только повысить FPS, но и создать универсальный подход к оптимизации нейросетевых вычислений на процессорах RISC-V, где стандартные инструменты глубокого обучения пока недоступны или работают с ограниченной эффективностью.

ЛИТЕРАТУРА

1. Губенко, Н. О. Федеративное обучение и частное масштабирование / Н. О. Губенко, Т. В. Коликова // 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. 19-21. – EDN UBAVKJ.
2. Губенко Н. О. Разработка классификатора для оптического распознавания музыкальной нотации / Н. О. Губенко, С. А. Молодяков, Т. В. Коликова // Современная наука: актуальные проблемы теории и практики. Серия: естественные и технические науки. – 2023. – №3. – С. 51-54. – DOI 10.37882/2223-2966.2023.03.09. – EDN WZYUWU.
3. YOLOv8-RMDA: Lightweight YOLOv8 Network for Early Detection of Small Target Diseases in Tea / R. Ye, G. Shao, Yu. He [et al.] // Sensors. – 2024. – Vol. 24, No. 9, - P.2896. – DOI 10.3390/s24092896. – EDN CLBKTH.
4. Авраменко, В. С. Оптимизация нейронных сетей для вычислительных средств ограниченной производительности / В. С. Авраменко, Е. С. Чичков // Региональная информатика (РИ-2024): Материалы XIX Санкт-Петербургской международной конференции, Санкт-Петербург, 23-25 октября 2024 года. – Санкт-Петербург: Санкт-Петербургское Общество информатики, вычислительной техники, систем связи и управления, 2024. – С. 44-46. – EDN XRKIRU.
5. Ермаков, Н. Архитектура RISC-V / Н. Ермаков // Шестнадцатая конференция разработчиков свободных программ : тезисы докладов, Калуга, 27-29 сентября 2019 года / отв. Ред. Черный В. Л. – Калуга: ООО "МАКС Пресс", 2019. – С. 17-21. – EDN JUIJG.
6. Неверовский, Н. А. Метод построения обогащенного метаданными изображения с использованием rest-микросервисов / Н. А. Неверовский, Н. И. Никифоров // Современные технологии в теории и практике программирования : сборник материалов научно-практической конференции, Санкт-Петербург: федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого», 2021. – С. 50-52. – EDN KARYHG.

ВЕКТОРИЗАЦИЯ АЛГОРИТМОВ ДЛЯ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛЕНИЙ НА ПРОЦЕССОРНОЙ АРХИТЕКТУРЕ RISC-V

Во многих сферах деятельности используются высокопроизводительные вычисления. Для достижения высокого качества результата выполнения таких вычислений требуется обрабатывать большой объём данных. Целью нашей работы является увеличение эффективности работы алгоритмов вычислительной математики с помощью векторизации на молодой процессорной архитектуре RISC-V. Для реализации обозначенной цели требуется выполнить следующие задачи:

- рассмотреть теорию работы RISC-V Vector extension (RVV) - векторного расширения для архитектуры RISC-V;
- представить сравнительный анализ существующих библиотек, которые используются в высокопроизводительных вычислениях;
- предложить подход повышения производительности вычислений на уровне кода для обработки больших массивов данных на архитектуре RISC-V, используя RVV;
- реализовать предложенный подход в программной библиотеке на языке C;
- провести экспериментальное исследование, демонстрирующее повышение производительности вычислений.

Для оптимизации вычислений с большим объёмом данных используется технология SIMD (Single Instruction Multiple Data), которая реализуется в RISC-V за счёт векторного расширения (RVV). Вследствие растущей популярности этой процессорной архитектуры инженеры со всего мира начали переписывать код библиотек, которые используются в высокопроизводительных вычислениях, например OpenBLAS (алгоритмы линейной алгебры). Однако, на момент выполнения данной работы отсутствует векторизация для библиотек, содержащих численные методы, которые также часто используются в такого рода вычислениях. Исходя из этого и выдвигается задача оптимизации алгоритмов численных методов с помощью векторизации данных.

Подход заключается в применении векторных инструкций вместо стандартных циклов для обработки массивов. При этом необходимо сохранить всю функциональность алгоритмов. Для реализации этого подхода было решено использовать интринсики - функции языка C, которые компилируются в инструкции RVV. В качестве стенда имеется одноплатный компьютер LicheePi 4A с поддержкой RVV 0.7.1. Для компиляции кода под данную версию используется тулчейн от Alibaba T-Head, который содержит gcc 8.4. Наглядная демонстрация подхода продемонстрирована на рисунке 1.

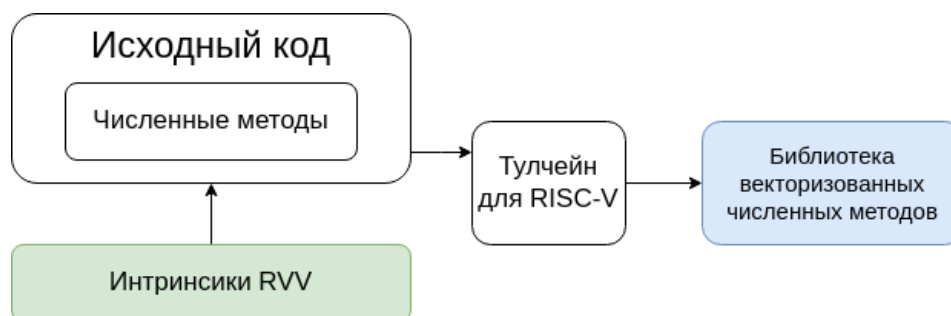


Рисунок 1 – Схема подхода оптимизации алгоритмов

Написанные алгоритмы численных методов собираются в библиотеку. Для каждого алгоритма существует 2 версии - с векторизацией и без. Составляются тесты для каждого

метода, настраивается конфигурация, пишется документация к установке и использованию библиотеки.

После успешного написания и тестирования проводится экспериментальное исследование, результатом которого является сравнительный анализ производительности алгоритмов численных методов с векторизацией и без.

ЛИТЕРАТУРА

1. RISC-V Vector Extension Intrinsic API Reference Manual. (год обращения: 2025) [Электронный ресурс]. URL: <https://occ-oss-prod.oss-cn-hangzhou.aliyuncs.com/resource/1836682/1638774209491/Xuantie+900+Series+RVV-0.7.1+Intrinsic+Manual.pdf>
2. Спецификация RISC-V "V" Vector Extension (год обращения: 2025) [Электронный ресурс]. URL: <https://storage.googleapis.com/shodan-public-artifacts/RVV-Specification-Docs/riscv-v-spec-1.0-rc2.pdf>
3. Volokitin V.D., Vasilyev E.P., Kozinov E.A., Kustikova V.D., Liniov A.V., Rodimkov Yu.A., Sysoyev A.V., Meyerov I.B. Improved Vectorization of OpenCV Algorithms for RISC-V CPU // Lobachevskii Journal of Mathematics. № 1. V. 45. 2024. pp. 130-142.
4. Zaytseva K.A., Puzikova V.V., Sokolov A.D. On Problems in OpenBLAS Library Usage in Productized Code on RISC-V. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 91-106.
5. Anna Pirova, Anastasia Vodeneeva, Konstantin Kovalev, Alexander Ustinov, Evgeny Kozinov, Alexey Liniov, Valentin Volokitin, Iosif Meyerov Performance optimization of BLAS algorithms with band matrices for RISC-V processors // Department of HPC and System Programming, Lobachevsky State University of Nizhny Novgorod. 2025. p. 13.
6. Губенко Н.О., Коликова Т.В., Федеративное обучение и частное масштабирование // Современные технологии в теории и практике программирования. Сборник материалов научно-практической конференции. Санкт-Петербург, 2021. с. 19-21.

УДК 004.45

С.С. Шерemet (4 курс бакалавриата),
Г.С. Мадьяров (2 курс бакалавриата),
И.В. Никифоров, к.т.н., доцент,
Н.О. Степина, ассистент

СИСТЕМА РЕЗЕРВНОГО ВОССТАНОВЛЕНИЯ ПРОГРАММНОЙ ИНФРАСТРУКТУРЫ НА ЭКСПЕРИМЕНТАЛЬНОМ СТЕНДЕ С RISC-V

Микросервисные архитектуры становятся все более популярными в современных приложениях. Такие приложения могут демонстрировать различную эффективность в зависимости от аппаратной платформы. В связи с этим становится актуальной задача портирования ПО на различные платформы, в частности на перспективную архитектуру – RISC-V. Данная архитектура отличается от ARM и x86 тем, что является открытой для использования. Архитектуры RISC имеют ряд преимуществ по сравнению с CISC [1]. Вот некоторые из них: производительность, энергоэффективность, простота программирования, меньший объем кода.

Программный стек RISC-V включает в себя все необходимые инструменты для разработки приложений. Операционная система (Linux), компилятор (gcc, Clang), основные библиотеки и программы [2].

Перед нами стоит задача по разработке системы резервного восстановления для автоматизации развертывания программной инфраструктуры. Для проведения данной работы предполагается произвести подготовку аппаратного и программного обеспечения, которое включает:

– собрать экспериментальный стенд, основанный на платах Lichee Pi 4A;

- выбрать и установить ОС для стенда;
- установить окружение для работы над поставленной задачей;
- установить необходимые компоненты для анализа портируемости;
- установить и настроить планировщик и систему для резервного копирования.
- я сборки тестового стенда необходимо воспользоваться комплектом сборки для 2 ПК,

в каждом из которых содержится:

- 1) lichee pi 4A;
- 2) корпус с Wi-Fi антенной;
- 3) кулер с термоинтерфейсом;
- 4) комплектный дисплей;
- 5) JTAG отладчик;
- 6) стойка для дисплея;

Был проведен сравнительный анализ операционных систем. В результате выбор остановился на Linux Debian, потому что его рекомендует производитель Lichee Pi 4A для своей платы, а также он открытый и поддерживает контейнеризацию. После установки ОС необходимо настроить и протестировать окружение, отвечающее нашей цели [3].

- компилятор GCC - для установки ПО[4];
- инструмент Docker - для контейнеризации микросервисов;
- утилита Docker Swarm – для оркестрации контейнеров;
- фреймворк K3s - для оркестрации контейнеров;

Пакеты были установлены и протестированы запуском простейших программ.

Система резервного восстановления представляет собой написанный бинарный файл, который запускается по stop-команде [5] в определенное, установленное пользователем время. Снепшоты системы делаются с помощью утилиты Timeshift [6]. Выполняемый бинарный файл вызывает команду по созданию снепшота и хранит его в выбранном месте, в данном случае SSD-диске. Количество снепшотов устанавливается пользователем по желанию.

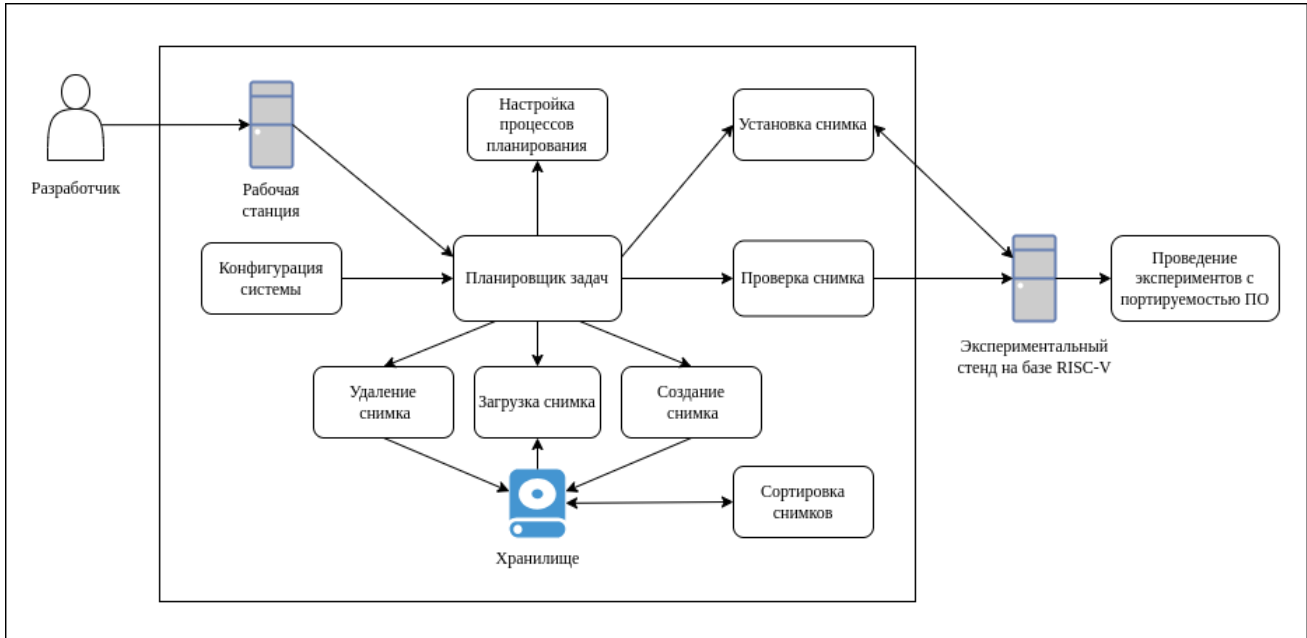


Рисунок 1 – Модульная схема системы резервного восстановления

Модель предоставляет интерфейс для работы с отсортированными по дате бекапами и позволяет выборочно удалять, или предотвращать автоматическое удаление выбранного пользователем бекапа [7]. В последствии, имея с собой носитель информации с сохраненными на него снепшотами системы новый пользователь может установить настроенное окружение и состояние эталонной машины на свою систему. Загрузка снепшота на новый стенд производится с помощью той же утилиты – Timeshift. Рабочая машина с установленным на

нее эталонным бекапом может использоваться для анализа портируемости микросервисных приложений на архитектуру RISC-V.

Использование данной системы позволяет сократить время на развертывание программной оболочки для тестирования портируемости в несколько раз.

ЛИТЕРАТУРА

1. Фролов В.А., Галактионов В.А., Санжаров В.В. Исследование технологии RISC-V. Труды ИСП РАН, том 32, вып. 2, 2020 г., стр. 81-98. DOI: 10.15514/ISPRAS-2020-32(2)-7.
2. Volokitin V., Kozinov E., Kustikova V., Liniov A., Meyerov I., Case Study for Running Memory-Bound Kernels on RISC-V CPUs, Lobachevsky State University of Nizhni Novgorod, Nizhni Novgorod, Russia 2024, с 52-53.
3. Dakić V., Mršić L., Kunić Z., Đambić G. Evaluating ARM and RISC-V Architectures for High-Performance Computing with Docker and Kubernetes, Algebra University, Zagreb, Croatia 2024 с.2-3.
4. Стивен Смит, под науч. ред. А.Ю. Романова, Программирование на языке Ассемблера, ДМК Пресс, Москва, 2025. с.44-45.
5. Cron-команды. [Электронный ресурс] Режим доступа. URL: <https://www.debian.org/doc/manuals/debian-handbook/sect.task-scheduling-cron-atd.ru.html>.
6. Timeshift. [Электронный ресурс] Режим доступа. URL: <https://github.com/linuxmint/timeshift/blob/master/docs/development.md>.
7. Коновалова, А. Г. Архитектура резервного копирования по модели Backup-as-a-Service и консолидация полученных данных / А. Г. Коновалова, В. И. Маракшин, И. В. Никифоров // 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. 140-141. – EDN ZRZTHT.

УДК 004.627

М.А. Щербаков (2 курс бакалавриата),
В.И. Гориховский, к.ф.-м.н., доцент

ПОВЫШЕНИЕ КАЧЕСТВА ДЕДУПЛИКАЦИИ С ПОМОЩЬЮ SIMILARITY BASED CHUNKING

Дедупликация — это метод сжатия данных, основанный на устранении повторяющихся фрагментов с целью снижения накладных расходов на хранение и передачу информации. Используется в системах хранения резервных копий, например TATLIN.BACKUP компании YADRO. Дедупликация бывает двух видов: файловая и блочная. Блочная дедупликация, она же чанковая, разбивает фрагмент данных на небольшие фрагменты и сохраняет лишь уникальные, заменяя остальные указателями на хранящийся. В рамках работы исследуется дедупликацию методом чанкирования.

Content Defined Chunking (CDC) — метод чанкирования, использующий контекст для определения границ блоков. Примерами таких методов являются RabinCDC, LeapCDC, SuperCDC [1]. Каждый из этих методов имеет недостатки. Для улучшения дедупликации существуют два подхода: Frequency Based Chunking и Similarity Based Chunking [2]. Качество методов исследовалось для RabinCDC, однако их применение возможно и для других вариаций CDC. Представленная работа посвящена анализу эффективности SBC.

Основная идея Similarity Based Chunking заключается в использовании дельта-кодирования для хранения похожих чанков. Для этого чанки группируются на основе их схожести, а после сохраняется лишь один чанк в группе, а остальные в виде дельта-кода. Для оценки качества применения SBC необходимо собирать информацию об уровне сжатия данных и о скорости работы в зависимости от параметров SBC и алгоритма CDC. Для этого

необходимо интеграционное тестирование внутри файловой системы, имеющей поддержку нескольких алгоритмов CDC.

Для проведения сравнения была выбрана ChunkFS — модельная файловая система, написанная на Rust, с встроенной поддержкой методов CDC и бенчмаркинга скорости чтения/записи, используемая для исследования алгоритмов дедупликации [3]. Целью работы является интеграция метода оптимизации дедупликации Similarity Based Chunking в ChunkFS для проведения исследований. Для её выполнения были поставлены следующие задачи:

1. Выполнение обзора существующих реализаций алгоритмов SBC с открытым кодом, предпочтительно написанных на Rust.
2. Интеграция SBC в ChunkFS.
3. Проведение экспериментов для определения повышения качества дедупликации.

В открытом доступе нет реализаций подобных алгоритмов на Rust, поэтому рассматривались, в основном, статьи с описанием математических моделей и реализации на других языках программирования, в частности, Python. В открытом репозитории на GitHub есть реализация Similarity Based Chunking на Python [4]. В этом проекте реализована ограниченная функциональность определения схожих чанков, не реализовано дельта-кодирование, отсутствует взаимодействие с файловой системой и возможность сравнивать эффективность алгоритма в зависимости от метода первичной дедупликации. Поэтому было принято решение реализовать алгоритм самостоятельно и интегрировать в ChunkFS.

Архитектура для интеграции была разработана и согласована совместно с разработчиком ChunkFS. В коде файловой системы были объявлены необходимые интерфейсы для поддержки оптимизаций дедупликации SBC и FBC. В коде алгоритма SBC были реализованы интерфейсы для хранения данных и применения алгоритма.

Наиболее важным параметром для анализа является коэффициент дедупликации, который вычисляется как отношение объема исходного датасета к сжатому алгоритмом дедупликации. Рассматривались датасеты с большим количеством повторяющихся фрагментов: две версии ядра Linux (3.4.6 и 3.4.7) и с малым количеством дубликатов: Enron Emails. В ChunkFS поддерживаются следующие CDC алгоритмы: SuperCDC, RabinCDC, SeqCDC, LeapCDC и UltraCDC, а также чанкирование фиксированного размера. Эксперименты проводились для всех этих чанкеров на выбранных датасетах.

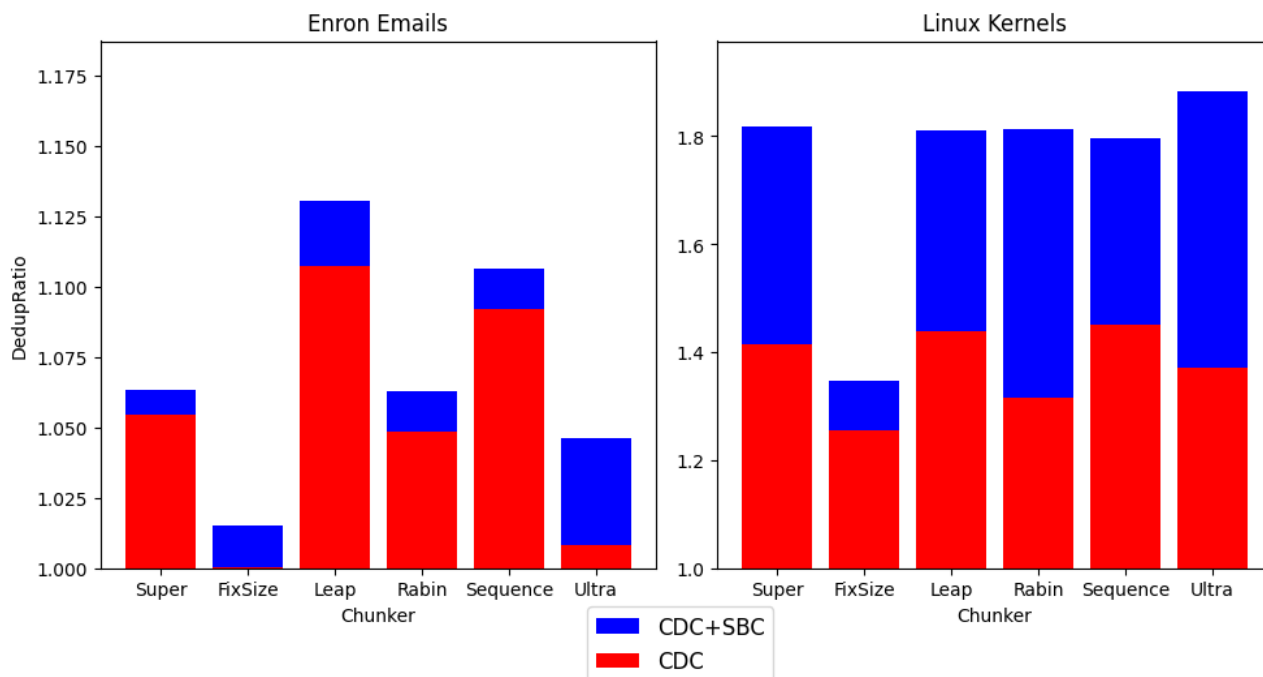


Рисунок 1 – Коэффициент дедупликации после применения SBC для различных чанкеров к датасету Enron Emails (слева) и Linux Kernels (справа)

Результаты экспериментов представлены на рисунке 1. На обоих датасетах применение SBC после чанкирования фиксированного размера показывает наименьший коэффициент дедупликации. На датасете Enron Emails применение SBC увеличивает коэффициент дедупликации на 1-2%, так как датасет содержит малое количество схожих фрагментов. Поэтому можно сделать вывод, что SBC не дает значимого результата для высокоэнтропийных данных. На датасете Linux Kernels применение SBC позволило дополнительно сжать данные примерно на 20%. Для различных чанкеров коэффициент примерно одинаковый, значит можно использовать в сочетании, например, с LeapCDC, для которого получена наилучшая производительность в наших экспериментах. Это говорит о том, что использование оптимизации SBC в бэкап системах может значительно уменьшить объем хранимых данных. Другие графики, иллюстрирующие результаты работы алгоритма SBC, хранятся в репозитории на GitHub [5].

ЛИТЕРАТУРА

1. Marcel Gregoriadis et al, "A Thorough Investigation of Content-Defined Chunking Algorithms for Data Deduplication", 28 сентября 2024 г.
2. L. Aronovich, D. R. Asher, and S. T. M. Klein Hersh, "Similarity based deduplication with small data chunks", Discrete Applied Mathematics, vol. 212, pp. 10–22, 2016.
3. Пилецкий О. А., Гориховский В. И. "Разработка инструмента для сравнения Content Defined Chunking на Rust" // Современные технологии в теории и практике программирования: сборник материалов конференции, Санкт-Петербург, 24 апреля 2024 года / Санкт-Петербургский политехнический университет Петра Великого. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2024. – С. 253-254.
4. Naveed Akthar. Web Scraping, Data Chunking, and Question Answering System. [электронный ресурс] Режим доступа: <https://github.com/Aktharnvdv/Retrieval-Augmented-Generation.git>
5. Similarity Based Chunking [Электронный ресурс] Режим доступа: https://github.com/maxscherbakov/sbc_algorithm.git

СОДЕРЖАНИЕ

Приветствие от компании YADRO	3
Тезисы докладов конкурса-конференции	4
Секция Программная инженерия: приложения, продукты и системы	4
<i>Акерма Э.М., Шмаков В.Э.</i> Разработка пользовательского чата на веб-сокетах	4
<i>Архипов В.Е., Воинов Н.В.</i> Разработка Telegram-Бота для скачивания и обработки контента с YouTube	5
<i>Алексеев Я.А., Андрианова Е.Е.</i> Метод настройки логической двунаправленной репликации в базе данных PostgreSQL	6
<i>Алпеева А.А., Корелин О.Н., Кобляков Д.А.</i> Взаимодействие операционных систем PL и PS частей SoC Zynq	8
<i>Батуева В.Б., Воскобойников С.П., Беляев С.А.</i> Разработка программы оценки забрасываемого веса ступени ракеты	10
<i>Бавыкин А.А., Савраненко А.С., Александрова О.В.</i> Оптимизация складских бизнес-процессов с использованием алгоритма назначения адресов	11
<i>Балунина Е.Е., Комарова Е.В.</i> Разработка веб-приложения для создания персонализированного меню и списка покупок на основе рецептов, добавленных методом веб-скрапинга по ссылке....	12
<i>Бендрышев С.А., Селин И.А.</i> Предсказание сахарного диабета I типа на основе медицинских показателей	14
<i>Беркоз М.А., Воинов Н.В.</i> Расширение модели управления доступом в системе хранения данных «Закрома.Хранение».....	16
<i>Береснев Р.А., Литвинов Ю.В.</i> Разработка веб-приложения для детекции дефектов ленты промышленного конвейера.....	18
<i>Боков Я.А., Яковлев А.Н., Скуднева Е.В.</i> Разработка операции обнаружения подобия текстов в рамках ССУБД Катарсис	20
<i>Борисов А.С., Туральчук К.А.</i> Разработка системы на основе LLM для глубокого чтения литературных текстов с контекстуальными аннотациями	21
<i>Борисов Д.К., Яковлев А.Н., Скуднева Е.В.</i> Реализация операций численного решения дифференциальных уравнений в ССУБД Катарсис	23
<i>Борисов А.А., Хлопин С.В.</i> Алгоритм и программа управления датчиком наличия и концентрации нефтесодержащих примесей.....	25
<i>Варнавский И.В., Прокофьев О.В.</i> Разработка библиотеки для работы с хронологическими типами данных в реляционных СУБД.....	27
<i>Виниченко Д.Н., Андрианова Е.Е.</i> Применение LSM-деревьев для локальных NoSQL-хранилищ в автономных системах.....	28
<i>Ворожко В.А., Маслаков А.П.</i> Разработка программного адаптера информационно-технического взаимодействия ведомственной автоматизированной системы с комплексом средств автоматизации системы управления движением судов Минтранс России (КСА СУДС) в интересах мониторинга морской обстановки	30
<i>Гавриков В.О., Яковлев А.Н., Скуднева Е.В.</i> Разработка замкнутого аппарата операций линейной алгебры в рамках ССУБД Катарсис.....	32

<i>Геворков К.М., Комарова Е.В.</i> Разработка системы управления ценовыми параметрами сделок для финансовых операций корпоративных клиентов в банкинге.....	34
<i>Гайле М.А., Асадулаев Ю.М., Зайцев И.В.</i> Распределённая система перебора хешей с использованием GPU кластеров.....	35
<i>Головин М.В., Молодяков С.А.</i> Разработка симулятора системы сборки, постановки и траления военных тралов	36
<i>Гладышев А.А., Щербakov Н.В.</i> Разработка API-сервиса для безопасного хранения и управления доступом к данным.....	38
<i>Дергунов Н.С., Ковалев А.Д.</i> Разработка системы дифференциальной диагностики заболеваний с использованием нейронных сетей типа «Трансформер».....	40
<i>Дмитриев А.В., Хахина А.М.</i> Разработка модели и системы анализа и прогнозирования продуктивности сотрудников на основе менеджера задач.....	42
<i>Доценко О.В., Смородников Г.В., Золотарев Р.А., Самочадин А.В.</i> Реализация средств прогнозирования облачных вычислений.....	44
<i>Дусеев И.Г., Андрианова Е.Е.</i> Исследование методов проектирования рекомендательной системы для веб-сервиса и мобильного Android-приложения по рецензированию и подбору книг.....	46
<i>Елакова В.О., Андрианова Е.Е.</i> Обзор методов разработки android приложения для подготовки к олимпиаде по СУБД PostgreSQL	48
<i>Емельянов П.А., Баженов К.М., Гончаров А.В.</i> Разработка интерактивного тренажёра по System Design.....	50
<i>Житнухина М.А., Литвинов Ю.В.</i> Sensor fusion для улучшения выкладывания модели дороги в 3D в ADAS	51
<i>Захаров П.А., Туральчук К.А., Ракова В.В.</i> Разработка программного обеспечения для дискретно-событийного моделирования движения общественного транспорта	53
<i>Загородняя А.Р., Шмаков В.Э.</i> Автоматизация управления складом на основе мобильного приложения	54
<i>Иванов З.Д., Медведев Б.М.</i> Разработка инструмента технической поддержки для отслеживания маршрута вызова на сети оператора связи.....	56
<i>Иванов Н.А., Маслаков А.П.</i> Разработка клиент-серверного решения для удалённого подключения к графической среде операционных систем на базе ядра Linux	58
<i>Иванова А.С., Шукин А.В., В.А. Пархоменко</i> Разработка средств оценки информативности и дизайна страниц маркетплейсов на основе айттрекинга	60
<i>Казанцева С.С., Силин М.С., Самочадин А.В.</i> Разработка веб-приложения для арбитража криптовалют.....	61
<i>Казимиров Н.А., Леонтьева Т.В.</i> Управление рисками на предприятии с помощью мобильной системы.....	63
<i>Карасева Е. О., Литвинов Ю.В.</i> Реализация мессенджера в веб-сервисе для помощи бездомным животным	65
<i>Картинин А.И., Прокофьев О.В.</i> Разработка веб-интерфейса и Telegram-бота для управления умной теплицей на основе IoT	67
<i>Клишева А.Д., Воскобойников С.П., Литвинов А.Н.</i> Разработка программного модуля для численного моделирования эффекта когерентного пленения населенностей в ячейках конечного размера в оптически плотной среде.....	68
<i>Клюкин С.А., Александрова О.В.</i> Автоматизированное тестирование трансформаторов с использованием технического зрения	69

<i>Кислов В.В., Лыченко Н.М.</i> Система автоматического мониторинга и анализа торговых предложений на маркетплейсах	71
<i>Козырев Ч.Э., Ковалев А.Д.</i> Разработка системы оценки платежеспособности банковских клиентов	72
<i>Козлова Е.А., Зайцев И.В.</i> Автоматизация составления расписания учебной сессии	74
<i>Колосовская А.А., Леонтьева Т.В.</i> Платформа для поиска и организации мероприятий по настольным играм: разработка и реализация.....	76
<i>Кошкарева Е.Д., Маслаков А.П.</i> Инструмент для удаленного управления конфигурацией приложений.....	78
<i>Коновалов М.А., Амосов В.В.</i> Разработка дизайн системы для приложений под платформу iOS .	80
<i>Кумище Р.А., Беккер Я.В., Хитров Е.Г.</i> Методы адаптации домена и переноса обучения для распознавания дефектов материалов по снимкам поверхности	82
<i>Лаврова Л.А., Туральчук К.А.</i> Обзор подходов к валидации LLM-моделей	84
<i>Лебедев Д.С., Маслаков А.П.</i> Интерактивное обучающее приложение для изучения основ программирования и робототехники.....	86
<i>Лисовик Р.А., Дробинцев П.Д.</i> Выбор архитектуры базы данных приложения автосервиса Street Legal Service	88
<i>Лихачева А.А., Пархоменко В.А.</i> Разработка системы поддержки анализа компьютерных игр для их ранжирования и составления рекомендаций по улучшению	89
<i>Майстренко Д., Малеев О.Г.</i> Виртуальный помощник для заказа продуктов питания.....	91
<i>Макаров М.С., Жолдошбаев К.А., Верзунов С.Н.</i> Интеллектуальная платформа мониторинга почв и водных ресурсов в сельском хозяйстве.....	93
<i>Марголис В.О., Кузнецова Л.В.</i> Разработка веб-сервиса для автоматизированного создания Docker-образов.....	95
<i>Матус А.С., Дробинцев П.Д.</i> Программная система автоматизации управления взаимодействия с поставщиками информационно-технологических решений	97
<i>Мещеряков Г.А., Воинов Н.В.</i> Разработка серверной составляющей программного продукта Polyhabr.....	99
<i>Мезенцева П.А., Прокофьев О.В.</i> Разработка системы автоматизации поддержки миграций	100
<i>Мясоедов Л.А., Тышкевич А.И.</i> Разработка системы для оформления комиссионных кредитных страховых и нестраховых продуктов для нужд банка	102
<i>Мухин Ф.А., Малеев О.Г.</i> Трекинг посетителей общественных пространств с помощью нейронных сетей.....	104
<i>Мэн Цзянин, Молодяков С.А.</i> Продвинутый редактор для художественной работы с изображением	106
<i>Новицкий И.В., Медведев Б.М.</i> Разработка программных средств системы контроля персонала на опасных производствах.....	108
<i>Нефедев В.К., Александрова О.В.</i> Разработка менеджера задач. Прогноз действий на основе ML	110
<i>Никифорова Е.А., Коликова Т.В.</i> Создание программной среды для местоопределения источника радиоизлучения.....	111
<i>Пахомов А.В., Комарова Е.В.</i> Разработка инструмента для создания интерактивных карт с пользовательскими коллекциями меток.....	113

<i>Пахомов А.В., Пархоменко В.А.</i> Разработка SBFL средства локализации ошибок в программном обеспечении	114
<i>Пашин Ф.А., Воинов Н.В.</i> Разработка iOS-приложения для поиска и бронирования автомоек...	116
<i>Пехтерев В.В., Пшеничная К.В.</i> Выбор оптимального маршрута передачи данных по компьютерной сети	117
<i>Пономаренко В.О., Артеменко Е.К., Багрей А.О., Кулаков К.В., Нейдорф П.Я., Ахпашев Р.В.</i> Разработка программного комплекса для сбора и анализа данных с реальных сетей мобильной связи.....	119
<i>Пономаренко В.О., Медведев Б.М.</i> Реализация кэширования новостной ленты для iOS-приложения «Одноклассники».....	121
<i>Парамонова А.П., Шмаков В.Э.</i> Мобильное приложение автоматизации управления расписанием	123
<i>Попов М.И., Воинов Н.В.</i> Разработка симулятора ПЭТ-сканера для лучевой терапии.....	125
<i>Попов Р.Р., Воинов Н.В., Павлов Е.А.</i> Разработка платформы для исследования свойств стекла	127
<i>Приймак М.С., Прокофьев О. В.</i> Пилотный проект системы контроля выходной продукции для линии ОиНКТ 3, линия 3	129
<i>Пятов Д.С., Леонтьева Т.В.</i> Разработка приложения для 3d печати по технологии селективного лазерного спекания.....	130
<i>Примак А.В., Гадаев Е.М., Пшеничная К.В.</i> Применение генетического алгоритма для выбора оптимальных параметров маломерного судна.....	132
<i>Ракитин Д.А., Воинов Н.В., Юсупов Ю.В.</i> Разработка платформы для управления задачами по работе с данными в распределенных хранилищах	133
<i>Рагимов Р.А., Маслаков А.П.</i> Сравнение стандартов сжатия видео и аудио контента для потоковой передачи и возможности, открываемые стандартом Wi-Fi 7 для потоковой передачи интерактивного контента в реальном времени.....	135
<i>Рыженко Д.А., Сараджишвили С.Э.</i> Разработка прототипа игры: контроллера персонажа и основных механик	137
<i>Сеферян Н.Э., Маслаков А.П.</i> Разработка десктопного приложения для управления информацией по комитетам и управления.....	138
<i>Свириденко С.А., Павлов Е.А.</i> Разработка серверной части для информационно-аналитической системы управления спортивной деятельностью.....	140
<i>Смола А.М., Литвинов Ю.В.</i> Распознавание направления дороги вдали по видео в ADAS	142
<i>Сиротин П.О., Шмаков В.Э.</i> Механизмы оптимизации WAN	144
<i>Соколов Н.А., Молодяков С.А.</i> Разработка алгоритма и программы для выявления читерства в онлайн-шахматах	146
<i>Сорин Н.В., Селин И.А.</i> Программное обеспечение симулятора рабочего места штурмана.....	148
<i>Стретенский Д.М., Воинов Н.В.</i> Разработка программы для работы с первичными бухгалтерскими документами с использованием JOOQ.....	150
<i>Старикевич Э.А., Самочадин А.В.</i> Автоматизация учёта и продаж товаров на борту пассажирских поездов.....	151
<i>Суханова Е.А., Пархоменко В.А.</i> Разработка рекомендательной системы по подбору фильмов.	153
<i>Соколов И.А., Максина Е.С., Старунская Е.И., Никифоров И.В.</i> Применение NLP для решения задачи автоматизации сбора и обработки обратной связи	155

<i>Сырбу Г.В., Гончаров А.В.</i> Использование языковой модели для генерации синтетических данных для дообучения модели рекомендательной системы площадки по продаже туров для путешествий.....	158
<i>Трубач К.И., Кулик М.И., Малявко Н.В., Шлык П.А., Иванюк А.А.</i> Реализация TRNG на основе DRAM PUF.....	159
<i>Фернандес А., Газизова А.М., Самочадин А.В.</i> Платформа для параллельного дискретно-событийного моделирования.....	161
<i>Фирсов С.В., Теренин А.А.</i> Layer-Independent Dynamic Patching (LIDP): методология динамического патчинга .NET приложений в контейнеризированных средах с применением IL Weaving.....	163
<i>Федорова С.К., Гончаров А.В.</i> Использование языковой модели в рамках рекомендательной системы туров пользователю на площадке по продаже туров для путешествий.....	166
<i>Фокин А.С., Ковалев А.Д.</i> Разработка системы музыкальных рекомендаций.....	167
<i>Цэнгэл Б.Л., Горавнева Т.С.</i> Разработка мобильного приложения для осуществления навигации по помещениям круизного судна.....	169
<i>Худоложский А. А., Прокофьев О. В.</i> Разработка расширения для СУБД PostgreSQL для интеграции нейронных сетей с использованием OpenNN.....	171
<i>Шабинский Д.Э., Воинов Н.В.</i> Разработка Android-клиента проекта PolyHabr	172
<i>Шапочкин Е.Г., Дементьев Н.А., Шошмина И.В.</i> Разработка веб-сервиса для аналитики и управления рекламными кампаниями на маркетплейсах	174
<i>Шкроба О.В., Прокофьев О.В., Дробинцев П.Д.</i> Разработка telegram-бота с интеграцией с облачным сервисом amoCRM для оптимизации работы международной детской школы программирования «Алгоритмика»	176
<i>Шульгин И.К., Медведев Б.М.</i> Программное средство валидации расчётов за роуминговые услуги	177
<i>Якубов Ш.К., Шмаков В.Э.</i> Проектирование платформы для создания NFT и Жетонов на блокчейне TON	179
Секция Программная инженерия: инструментальные средства и технологии проектирования и разработки	181
<i>Айнализев Н.З., Шукин А.В., Пархоменко В.А.</i> Разработка плагина для выявления причин flaky-тестов в Playwright.....	181
<i>Антипенко С.В., Пархоменко В.А.</i> Разработка цепочки инкрементальной обработки больших данных и сравнительная оценка ETL-решений для её реализации	183
<i>Арбузов Б.Е., Петров О.Н.</i> Разработка шаблона виртуального инструктора для систем расширенной реальности	184
<i>Барковская М.А., Шалымов Д.С.</i> Code Search Toolkit: Разработка системы поиска фрагментов программного кода на основе методов обнаружения дубликатов.....	186
<i>Беляев Д.С., Медведев Б.М.</i> Разработка программных средств морской навигационной системы на базе протоколов NMEA.....	187
<i>Батушева О.В., Александрова О.В.</i> Автоматизированное средство распознавания и проведения анализа заявок клиентов	189
<i>Булатов Д.Е., Вяземский А.А., Шожат А.А., Коликова Т.В.</i> Анализ сведений о шахматных партиях и игроках Lichess.org	191
<i>Борисов А.А., Нестеров С.А.</i> Утилита миграции баз данных из Oracle Database в Ред Базу Данных	193

<i>Булгаков И.Н., Петров А.В.</i> Адаптация автоматизированного сервиса для анализа результатов нагрузочного тестирования в системе облачного гейминга.....	195
<i>Брежо М.Г., Воинов Н.В.</i> Разработка механизма передачи событий системы Закрома.Хранение для интеграции с siem-платформой ArcSight.....	197
<i>Возиян А.И., Дробинцев П.Д.</i> Автоматизация проверки корректности развертывания серверной части 1С в облачной среде разработки.....	198
<i>Дружков С.А., Кознов Д.В.</i> Поиск клонов и автоматический рефакторинг Lua-программ для уменьшения размера кода сетевых устройств	200
<i>Доманцевич Е.Д., Муравьев Е.А.</i> Обработка ошибок в компиляторе с контролируемого естественного языка спецификации требований.....	202
<i>Емельянов В.Д., Пархоменко В.А.</i> Разработка анализатора компонентов исходного кода на примере web-сервисов, использующих Spring Boot.....	204
<i>Епишин И.А., Щукин А.В., Пархоменко В.А.</i> Исследование средств трекинга взгляда при выполнении заданий по программированию и разработка плагина для выявления аномального поведения	206
<i>Ефимов М.А., Каширин К.А., Самочадин А.В.</i> Средство создания имитационных моделей центров облачных вычислений.....	208
<i>Жуков Г.М., Перов И.А., Дробинцев П.Д.</i> Оптимизация нагрузки на бэкэнд при массовых изменениях данных с использованием асинхронных технологий и кеширования.....	210
<i>Каралюс Н.Ю., Туральчук К.А.</i> Разработка инкрементального генератора кода для получения информации о типах в .NET приложениях	212
<i>Карпова С.С., Александрова О.В.</i> Адаптация автоматизированного таск-трекера для управления бизнес-процессами	213
<i>Кривошапкина Р.И., Леонтьева Т.В.</i> Интеграция искусственного интеллекта в приложение для управления проектами	215
<i>Кочергин В.Ю., Саранулов В.А., Смирнов К.К.</i> Рекомпиляция с помощью Ghidra.....	217
<i>Киселева А.А., Никифоров И.В.</i> Подход автоматизированного развертывания облачных сервисов «Р7-офис» в информационно-технологической инфраструктуре предприятия с использованием инструмента Ansible.....	219
<i>Кубов Н.А., Сабуткевич А.М., Никифоров И.В.</i> Применение метода горизонтального масштабирования для исполнения имитационных моделей в облачной среде.....	221
<i>Ларин А.В., Амосов В.В.</i> Сборщик базы знаний и ИИ-ассистент по вопросам пользователей на основе Web3.....	223
<i>Лихачев М.С., Пархоменко В.А.</i> Экспериментальное сравнение средств property-based тестирования на JavaScript.....	224
<i>Мазин Я.П., Павлов Е.А.</i> Использование архитектуры трансформеров для прогнозирования показателя преломления халькогенидных стёкол	226
<i>Мбеле Оссийи Л.П., Дробинцев П.Д.</i> Автоматическое выявление ненавистнических высказываний с помощью нейронных сетей.....	228
<i>Мацигудра П.Е., Шмаков В.Э.</i> Разработка системы интеллектуального управления и балансировки нагрузки для API-сервисов искусственного интеллекта	230
<i>Мейник А.В., Воскобойников С.П., Попов Е.Н.</i> Разработка и применение распределённой модели вычислений для моделирования системы диполей.....	232
<i>Мосолов М.А., Смородников Г.В., Золотарев Р.А., Самочадин А.В.</i> Средства краткосрочного прогнозирования для облачных вычислений.....	234

<i>Осетров Д.А., Шошмина И.В.</i> Разработка сервиса поддержки клиентов в системе обработки звонков.....	236
<i>Набиуллин Д.Ф., Жуков Г.М., Перов И.А., Александрова О.В.</i> Реализация инструмента сжатия файлов на основе особенностей цветовосприятия	238
<i>Палий А.А., Нестеров С.А.</i> Анализ методов оптимизации производительности PostgreSQL при работе со сложными типами данных	239
<i>Перов И.А., Жуков Г.М., Дробинцев П.Д.</i> Выбор фреймворка для инъекции зависимостей при разработке Android-приложений на основе оценки влияния на производительность и скорости сборки	241
<i>Петров А.А., Щукин А.В., Пархоменко В.А.</i> Разработка плагина сбора метрик для Drone и сравнение его с аналогами в других инструментах CI/CD	243
<i>Рафальская М.В., Семенова-Тян-Шанская В.А.</i> Разработка приложения для расчета трудоемкости постройки строительства транспортных судов	245
<i>Ромашов В.А., Никифоров И.В., Юсупова О.А.</i> Проектирование и разработка AI-агента для автоматизации разработки программного кода.....	246
<i>Радько В.В., Туральчук К.А.</i> Разработка плагина Android Gradle для генерации демо-приложения дизайн-системы проекта	248
<i>Рыкова А.А., Золотарев Р.А., Смородников Г.В., Самочадин А.В.</i> Разработка средств распределения ресурсов для облачных хранилищ	250
<i>Сергеев Д.В., Каширин К.А., Самочадин А.В.</i> Средства генерации синтетических данных для формирования датасета для работы с облачными вычислениями.....	252
<i>Сафронов А.А., Сараджиев С.Э.</i> Оценка применимости гибридного подхода к анализу логов: сравнение supervised, semi-supervised и unsupervised методов	253
<i>Сидоркина М.А., Пархоменко В.А.</i> Разработка приложения для оценки пользовательского интерфейса на основе анализа совместных движений глаз и мыши	254
<i>Стопченко М.В., Шмаков В.Э.</i> Разработка системы умного дома с голосовым управлением	256
<i>Сыпачёв К.В., Пархоменко В.А.</i> Экспериментальное сравнение возможностей LLM для поиска семантических ошибок	257
<i>Тимошин Е.Е., Воинов Н.В.</i> Сравнение паттернов отказоустойчивости на примере разработки микросервисных приложений	259
<i>Тампио И.С., Селин И.А.</i> Система мониторинга безопасности рабочего процесса с применением технологий компьютерного зрения	260
<i>Ткаченко А.А., Никифоров И.В.</i> Подход консолидации данных для автоматизации анализа и управления ИТ-ландшафтом компании	262
<i>Шаровагин А.Н., Семенова-Тян-Шанская В.А.</i> Применение метода К-средних для анализа эксплуатационных характеристик судна.....	265
<i>Якишева Л.М., Серазитдинова А., Каширин К.А., Самочадин А.В.</i> Backend и frontend стенда для исследования алгоритмов прогнозирования и распределения ресурсов облачных вычислений	267
Секция Программная инженерия: методы и алгоритмы теории программирования	269
<i>Багрова А.В., Мальцев А.Л., Романчук К.Д., Амосов В.В.</i> Программная и аппаратная реализация алгоритма Дейкстры.....	269
<i>Бородулин М.Д., Петров А.В.</i> Тестирование на основе свойств (property-based testing): искусство проверки программного обеспечения через универсальные свойства.....	270
<i>Волкова Д.В., Шагалова П.А.</i> Исследование инструментов оптимизации кода в задаче повышения производительности обработки изображений	271

<i>Волосов Е.А., Нестеров С.А.</i> Применение метода Плаккетта-Бермана для анализа IT-систем на базе PostgreSQL	272
<i>Ерошин И.П., Жукин А.В., Пархоменко В.А.</i> Экспериментальное исследование алгоритмов компьютерного зрения для оценки удовлетворённости пользовательским интерфейсом с учётом индивидуального эффекта воздействия	274
<i>Демчук П.Д., Шошмина И.В.</i> Моделирование вычислительного конвейера GPU с поиском оптимальных параметров с помощью model checking	276
<i>Королев Д.О., Никифоров И.В., Юсупова О.А.</i> Методы дискретно-событийного моделирования для изучения и предотвращения сбоев и аномалий в работе облачных инфраструктур.....	278
<i>Климков А.В., Герасимов А.С.</i> Генерация учебных заданий на топологическую сортировку	280
<i>Кулаков Н.В.</i> Алгоритмическое исследование стратегий инвалидации кеша при взаимодействии с высоколатентным и непрозрачным источником данных	282
<i>Лонишин М.Р., Сабуткевич А.М., Никифоров И.В.</i> Метод модели-ориентированной разработки на основе дискретно-событийного моделирования	284
<i>Чолак Э.А., Петров А.В.</i> Интеграция DevOps и тестирования: как обеспечить качество ПО на каждом этапе.....	286
<i>Мехоношин Н.Н., Миниахметов А.У., Пархоменко В.А.</i> Исследование и расширение возможностей инструмента поиска семантических ошибок Avicenna	287
<i>Московских Д.П., Мелентьев О.Г.</i> Разработка блока управления FFT на основе конечных автоматов.....	289
<i>Норицин И.А., Брагин К.И.</i> Имитационное моделирование алгоритмов планирования ресурсов мобильной сети при передвижении абонентов.....	291
<i>Палий Е.А., Нестеров С.А.</i> Совместное использование реляционных и NoSQL СУБД в современных программных продуктах на примере PostgreSQL и Redis.....	293
<i>Савченко А.В., Петров О.Н.</i> Разработка плагина для написания и выполнения логических программ	295
<i>Слугин А.Н., Сартасов С.Ю.</i> Методы работы с утечками памяти в JVM.....	296
<i>Сопрачев А.К., Дробинцев П.Д.</i> Преимущества применения Serverless-подхода при разработке микросервисных приложений	298
<i>Селиверстова А.К., Пархоменко В.А.</i> Исследование и экспериментальное сравнение средств тестирования инфраструктуры с помощью InSpec, Terratest, Test Kitchen.....	300
Секция Подходы к разработке программного обеспечения на основе технологий компании YADRO	302
<i>Андрианов А.А., Никифоров И.В.</i> Метод организации управления хранилищами пользовательских записей в Linux дистрибутивах	302
<i>Гавриленко М., Васенина А.И.</i> Реализация лог-структурированного блочного устройства в ядре Linux.....	304
<i>Денисенко Б.А., Юсупова О.А., Никифоров И.В.</i> Реактивная система на основе Netlink нотификаций: gRPC-доставка и HotPlug для управления сетевой конфигурацией	305
<i>Дмитриевцев А.С., Гориховский В.И.</i> Повышение качества дедупликации с помощью Frequency Based Chunking.....	307
<i>Дюльдин Г., Райкевич И.Ф., Соломонов М.С.</i> Архитектура CSI-драйвера для Kubernetes с поддержкой спецификации Swordfish	309
<i>Ерин И.А., Гориховский В.И.</i> Исследование производительности асинхронного рантайма tokio	312

<i>Истаев Э.Э., Фоломкин Д.А., Александрова О.В.</i> Разработка динамической ролевой модели управления доступом для сервера-эмулятора Swordfish	313
<i>Клеблеева Э.М., Вайндрах Д.М., Селин И.А.</i> Анализ архитектурной документации сравнение PlantUML-диаграмм с исходными файлами на языке Golang.....	316
<i>Кузьмин Р.С., Шагалова П.А.</i> Разработка и оптимизация алгоритма поиска угловых точек на изображении.....	318
<i>Лебедева Е.П., Никифоров И.В.</i> Разработка системы управления репутацией узлов объектного хранилища FrostFS	319
<i>Лонишин М.Р., Богданов Н.Р., Селин И.А.</i> Разработка программы для сравнения классовых диаграмм PlantUML и protobuf файлов	321
<i>Луконенко Н.И., Смирнов К.К.</i> Создание SnK на базе процессора SCR1 на плате Tang Primer 20K	323
<i>Мамонова А.С., Тронеv В.В., Селин И.А.</i> Анализ архитектурной документации: Сравнение C4-диаграмм с каталогами проекта	325
<i>Попович И.А., Пушница В.А., Калачиков А.А.</i> Экспериментальная реализация алгоритмов синхронизации OFDM для бюджетных SDR.....	327
<i>Пономарев А.А., Сабуткевич А.М., Никифоров И.В.</i> Применение событийно-ориентированной архитектуры (EDA) в системах ОАМ для управления сетевыми устройствами.....	329
<i>Росенко А.С., Молодяков С.А.</i> Внедрение Golang в RISC-V разработку: анализ актуальных разработок и перспектив.....	331
<i>Савкин А.Е., Шагалова П.А.</i> Применение нейросетевых моделей в разработке систем компьютерного зрения на базе RISC-V	333
<i>Семичев А.С., Александрова О.В.</i> Разработка высокопроизводительной библиотеки для асинхронного взаимодействия с PostgreSQL	334
<i>Симонова Д.В., Шагалова П.А.</i> Исследование и реализация подключения устройств с использованием GPIO интерфейса на платах с процессорной архитектурой RISC-V	336
<i>Соболева А.Е., Кладковой М.Д., Соломонов М.С., Никифоров И.В.</i> Автоматизация управления системами хранения данных в Kubernetes с использованием CSI-драйвера и спецификации Swordfish.....	338
<i>Солодовников С.Ф., Никифоров И.В., Устинов С.М.</i> Исследование нагрузки на блочные устройства для тестирования целостности СХД.....	340
<i>Суворов Р.М., Григорьев С.В., Кутуев В.А.</i> Оптимизация функции умножения матриц библиотеки SuiteSparse:GraphBLAS с использованием векторного расширения RISC-V	342
<i>Тронеv В.В., Селин И.А.</i> Сравнение технических документаций сетевой инфраструктуры в форматах PlantUML и C4.....	344
<i>Тесленко А.Р., Ролецкая С.Н., Коликова Т.В.</i> Автоматизация сбора диагностических данных с серверных систем с помощью консольной утилиты с использованием декларативной конфигурации	346
<i>Тянутов М.В., Юсупова О.А.</i> Автоматизация тестирования и управления шаблонов Zabbix	347
<i>Фоломкин Д.А., Никифоров И.В., Устинов С.М.</i> Методика развертывания окружения для разработки систем хранения данных с использованием инструмента Ansible	349
<i>Худяков Г.А., Макейкин А.С., Соломонов М.С.</i> Автоматизация конфигурирования и управления системами хранения данных за счёт Ansible-модулей для спецификации Swordfish.....	351
<i>Черепанов Н.И., Чистяков В.М., Малеев О.Г., Степина Н.О.</i> Обработка изображений на платформе RISC-V	354

<i>Шемаев К.Е., Ребдев П.А., Степина Н.О.</i> Векторизация алгоритмов для повышения производительности вычислений на процессорной архитектуре RISC-V	356
<i>Шеремет С.С., Мадьяров Г.С., Никифоров И.В., Степина Н.О.</i> Система резервного восстановления программной инфраструктуры на экспериментальном стенде с RISC-V	357
<i>Щербаков М.А., Гориховский В.И.</i> Повышение качества дедупликации с помощью Similarity Based Chunking.....	359

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

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

23–24 апреля 2025 года

Налоговая льгота – Общероссийский классификатор продукции
ОК 005-93, т. 2; 95 3004 – научная и производственная литература

Подписано в печать 16.04.2025. Формат 60×84/16. Печать цифровая.

Усл. печ. л. 23,25. Тираж 200. Заказ 1941.

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.

