Министерство науки и высшего образования Российской Федерации

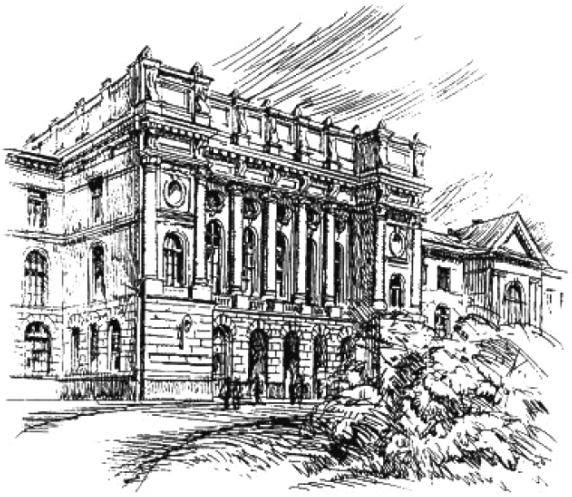
САНКТ-ПЕТЕРБУРГСКИЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

DELL TECHNOLOGIES

EPAM SYSTEMS

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

Сборник материалов конференции
23 апреля 2020 года



ПОЛИТЕХ-ПРЕСС
Санкт-Петербургский
политехнический университет
Петра Великого

Санкт-Петербург
2020

УДК 004
ББК 32.973
С56

Современные технологии в теории и практике программирования : сборник материалов конференции, 23 апреля 2020 г. – СПб. : ПОЛИТЕХ-ПРЕСС, 2020. – 175 с.

В сборнике публикуются материалы докладов студентов, представленные на научно-практическую конференцию, проводимую Санкт-Петербургским политехническим университетом Петра Великого и организованную Институтом компьютерных наук и технологий при поддержке Санкт-Петербургского центра разработок «Dell Technologies» и компании «EPAM Systems». Доклады отражают уровень подготовленности студентов – участников конференции в области применения современных средств и технологий разработки программного обеспечения.

Представляет интерес для специалистов в области информационных технологий, методов разработки программных проектов различного назначения, систем и средств автоматизации инженерного проектирования, а также для учащихся и работников системы высшего образования.

ISBN 978-5-7422-6935-9

Приветствие от Санкт-Петербургского Центра Разработок Dell Technologies

Уважаемые участники конференции
«Современные технологии в теории и практике программирования»!

С тезисом о том, что цифровая трансформация неизбежна, сегодня согласны даже скептики. Готовы меняться и модернизироваться компании с любыми бизнес-моделями во всех странах и отраслях. За пять предыдущих лет человечеством было произведено информации больше, чем за всю предшествующую историю. Компании по всему миру изобретают все новые технологии для обработки этой информации, превращения ее в данные и безопасного хранения. Инновации являются неотъемлемой частью жизни и развития современной IT-компании.

Это и есть одна из причин, по которой Dell Technologies с 2013-го года поддерживает конференцию «Современные технологии в теории и практике программирования». Данная конференция позволяет молодыми учеными продемонстрировать результаты их исследований в области информационных технологий.

Санкт-Петербургский центр разработок Dell Technologies и Политехнический университет сотрудничают уже более одиннадцати лет. За эти годы нами было реализовано более пятнадцати совместных образовательных проектов. Стоит отметить, что с 2016-го года в Политехническом университете реализуются студенческие проекты с открытым исходным кодом, в рамках которых участники вносят свой вклад в продукты, которыми пользуются ведущие ИТ-компании всего мира. Сотрудники компании проводят спецкурсы для студентов-политехников по тематикам современного программирования, управления программным продуктом, хранению данных, тестированию программного обеспечения и DevOps подходам. Помимо этого, выпускники Политеха проходят стажировки в компании Dell Technologies, по окончании которых, лучшие студенты получают возможность стать инженерами нашего центра.

Целью как данной конференции в целом, так и в частности секции Dell Technologies является, в первую очередь, поддержка молодых ученых, их научных исследований в области информатики и информационных технологий; информирование молодых ученых о передовых технологических трендах в индустрии, о современных программных продуктах и решениях, представленных крупнейшими мировыми ИТ-корпорациями. Для компаний, занимающихся разработкой и сопровождением ПО, равно как и другой деятельностью в области ИТ, сотрудничество с университетами жизненно важно для дальнейшего устойчивого развития.

Желаю вам успешной работы на конференции, профессионального роста и творческих побед!

С уважением,

Павел Борисович Егоров

Генеральный директор

Санкт-Петербургского Центра Разработок Dell Technologies

Выпускник Санкт-Петербургского политехнического университета

Приветствие от компании EPAM

Уважаемые гости конференции!

От себя лично и от имени компании EPAM я рада приветствовать вас на научно-практической конференции студентов, аспирантов и молодых ученых «Современные технологии в теории и практике программирования».

Вот уже третий год подряд компания EPAM становится партнером мероприятия. Мы обладаем многолетним опытом и экспертизой в области разработки программного обеспечения, решаем реальные бизнес-задачи с помощью ИТ для ведущих мировых компаний, и мы рады поделиться с вами этим опытом, нашим пониманием тенденций развития отрасли.

Информационные технологии развиваются по экспоненте. Сегодня компании, которые стремятся быть на шаг впереди, возлагают большие надежды на такие технологии, как Big Data, искусственный интеллект и облачные решения. Завтра новые технологии будут менять ландшафт всех отраслей: виртуальная и дополненная реальность, квантовые технологии, автономные «вещи».

Эти тренды создают спрос на новых специалистов — людей будущего, которые помимо исключительных знаний различных технологий должны будут обладать гибкостью: способностью подстраиваться под новые вызовы, осваивать смежные направления, уметь находить нестандартные решения.

EPAM всегда поддерживает практическую направленность образовательных инициатив. Мы сотрудничаем с 18 вузами по всей России, приглашаем студентов и начинающих специалистов на тренинги, чтобы дать более глубокие знания в информационных технологиях и предоставить возможность закрепить полученные в вузе знания на практике — на реальных проектах. Эта конференция даёт возможность поделиться с вами опытом в ИТ-разработке, рассказать о том, как мировые лидеры в различных сферах бизнеса решают вопросы цифровой трансформации.

Надеемся, эта конференция поможет вам раскрыть свой потенциал и сделать первый шаг в карьере в сфере информационных технологий.

С уважением,

директор офиса в Санкт-Петербурге

Марианна Округина

Информационное сообщение о проведении научно-практической конференции студентов, аспирантов и молодых ученых «Современные технологии в теории и практике программирования»

В целях содействия подготовке к будущей работе в профессиональных программистских коллективах, обеспечивающих высокое качество программного продукта, в целях поддержки изучения современных информационных технологий и инструментальных средств в соответствии с мировыми стандартами и действующими международными сертификационными требованиями, а также для выявления талантливых молодых специалистов в области разработки и использования программных систем, инженерных проектов и моделей, мировые лидеры в разработке программного продукта компании Dell Technologies и EPAM совместно с Санкт-Петербургским Политехническим Университетом объявляют научно-практическую конференцию **«Современные технологии в теории и практике программирования».**

1. В рамках мероприятия планируется провести:

   Пленарное заседание и открытые лекции ведущих специалистов компаний Dell Technologies и EPAM

   Конкурс-конференцию теоретических и практических работ - претендентов на именные дипломы компаний Dell Technologies и EPAM.

2. Сроки проведения мероприятия: **23 апреля 2020 года**.

3. Место проведения:

   Санкт–Петербургский Политехнический Университет

4. Предлагаемая тематика конкурса:

   - Программная инженерия: приложения, продукты и системы
   - Программная инженерия: инструментальные средства и технологии проектирования и разработки
   - Программная инженерия: методы и алгоритмы теории программирования
   - Технологии обработки и анализа больших массивов данных на основе работ EPAM
   - Подходы к разработке ПО на основе технологий Dell Technologies.

   Данный конкурс рассчитан на студентов 1-6 курсов, аспирантов и молодых ученых, обучающихся или работающих по направлениям информатики, вычислительной техники, и по направлениям использования информационных технологий, владеющих основами современных промышленных технологий и инструментарием разработки программных продуктов и проектов. Предполагается, что участник должен проявить свои знания и умения не столько в области программирования различных математических головоломок, сколько в области разработки и использования программных продуктов и систем в условиях, максимально приближенных к реальным процессам проектирования и разработки современных систем различной степени сложности.

   Темы работ могут быть взяты в соответствии с тематикой секции конференции.

   На конкурс принимаются работы, оформленные в соответствии с заданными требованиями и представленные в организационный комитет конференции не позднее **18 марта 2020 года**. Требования к представлению проектов представлены в документах [I-II].

5. Количество работ конкурсантов не ограничено.

6. Конкурс принятых к участию работ проводится в один этап и протекает в виде представления презентаций и секционных докладов. Приглашения авторам отобранных работ будут рассылаться в период с 10 по 17 апреля 2020г. Решение о награждении участников конкурса принимается конкурсной комиссией. Требования к проведению докладов представлены в документе «Порядок проведения презентаций и выступлений на конкурсе-конференции» [III].
7. Объявления о условиях проведения конкурса и вся дополнительная информация будут представлены на сайте ИКНТ СПбПУ в разделе Конференции. Приглашения на участие в конкурсе-конференции будут разосланы в ведущие университеты Северо-Запада.

ЖЕЛАЕМ УСПЕХОВ И ИНТЕРЕСНОЙ РАБОТЫ!

Научные руководители конференции:

Директор ВШПИ ИКНТ Дробинцев Павел Дмитриевич,

профессор Чернoруцкий Игорь Георгиевич

секретарь Эламик Татьяна Николаевна, тел. 552-76-66, часы приема: понедельник и среда с 15 до 17 часов.

Адрес: Политехническая ул., 29, 2 уч.корпус, к.359а

E-mail: **icc_2020@mail.ru**

I. Порядок представления проектов на конкурс-конференцию.

Для представления разработанных проектов в Конкурсный центр устанавливаются следующие правила:

- Желающие принять участие в конкурсе – конференции на этапе предварительного сбора материалов на почтовый адрес icc_2020@mail.ru присылают материалы секционного доклада и анкету участника.
- На заочном этапе – первом туре - конкурсная комиссия на основе тезисов отбирает работы для непосредственного участия в конкурсе, после чего высылает приглашение по электронной почте (при наличии) или по телефону.
- Очный этап проводится в **виде секционных докладов для теоретических или прикладных поисковых работ.**
- Конкурсная комиссия оценивает представленные доклады по представлению руководителей секций и отбирают наиболее значимые из них для награждения. Участники очного этапа награждаются специальными дипломами Dell Technologies или EPAM.

II. Требования к содержанию конкурсных работ.

Предполагается, что конференция будет проходить в рамках пяти секций:

С1. Программная инженерия: приложения, продукты и системы

Задача конкурса в разделе С1: продемонстрировать в тезисах и презентации знания и умения в создании программного продукта на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ по сравнению с существующими.

(Например, программные проекты, представляющие завершенный продукт или незавершенный, но исполняемый в соответствии с требованиями к промышленному продукту)

Рекомендуется особо отмечать передовые технологии, поддержанные в программных продуктах и оценку эффективности их применения.

С2. Программная инженерия: инструментальные средства и технологии проектирования и разработки

Задача конкурса в разделе С2: продемонстрировать знания и умения в создании и/или применении перспективных методов и технологий разработки программного обеспечения на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ по сравнению с существующими. Наряду с технологиями разработки программных продуктов на секции рассматриваются программные средства систем управления жизненным циклом информации.

(Например, автоматизация разработки спецификаций, доказательства корректности спецификаций, их использование при генерации кода, автоматизация тестирования, средства обеспечения качества программного продукта, средства управления разработкой программного продукта, средства идентификации и информационной безопасности, управляющие и встроенные применения, беспроводная телекоммуникация, моделирование контроллеров и других аппаратно-программных решений, средства управления качеством реализации программного продукта, системы управления электронным документооборотом, безопасностью хранения данных, виртуализация вычислительных ресурсов.)

Рекомендуется особо отмечать передовые технологии, поддержанные в программных продуктах и оценку эффективности их применения.

С3. Программная инженерия: методы и алгоритмы теории программирования

Задача конкурса в разделе С3: продемонстрировать знания и умения в разработке и применении формальных методов при создании и/или улучшении либо оптимизации характеристик программного обеспечения на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ по сравнению с существующими.

(Например, алгоритмы и методы для проверки корректности программного продукта, исполнимые спецификации, формальные методы для применения в программировании и т.п.)

Оценка применимости предлагаемых подходов на практике и оценка эффективности применения.

С4. Технологии обработки и анализа больших массивов данных на основе работ EPAM

Задача конкурса в разделе С4: продемонстрировать знания и умения в разработке ПО с применением новых технологий EPAM в области обработки и анализа больших массивов данных, на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ.

С5. Подходы к разработке программного обеспечения на основе технологий Dell Technologies

Задача конкурса в разделе С5: продемонстрировать знания и умения в разработке ПО с применением новых технологий Dell Technologies. на основе доклада, содержащего обоснование предлагаемого решения и анализа преимуществ.

III. Порядок проведения презентаций и выступлений.

В процессе сбора материалов на конкурс-конференцию конкурсная комиссия отбирает лучшие работы для представления презентаций и выступлений, о чем участник конкурса ставится в известность.

Порядок проведения выступления на конкурсе-конференции:

1. В ходе конференции используется проекционное оборудование.
2. Для доклада при помощи Microsoft PowerPoint готовится презентация.
3. Презентация должна содержать не менее 10 слайдов. Обязательными являются следующие слайды:
   - Титульный, на котором должна быть представлена следующая информация:
     - Название проекта.
     - Фамилия и имя докладчика;
     - Учебное заведение, которое он представляет;
     - Фамилия И.О. научного руководителя.
   - Область применения проекта.
   - Какие задачи решаются проектом.
   - 2 слайда на описание концепции проекта.
   - 2 слайда на особенности реализации проекта.
   - Представление полученных результатов проекта.

- Перечень использованных для реализации продуктов (указать, какие технологии и для чего использовались в проекте).
- Заключение.

4. Длительность доклада не должна превышать 10 минут.

IV. Положение о системе награждения победителей конкурса – конференции.

Победителей конкурса определяет конкурсное жюри, в состав которого включены ведущие преподаватели вузов. Жюри оценивает представленные материалы и отбирает лучшие из них для участия в работе конференции. Работы будут премироваться следующим образом:

1. Тезисы докладов по всем отобранным комиссией проектам будут опубликованы в сборнике тезисов докладов конкурса-конференции.
2. Победители получат именные дипломы.

Тезисы докладов конкурса-конференции

Секция «Программная инженерия: приложения, продукты и системы»

УДК 004.4

Э. Г. Амирасланов (4 курс бакалавриата)
Т. А. Вишневская, ст. преподаватель

РАЗРАБОТКА TELEGRAM-БОТА ДЛЯ ПОИСКА И ПРОСМОТРА ВИДЕОМАТЕРИАЛОВ

Целью данной работы является создание бота на широко известной платформе Telegram. Бот должен обеспечивать взаимодействие между пользователем и сервером путём обмена командами, результатами их выполнения и информационными сообщениями. В его функционал должен входить следующий перечень возможностей:

- Вывод фильмов по запросам пользователя;
- Возможность просмотра фильма;
- Вывод списка кинотеатров (относительно геолокации пользователя);
- Добавление любимых фильмов в Избранное.

Аналогами данного бота являются такие сервисы, как:
Kinopoisk.ru - крупнейший русскоязычный интернет-сервис о кино.
Kinoafisha.info – сервис с полной информацией о кино и кинотеатрах России.

Сравним эти два сервиса между собой:

	Kinopoisk.ru	Kinoafisha.info
Информация о фильмах	Имеется	Имеется
Просмотр трейлера фильмов	Имеется	Имеется
Просмотр фильмов	Только по платной подписке	Не имеется
Покупка билетов	Имеется	Имеется
Приложение на Android и IOS	Имеется	Имеется
Список кинотеатров	Имеется	Имеется

Данные сервисы имеет обширную базу фильмов и сериалов для просмотра, чем и привлекают большинство пользователей.

Преимуществами Telegram-бота является его удобный интерфейс, быстродействие и малое потреблений интернет трафика. Написав боту, он предоставляет вам полную информацию о фильме, а также дает возможность перейти по ссылке для его просмотра. Отправив свою геолокацию, бот показывает ближайшие кинотеатры, в которых показывают новые фильмы.

Особенности реализации:

Telegram-бот реализован с помощью API - это встроенный интерфейс, который занимается отправкой компьютеру или ОС запросов на доступном ему языке. API упростил написание различных приложений и программ, так как приложения становились более модульными. Изменения могли вноситься в одну часть программы без нарушения в других ее частях.

Преимуществами API можно считать:

- Мобильность.
- Постоянство.
- Простота.

Проект реализуется на платформе Node.js с использованием фреймворка node-telegram-bot-api. Он служит для упрощения работы с Telegram API. Данный фреймворк – это некий интерфейс для работы с ботами. Грамотное распределение ресурсов помогает выдержать большее количество соединений к серверу. Такая модель идеально подходит создания бота. Имеются и другие преимущества данной платформы – скорость, синтаксис JavaScript, а также данная технология стремительно улучшается.

Для создания Telegram-бота идеально походит удобная база данных MongoDb. Она имеет гибкую структуру и отлично работает с большими объемами данных. В данной БД могут храниться данные различных типов. Проект работает с файлами JSON-формата и MongoDb идеально интегрируется с данным форматом, что и является её преимуществом.

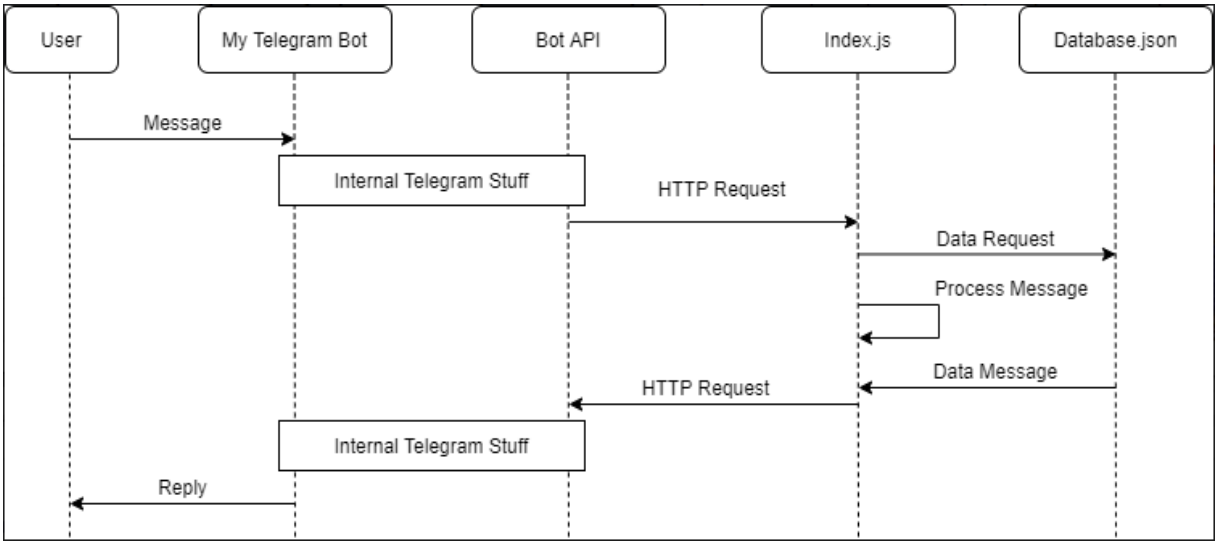


Рисунок 1 – Архитектура проекта.

ЛИТЕРАТУРА

1. Васильев, А.Н. JavaScript в примерах и задачах/ А.Н.Васильев. – М.: Эксмо, 2017. – 720 с.
2. Робсон, Э. Изучаем программирование на JavaScript/ Э.Робсон, Э.Фримен; пер. с англ. под общ. ред. Е.Матвеевой. – Санкт-Петербург: Питер, 2016. – 620 с.
3. Салмре И. Программирование мобильных устройств на платформе .Net Compact Framework/ И.Салмре. – М.Вильямс, 2006. Кооперайт на английском языке – Pearson Education, Inc., 2005.
4. Андриянова С.С. Использование мессенджера Telegram для продвижения бренда/ С.С.Андриянова, А.А.Веретено// Economics. – 2018. – с. 54-56.
5. Матвеева, Н.Ю. Технологии создания и применения чат-ботов/ Н.Ю. Матвеева, А.В. Золотарюк// Научные записки молодых исследователей. - 2018. - №1. - С.28-30.

УДК 004.9

И. А. Васеха, О. Д. Субцельная (3 курс бакалавриата)
И. В. Зайцев, ст. преподаватель

О ПРИМЕНЕНИИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ДЛЯ СНИЖЕНИЯ ЗАТРАТ ВРЕМЕНИ ПРЕПОДАВАТЕЛЕЙ ПРИ ИСПОЛЬЗОВАНИИ «МНОГОМЕРНОЙ» БАЛЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЫ.

Балльно рейтинговая система при творческом подходе позволяет повысить эффективность процесса обучения, в частности, за счет повышения мотивированности студентов более равномерно распределять усилия в течение семестра, готовиться к каждому занятию, выполнять домашние задания. Для достижения этой цели требуется следующее:

1. сформировать правила, дающие возможность студентам увеличивать свой рейтинг путем приложения усилий и достижения результатов в нужных для освоения материала курса направлениях. Данные правила должны обеспечивать приблизительную сбалансированность усилий, затрачиваемых студентами и соответствующий прирост баллов их рейтингов. Для оценки сложности выполнения заданий можно использовать накопленные данные и современную теорию тестирования (Item Response Theory).
2. обеспечить прозрачность формирования рейтингов студентов (предоставить студентам информацию для понимания, за что начислены или не начислены баллы) — студенты не должны сомневаться в «справедливости» формирования рейтинга, иначе их мотивация может снижаться.

Прозрачность предполагает возможность для студентов посмотреть подробную историю начисления собственных баллов, а открытость историй всех студентов может дополнительно увеличить объективную и субъективную справедливость применяемой рейтинговой системы.

Субъективная (воспринимаемая) справедливость, как показывает практика, важна для обучающихся и способствует повышению мотивированности студентов в учебном процессе.

Кроме того, прозрачность помогает быстро устранять возможные ошибки, благодаря большому числу «заинтересованных проверяющих».

И наконец, опыт работы в условиях прозрачной системы может быть полезен студентам в будущем в профессиональной деятельности для эффективной и коррупционно-безопасной организации работ в других сферах деятельности.

Для устранения эффекта потери мотивации отставших студентов предлагается применить «многомерную» балльно-рейтинговую систему, в которой баллы по каждой изучаемой теме фиксируются и дают эффект для итогового зачета или экзамена отдельно. При таком подходе студенты, по каким-то причинам набравшие мало баллов в одной теме, сохраняют все возможности на успех в следующей теме, и это положительно влияет на их мотивацию.

Однако подобный учет, обработка и публикация подробных результатов для участников учебного курса требует от преподавателя значительных затрат времени, которого обычно катастрофически не хватает, что тормозит внедрение «многомерной» балльно-рейтинговой системы. Логичным выходом является создание системы, позволяющей удобно фиксировать (с помощью мобильного устройства) успехи студентов, полученные на очных занятиях, автоматически получать данные из систем дистанционного обучения, например, с помощью HtmlEdit или Selenium ChromeDriver, объединять их по заданным правилам и публиковать. Дополнительной функцией системы, снижающей временные затраты преподавателей, может быть автоматическая генерация отчетов о текущей успеваемости для деканатов, то есть аттестестации. Ключевым моментом является обеспечение прозрачности начисления баллов для студентов путем публикации не только агрегированных текущих результатов, но и данных, из которых они получаются. Система должна иметь гибкую систему конфигурации, позволяющую без перекомпиляции настраивать работу с конкретными системами дистанционного обучения, алгоритмы обработки получаемых результатов, правила

формирования отчетов. В дальнейшем файлы конфигурации могут создаваться пользователями-преподавателями и удачные варианты правил и конфигураций, хорошо зарекомендовавшие себя в смысле увеличения мотивации студентов и удобства преподавателей, могут тиражироваться для экономии времени на настройку системы.

В частности, мотивации способствует большая «линейность» бонусов по результатам итогового рейтинга. Если студент получает бонусы при достижении какого-то уровня, то у тех, у кого мало шансов достичь данного уровня, может снижаться мотивация.

Варианты правил и конфигурации системы могут учитывать специфику преподавания различных дисциплин.

В случае широкого применения такой системы появляется дополнительная полезная возможность для статистических исследований и оптимизаций, на основе создаваемых в ходе учебного процесса данных.

Сама по себе система может состоять из нескольких (веб-)приложений, связанных между собой посредством веб-служб:

1. Веб-приложение учета оффлайновых успехов и посещаемости студентов с помощью мобильных устройств. Основные функции: предоставление возможности удобно фиксировать в базе данных посещаемость и успехи студентов с помощью веб-браузера мобильного устройства и предоставление данной информации через веб-службу другим приложениям для обработки, публикации и формирования отчетов.
2. Веб-приложение, объединяющее (регулярно, например, раз в сутки) данные оффлайнового учебного процесса с данными систем дистанционного обучения (с помощью веб-служб и/или вышеупомянутых Selenium ChromeDriver или HtmlEdit), если они также применяются в учебном процессе, например, для домашних заданий или тестирования во время очных занятий.
3. Веб-приложение, обеспечивающее обработку и публикацию рейтингов студентов, а также формирование отчетов о текущей успеваемости на основе данных, полученных через веб-службы от приложений 1 или 2 (приложение 2 требуется не для всех учебных курсов).
4. Другие приложения, компенсирующие недостатки существующих зарубежных систем типа Moodle в российских условиях (см. 4).

Очевидно, что каждое из веб-приложений должно иметь информацию о списках групп, учебных курсах и преподавателях, имеющих право начислять баллы, которую естественно хранить в центральной базе данных. С другой стороны, необходимость обеспечения взаимодействия с существующими в вузах информационными системами способна сильно замедлить внедрение обсуждаемой системы по техническим и бюрократическим причинам. Поэтому для ускорения распространения желательно обеспечить возможность независимого применения такой системы даже единичными преподавателями-энтузиастами (которые могут сами загрузить списки «своих» групп), для чего следует предусмотреть вариант «автономного» использования необходимых в конкретном случае приложений (то есть без специальной (требующей согласований и официальных решений) организации подключения к информационным системам вуза). При этом для автоматизации обработки баллов вполне достаточно настройки подключения к системе дистанционного обучения вуза с правами и учетными данными конкретного преподавателя, полученных в ходе работы студентов с онлайн-заданиями по конкретному курсу.

Целесообразность разработки такой системы можно обосновать возможностью более широкого внедрения многомерной балльно-рейтинговой системы для повышения мотивации студентов к регулярной работе за счет более подробного и прозрачного учета их текущих успехов по освоению учебного материала.

ЛИТЕРАТУРА

1. http://free-elearning.ru
2. http://moodle.org
3. http://olat.org

4. Функциональные лакуны в информационных системах поддержки образовательного процесса. Петров Д.А., Зайцев И.В., Сборник докладов конференциии ИКНТ «Современные технологии в теории и практике программирования» Санкт-Петербург, 2018

УДК 004.623

П. В. Елисеев (2 курс бакалавриата)
О. Н. Петров, к.т.н., доцент

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ГАЛЕРЕИ УНИКАЛЬНЫХ ИЗОБРАЖЕНИЙ

Целью работы является создание системы, позволяющей собирать большое количество изображений тематического характера в автоматизированном режиме.

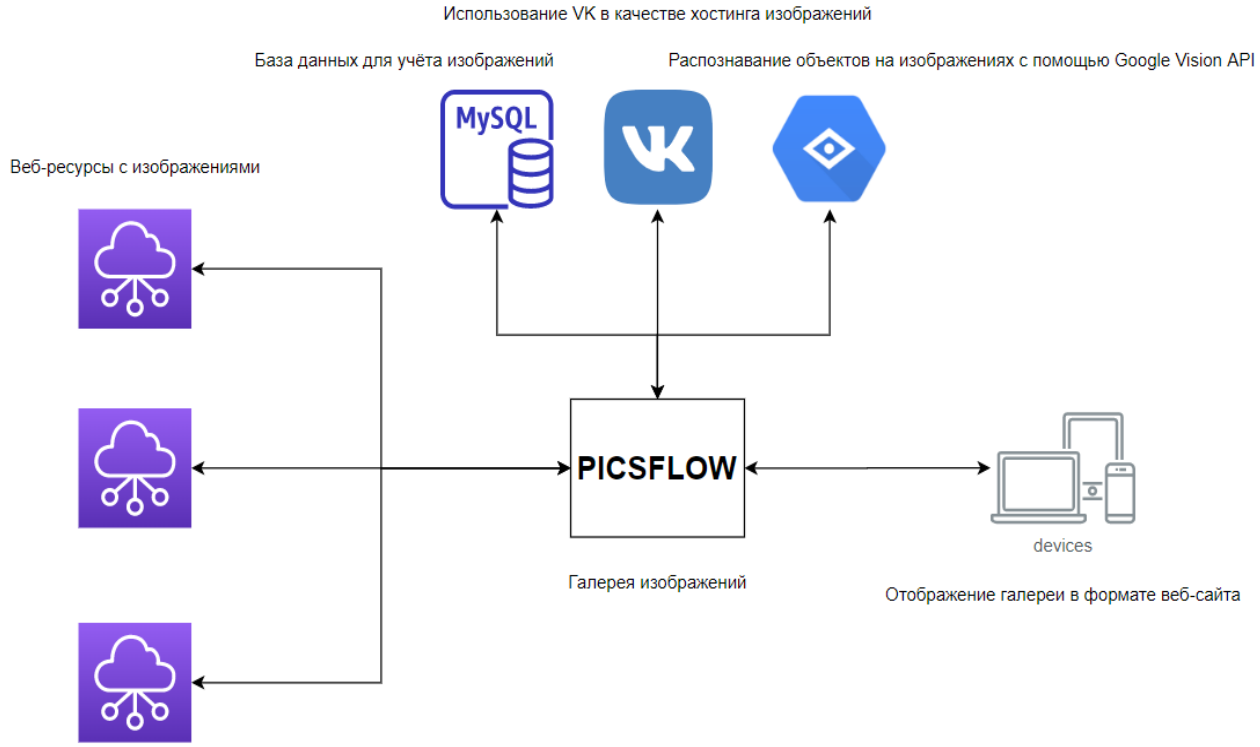


Рисунок 1 – Схема работы Галереи изображений.

Галерея изображений работает по следующему алгоритму: сначала вручную указываются источники изображений (в формате URL) и вносятся различные настройки, как, например, ключ для Google Vision API, данные аккаунтов VK, настройки модулей веб-приложения и т.д., затем настраивается Cron-задача, которая запускает входной скрипт, распределяющий доступное время ожидания при вызове. Затем происходит решение 3-х основных задач:

1) парсинг веб-страниц на наличие изображений;
2) валидация и учёт найденных изображений;
3) сохранение изображений на серверах Вконтакте.

Таким образом, при создании данного проекта, были выявлены и решены проблемы хранения большого количества изображений, индексации изображений и игнорирования дубликатов, автоматического создания тегов, а также в целом проблемы создания системы, работающей циклично и независимо от моей локальной машины.

Решить эти проблемы удалось с использованием следующих технологий:

1) VK API

2) Google Vision API

3) Cron

4) PHP OOP

5) Perceptual hashing

6) CMS Wordpress

УДК 004.415.25

С. Д. Елисеев (3 курс бакалавриата)
Т. В. Леонтьева, к.т.н., доцент

ПРОГРАММНЫЕ СРЕДСТВА ПРОСМОТРА, АНАЛИЗА И ОБРАБОТКИ СИГНАЛОВ МАГНИТНОГО ПОЛЯ ДЛЯ НЕРАЗРУШАЮЩЕГО КОНТРОЛЯ ТРУБОПРОВОДОВ.

На сегодняшний день сеть магистральных трубопроводов покрывает 35% территории России, на которой проживает около 60% населения, пересекая автомобильные и железные дороги, водоемы и лесные массивы, иногда проходя в непосредственной близости от населенных пунктов [1]. При этом крайне важное значение имеет обеспечение надежности и безопасности магистральных трубопроводов. Для диагностики нефтетрубопровода используют методы неразрушающего контроля. Неразрушающий контроль (НК) – это совокупность таких видов контроля, которые производятся непосредственно на объекте, при этом исправный объект сохраняет работоспособность без какого-либо повреждения материала.

Основными методами неразрушающего контроля являются [2]:
- магнитный;
- вихретоковый;
- радиоволновой;
- оптический;
- акустический (ультразвуковой);
- радиационный;
- тепловой;
- электрический;
- проникающими веществами.

В качестве метода НК предлагается использовать анализ сигналов магнитного поля, на основе которых можно определить состояние трубы и наличия в ней дефектов. Суть метода состоит в следующем: с помощью прибора записываются показания датчиков магнитного поля во время его движения вдоль трубы. При этом устройство может нести оператор или оно может двигаться внутри трубы вместе с потоком нефти. Данные заносятся в файл, который затем передаётся для обработки и анализа сигналов. Для этого используется программа обработки, которая читает содержимое файла и позволяет строить графики сигналов, применять различные фильтры и выявлять возможные дефекты трубопровода. Например, на рисунке 1 представлен график сигнала магнитного поля, записанный на трубе длинной 27 метров.

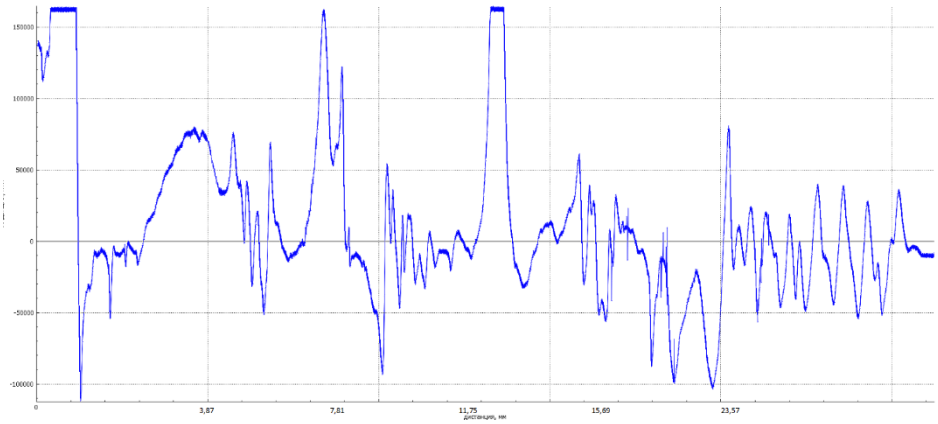


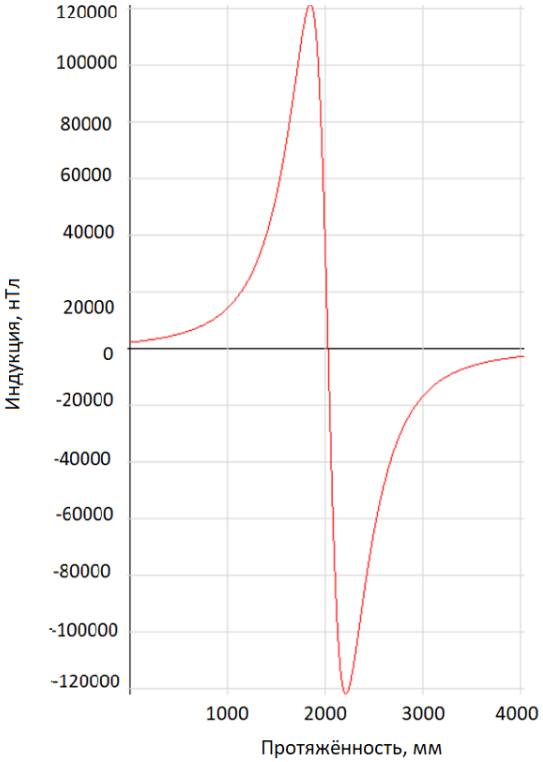
Рисунок 1 – Сигнал магнитного поля.

Разрабатываемая программа предназначена для использования в ОС Windows 7 и ОС Windows 10, требованиям к языку является возможность компиляции программы на данных операционных системах, о также возможность реализации пользовательского графического интерфейса. Язык c# удовлетворяет вышеперечисленным условиям, и он был выбран в качестве основного языка программирования. В качестве библиотеки векторной графики была выбрана SharpGL, так как она интегрирована в данный язык.

Процесс анализа сигнала заключается в сравнении графиков полученных сигналов с эталонной моделью сигнала. Например, в качестве такой модели может использоваться формула Ферстера [3]

$$H_y = m\left(\frac{x}{x^2 + y^2} - \frac{x}{x^2 + (y + d)^2}\right)$$

В формуле H_y – поперечная компонента индукции магнитного поля. Глубина дефекта и расстояние до него – d и y. На рисунке 2 представлен график поперечной компоненты при d = 3 и y = 2.



Риунок 2 – Поперечная компонента индукции магнитного поля [4]

Метод эффективен для определения ручейковой коррозии, когда происходит локальное утончение стенки трубы, и в некоторых местах появляются коррозионные отверстия.

На данный момент в программе реализованы следующие функции:

- Построение графиков на основе данных, полученных из файла
- Масштабирование и просмотр части графика
- Возможность производить стандартные арифметические операции над сигналами
- Применения фильтров к сигналу (Баттерворта, Чебышева и эллиптический)
- Построение развёрток, на основе изолиний, по данным от нескольких датчиков, образующих кольцо.

ЛИТЕРАТУРА

1. Мазур И.И., Иванцов О.М. и др. Безопасность России. Правовые, социально-экономические и научно-технические аспекты. Безопасность трубопроводного транспорта. М.: МГФ «Знание», 2002. – 629 с.
2. ГОСТ 18353-79. Контроль неразрушающий. Классификация видов и методов
3. Zhang Y., Sekine K., Watanabe S. Magnetic leakage field due to sub-surface defects in ferromagnetic specimens. NDT & E International, 1995, 28(2), p.67–71
4. Suresh V., Abudhahir A, Jackson Daniel. Determination Scheme for Accurate Defect Depth in Underground Pipeline Inspection by Using Magnetic Flux Leakage Sensors.- IEEE Transactions on Magnetics, 2018

УДК 004.415.04

В. О. Ерёменко (4 курс бакалавриата)
С. А. Молодяков, д.т.н., профессор

УПРАВЛЕНИЕ УМНЫМ ДОМОМ С ПОМОЩЬЮ ДИЛОГОВЫХ КОМАНД ГОЛОСОВОГО УПРАВЕНИЯ

С каждым годом увеличивается количество устройств в системах Умный Дом. Они находят все более широкое применение. Одним из способов управления системами является голосовое управление. Поэтому разработка и анализ систем и алгоритмов голосового управления является актуальной. В настоящее время используются следующие системы для управления голосом: Google Speech Recognition API, Siri , Amazon Transcribe, IBM Speech to Text, Yandex Speech Kit, Dragon Mobile SDK, Microsoft Speech API. Они способны решать огромное количество задач и освобождают разработчика от необходимости создавать сложную и комплексную систему автоматического распознавания речи. Например, распознавание речи с помощью проверенных временем алгоритмов, демонстрирующих высокие результаты, можно реализовать с использованием высокопроизводительных вычислительных ресурсов облачных систем. Облачные платформы предоставляют отказоустойчивый программный интерфейс приложения (API, Application Programming Interface) для распознавания речи, а также имеют экосистемы с большим количеством пользователей, куда легко встраивать собственные приложения с голосовым интерфейсом.

Для работы с голосовыми ассистентами необходимо подключение к сети Интернет для обработки получаемых данных в «облаке», а также для синхронизации и управления устройствами пользователей. Огромный практический интерес могут иметь голосовые ассистенты, которым не требуется наличие подключение к сети Интернет. Был проведен анализ некоторых облачных систем.

Система Microsoft Speech API ориентирована на платформу NET и соответственно на использование в приложениях, написанных под операционную систему Windows. Другие облачные сервисы имеют готовые SDK и использующие платформу Java. Такие сервисы, как Dragon Mobile SDK и Yandex Speech Kit, имеют в наличии SDK, представленные на официальных сайтах разработчиков. Облачный сервис Google Speech Recognition API также имеет в наличии готовый SDK, но написанное сторонним разработчиком(JARVIS). Такой SDK имеет ряд недостатков, таких как отсутствие официальной поддержки, но также и ряд

преимуществ, связанных с открытым исходным кодом, который может быть модифицирован под нужды разработчика. Так же можно рассмотреть наиболее популярную систему распознавания речи. Такой продукт как Dragon Mobile SDK имеет хорошую документацию, с менее сложным кодом для встраивания чем у других, поэтому и является более удобной для её использования. Тем не менее система имеет не простые лицензионные правила в использовании что усложняет её внедрение.

Наиболее широкие возможности для голосового управления имеет Google Speech Recognition API, за счет больших сложно вычислительных систем корпорации Google, облачная система имеет отличную встраиваемость и быстродействие. Одно из огромных преимуществ в использовании этого сервиса является наличие в ней фильтрации и подавление внешних шумов, что помогает увеличить точность распознавания голоса. Для преобразования голосовой команды в текст, посредствам данной системы, нужно записать аудиофайл в формате одноканальной записи (моно), используя такой формат файла аудиофайлов FLAC (свободный кодек, предназначенный для сжатия без потерь) и частотой дискретизации 8кГц. Поле чего данный аудиофайл может быть отправлен на серверные системы Google для дальнейшего преобразования в текст. Одна из важнейших задач вышеуказанной библиотеки JARVIS является поиск начала и так же конца фразы в беспрерывном звуковом потоке. С помощью стандартов этой библиотеке можно реализовать простейший алгоритм поиска тех самых нужных моментов, в которых необходимо начать запись аудиофайла и закончить её для дальнейшей отправки в облачный сервис. Суть алгоритма в том, чтобы вести постоянное отслеживание уровня сигнала, а точнее его громкости: в том случае если уровень сигнала превысил некоторое граничное значение, то это значит, что необходимо начать запись, если происходит обратное действие, то есть уровень сигнала становится ниже некоторого значения, необходимо прекратить запись и отправить аудио на обработку.

Так же у этого алгоритма есть и свои недостатки, такие как ложные срабатывания и возникновение внешних шумов (всесторонний окружающий звук) а также голоса посторонних людей, которые не зарегистрированы в системе Умного дома. Однако есть решение этой проблемы, а именно, использование полосового фильтра, который позволяет фильтровать источники внешних шумов. А благодаря открытости исходного кода данной библиотеки, не возникнет никаких проблем с данными изменениями.

В данной работе разрабатывается приложение для управления Умным домом, которые будет написано на платформе Java в нем используются основные функции Google Speech Recognition API для распознавание голосовых команд, так же для объединения устройств Умного дома будет использована протокол MQTT. Индивидуальность этого приложения заключается в том, что в данной программе можно использовать сложные диалоговые команды, в отличие от простых других наработанных фраз. Приложение можно обучать, добавляя вручную новые сложные команды (состоящие из нескольких слов или целого предложения) а данная программа распознает ту или иную команду и произведет взаимодействие с девайсами Умного дома. Основная концепция заключается в том, чтобы объединить все устройства Умного дома, имеющие подключение к сети Интернет и управлять ими с помощью голосовых команд.

ЛИТЕРАТУРА

1. Беленко М.В., Балакшин П.В. Сравнительный анализ систем распознавания речи с открытым кодом // Международный научно-исследовательский журнал. 2017. № 4-4. С. 13–18. doi: 10.23670/IRJ.2017.58.141
2. Дикий Д.И., Артемьева В.Д. Протокол передачи данных MQTT в модели удаленного управления правами доступа для сетей интернета // Научно-технический вестник информационных технологий, механики и оптики. 2019. Т. 19. № 1. С. 109–117. doi: 10.17586/2226-1494-2019-19-1-109-117

УДК 519.872.4

Е. И. Жарко (2 курс магистратуры)
И. Г. Черноруцкий, д.т.н., профессор

РАЗРАБОТКА ПРОГРАММЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ ЕДИНИЧНЫХ И МЕЛКОСЕРИЙНЫХ ПРОИЗВОДСТВ

Технологическая подготовка производства является одним из важнейших этапов процесса по выпуску изделий. В рамках такой подготовки необходимо построить оптимальный технологический маршрут обработки, выбрать параметры обработки и средства технологического оснащения. От сложности детали данный процесс может занять значительную часть всего производственного цикла.

На данный момент, автоматизация производственных процессов является одним из наиболее востребованных направлений оптимизации производственных процессов, так как способствует повышению эффективности работы и значительно увеличивает конкурентоспособность предприятия [1].

Как правило, автоматизация производства и технологической подготовки сводятся к типизации конструкторских элементов и технологических процессов [2, 3], хотя в условиях единичных и мелкосерийных производств данный подход может показывать себя не эффективно вследствие больших затрат на выполнение подготовительных работ [4].

Имитационную модель технологической подготовки производства можно представить в виде трехуровневой модели с сетецентрическим управлением [5]. В рамках данной работы, рассматривается третий уровень модели (далее модель) на котором производится моделирование и анализ различных вариантов технологических маршрутов обработки и применяемых средств технологического оснащения для каждой партии деталей.

Рассматриваемая модель реализована как система массового обслуживания [6]. В процессе моделирования генерируются заявки – отдельный этап технологического процесса. Одновременно в системе обрабатывается один экземпляр каждого технологического процесса. Таким образом, количество заявок в системе не превышающее общее число технологических процессов.

Основная задача модели состоит в выборе оптимального сценария работы производственного участка по заданным данным. Для решения этой задачи, был построен генератора реализаций технологических процессов по изготовлению заданной номенклатуры изделий на заданном оборудовании (далее ГРТ). Основная задача ГРТ – генерация наибольшего числа возможных вариантов реализации.

Генерация различных реализаций происходит за счёт перебора всех возможных приоритетов операций обработки заданных технологических процессов. Приоритеты задают последовательность выбора заявок из источников, то есть деталей, соответствующих определённым технологическим процессам, а также последовательность их обработки на заданном оборудовании (станках).

Входом модели является набор технологических процессов – список из пар (тип операции и время реализации) и количество реализаций каждого из них (Таблица 1), а также набор станков и их количество. Каждому станку соответствует один или несколько типов выполняемых операций (Таблица 2).

Таблица 1 – Входные данные модели для заданного технологического процесса

Технологический процесс
(:device 0:task ((:process "ТР-Т":line (("op4" 40) ("op3" 12) ("op5" 16) ("op6" 29) ("op2" 18) ("op3" 12) ("op2" 36) ("op4" 100) ("op2" 18) ("op3" 12) ("op2" 18) ("op3" 12) ("op5" 16) ("op2" 18) ("op3" 12) ("op4" 40) ("op3" 12) ("op2" 18) ("op1" 32)):) 1):)
Пояснения
(:device id технологического процесса: task ((:process "имя технологического процесса": line (("операция" время выполнения)) :) количество деталей) :)

Таблица 2. Входные данные модели для заданного оборудования

Оборудование
(:consumer 0:worker (:machine "mach1":line ("op1" "op2" "op3" "op4" "op5"):):) (:consumer 1:worker (:machine "mach2":line ("op1" "op2" "op3" "op4" "op5"):):) (:consumer 2:worker (:machine "mach3":line ("op1" "op2" "op3" "op4" "op5"):):) (:consumer 3:worker (:machine "mach4":line ("op6"):):)
Пояснения
(:consumer id станка:worker (:machine "название станка":line ("тип выполняемой операции") :) :)

Результатом работы модели является оптимальный сценарий работы производственного участка по заданным данным (Таблица 3). Сценарий представляет собой последовательный список из этапов обработки детали с указанием места, времени начала и конца обработки.

Таблица 3. Пример выходных данных модели

Оптимальный сценарий работы
... (:id (:name TP-T:num 0:step 1:type op3:):birth 40:wait 40:executor (:name mach1:num 0:execution 52:):) (:id (:name TP-T:num 0:step 2:type op5:):birth 52:wait 52:executor (:name mach1:num 0:execution 68:):) (:id (:name TP-T:num 0:step 4:type op2:):birth 97:wait 97:executor (:name mach1:num 0:execution 115:):) ...
Пояснения
(:id (:name имя технологического процесса:num номер детали:step текущий шаг реализации:type тип операции:):birth время появления заявки:wait время постановки заявки:executor (:name название станка:num номер детали:execution время окончания операции:):)

Таким образом, была разработана программа имитационного моделирования для единичных и мелкосерийных производств. Построенная программа позволяет автоматизировать процесс технологической обработки путём автоматической генерации наибольшего числа сценариев работы и дальнейшего выбора оптимального сценария работы производственного участка по заданным данным. Дальнейшая работа направлена на внедрение в модель возможность выбора оптимального сценария работы по нескольким критериям.

ЛИТЕРАТУРА

1. Stephen Kalas. Small-Scale Automation in Shipbuilding. Norwegian University of Science and Technology, 2015. 92 p.
2. Свирский Д.Н., Климентьев А.Л. Автоматизация принятия технологических решений в компактном производстве машиностроительной продукции // Вестник полоцкого государственного университета. Серия В: Промышленность. Прикладные науки. 2010. № 2. С. 54—62.
3. Трусов А.Н. Методика выбора оптимального варианта автоматизации конструкторскотехнологической подготовки машиностроительного производства // Вестник Кузбасского государственного технического университета. 2012. № 1(89). С. 81—83.
4. Жуков Э.Л., Козарь И.И., Мурашкин С.Л., Розовский Б.Я., Дегтярев В.В., Соловейчик А.М. Основы технологии машиностроения. Производство деталей машин / Под общ. ред. С.Л. Мурашкина. СПб.: Изд-во Политехн. ун-та, 2008. 412 с.
5. V. Kotlyarov, I. Chernorutsky, P. Drobintsev, A. Tolstoles, I. Khrustaleva, L. Kotlyarova Net-centric Internet of Things for industrial machinery workshop // Proceedings of the 4th Ural Workshop on Parallel, Distributed, and Cloud Computing for Young Scientists. 2017. 112-122 pp.
6. Александрова О. В., Котляров В. П., Федотова Д. А. Архитектура вычислительных систем. Методические указания для курсового проектирования – 2013, электронное издание.

УДК 004.4'22

Е. Е. Зайнуллин (4 курс бакалавриата)
Ю. В. Литвинов, к.т.н, доцент

РЕАЛИЗАЦИЯ ОБРАЗОВАТЕЛЬНОЙ СРЕДЫ ПРОГРАММИРОВАНИЯ НА ОСНОВЕ ПЛАТФОРМЫ REAL.NET

Целью работы является создание набора графических языков для программирования исполнителей[1], таких как, например, Лого [1], а также среды разработки для них. Создание такой системы позволит понизить порог вхождения и возраст детей для обучения их базовым принципам программирования, а также развить абстрактное мышление и способности к творчеству. Идеей этой платформы заинтересовался 419 лицей Санкт-Петербурга и предложил нашему Университету заняться ее разработкой.

В частности, для создания системы такого рода необходимо для каждого типа исполнителя создать собственный язык, который позволял бы использовать базовые команды исполнителя, переменные, циклы, условные операторы. Также учителя лицея попросили реализовать визуализатор для каждого исполнителя: Черепашки, Чертежника и Робота.

– Черепашка может ездить вперед и назад, а также поворачивать на угол, чертить линию при перемещении;

– Чертежник рисует фигуры по заданным координатам;

– Робот умеет ездить по клеткам клетчатого прямоугольника, передвигаясь на каждом ходу в соседнюю.

На данный момент существуют несколько сред для разработки программ для исполнителей, рассмотрим некоторые их них.

– FMSLogo [2] – редактор с компилятором и визуализатором для Лого. Имеет простой текстовой редактор, поддерживает многопоточность для работы с несколькими черепашками, также доступна документация на английском с примерами. Также возможна локализация самого языка с помощью расширений;

– ЛогоМиры [3] – интегрированная среда разработки для программирования на языке Лого. Редактор подсвечивает синтаксис, позволяет производить отладку по шагам. Можно программировать полноценное приложение с развитым графическим интерфейсом, есть подробная справка и примеры, кроссплатформенна и полностью на русском языке. Работает только с Черепашкой, является проприетарным проектом;

– КуМир [4] – система для программирования исполнителей. Проект с открытым исходным кодом, разрабатываемый РАН. Имеет свой собственный язык, поддерживает множество исполнителей и позволяет добавлять своих собственных. Полностью на русском языке, обладает хорошей документацией. В качестве недостатка стоит выделить сложность предоставленного языка для детей.

В основе проекта лежит DSM-платформа[2] REAL.NET [5] для упрощения создания языков для исполнителей и редактора для них. REAL.NET написана на .NET и для графического интерфейса использует подсистему Windows Presentation Foundation (WPF).

В данный момент работа идет по созданию визуализатора и графического языка для Черепашки. Для визуализатора надо было создать приложение, которое бы отображало поведение Черепашки на экране в соответствии с базовыми командами: движение взад-вперед, поворота на заданные углы, поднятие и опускание пера для чертежа.

Визуализатор – сцена, на которой находится Черепашка. Клиент управляет Черепашкой посредством класса, который запоминает положение исполнителя, вызывает событие о том,

[1] Абстрактный робот, умеющий выполнять команды
[2] Инструмент для быстрого создания визуальных языков

что необходимо начать движение. После того, как Черепашка перестанет двигаться, значение положения обновится и об этом узнает клиент.

Язык состоит из базовых команд исполнителя и цикла. Также есть возможность создать переменную и присвоить ей значение, посчитав арифметическое выражение.

Для апробации выполнены программы, строящие простейшие геометрические фигуры (рисунок 1 и рисунок 2).

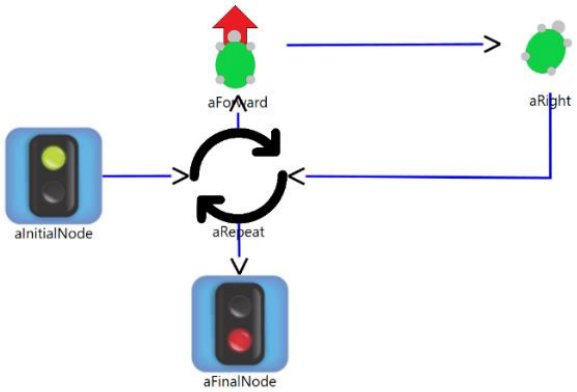


Рисунок 1 – Программа для Черепашки.

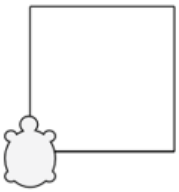


Рисунок 2 – Результат выполнения

На данный момент сделан язык для исполнителя Черепашка, а также визуализатор, анимирующий его движения.

ЛИТЕРАТУРА

1. Wikipedia. Лого (язык программирования // Википедия. – URL: https://ru.wikipedia.org/wiki/Лого_(язык_программирования) (дата обращения: 14.12.2019).
2. FMSLogo // Internet. – URL: http://fmslogo.sourceforge.net/manual (online; accessed: 14.12.2019).
3. Пирогова Е.Ю. Создание проектов в интерактивной творческой среде ЛогоМиры. – ВИРО, 2013. – ISBN: 978-5-87590-398-4. –http://viro.edu.ru/attachments/article/2495/LogoMiry.pdf.
4. Леонов А. Г., Первин Ю. А. Переход от непосредственного управления исполнителями к составлению программ в пропедевтическом курсе информатики // Ярославский педагогический вестник. – 2013. – Т. 3. – С. 20 –30. – https://cyberleninka.ru/article/n/perehodot-neposredstvennogo-upravleniya-ispolnitelyami-k-sostavleniyuprogramm-v-propedevticheskom-kurse-informatiki/viewer.
5. Литвинов Ю.В., Кузьмина Е.В., Небогатиков И.Ю., Алымова Д.А. Среда предметно-ориентированного программирования REAL.NET // Список. — 2017. — URL: https://github.com/yurii-litvinov/articles/blob/master/2017-realNet/realNet.pdf (дата обращения: 09.12.2018).

УДК 004.416.6

В. А. Ивлев (2 курс магистратуры)
И. В. Никифоров, к.т.н., доцент

АНАЛИЗ, ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПОДХОДА ПО СОВЕРШЕНСТОВАНИЮ ГЕОИНФОРМАЦИОННОЙ СИСТЕМЫ «СПУТНИК ЛЭП»

Область применения геоинформационных систем [1] (далее ГИС) имеет широкий спектр задач, которые в свою очередь определяют тип проектируемой ГИС [2]. Например, для задач, требующих больших вычислительных мощностей, используются настольные или серверные ГИС [2], одной из которых является ГИС «Спутник ЛЭП», принадлежащая ГК «Геоскан» [3]. Данный продукт является производным от семейства ГИС «Спутник» [4] и создан для решения задач по автоматизации расчета данных для обслуживания линий электропередач (далее ЛЭП). Обслуживание включает в себя расчистку территории вокруг ЛЭП от угрожающих ей лесных массивов и древесно-кустарниковой растительности (далее ДКР).

Актуальность задачи автоматизации расчета данный для обслуживания ЛЭП обоснована рядом факторов:

1. Территория Российской Федерации огромная и общая протяженность эксплуатируемых ЛЭП составляет около 3 млн. км. [5];
2. В некоторых районах расстояние между населенными пунктами может достигать 100 – 1000 км.

Исходя из вышеперечисленных факторов встает задача об обслуживании состояния ЛЭП. Существующий подход визуального осмотра экспертом с дальнейшими расчетами требует огромных кадровых затрат, а также является некорректным ввиду высокой неточности и возможной потасовки предоставляемых данных.

Подход с получением цифровой модели рельефа [6] (далее ЦМР) и ЛЭП с использованием метода фотограмметрии [7] при помощи беспилотного летательного аппарата (далее БПЛА), построением модели на основе облака точек [8] с использованием программного обеспечения (далее ПО) Agisoft Metashape [9] и дальнейшем расчете данных на основе полученной модели в ГИС «Спутник ЛЭП» — автоматизирует процесс контроля за состоянием ЛЭП, значительно снижает кадровые затраты и снижает вероятность ошибки с участием человеческого фактора.

Исходя из этого актуальным является разработка и улучшение качества существующей ГИС «Спутник ЛЭП». Для выявления недостатков был проведен анализ существующей системы и выявлен ряд недостатков (см. таблицу 1):

Таблица 1 – Недостатки ГИС «Спутник ЛЭП»

Недостатки	**Описание**
Производительность системы требует улучшения	Множество расчетов не переиспользуются, а выполняется заново
Нет единой сборки продукта	Для каждой теплоэлектроцентрали (далее ТЭЦ) используется собственная сборка ПО
Отсутствует покрытие модульным и интеграционным тестированием	Повышен риск отказа ПО, а также его нестабильность
User experience design (далее UxD) требует нового подхода	После проведения исследования среди специалистов, использующих данный продукт было выявлено, что ряд интерфейс решений неудобен или не используется
Отсутствует непрерывная интеграция (далее CI) продукта	Нет автоматической поставки продукта конечному пользователю

После детального разбора каждого пункта был смоделирован текущий процесс работы ГИС «Спутник ЛЭП», а также спроектирован подход, реализация которого увеличит

производительность работы алгоритмов и повысит качество ПО за счет увеличения процента покрытия кода модульными и интеграционными тестами.

Далее спроектированный подход успешно внедрен в исходных код ГИС «Спутник ЛЭП». После этого был проведен анализ качества Quality assurance (QA) инженером. Ниже представлены результаты:

1. Повышение производительности в 6 раз происходит за счет переиспользования ранее посчитанных данных, которые хранятся в XML подобном файле с расширением *.ptl;
2. Т.к. данные для каждого пролета ЛЭП независимы, то было принято решение выполнять расчеты параллельно, что значительно уменьшает временные затраты и увеличивает производительность экспоненциально в зависимости от количества обрабатываемых данных;
3. После глобального перепроектирования кода индивидуальные расчеты ТЭЦ переопределяются в собственных реализациях классов, что делает возможным создание единой сборки ПО;
4. Проведение статического анализа кода с дальнейшим покрытием его модульными и интеграционными тестами до 80% значительно увеличило качество кода и избавило ПО от сбоев;
5. Проведение UxD анализа выявило свыше 17 недостатков в интерфейсе пользователя (речь идет только о интерфейсе для расчета данных ЛЭП), исправление которых ускорило работу специалиста при обработки данных ЛЭП;
6. Реализация для продукта CI дало возможность своевременно предоставлять специалистам последние сборки продукта, имеющие все актуальные изменения.

Опираясь на результаты, изложенные в вышеперечисленных пунктах можно сделать вывод, что проведение анализа по выявлению недостатков ГИС «Спутник ЛЭП» и проектирование на его основе подхода повысило производительность и качество системы до уровня, допустимого для предоставления его конечному пользователю.

ЛИТЕРАТУРА

1. Бугаевский, Лев Моисеевич. Геоинформационные системы: Учебное пособие для вузов / Бугаевский Л. М., Цветков В. Я. Москва : «Златоуст», 2000 – 224с. : ил. 28. ISBN 5-7259-0057-3.
2. В. А. Ивлев, И. В. Никифоров. Обработка данных в геоинформационных системах для выбора местоположения рекламы. (с. 27) / Современные технологии в теории и практике программирования: сборник материалов конференции, 19 апреля 2019 года / Санкт-Петербургский политехнический университет Петра Великого, [Институт компьютерных наук и технологий [и др.] ; Санкт-Петербург : [ПОЛИТЕХ-ПРЕСС], 2019. 247 с.: ил., табл. ; 20 см. ISBN 978-5-7422-6508-5.
3. Официальный сайт ГК Геоскан [Электронный ресурс] URL: https://www.geoscan.aero/ru (дата обращения 17.03.2020 г.).
4. Руководство пользователя ГИС Спутник, версия 1.4 [Электронный ресурс] URL: https://download.geoscan.aero/sputnik/Sputnik_Manual_1.4_ru.pdf (дата обращения 17.03.2020 г.).
5. Линия электропередачи. Большая российская энциклопедия [Эллектронный ресурс] URL: https://bigenc.ru/technology_and_technique/text/2146410 (дата обращения 17.03.2020 г.).
6. Campbell, D. M. H.; White, B.; Arp, P. A. "Modeling and mapping soil resistance to penetration and rutting using LiDAR-derived digital elevation data". Journal of Soil and Water Conservation. 68 (6): 460–473. doi:10.2489/jswc.68.6.460. ISSN 0022-4561.
7. Pratt, William K. Digital image processing : PIKS inside / William K. Pratt. 3rd ed. New York [et al.] : Wiley, 2001. xix, 735 p. : ill. ; 24 cm + 1 electronic optical disk (CD-ROM). ISBN 0471374075.
8. В. А. Ивлев, И. В. Никифоров. Восстановление облака точек и построение 3D модели. (с. 61) / Санкт-Петербургский политехнический университет Петра Великого. Материалы научной конференции с международным участием "Неделя науки СПбПУ", 18-23 ноября 2019 года. Институт компьютерных наук и технологий / Санкт-Петербург : ПОЛИТЕХ-ПРЕСС, 2019. 144 с. : ил., табл. ; 20 см. ISBN 978-5-7422-6804-8.
9. Agisoft Metashape User Manual Professional Edition, Version 1.5 [Электронный ресурс] URL: https://www.agisoft.com/pdf/metashape_1_5_en.pdf (дата обращения 17.03.2020 г.).

УДК 004.45

Д. Г. Казанцева, Н. А. Чикин (4 курс бакалавриата)
Н. В. Воинов, к.т.н., доцент

ПРИЛОЖЕНИЕ ДЛЯ ХРАНЕНИЯ ПАРОЛЕЙ С ДОСТУПОМ ЧЕРЕЗ БРАУЗЕР

Целью работы было создание облачного хранилища паролей, к которому можно иметь доступ с любого устройства с веб-браузером.

На 2020 год самыми популярными паролями оставались такие простые комбинации, как “123456” или “qwerty”. Однако использование одного сложного пароля во всех системах также не увеличивает безопасность ваших данных. Запоминание же большого количество уникальных паролей зачастую довольно сложно, так как в некоторые системы мы входим крайне редко.

Облачное хранилище позволяет хранить пароли в одном месте, тем самым, не требуя их запоминания и наложений ограничения на длину или сложность пароля. На основании анализа преимуществ и недостатков существующих облачных сервисов хранения особое внимание при реализации уделено защите данных на сайте, что обеспечивается тремя этапами.

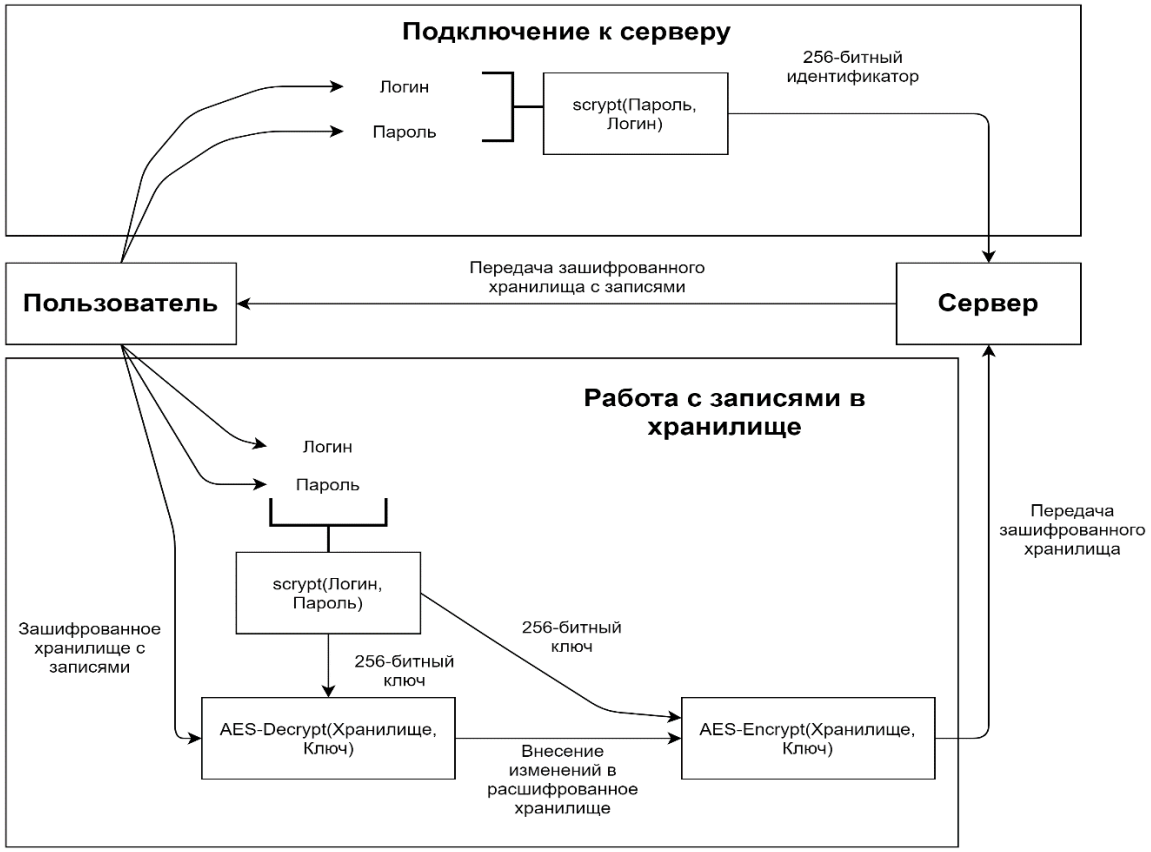


Рисунок 1 – Принцип работы приложения

Связь с сайтом устанавливается исключительно через безопасное (HTTPS) соединение, а значит исключается атака человек посередине (MITM). Сертификат для безопасного соединения был получен у Let's Encrypt. Для увеличения безопасности в дальнейшем необходимо будет активировать протокол HSTS.

Все данные зашифровываются через симметричное шифрование (AES-256) с ключом, основанным на связке логин-пароль. Выбранный нами алгоритм AES-256 на текущий момент является практически стандартом в сфере симметричного шифрования данных, а также крайне надёжным алгоритмом.

Данные, передающиеся по сети, зашифрованы. Расшифровка паролей происходит на клиентской стороне. Следовательно, при утечке данных с сервера, либо подмене сертификата

на стороне клиента или Интернет-провайдера, злоумышленник сможет получить только закодированные данные, которые невозможно расшифровать.

Для реализации проекта нами выбран был язык программирования Python, поскольку язык хорошо подходит для быстрого прототипирования, возможно использование разносторонних библиотек (для работы с передачей данных – SQL, JSON). Однако в качестве дальнейшего улучшения проекта можно переписать с интерпретируемого языка Python на более быстрый компилируемый: C++, Java, Rust.

Настоящая реализация является решением с открытым исходным кодом. Если пользователь захочет, то он сможет прочитать или предложить изменения в исходный код. Таким образом сервис сможет активно развиваться с поддержкой сообщества и быстрее находить, и исправлять возможные ошибки.

ЛИТЕРАТУРА

1. Жуков, К.Д. Прикладная дискретная математика: Обзор атак на AES-128: к пятнадцатилетию стандарта AES. 2017. №35. С.48 – 62.
2. Камер, Д. Э. Сети TCP/IP: Разработка приложений типа клиент/сервер для Linux/POSIX, том 3, 2002. – 592 с.

УДК 004.42

Л. А. Каравашкин, А. Е. Шкригунов (4 курс бакалавриата)
С. А. Молодяков, д.т.н., профессор

ВЕБ СЕРВИС ДЛЯ РАБОТЫ С РАЗЛИЧНЫМИ АУДИО ПЛАТФОРМАМИ.

Целью работы является создание клиент-серверного веб приложения, позволяющего находить и прослушивать аудио с различных платформ, а также создавать кроссплатформенные плейлисты.

Сервер предоставляет единый интерфейс для работы с различными аудио сервисами, а также отвечает за авторизацию и хранение пользовательских данных. Клиент представляет из себя одностраничное веб приложение, предоставляющее пользователю удобные методы для работы с сервером.

Особенности реализации сервера:

Сервер написан на языке Java с использованием фреймворка Spring boot, представляет из себя REST API. На данный момент поддерживаются такие площадки для прослушивания музыки, как vk и youtube. Поддерживается два типа плейлистов - динамические (плейлисты напрямую с площадок, которые могут обновляться извне) и статические (в которых могут содержаться аудиозаписи с различных платформ, не обновляются извне). Внутри себя содержит единый интерфейс для работы с аудио платформами, реализовав который можно с лёгкостью добавить поддержку дополнительных платформ.

Особенности реализации клиента:

Клиент написан с использованием библиотеки React в связке с Redux. Первая отвечает за UI приложения, а вторая за BLL(Business Logic Layer).

Был выбран React, так как он отвечает только за эффективную отрисовку приложения, не накладывая при этом ограничения на архитектуру, что дает гибкость при разработке. Высокая скорость работы приложения обеспечена использованием Virtual DOM и перерисовки только изменяемых частей приложения. Всё приложение разделено на компоненты, что упрощает тестирование отдельных частей, а также позволяет ускорить разработку.

Redux отвечает за менеджмент состояния приложения по архитектурному подходу Flux. При такой архитектуре в приложении присутствует только один источник данных (Store), в

зависимости от которых и происходит отрисовка интерфейса. Изменение данных происходит только через специальные функции (Actions).

В приложении использована относительно новая технология Progressive Web App (PWA). С весны 2018 года приложения этого класса поддерживаются всеми основными браузерами. PWA – это веб-технология, которая трансформирует сайт в приложение. Прямо из браузера его можно поставить на главный экран телефона, и оно сможет отправлять push-уведомления и получит доступ к аппаратным средствам гаджета. PWA-приложения ставятся на смартфон пользователя в обход официальных магазинов приложений и несмотря на запрет ставить приложения с неизвестных источников. После установки PWA создает кэш сайта. Это решает две задачи: повышает скорость загрузки и делает сайт доступным офлайн. То есть благодаря технологии PWA сайтом можно пользоваться даже без подключения к интернету.

Однако кэширование сайта может привести к проблемам с актуальностью хранимых данных. Для решения этой проблемы к различным ресурсам приложения следует применить различные стратегии кэширования. В этом поможет Workbox библиотека, которая поддерживает:

– предварительное кэширование;
– кэширование во время выполнения;
– фоновую синхронизацию;
– стратегии кэширования: Stale-While-Revalidate, Cache First, Network First, Network Only, Cache Only;
– настройки времени жизни кэша для каждого ресурса.

УДК 004.4

Т. Г. Кочарова, Д. А. Лосева (4 курс бакалавриата)
Т. А. Вишневская, ст. преподаватель

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ОБРАЗОВАТЕЛЬНЫМ ПРОЦЕССОМ ВУЗА

Одной из важных задач в университете является эффективное управление учебной деятельностью. Студенты, преподаватели, административно-управленческий и учебно-вспомогательный персонал нуждаются в единой системе организации процесса обучения. Проведение мероприятий по автоматизации процедур управления учебным процессом способно существенно улучшить качество управления и сократить его трудоемкость. Как правило, в настоящее время в вузах уже существуют готовые системы управления учебным процессом. Однако они в основной части направлены на организацию управления методическими материалами образовательных программ, составление расписания, автоматизацию работы сотрудников деканата (частично). Тем не менее существует множество узких мест в учебном процессе менее глобальных, чем вышеописанные, но которые требуют немалых временных затрат.

Примерами таких процессов могут быть, например,

– задача организации взаимодействия преподавателя с обучающимися у него студентами, в том числе в части согласования и корректировки расписания занятий и экзаменов;
– задача организации процесса дополнительной сессии для студентов, не сдавших экзамены и зачеты вовремя, а также создание преподавательских комиссий для приема экзаменов и зачетов у студентов должников;
– задача формирования студенческих групп для занятий по дисциплинам по выбору студента и другие.

Целью данной работы является разработка подсистем управления отдельными процессами на примере автоматизации учебного процесса высшей школы программной

инженерии ИКНТ и рассмотрение особенностей структуры связи и взаимодействия подсистем друг с другом.

Проектируемые подсистемы позволят преподавателям и административно-управленческому персоналу автоматически оповещать студентов об актуальной информации, касающейся учебного процесса и формировать документацию, необходимую, для организации дополнительной экзаменационной сессии.

Система представляет собой web-приложение, в обобщенном виде предоставляющее следующие возможности:

– вход в систему под корпоративным логином и паролем, в зависимости от должности и целей использования приложения

– рассылка студентам сообщений с информацией, касающейся изменений в учебном процессе

– корректировка информации о студентах, например, изменение их статуса или личных данных

– составление расписания комиссии для приема студентов с академическими задолженностями

– обновление данных о преподавателях, расписании и структуре университета

Для реализации приложения предлагается использовать следующие технологии: MS SQL Server для управления базой данных, содержащей информацию о студентах, преподавателях, расписании занятий и иные необходимые данные, и язык Python в среде разработки PyCharm для взаимодействия с базой данных. Общение пользователя с приложением осуществляется посредством браузера.

Схема данных системы представлена на рис. 1.

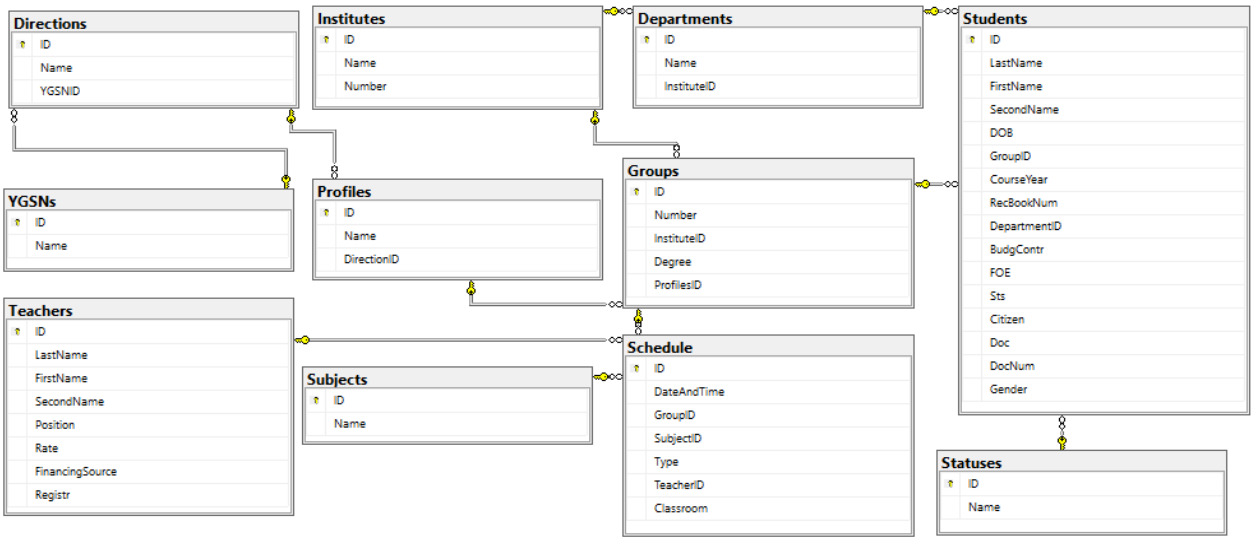


Рисунок 1

Для реализации системы применяется следующая иерархическая структура:

– Модуль для API.

– Модуль для базы данных.

– Модуль для пользовательского интерфейса.

– Протокол для взаимодействия Web – сервера и пользовательского интерфейса.

– Интерфейс для взаимосвязи Web – сервера и пользовательского интерфейса.

– Интерфейс для взаимодействия Web – сервера и базы данных.

Модуль пользовательского интерфейса включает в себя реализацию методов для отправки HTTP-запросов на сервер и принятия ответов с сервера и реализацию пользовательского интерфейса.

Модуль базы данных представляет собой перечень запросов, триггеров, функций и процедур для формирования таблиц, связей, внесения изменений в базу и иного взаимодействия с ней на уровне SQL-запросов.

Модуль API содержит в себе методы для принятия запросов от клиента, извлечения параметров запроса и формирования SQL-запросов для обращения к базе данных, а также методы для получения возвращаемых значений из базы, их обработки и предоставления клиенту.

Схема взаимодействия модулей системы представлена на рис. 2

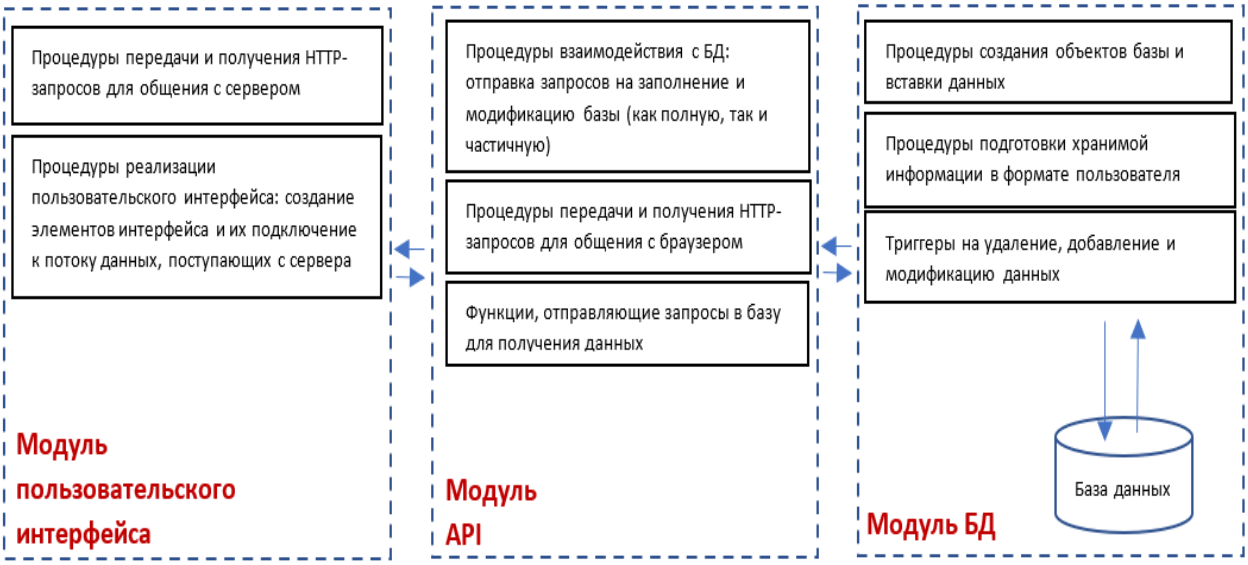


Рисунок 2

В будущем планируется расширение данной системы. На данный момент рассматривается система, доступная только для ИКНТ, а оповещение студентов предусмотрено только по электронной почте. В планах расширить систему и обеспечить оповещение посредством SMS-сообщений, так как такое сообщение с большей вероятностью будет прочитано в самые кратчайшие сроки, а значит информация будет передаваться быстрее, и уверенность в том, что адресат получил необходимую информацию, будет выше.

УДК 004.64

В. А. Кузьмин, М. А. Балдус (4 курс бакалавриата)
Т. А. Вишневская, ст. преподаватель

СИСТЕМЫ «РОДИТЕЛЬСКИЙ КОНТРОЛЬ» ПОД УПРАВЛЕНИЕМ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID

Смартфон стал неотъемлемой частью нашей жизни, и мы его используем постоянно: переписываемся в социальных сетях, воспринимаем медиаконтент, скачиваем различные приложения. Большинство людей не расстаются с телефоном на протяжении целого дня, что в частности дает возможность полностью отследить их жизнедеятельность. Все чаще можно услышать такое выражение: “зависимость от смартфона”. Особенно это касается детей, которые не в состоянии контролировать свою мобильную активность.

Целью нашей работы является создание приложения, с помощью которого родители смогут отследить и проконтролировать жизнедеятельность ребенка как в ночное время суток, так и днем. Подобные приложения существуют от двух больших компаний- Kaspersky и Google.

Таблица 1 – Анализ функциональности трех приложений.

	Google	Kaspersky	Наш проект
Установка будильника	+	+	+
Блокировка телефона	+	+	+
Определение геолокации	+	+	+
Активность приложениях	+	+	+
Доступ к фотографиям	+	-	-
Ограничение времени пользования	+	-	-
Рекомендации приложений	+	-	-
Установка расписания	-	-	+
Чат с ребёнком	-	-	+
Экстренная кнопка для ребёнка	-	-	+
Анализ освещённости помещения	-	-	+

Проведенный анализ подобных приложений показал, что некоторые функции, которые родители хотели бы иметь у себя в приложении, в них отсутствуют (см.табл.1). Например, с помощью функции “Ночная блокировка” можно блокировать возможность использовать телефон за час до сна, что поможет ребенку успокоиться и подготовиться ко сну. А функция “Анализ освещенности помещения” даст родителям знать, если в комнате ребенка слишком светло, что плохо сказывается на выработке мелатонина, и нужно принять меры. Используя функцию “Контроль физической активности”, родители устанавливают определенные цели на день, которые ребенку стоит выполнить. В случае недостаточной активности в течение дня, приложение уведомит об этом родителей.

Требования и архитектура приложения

Для работы приложения необходимо наличие следующих устройств:

– Два смартфона, с установленной на них операционной системой Android;

– Фитнес браслет для ребенка;

Разрабатываемое приложение должно состоять из двух подсистем с общей базой данных:

– Подсистема родительского контроля мобильной деятельности ребенка, выполняющая следующие функции;

– Составление списка дел для ребенка;

– Установка целей по физической активности;

– Составление расписания дня ребенка;

Подсистема сбора и анализа информации о жизнедеятельности ребенка, выполняющая следующие функции:

– Сбор данных по физической активности;

– Анализ данных о качестве сна;

– Сбор данных о использовании смартфона;

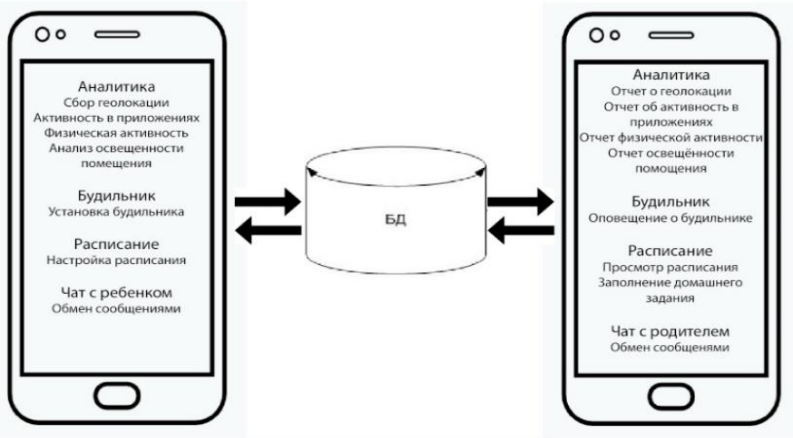


Рисунок 1 – Схема взаимодействия компонентов приложения.

Для хранения информации о деятельности ребенка используется облачная NoSQL база данных Firebase realtime database.

Для разработки подсистем будет использоваться язык Java, в качестве средств разработки выбраны следующие инструменты: Android studio, IntelIj idea.

Как уже было выше сказано, в качестве платформ разработки будут применяться Android, Firebase.

Дополнительно используем гибкую методологию разработки Agile. Применяя эти методы разработки, мы можем добавлять функции в проект, когда в них возникнет необходимость.

ЛИТЕРАТУРА

1. Documentation | Friebase.[Электронный ресурс] URL: https://firebase.google.com/docs/database
2. Documentation | Android Developers [Электронный ресурс] URL: https://developer.android.com/docs
3. Amazon.com. Статья «Создание приложений с бессерверными архитектурами». [Электронный ресурс]. URL: https://aws.amazon.com/ru/lambda/serverless-architectures-learn-more/

УДК 004.428

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

РАЗРАБОТКА СИСТЕМЫ СОЗДАНИЯ ТЕХНОЛОГИЧЕСКИХ МАРШРУТОВ ДЛЯ МЕЛКОСЕРИЙНОГО ПРОИЗВОДСТВА

На данный момент в промышленности ведется активная работа по оптимизации и повышению эффективности мелкосерийного и единичного производства. [1] Одним из проблемных мест в производстве являются большие траты времени на проектирование технологического процесса и технологических маршрутов. [2] Моя работа предлагает способ автоматизации процессов путем использования веб-приложения в производстве.

Целью работы является разработка веб-приложения для автоматизации создания технологический маршрутов по изготовлению деталей с возможностью создания, копирования, редактирования и удаления деталей для мелкосерийного производства.

В качестве основного фреймворка для создания приложения был выбран React, один из самых популярных веб фреймворков [3], при разработке также использовались такие технологии, как SCSS для упрощения работы с CSS кодом [5], фреймворк Material UI, имеющий в своем составе готовые компоненты, поддерживающие полную кастомизацию [4] для работы с дизайном и Selenium Web Driver [6] для автоматизации тестирования.

Архитектурно приложение разделено на 12 модулей, согласно выполняемым задачам. Каждый из модулей состоит из *.js-файла, в котором описана логика его работы и структура, а также файла стилей, содержащего в себе информацию о внешнем виде элементов, используемых в качестве составных частей для модулей. В отдельный модуль вынесен файл, содержащий основную логику обратных вызовов функций для передачи информации между страницами и переключения между страницами. Также отдельно вынесен основной корневой модуль приложения App, в котором хранятся все остальные модули.

В отдельном разделе хранятся файлы моделей для упорядочивания поступающей информации и ее структуризации, которые также включают в себя конструкторы для создания каждой из моделей, в соответствии с данными моделями были созданы тестовые данные для эмуляции.

Раздел страниц содержит информацию о страницах и состоит из 10 частей. Каждая страница состоит из *.js-файла файла, в котором находится ее структура, и данных об отображении внешнего виды страницы, находящихся в файле стилей.

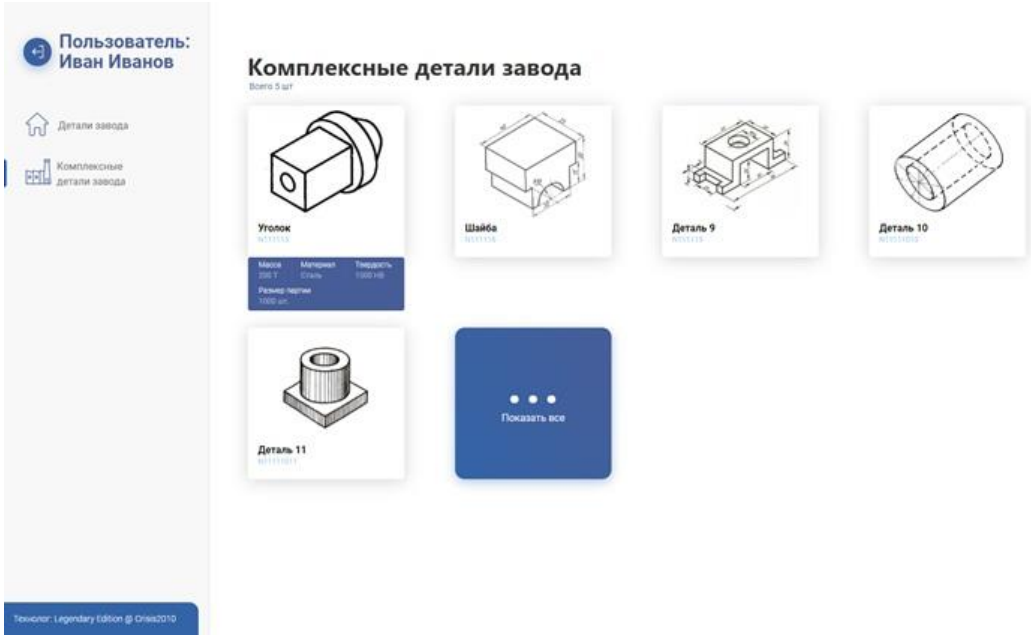


Рисунок 1 – Просмотр комплексных деталей в приложении

В избежание ошибок в основной ветке разработки были применены техники unit-тестирования и автоматических сценариев. В работе использовался selenium-web driver и selenium ide для создания тестов.

Автоматическими сценариями покрыты основные способы использования этой программы, что делает безопасным и удобным доработку существующего функционала при необходимости.

Таким образом, в работе была рассмотрена проблема повышения эффективности мелкосерийного производства, и предложено решение по снижению затрат времени на проектирование технологического процесса и технологических маршрутов. На данный момент приложение создано и полностью функционирует, оно позволяет оптимизировать работу по созданию и редактированию деталей с возможностью редактирования параметров, таких как наборы поверхностей и блоков.

ЛИТЕРАТУРА

1. Белашевский, Г.Е. Метод оценки управления системной технологической подготовки производства / Г.Е. Белошевский, И.Г. Абрамова, Д.А. Абрамов, В.Н. Бородин // Вестник Самарского государственного аэрокосмического университета им. Академика Королева. - 2008. - №3(16). - С.146-156.
2. Берлинер Ю.М. САПР технолога машиностроителя: учебник / Берлинер Ю.М., Таратынов О.В. – М.: ФОРУМ : ИНФРА-М, 2015.-336 с.: илл.
3. Справочник по фреймворку React// [официальная документация]/— URL: https://ru.reactjs.org/tutorial/tutorial.html. (дата обращения:13.11.2019).
4. Справочник по компонентам, входящим в состав Material UI// [официальная документация]/— URL: https://material-ui.com/ru/components.(дата обращения:08.12.2019).
5. Справочник по SCSS// [официальная документация]/— URL: https://sass-scss.ru/guide/.(дата обращения:26.12.2019).
6. Справочник по Selenium IDE// [официальная документация]/— URL: https://www.selenium.dev/selenium-ide/. (дата обращения:15.12.2019).

УДК 004.514.6

О. О. Кузьмин (4 курс бакавриата)
А. П. Маслаков, ст. преподаватель

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ ПЛАНИРОВАНИЯ СОБЫТИЙ НА ВРЕМЕННОЙ ШКАЛЕ

Основная цель работы – разработка встраиваемого веб-компонента экосистемы Angular [1] для работы с событиями на временной шкале в графическом режиме. Данный компонент может иметь применения в различных областях, таких как: планирование жизненного цикла программного обеспечения, создания личного расписания, управление использованием активов.

Для удобного взаимодействия конечного пользователя с компонентом требуется поддержка следующего функционала:

– информативное графическое отображение как календаря, так и событий, позволяющее сопоставить календарное время и период протекания события;
– интуитивное управление: создание, изменение размеров и перемещение события с использованием drag-and-drop, контекстное меню;
– наличие управляющих элементов для манипуляции отображаемым временным промежутком;
– отсутствие снижения производительности при работе с временными промежутками продолжительностью в несколько месяцев.

При этом, поскольку данный компонент является встраиваемым и переиспользуемым, на него налагаются дополнительные требования, отвечающие за удобство его использования инженером-программистом:

– адаптация интерфейса под различные размеры окна или выделяемой области;
– интеграция с используемой платформой разработки;
– возможность получать оповещения о производимых пользователем операциях и влиять на результат этих операций;
– удобный формат работы с датами и часовыми поясами;
– настройка доступного пользовательского интерфейса без ущерба его внешнему виду и, в том числе, возможность замены стандартных частей интерфейса;
– отсутствие искусственных ограничений на количество экземпляров виджета на странице при сохранении их функциональной и информационной независимости.

Существуют библиотеки, предоставляющие схожие по функционалу компоненты. Далее приведена таблица, отображающая их соответствие основным характеристикам.

Таблица 1 – Сравнение аналогов

Характеристика	Бесплатные			Коммерческие		
	angular-calendar	ngx-time-scheduler	fullcalendar-angular	jqWidgets	Kendo UI	Syncfusion JavaScript Controls
Поддержка разработчиками	-/+	-/+	+	+	+	+
Интуитивное управление	-/+	-	+	+	+	-/+
Отображение произвольных (длительных) промежутков времени	-	-	-	-	-	-
Взаимодействие пользователя с событиями	+	-/+	-/+	+	-/+	+
Компактный объём пакета	+	+	+	-	-	-
Отзывчивый дизайн	+	-	-/+	-	+	+

Реализованный продукт представляет собой 3 проекта, каждый из которых является полноценным пакетом для экосистемы Angular 9. Проекты делят между собой логическую часть и предоставляемый интерфейс, в то время как различие заключается в используемых библиотеках графического интерфейса. Внешний программный интерфейс представлен асинхронными вызовами. Разработчик может обрабатывать их в соответствии с требуемой логикой, включающей взаимодействие с пользователем или удалёнными ресурсами.

Пакет core предоставляет временную шкалу с отображаемыми на ней событиями и с возможностями взаимодействия на основе drag-and-drop и контекстного меню.

Пакеты material и bootstrap имеют дополнительные зависимости на Angular Material UI [2] и Bootstrap [3] соответственно. Кроме временной шкалы, они предоставляют панель управления, реализованную с помощью возможностей фреймворка Angular и графических библиотек Angular Material UI и Bootstrap. Наличие панели контролируется через конфигурацию компонента.

Такая организация пакетов позволяет совместить удобство использования компонентов с отсутствием необязательных зависимостей. Единый программный интерфейс позволяет без трудозатрат переключаться между пакетами и используемыми графическими библиотеками.

Отзывчивость интерфейса и возможность его подстраивания под различные размеры доступной области экрана без ущерба производительности реализуется с помощью двух решений.

Первое заключается в использовании возможностей CSS (Cascading Style Sheets) [4] современных браузеров – Flexible Box Layout. Данная спецификация позволяет создавать гибкие макеты с указанием безразмерных значений пропорций [5]. Кроме того, она позволяет задавать последовательность отображения элементов через свойства стилей без использования JavaScript, что положительно сказывается на производительности и уменьшает количество кода.

Вторым является концепция CSS Element Queries - условное применении CSS стилей к HTML элементу в зависимости от размеров его родителя [6]. Данная концепция не является частью стандарта, но имеет некоторое количество реализаций, основанных на возможностях языка JavaScript. В данном случае используется реализация marcj/css-element-queries, которая позволяет отслеживать события изменения конкретных элементов разметки. Это обеспечивает высокую производительность и, вместе Flexible Box Layout, позволяет перестраивать интерфейс во время исполнения, не опираясь на размер окна и пользуясь механизмами, схожими точками перехода (breakpoints) [7].

Вышеозначенные технологии и архитектурные решения позволили создать удобный графический веб-интерфейс для взаимодействия с событиями, обладающий лёгкостью применения и переиспользования в различных продуктах. Данный компонент, при соответствии поставленным требованиям и задачам, имеет потенциал для расширения функционала и областей применения.

ЛИТЕРАТУРА

1. Angular - https://angular.io/
2. Angular Material UI - https://material.angular.io/
3. Bootstrap - https://getbootstrap.com/docs/4.4/getting-started/introduction/
4. Cascading Style Sheets - https://www.w3.org/Style/CSS/Overview.en.html
5. CSS Flexible Box Layout - https://www.w3.org/TR/css-flexbox-1/
6. CSS Element Queries proposed specification - https://tomhodgins.github.io/element-queries-spec/element-queries.html
7. Eric A. Meyer “CSS: The Definitive Guide” (2017)

УДК 004.738.5: 004.451.55

Мин Хтет Джо (1 курс магистратуры)
Т. С. Горавнева, к.т.н., доцент

РАЗРАБОТКА WEB-ИНТЕРФЕЙСА ЦИФРОВИЗАЦИИ СУДОВЫХ ТЕРМИНОВ

Задачей данного Интернет-проекта является предоставление полезной и интересной информации о судовых терминах в адаптивном и современном видах отображения на различных устройствах, чтобы она позволила оценить популярность и привлекательность специальностей, которые обеспечиваются федеральными государственными образовательными стандартами высшего образования СПбГМТУ.

Таким образом, в первую очередь web-проект предназначен для абитуриентов, поступающих в Морской технический университет, но также может быть полезен другим пользователям.

Поэтому целью данной работы было применение возможностей клиентского web-программирования для цифровизации информации судовой тематики, разработки хранилища терминов (русская, английская и мьянманская терминологии морской технической области) в виде XML-документа, а также разработка интерфейса доступа к хранилищу средствами web-технологий.

Веб-приложение состоит из главной страницы, имеющей навигационное меню и подменю разделов веб-сайта, с помощью которых осуществляется переход между страницами веб-сайта. Веб-сайт имеет приятный внешний вид за счет хорошо оформленных страниц с помощью CSS-таблиц. На рис. 1 представлена часть главной страницы, по сути, она является введением в судостроительную терминологию, содержание которой написано на мьянманском языке.

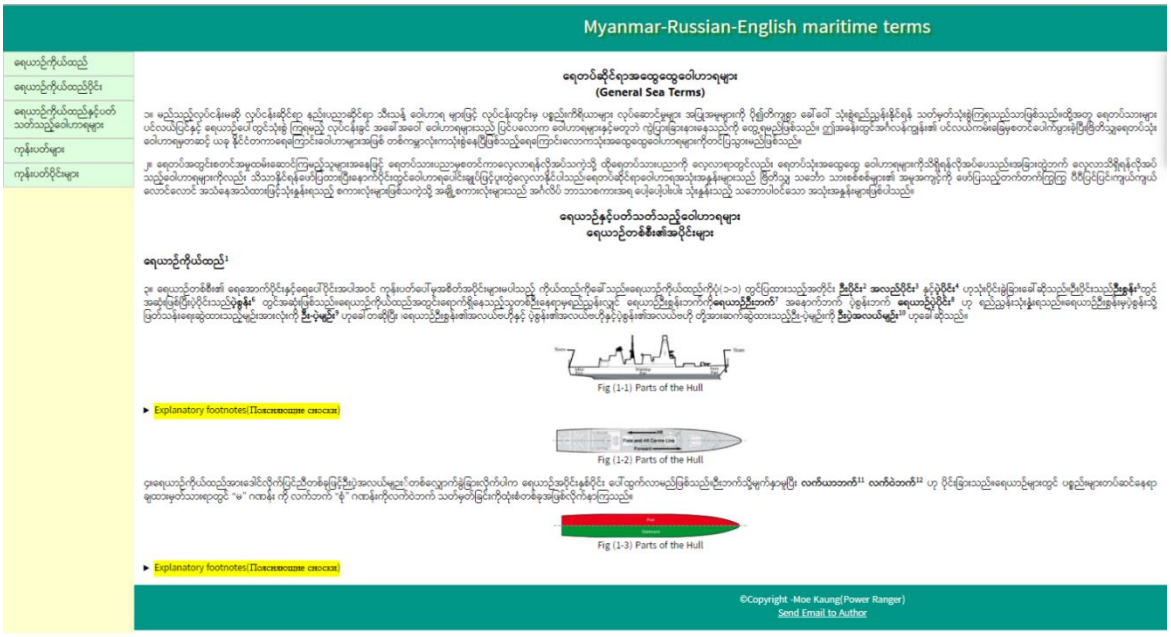


Рисунок 1 – Главная страница сайта

XML-хранилище данных. Данное XML-хранилище необходимо для реализации раздела, с помощью которого можно ознакомиться с терминами судовой и морской тематик на разных языках.

Средствами создания программного обеспечения данной разработки являются программные средства, позволяющие сформировать структуру хранилища (XML, XML Schema), а также средства программирования динамического web-интерфейса, основанные на

языке гипертекста HTML, каскадных таблицах стилей CSS, сценариев JavaScript с использованием объектной модели документа браузера Microsoft Internet Explorer.

Разработанные программные средства функционируют под управлением любой из операционных систем Windows и предназначены для работы в локальной сети университета или глобальной сети Интернет.

УДК 004.457

А. Г. Лезов (3 курс бакалавриата)
А. П. Маслаков, ст. преподаватель

ПРИЛОЖЕНИЕ ДЛЯ ПОИСКА ИНФОРМАЦИИ В СПРАВОЧНИКАХ ДЛЯ ТЕХНОЛОГОВ МАШИНОСТРОЕНИЯ

В ходе работы технологов машиностроения перед ними часто возникает необходимость в использовании различных видов справочников по деталям и режущему инструменту. Размер каждого такого справочника может доходить до 1000 - 1500 страниц. Работа с каждым из них по отдельности довольно трудоемкая, рутинная и занимает много времени. В связи с этим возникает необходимость в утилите, позволяющей ускорить данный процесс.

Основная цель разрабатываемого приложения – обеспечить простоту и комфорт при работе с такими справочниками. Осуществить это можно с помощью веб ресурса с личным кабинетом для каждого пользователя. В приложении должен присутствовать простой графический интерфейс, с которым будет удобно работать.

Вначале пользователю будет предложено пройти форму регистрации для создания личного кабинета. Далее будет возможность загрузить все необходимые документы. После загрузки, пользователь в предоставленную форму ввода сможет внести названия типов деталей, информацию по которым он бы хотел получить. Программа анализирует каждый загруженный справочник, находит всю информацию по заданным деталям и формирует pdf файл, куда вносит все результаты поиска. В результате, пользователь получает новый файл, содержащий только необходимую информацию. Все загружаемые раннее файлы и все новые pdf с результатами можно хранить в личном кабинете.

Программа представляет собой архитектуру, представленную на рисунке 1 и функциональную модель, представленную на рисунке 2.

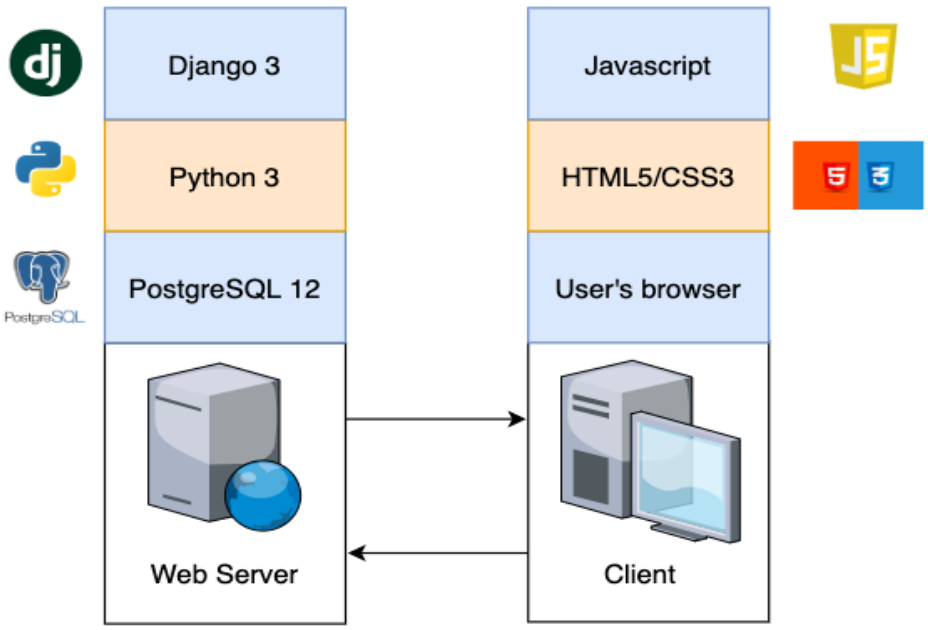


Рисунок 1 – Архитектура системы.

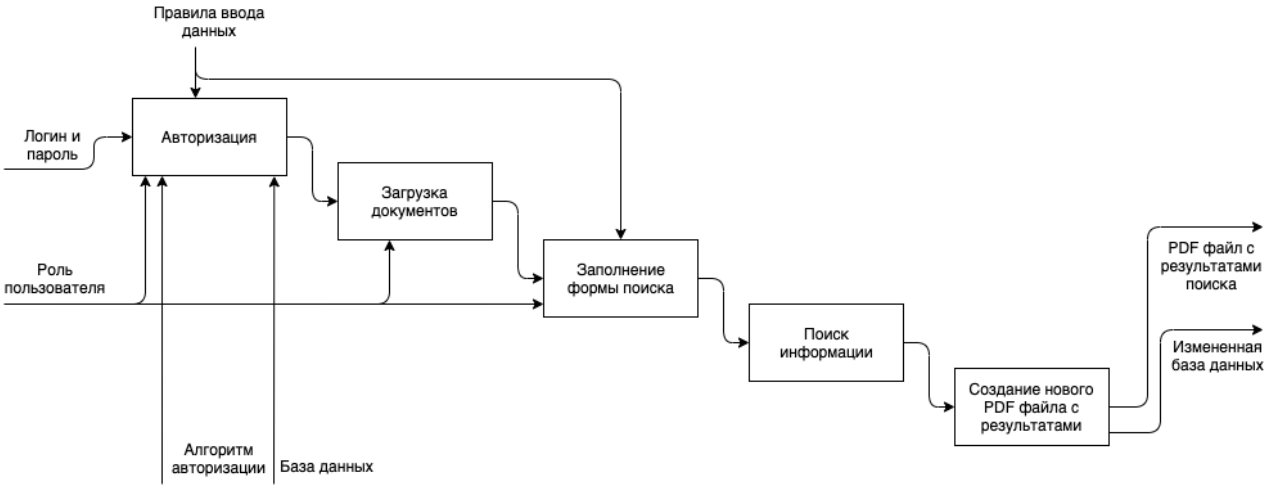


Рисунок 2 – Функциональная модель. Работа с системой.

Интерпретированный серверный сценарий программного средства должен обладать свойством кроссплатформенности. Пользовательские интерфейсы должны корректно и одинаково отображаться во всех актуальных версиях браузеров Google Chrome, Opera, Mozilla, Safari.

Реализованное в ходе работы приложение отвечает поставленным целям, а именно – сокращает время и затраты технолога на работу со справочниками и позволяет быстро искать нужную информацию.

Возможные направления развития проекта:

– реализовать возможность поиска по картинкам

– реализовать удобный фильтр для поиска

УДК 004.67

В. В. Монастырев (2 курс магистратуры)
П. Д. Дробинцев, к.т.н, доцент

СИСТЕМА АНАЛИЗА ИНТЕРЕСОВ ПОЛЬЗОВАТЕЛЯ НА ОСНОВЕ ДЕЙСТВИЙ В СОЦИАЛЬНОЙ СЕТИ

В настоящий момент практически каждый человек пользуется различными социальными сетями, мессенджерами, фотохостингами и т.п. При этом, как правило, пользователи оставляют множество информации о себе в свободном доступе. Данную информацию можно использовать для анализа интересов пользователей. На основе интересов можно создавать рекомендательные ленты новостей, таргетировать рекламу и рекомендовать интересных авторов.

Практически все крупные сервисы имеют собственные рекомендательные системы – YoutTube [1], Netflix [2], Yandex [3] и т.п. Подобные системы не являются open-source решениями, так как каждая компания разрабатывает их под свои цели (будь то рекомендация фильмов, рекомендация новых друзей и т.п.) и затачивает их под свои наборы данных. В данной работе будет рассматриваться система для анализа интересов пользователей на основе следующих наборов данных: опубликованные изображения и комментарии, позитивно и

негативно оцененные изображения и комментарии, геометки. При этом рассматривался набор данных примерно 100 пользователей за период около 1 месяца.

Архитектура системы анализа интересов представлена на рисунке 1:

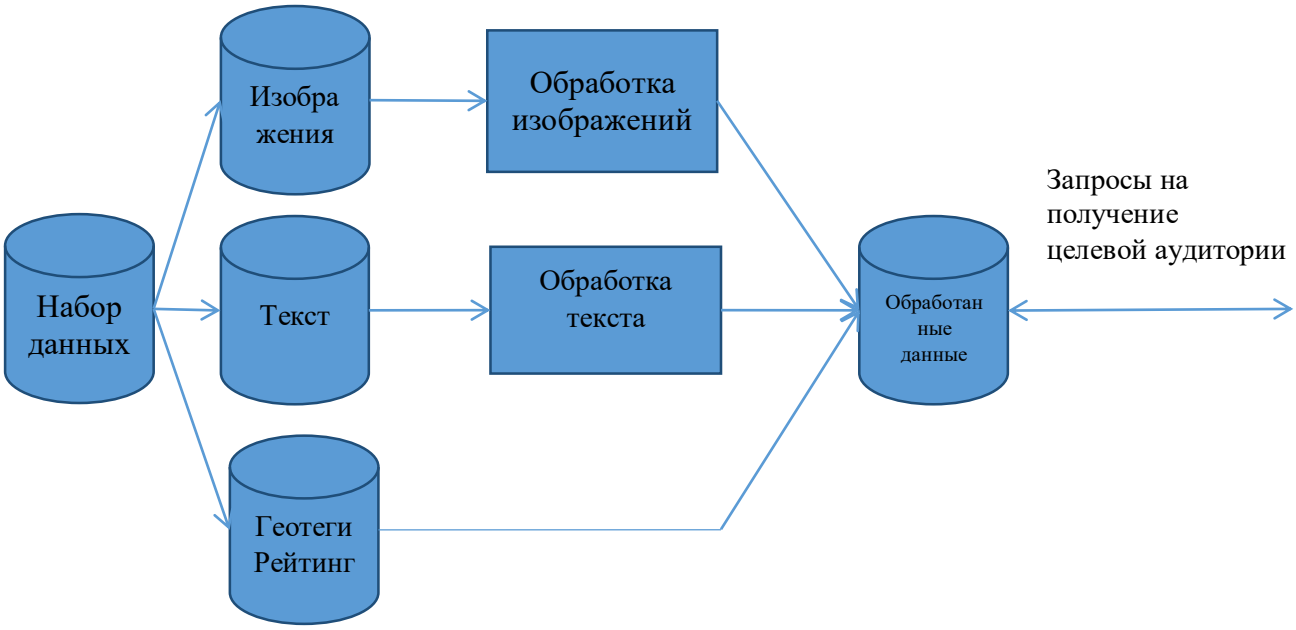


Рисунок 1 – Архитектура системы анализа интересов пользователя.

Изначальный набор данных представляет собой дамп из базы данных MySQL. Как было сказано выше, данный набор данных содержит информацию об изображениях и текстах, которые выкладывал пользователь, о понравившихся и не понравившихся изображениях и текстах, а также информацию о геопозиции пользователя. Из данного набора данных выделяются отдельно изображения и текст. Они будут подвергаться дополнительной обработке. На изображениях будут распознаваться объекты. Текст будет анализироваться на тональность (позитивный или негативный), а также на содержание.

Для распознавания объектов на изображениях использовалась модель Inception-v3[4]. Данная модель обучена на 1000 классов [5] и представлена в свободном доступе в официальном репозитории TensorFlow на GitHub [6]. Для анализа изображений была выбрана данная модель, так как она имеет простое API, уже обучена на большом количестве классов (что позволяет сэкономить ресурсы на обучении и сборе датасета), работает достаточно быстро (1-2 секунды на 1 изображение). Пример работы модели представлен на рисунке 2:

```
flags =  Namespace(image_file='server_uploads/
30/8b10a257-7569-49c8-96e2-6d15676c9169.jpeg',
model_dir='tmp/imagenet', num_top_predictions=
5)
unparsed =  ['D:\\Users\\WinUser\\Anaconda3\\l
ib\\site-packages\\ipykernel_launcher.py', '-
f', 'C:\\Users\\WinUser\\AppData\\Roaming\\jup
yter\\runtime\\kernel-fafb8a94-6d99-4718-bbe4-
80870508319a.json']
television, television system (score = 0.8473
8)
```

Рисунок 2 – Пример работы модели по распознаванию изображений.

При анализе изображения данная модель выводит топ классов предсказаний с самым высоким значением вероятности (score). На данном этапе принимался только один предсказанный класс с самым высоким значением score. Данные о предсказанных классах также записываются в MySQL.

Следующий пункт – анализ текста. Анализ текста представлен 2-мя этапами – выделение основной тематики и выделение тональности (написан текст в позитивном ключе или негативном).

Для анализа тематики текста использовалась модель на основе алгоритма Latent Dirichlet Allocation (LDA) [7] в пакете Gensim. Смысл алгоритма LDA в том, что каждый документ рассматривается как набор тем в определенной пропорции, которые в свою очередь состоят из определенного набора слов. Дополнительно были использованы стоп слова из NLTK и модель ru2 из пакета spacy, так как в представленном наборе данных все тексты представлены на русском.

Предварительно входной текст был дополнительно обработан – удалены лишние пробелы и кавычки при помощи регулярных выражений. После этого была произведена очистка текста и его токенизация при помощи метода simple_preprocess() пакета Gensim, а также удалены стоп слова. На следующем шаге были созданы модели биграмм (2 слова, встречающиеся в тексте вместе) и триграмм (3 слова, встречающиеся в тексте вместе). После этого строилась LDA модель. Данная модель строилась как для всего текста, чтобы рассмотреть темы, так и для одного пользователя. Пример визуализации модели представлен на рисунке 3:

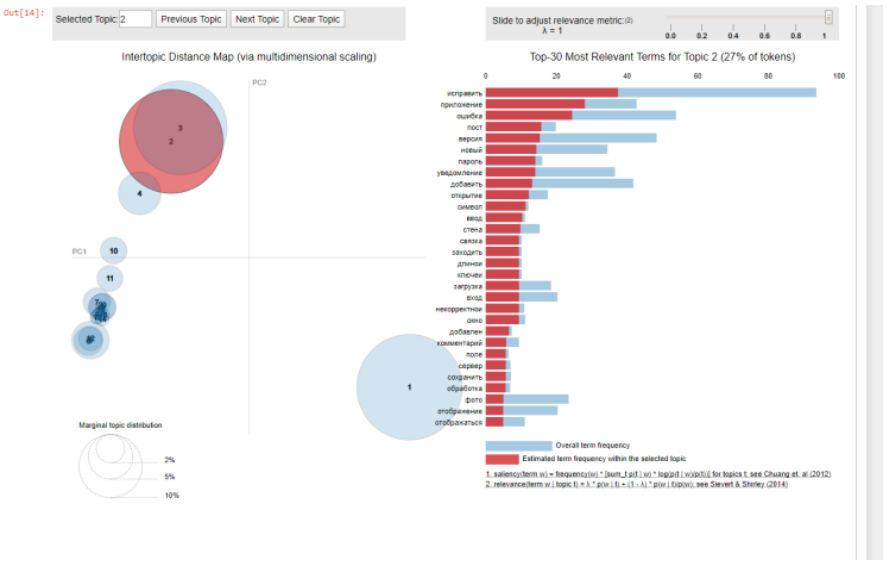


Рисунок 3 – Визуализация работы модели LDA.

Для анализа тональности текста использовалась сверточная нейронная сеть (Convolutional Neural Networks, CNN) [8]. Для формирования признакового пространства использовался Word2Vec. Обучение проводилось на корпусе слов на основе русскоязычных сообщений из Twitter, который содержит 114991 положительных и 111923 отрицательных твитов, а также 17639674 неразмеченных твитов. Перед обучением все данные прошли предварительную обработку (приведение к нижнему регистру, замена ссылок на токен и т.д.). Обучение модели Word2Vec проводилось при помощи библиотеки Gensim. Для построения нейронной сети использовалась библиотека Keras. Данная модель, обученная на твитах, была применена для текстовых сообщений из рассматриваемого набора данных. Метрики модели представлены на рисунке 4:

	precision	recall	f1-score	support
0	0.76179	0.80437	0.78250	22236
1	0.79569	0.75180	0.77312	22534
accuracy			0.77791	44770
macro avg	0.77874	0.77808	0.77781	44770
weighted avg	0.77885	0.77791	0.77778	44770

Рисунок 4 – Метрики модели тональности текста.

Данные о текстах также были записаны в базу данных MySQL. По итогам все обработанные данные были объединены на основе id. В итоговой таблице была получена информация о том, что выкладывает пользователь, о чем он пишет и в каком ключе, а также о точках, которые пользователь отмечал на карте. На основе этой таблицы легко составлять таргет группу по заданным параметрам. К примеру, мы хотим прорекламировать магазин велосипедов, который находится в некотором районе города. Таким образом, нам достаточно просто сделать запрос в таблицу с поиском пользователей, которые выкладывают фото велосипедов, либо оценивают записи с велосипедами, либо пишут о велосипедах в позитивном ключе и которые оставляли геотеги в нужной нам области.

В будущих планах – автоматизация данного алгоритма, так как на данный момент он не встроен в целевую систему. Также планируется автоматизация процесса дообучения моделей и создания платформы с web-интерфейсом для поиска таргет-групп.

ЛИТЕРАТУРА

1. Covington P., Adams J., Sargin E.: Deep Neural Networks for YouTube Recommendations. Proceedings of the 10th ACM Conference on Recommender Systems, ACM, New York, NY, USA (2016).
2. Chandrashekar A., Amat F., Basilico J, Jebara T.: Artwork Personalization at Netflix. The Netflix Tech Blog, 2017.
3. Рекомендательная технология Диско [Электронный ресурс]. – Режим доступа: https://yandex.ru/company/technologies/disco/
4. Распознавание изображений предобученной моделью Inception-v3 с Python API на CPU [Электронный ресурс]. – Режим доступа: https://neurohive.io/ru/tutorial/tutorial-raspoznavanie-izobrazhenij-s-tensorflow-i-python-api-na-cpu/
5. 1000 synsets for Task 2 (same as in ILSVRC2012) [Электронный ресурс]. – Режим доступа: http://image-net.org/challenges/LSVRC/2014/browse-synsets
6. Tensorflow GitHub [Электронный ресурс]. – Режим доступа: https://github.com/tensorflow/models
7. Тематическое моделирование с помощью Gensim (Python) [Электронный ресурс]. – Режим доступа: https://webdevblog.ru/tematicheskoe-modelirovanie-s-pomoshhju-gensim-python/
8. Анализ тональности текстов с помощью сверточных нейронных сетей [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/mailru/blog/417767/

УДК 004.457

С. О. Полевич (4 курс бакалавриата)
Н. В. Воинов, к.т.н., доцент

РЕАЛИЗАЦИЯ ПОДДЕРЖКИ СЕРВИСА ПО УПРАВЛЕНИЮ ИНТЕРАКТИВНЫМИ ПОДПИСКАМИ НА МОБИЛЬНЫХ ПЛАТФОРМАХ

Целью работы является исследование возможностей фреймворка React Native для разработки кроссплатформенных приложений для iOS и Android. Во времена, когда еще не было технологий для кроссплатформенной разработки (таких как Apache Cordova и React Native) приходилось изучать язык программирования Java, чтобы написать приложение для Android, и Objective C, чтобы написать приложение iOS. В ходе решения данной проблемы была придумать Apache Cordova. Мы пишем приложение на HTML, CSS, Java, а Cordova запускает его в браузере на телефоне. Такой подход значительно ускорил процесс разработки приложений, но он имеет некоторые недостатки (существующие плагины для Cordova быстро устаревают. Так как приложение запускается в браузере, то возникают сложности в получении обратной связи с нативным приложением). Результатом поисков решения данных проблем, Facebook представил React Native. В данном случае приложения написаны на языке JavaScript, но итоговое приложение, то, которое мы запускаем на телефоне, использует нативный код (Java для Android, Objective C для iOS). В итоге мы получаем такое же приложение, как если бы писали его на нативном языке: элементы управления, внешний вид и жесты также работают, как в нативном приложении. Как же это работает?

Каждое React Native приложение имеет два важных потока. Основной поток – он запускается абсолютно в каждом нативном приложении, Он обрабатывает отображение элементов пользовательского интерфейса и жесты пользователя. JavaScript поток – он выполняет код JavaScript в отдельном движке. JavaScript определяет как работает приложение.

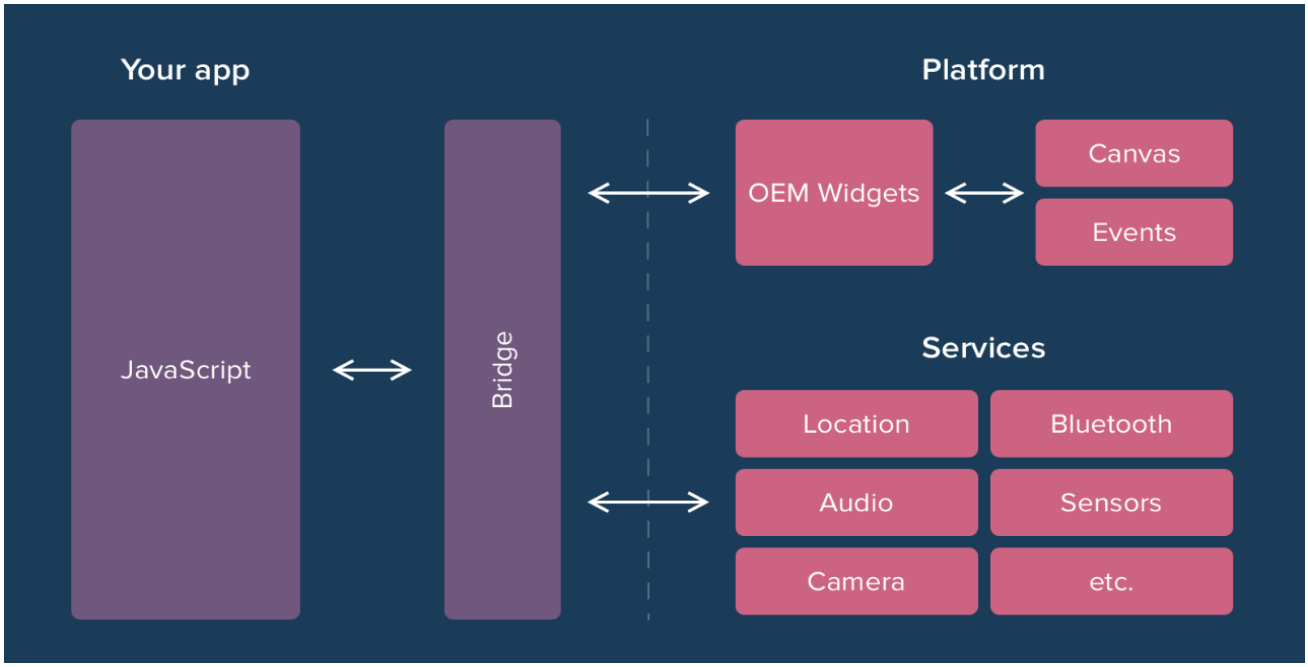


Рисунок 1 – Как React Native взаимодействует с нативными компонентами

За сообщение между двумя потоками отвечает так называемый мост (Bridge) – это ядро React Native. Мост позволяет потокам общаться наилучшим, оптимизированным способом. Он служит посредником, который распределяет запросы и входящие данные от двух потоков. Такой подход позволяет им общаться асинхронно, что приводит к стабильной работе, потому что потоки никогда не смогут заблокировать друг друга.

УДК 004.65

И. В. Попугаев (3 курс бакалавриата)
О. Н. Петров, к.т.н., доцент

РАЗРАБОТКА ПЛАТЕЖНОЙ СИСТЕМЫ

Целью работы является проектирование платежной системы (Single Page Application) под веб-браузер. Данная система должна обеспечивать возможность пополнения баланса, перевода денежных средств, а также возможность заполнения и изменения персональных данных пользователя. Все операции, связанные с денежными средствами, должны фиксироваться системой.

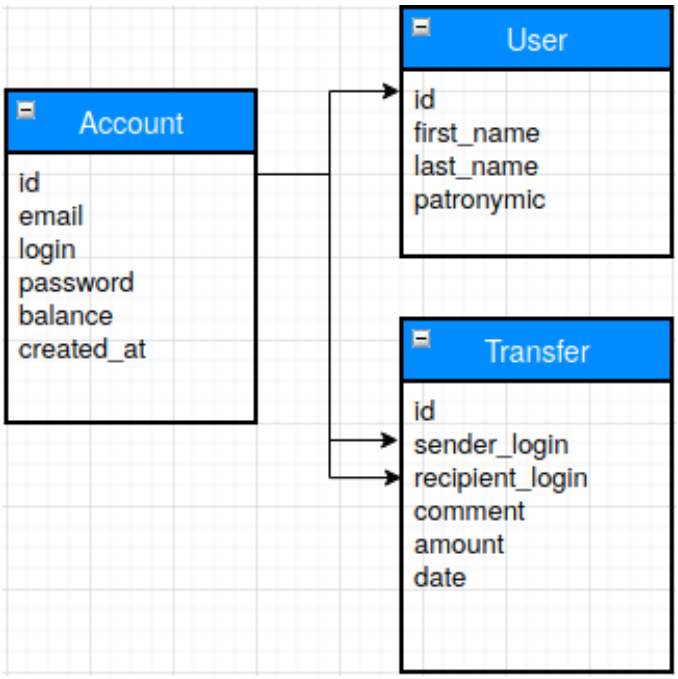


Рисунок 1 – Сущности платежной системы и их взаимоотношение

Для проектирования серверной части выбран язык “Go”, потому что он один из самых высокопроизводительных, а также прост в организации конкурентного исполнения программы за счет “горутин”. В качестве базы данных используется “Postgres”, она отлично подходит к объектно-ориентированному программированию, так как является объектно-реляционной. Для построения пользовательских интерфейсов выбран язык “typescript” с библиотекой “react”, потому что она позволяет писать декларативный код, используя функциональный подход. Также он вносит понятие Virtual DOM. Virtual DOM – это концепция программирования, в которой «виртуальное» представление пользовательского интерфейса хранится в памяти и синхронизируется с «настоящим» DOM. Этот процесс называется согласованием.

Все денежные операции являются атомарными и исполняются в виде транзакций, что гарантирует корректную работу программы, а в точности то, что операции не будут выполнены частично. Это позволяет устранить такие ситуации, когда пользователь осуществляет перевод, а в момент снятия денег с его баланса у него обрывается сеть, в результате чего списанные средства не начисляются другому пользователю.

ЛИТЕРАТУРА

1. Коннолли Т., Бегг К. Базы данных: проектирование, реализация, сопровождение. Теория и практика, 3-е изд. : Пер. с англ. : Уч. пос. – М.: Изд. дом "Вильямс", 2003. – 1440 с.
2. Донован, Алан А. А., Керниган, Брайан, У.Д67 Язык программирования Go. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2016. — 432 с. : ил. — Парал. тит. англ.
3. Бэнкс Алекс, Порселло Ева. React и Redux: функциональная веб-разработка. — СПб.: Питер, 2018. — 336 с.

УДК 004.457

М. К. Рябикин (4 курс бакалавриата)
А. П. Маслаков, ст. преподаватель

РАЗРАБОТКА И АВТОМАТИЗАЦИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ПРОВЕРКИ ЛАБОРАТОРНЫХ РАБОТ В РАМКАХ КУРСА ООП НА ЯЗЫКЕ JAVA

При прохождении курса по программированию студенты выполняют практические задания. Большая часть рутинной работы преподавателя по оценке программ может быть автоматизирована, оставляя только написание тестов и окончательную приёмку работы. Для сбора, тестирования и проверки лабораторных работ можно использовать различные программные средства и сервисы.

Распространённым способом является использование систем контроля версий для организации текстов программ студентов в одном репозитории. Преподаватель добавляет студентов в приватный репозиторий, где для каждого студента создана директория. Студент создаёт свою ветку проекта и добавляет преподавателя проверяющим к своим merge-requests. После успешной сдачи работы она добавляется в общий репозиторий.

Обучение программированию включает в себя также обучение написанию легко-читаемых, безопасных программ, а также использованию современных систем непрерывной интеграции и оценки (code review) программ. В данной статье предлагается описание последовательности задач по проверке лабораторных работ, организованной при помощи современных средств контроля версий и непрерывной интеграции.

Для минимизации списывания среди студентов следует изолировать тексты программ каждого студента в приватном репозитории, к которому при этом должен иметь доступ преподаватель. Для этих целей хорошо подойдёт недавно добавленная в систему контроля версий GitHub функциональность для учебных заведений — GitHub Classroom. При предоставлении преподавателем подтверждения своего статуса, платформа бесплатно предоставляет возможность неограниченного создания приватных репозиториев, а также организации их внутри классов соответственно группам [1].

Несмотря на изоляцию текстов программ студентов, дополнительным шагом пресечения прямого списывания может выступать проверка на плагиат. Система должна накапливать тексты программ и, основываясь на результатах сравнения их с проверяемой программой, оценить работу на плагиат. Такая проверка не должна пропустить работы отличающиеся, например, только именами переменных или функций.

Чтобы уменьшить количество ручных проверок, нужно, как минимум проводить предварительную проверку программ на компиляцию. Для этого к системе контроля версий возможно подключить систему сборки. Для GitHub это GitHub Actions, позволяющие задать последовательность действий для событий внутри репозитория [2]. Последовательность шагов задаётся на достаточно простом языке конфигурации в .YAML файле [3]. Для контроля сроков выполнения сроки сдачи могут быть добавлены как критерий продолжения тестов. Для более тщательной фильтрации некачественных работ возможно использование различных ключей компиляции, например, запрещающие устаревшие функции языка.

Следующим этапом проверки может выступать статический анализ текста программы. Такой анализ может включать в себя проверки дублирования кода, именования классов, переменных, методов, соблюдение размеров функций, излишнего ветвления кода и множество других [4]. Для статического анализа программ на Java в рамках учебного курса предполагается использование свободно распространяемых средств, например, PMD.

В случае, если к лабораторной работе студент должен написать модульные тесты, их прохождение должно также быть условием перехода на следующий уровень проверки. Данный шаг может быть добавлен в GitHub Actions с обязательным условием прохождения. Также в случае предъявления к лабораторной работе требований по покрытию кода модульными

тестами, возможно использование утилиты Jacoco, которая позволяет выполнять проверку на своих серверах и анализировать результат уже в GitHub Actions [5].

Последним этапом программа должна проходить тестирование на различных входных данных, если это возможно по условию задания. Для этого можно использовать контейнер, в котором запущен сам GitHub Actions, и полученный на этапе сборки проекта его артефакт. Тогда будет необходимо указать в самом тексте .yaml файла входные данные и проверку результата. Для упрощения этого шага возможно организовать все тесты в одном bash-скрипте. Лучшим способом будет использовать собственный, выложенный на DockerHub docker-контейнер, в котором будет настроено тестовое окружение, набор тестов. Отдельным этапом можно выполнить скрытые от внешнего наблюдателя при помощи шифрования тесты, для запуска которых требуется ключ. Ключ шифрования преподаватель может сохранить в разделе secrets» настроек проекта. Сохранённый таким образом ключ может быть использован как параметр при запуске контейнера, однако не будет виден ни в настройках проекта, ни в процессе выполнения. Если у преподавателя имеются собственные вычислительные мощности, настроить скрытые тесты можно как webhook на сервер преподавателя с возвратом отчёта о прохождении. Например, для этих целей хорошо подходит Jenkins – это свободно распространяемое ПО, имеющее интеграцию с GitHub [6].

В случае прохождения всех предварительных проверок лабораторная работа переходит к этапу приёмки преподавателем. Для связи с автором работы можно использовать встроенную в систему контроля версий функцию комментирования кода.

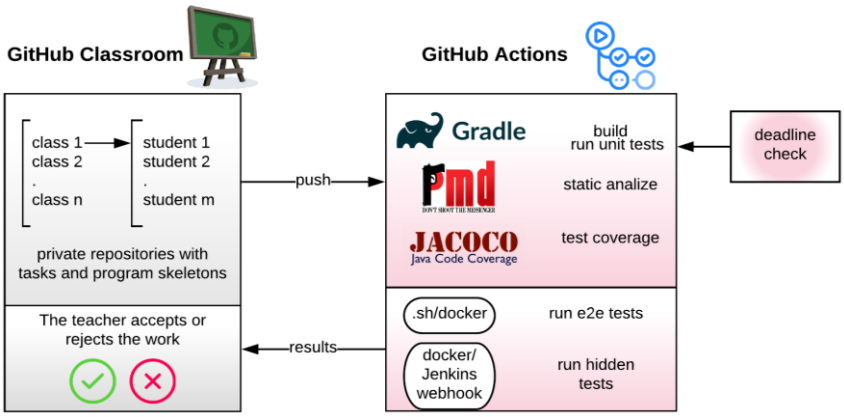


Рисунок 1 – Архитектура системы

Недостатком такой системы является ограничение использования статусом PRO на GitHub, который выдаётся студентам только на время обучения. Также дополнительной сложностью может оказаться добавление секретных ключей для настройки скрытых тестов.

Система, сформированная в соответствии с описанными принципами будет являться полностью бесплатной за счёт учебных лицензий, предоставляемых сервисами, а также использования свободно распространяемого ПО. Реализация такой последовательности проверки позволит повысить качество программ студентов, автоматизировать их приёмку и даст преподавателю возможность уделять больше времени непосредственно ответам на вопросы учащихся.

ЛИТЕРАТУРА

1. Страница помощи GitHub Classroom [Электронный ресурс] URL: https://classroom.github.com/help
2. Документация GitHub Actions [Электронный ресурс] URL: https://help.github.com/en/actions
3. Документация языка YAML [Электронный ресурс] URL: https://yaml.org/spec/1.2/spec.html
4. Документация библиотеки статического анализа PMD [Электронный ресурс] URL: https://pmd.github.io/pmd-6.22.0/index.html
5. Описание использования Jacoco в GitHub Actions [Электронный ресурс] URL: https://github.com/codecov/codecov-action
6. Описание добавления Jenkins как webhook для GitHub Actions [Электронный ресурс] URL: https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project/

УДК 621.319

И. О. Селиванов (4 курс бакалавриата)
И. Г. Черноруцкий д.т.н., профессор

ОБНАРУЖЕНИЕ ПОДДЕЛЬНЫХ ВИДЕОИЗОБРАЖЕНИЙ ЧЕЛОВЕЧЕСКИХ ЛИЦ ПРИ ПОМОЩИ МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ НА ОСНОВЕ ВРЕМЕННЫХ ХАРАКТЕРИСТИК В ВИДЕОКАДРАХ

Целью работы является разработка приложения для обнаружения поддельных видеоизображений человеческих лиц при помощи методов глубокого обучения на основе временных характеристик в видеокадрах.

Приложение получает на вход видеоряд в формате mp4, и ПО на основе временных характеристик видеокадров выносит решение – поддельное это видео или нет.

Алгоритм работы приложения построен на том, что нейронные сети не могут точно воспроизвести лицо другого человека на всем промежутке видео, и даже если различить разницу на глаз невозможно, то при помощи методов глубокого обучения возможно научить обнаруживать различия скрытых параметров лица.

Приложение состоит из двух модулей: предварительная обработка видео (обнаружения, обрезания и выравнивания обнаруженного в кадрах лица) и классификация изображений на поддельные и оригинальные при помощи свёрточных и рекуррентных нейронных сетей.

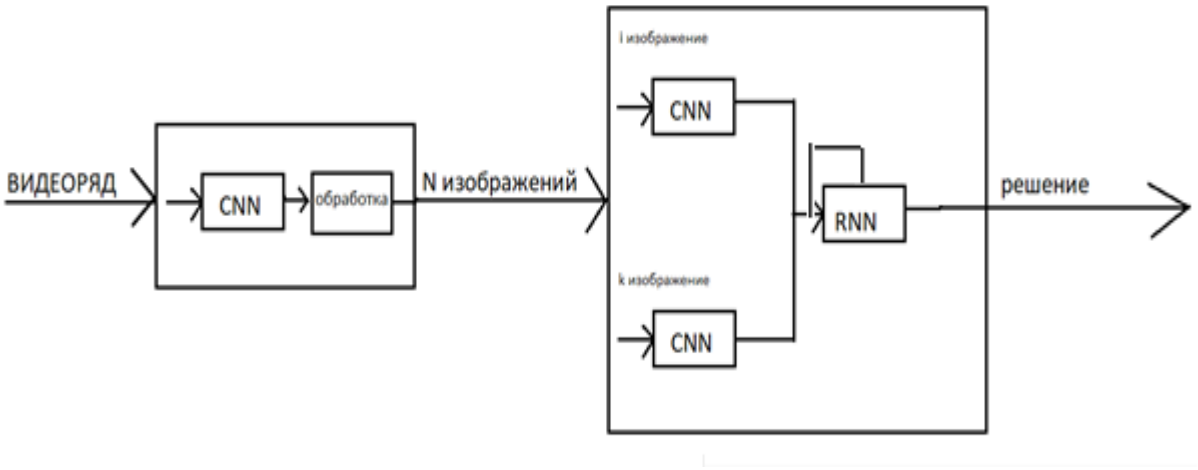


Рисунок 1 – Архитектура приложения.

Особенности реализации:

В качестве свёрточной нейронной сети в первом модуле была использована уже обученная MTCNN, покадрово принимающая на вход стоп-кадры и возвращающая координаты, длину и ширину описывающего лица прямоугольника для каждого лица, использующиеся для последующей обработки кадра при помощи opencv.

Второй блок принимает на вход обработанные изображения, и при помощи все того же MTCNN (без последних полносвязных слоев) выделяет признаки для каждого изображения лица и попарно сравнивает их при помощи GRU архитектуры рекуррентной сети и выносит решение для каждой пары, которое подается на вход вместе со следующей парой признаков и тем самым аккумулирует решение и по окончанию выносит вердикт на основе всех попарно сравненных изображений.

УДК 004.42

А. И. Станкеев (4 курс бакалавриата)
А. В. Самочадин, к.т.н., профессор

МУЛЬТИМОДАЛЬНЫЙ ЧАТ ДЛЯ КОРПОРАТИВНОЙ СИСТЕМЫ

Работа представляет собой часть проекта, разрабатываемого в СПбПУ, по созданию платформы для повышения эффективности деятельности компании на основе технологии гибридного интеллекта. Для организации коммуникации и взаимодействия между сотрудниками используется модуль чата. Также в нём происходит общение с модулем бота – ключевой частью системы. Упрощенная схема работы проиллюстрирована на рис. 1.

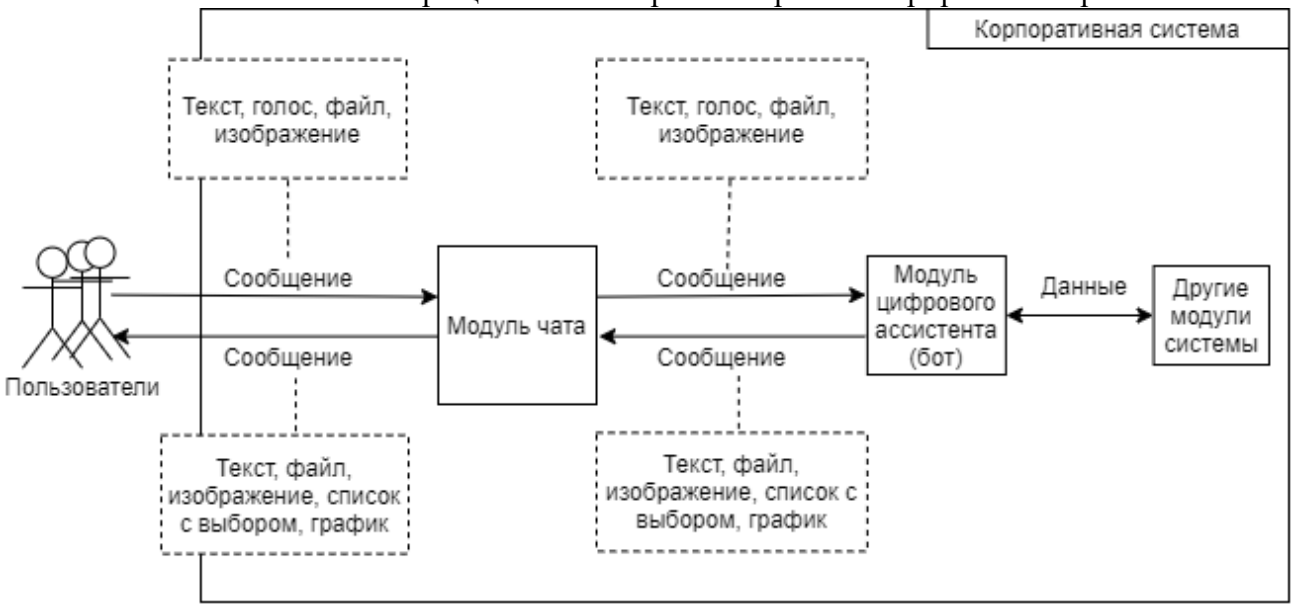


Рисунок 1 – Схема взаимодействия пользователей с корпоративной системой.

Технологические требования к разрабатываемому модулю:
– возможность локальной работы (без подключения к Интернет);
– поддержка мультимодальности (возможность взаимодействия с помощью сообщений разных типов);
– поддержка мультиязычности;
– клиент-серверная архитектура;
– взаимодействие между клиентом и сервером происходит посредством протокола HTTP, WebSocket и STOMP.

Мультимодальность обеспечивается поддержкой нескольких типов сообщений: текстовое, голосовое, произвольный документ, изображение, график и список с одиночным или множественным выбором.

Модуль должен обеспечивать создание групповых чатов. В группу, кроме пользователей, могут входить боты, обеспечивающие доступ к информации, которая хранится в информационной системе. Также бот может по команде предоставлять запрошенную информацию в виде графиков, взаимодействовать со сторонними сервисами (YouTrack, GitLab и т.д.).

В модуле чата есть несколько основных сущностей. Обобщенная диаграмма классов приведена на рис. 2

Для разработки Web-приложения был использован Angular (фреймворк для создания эффективных и сложных одностраничных приложений [Цит. по: 1]).

Разработанный модуль чата обладает следующей функциональностью:
– просмотр списка диалогов;
– просмотр диалога;
– создание нового личного/группового чата;

– общение в чате;
– общение с ботом;
– изменение настроек чата;
– добавление/удаление пользователей из чата;
– отображение разных типов сообщений;

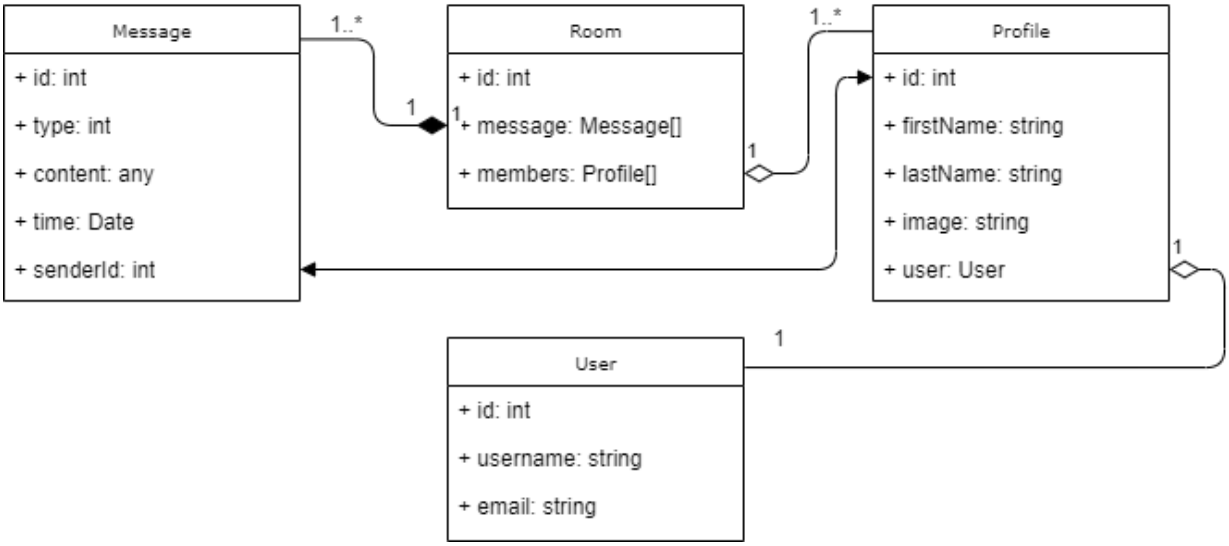


Рисунок 2 – Диаграмма классов модуля чата.

На данный момент модуль чата реализован, протестирован и поддерживает сообщения следующих типов: текстовые, голосовые, графики, файлы. Результаты тестирования показали, что он удовлетворяет предъявленным требованиям и обеспечивает корректную работу пользователей системы.

ЛИТЕРАТУРА

1. Introduction to the Angular Docs [Электронный ресурс]. – Режим доступа: https:// https://angular.io/docs– (Дата обращения: 05.03.2020).

УДК 004.896

Н. Стойкоски (4 курс бакалавриата)
И. А. Селин, ст. преподаватель

СЕМАНТИЧЕСКАЯ СЕГМЕНТАЦИЯ КАДРОВ АВТОДОРОЖНОГО ТРАНСПОРТА

Целью работы является разработки системы зрения для беспилотного автомобиля, позволяющая воспринять окружающее пространства из видеопотока. Данная система на входе должна принимать кадры из видеопотока, при этом предполагается что камера находится в передней части автомобиля и направлена вперед, глядя на пространство перед автомобилем. В качестве выхода система должна отдавать размеченные данные дороги.

Семантическая сегментация изображений означает разделение изображения на отдельные группы пикселей, области, соответствующие одному объекту с одновременным определением типа объекта в каждой области. В частности, это означает проставление меток каждому пикселю изображения – является ли он частью дороги или нет.

В качестве исходных данных присутствуют два набора. Первым является набор собранных вручную видео файлов с высоким разрешением со съемкой гоночной трассы. Вторым является открытый набор данных KITTI Vision Benchmark [1] для семантической

сегментации изображения дорожного транспорта. Примеры исходных данных можно увидеть на Рис. 1.

Рисунок 1 – Кадры из исходных видео файлов, образцы из набор данных KITTI.

Алгоритм на основе полносвёрточной нейронной сети для обнаружения краев.

Выделение границ на изображении включает в себя множество математических методов, которые направлены на определение точек на цифровом изображении, в которых яркость изображения резко изменяется, т.е. имеет разрывы. Результатом выделения границ является набор связанных кривых, обозначающих границы объектов, граней и оттисков на поверхности.

Для сегментации дороги был разработан алгоритм в начале которого выполняется обнаружение границ в исходном кадре с использованием полносвёрточной нейронной сети HED [2]. При этом была использована модель, предварительно обученная на датасете BSDS500 [3]. Алгоритм был реализован с помощью библиотеки pytorch на языке python. На выходе сети получается карта краев, к которой применяется порог для получения бинарного изображения.

Предполагается что автомобиль находится на дороге, по которой он движется и, соответственно, были выбраны фиксированные координаты на кадре в качестве начального пикселя для выполнения алгоритма floodfill библиотеки opencv. В результате имеется бинарная маска дороги т.е. для каждого пикселя входного кадра есть информация о том, является ли он частью дороги или нет. Работа алгоритма представлена на Рис. 2.

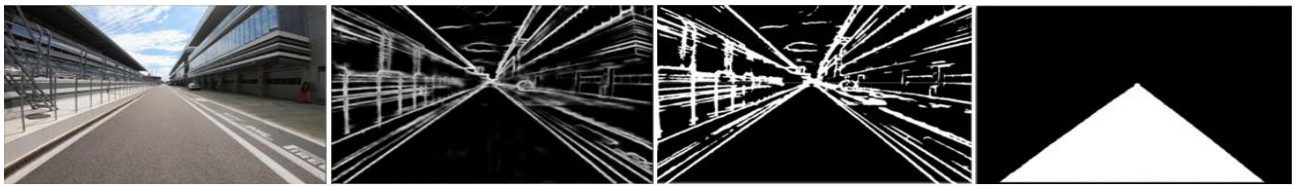


Рисунок 2 – Этапы работы алгоритма распознавания дороги

Данный алгоритм имеет недостаток в том, что полученная карта краев не всегда полностью закрывает дорогу в рамку и, следовательно, floodfill "вытекает" из ее границы, в результате чего получается неверная маска дороги. Для борьбы с этой проблемой были использованы ограничения по x, y при выполнении алгоритма floodfill.

Когда изображение дороги четкое, этот метод может успешно обрабатывать около 95% кадров, однако могут быть некоторые проблемы при крутых поворотах, туннелях и т.д. Это одна из причин, по которой предпочтительна обработка изображений и непосредственное получение маски дороги в нейронных сетях для семантической сегментации.

Алгоритм на основе полносвёрточных сетей для семантической сегментации

Полносвёрточные сети могут эффективно научиться делать прогнозы для попиксельных задач, таких как семантическая сегментация. Для решения данной задачи была выбрана модель полносвёрточной нейронной сети FCN [4], основанный на предобученной VGG16 [5], настроенной на задачу сегментации. Это одна из самых простых и популярных архитектур, используемых для семантической сегментации. Данная архитектура была реализована с помощью библиотеки pytorch на языке python.

Был создан датасет для обучения модели состоящий из N=246 пары изображения-меток с разрешением 960x640. Количество классов = 2 (дорога / не дорога). При этом были отобраны

и первоначально обработаны 116 образцов из датасета KITTI и 130 образцов, полученных из исходных видео файлов после их разметки в полуавтоматическом режиме.

Для обучения была использована функция потери torch.nn.BCEWithLogitsLoss. Обучение проводилось в среде Google Colaboratory, предоставляющий доступ к Jupyter ноутбукам, с возможностью использовать GPU Tesla K80. Были использованы следующие гиперпараметры: n_class = 2, momentum = 0, batch_size = 6, w_decay = 1e-5, epochs = 25, step_size = 50, lr = 1e-4, gamma = 0.5. Точность обучения: pix_acc = 0.983, mean IoU = 0.964, IoUs = [0.973 0.955].

Результирующие метки качественно выделяют дорогу в более 99% кадров даже при плохих условиях, таких как тени, препятствия, несколько полос, и являются достаточными для использования в дальнейших этапах систем зрения в беспилотных автомобилях. Результаты сегментации представлены на Рис. 3. Недостатком является низкая скорость обработки кадра, занимающая в среднем 0.33 секунды на GPU Tesla K80.

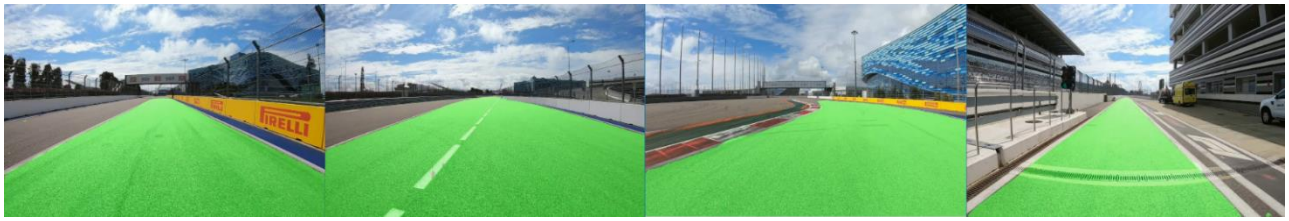


Рисунок 3 – Примеры результатов сегментации

ЛИТЕРАТУРА

1. Geiger, A, Lenz, P, Urtasun, R (2012a) Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Conference on computer vision and pattern recognition (CVPR).
2. Xie, Saining & Tu, Z. (2017). Holistically-Nested Edge Detection. International Journal of Computer Vision. 125. 1-16. 10.1007/s11263-017-1004-z.
3. Contour Detection and Hierarchical Image Segmentation P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011.
4. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
5. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

УДК 004.75

С. С. Ткаченко (2 курс магистратуры)
П. Д. Дробинцев, к.т.н., доцент

РАЗРАБОТКА ОБЛАЧНОГО ПРИЛОЖЕНИЯ ФИНАНСОВОГО И УПРАВЛЕНЧЕСКОГО УЧЕТА ДЛЯ МАЛОГО БИЗНЕСА, ИНДИВИДУАЛЬНЫХ ПРЕДПРИНИМАТЕЛЕЙ И САМОЗАНЯТЫХ ГРАЖДАН

Актуальность использования распределенных и облачных систем для обработки данных с каждым годом возрастает из-за возможности быстрого горизонтального масштабирования и потенциально более высокой отказоустойчивости подобных систем. В настоящее время на рынке существует ряд приложений, предоставляющих основанные на микросервисной (многомодульной) архитектуре облачные SaaS (Software-as-a-Service) решения, для использования в экономических процессах финансового и управленческого учета. Анализ данного сегмента рынка показал, что зачастую пользователи сталкиваются с проблемой недостаточной гибкости приложений. Лишь немногие решения предоставляют API, а интеграция дополнительного сервиса в существующий практически невозможна, что

приводит к необходимости использования сторонних средств и приложений - Excel, Google Tables, другие решений по финансовому и управленческому учету.

Целью данной работы является разработка системы в облаке, позволяющей использовать принципы масштабируемости и простой интеграции с внешними системами. Данная система предназначена для ведения экономического и финансового учета в сфере малого бизнеса и подразумевает возможность быстрого масштабирования и адаптации в условиях быстро меняющегося рынка и стремительного роста компании.

Само приложение можно условно поделить на несколько ключевых частей: инфраструктурная часть, базовая функциональная часть и интеграционная (модульная) часть. На рисунке 1 изображена концептуальная архитектура приложения.

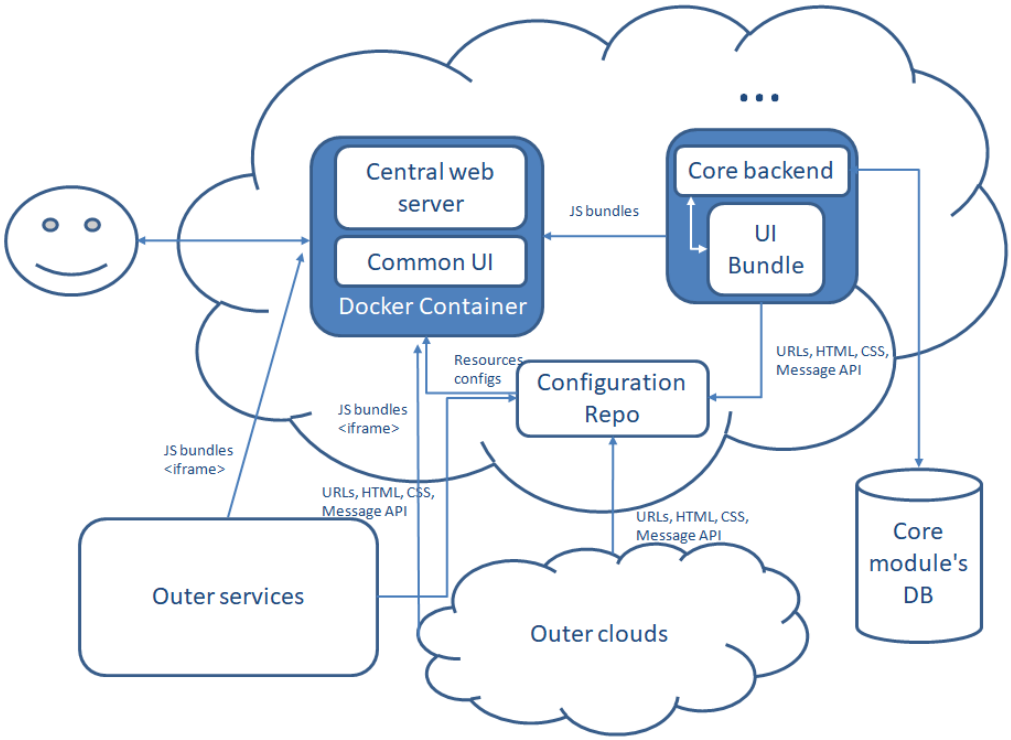


Рисунок 1 – Концептуальная схема архитектуры приложения в облаке.

Инфраструктурная часть представлена облачной платформой RedHat OpenShift Online, включающей расширенные конфигурации оркестратора Kubernetes и тонкую настройку процесса сборки образов ключевых Docker-контейнеров. Базовая функциональная часть состоит из нескольких микросервисов, каждый из которых отвечает за определенный подпроцесс финансового или экономического учета. В качестве базы данных используется PostgreSQL, в качестве инструментов реализации серверной части используется язык Kotlin, Spring Boot. В качестве основных технологий разработки пользовательского интерфейса используется язык TypeScript, фреймворк Angular, а также RxJS, NgRx. Ключевой концепцией взаимодействия сервисов между собой является понятие веб-компоненты.

В облаке существует специальный репозиторий конфигураций, в котором указываются все необходимые сервисы, которые планируется использовать в ходе работы приложения. В каждой конфигурации содержится HTML код элементов, пользовательский селектор – уникальный идентификатор компонента, а также указание всех возможных сообщений (Message API), которые используются для взаимодействия компонентов между собой. На рисунке 2 изображен макет главного экрана приложения. Вкладки APP1-APP4 символизируют интегрированные с приложением независимые сервисы. Вкладка APP1, например, может отвечать за анализ доходов/расходов за некоторый период, в то время как вкладка APP2 может отвечать за учет информации о контрагентах.

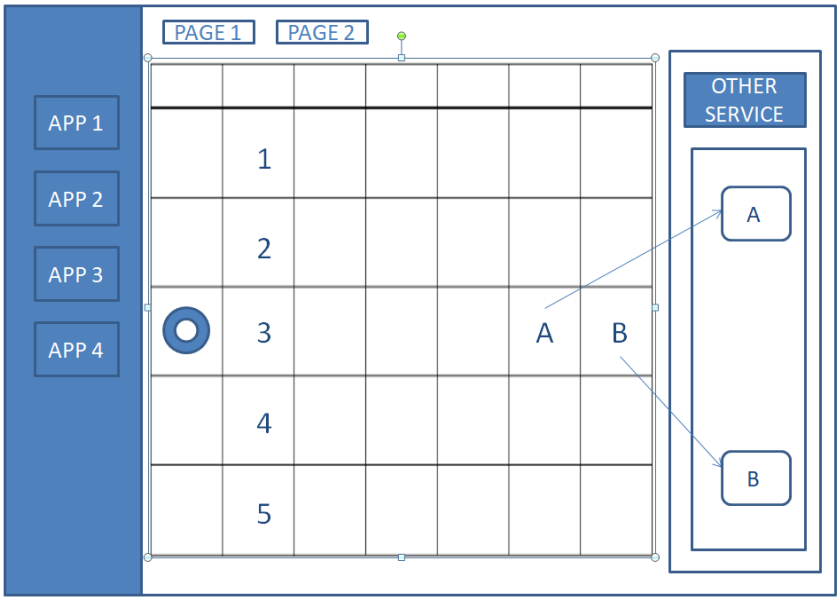


Рисунок 2 – Концептуальная схема работы приложения

Каждая вкладка на левой панели представляет собой пакет кода на языке JavaScript, который отвечает за запрос разрешений на отображение данной вкладки у микросервиса авторизации. Например, существуют роли администратора и пользователя с разными наборами доступных вкладок. При нажатии на каждую вкладку на левой панели открывается соответствующий ей функциональный экран. На рисунке 2 изображена таблица в качестве основного функционального компонента, при нажатии на некоторую строку этой таблицы есть возможность открыть дополнительный экран в виде всплывающего окна или полноразмерного представления с сохраняемым состоянием. Данная операция осуществляется за счет интерфейса сообщений (Message API), который генерирует события, перехватываемые другими компонентами. Таким образом, можно, например, вызвать метод из внешнего пакета JavaScript кода или выполнять HTTP GET-запрос и открывать внешний компонент как <iframe> - элемент, позволяющий встраивать внешние HTML страницы в разметку текущей страницы.

Подобный механизм взаимодействия между элементами обеспечивает быструю интеграцию и возможность горизонтального масштабирования основных компонентов за счет использования микросервисов в облаке. Также данный подход позволяет создать единую точку входа для всевозможных внешних компонентов и выстроить произвольную архитектуру их взаимодействия.

УДК 004

Умар Диаби (2 курс магистратуры)
Н. В. Воинов, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ РАСПОЗНАВАНИЯ ВОЗГОРАНИЙ ПО ВИДЕОПОТОКУ НА ОСНОВЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Целью данной работы является изучение возможностей построения модели с фреймворком для глубокого обучения Keras и использование его последовательного API (англ. Sequential API) для построения модели, использующей depthwise separable convolution к решению задачи обнаружения возгораний по видеопотоку на основе сверточных нейронных сетей.

Данная система должна обеспечивать обнаружение возгораний путём обработки видеопотоков в режиме реального времени.

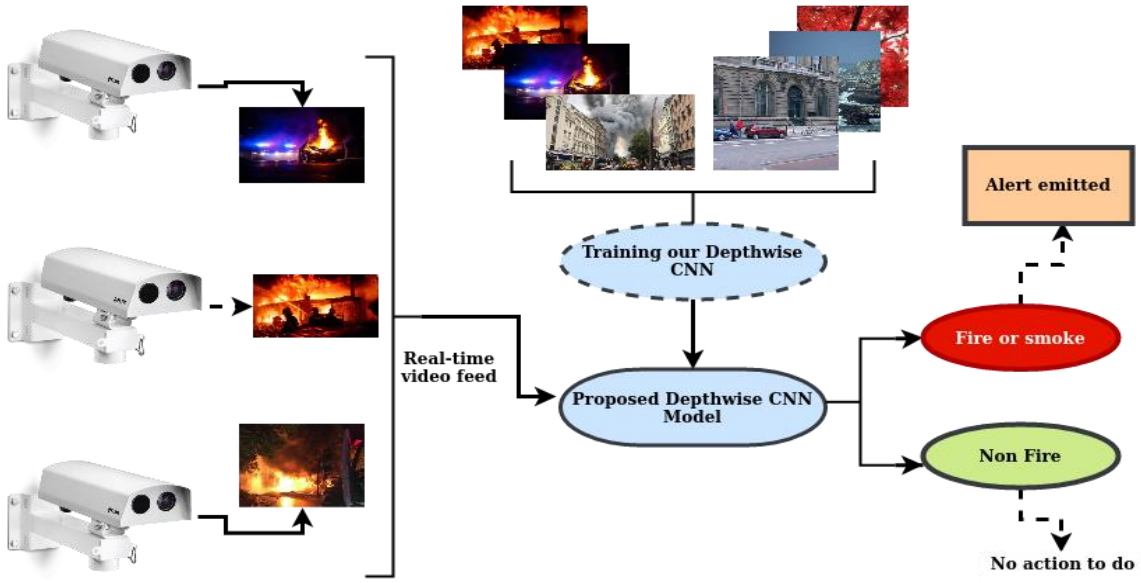


Рисунок 1 – Система распознавания возгораний по видео на основе сверточных нейронных сетей.

Особенности реализации модели:

Модель реализована с помощью фреймворка Keras и его sequential API. Этот API подходит для построения большинства моделей. Важным существенным изменением в нашей модели по сравнению с многими другими моделями, решающими такие задачи, является использование depthwise separable convolutions вместо стандартных сверток. Этот тип сверток делает сеть компактнее и существенно уменьшает количество операций, которые необходимо выполнить для вычисления активаций одного слоя. Они позволяют достигать интересных результатов в некоторых задачах, и нашей в том числе. Depthwise convolution можно представить следующим образом:

$$\hat{G_{k,l,m}} = \sum_{i,j} a \cdot \hat{K_{i,j,m}} \cdot F_{k+i-1,l+j-1,m}$$

где $\hat{K}$ обозначает ядро depthwise свертки , $\hat{G}$ выход карти признаков (англ. feature map) и a- это коэффициент канала. Этот коэффициент можно представить $[a_r, a_g, a_b]$, где a_r, a_g, a_bявляются неотрицательными числами.

Посмотрим, как это все выглядит на простом примере. Пусть приходится сворачивать изображение с 16 каналами стандартным свёрточным слоем с 32 фильтрами размерности 3×3. Этот свёрточный слой будет иметь $32 \times 16 \times 3 \times 3 = 4608$ весов. Далее сравниваем количество, полученное с тем, что получается в нашей модели с depthwise separable convolution. Поскольку идея depthwise separable convolution заключает в том, что мы сначала делаем сверту по каналам а потом 1×1 сверту, посмотрим сколько у нас весов будет в аналогичном же блоке с depthwise separable convolution. В первом шаге получаем $16 \times 32 \times 1 \times 1 = 512$весов и во втором шаге получаем $32 \times 3 \times 3 = 288$ весов и мы поучаем в итоге 800 весов, это количество весов намного меньше чем то что мы получили в обычном сверточном слое.

Наша модель построенна следуюшим образом : Мы определили наш первый depthwise convolution блок сверточным слоем с 16 каналами размером 7×7 каждое. Здесь большие размеры ядра позволяют с одной стороны уменьшить пространственные размеры входного изображения, а с другой стороны обнаружить цветные пятна, которые дают серьезное первое указание на присутствие огня или дыма в изображении. Следует нелинейная функция активации ReLU, обеспечивающая быстрое выполнение процесса обучения. Затем мы добавляем нормализацию для стабилизации процесса обучения и пулинг или слой субдискретизации для уменьшения размерности входного изображения. Далее однако мы определяем второй наш depthwise convolution блок сверточным слоем с 32 каналами

размерности 3 × 3 каждое. Далее мы применяем два набора сверточных слоев с 64 каналами. Эти два слоя, сложенные вместе позволяют модели найти интересные признаки. Далее мы добавляем два набора полносвязных слоев перед тем как добавить наш классификатор softmax с нашими двумя классами.

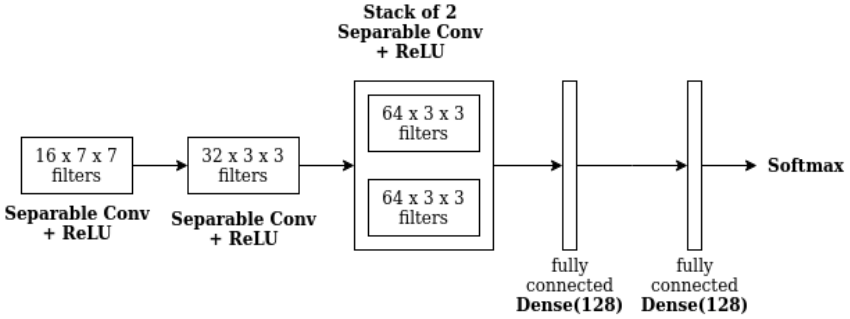


Рисунок 2 – Модель.

Для обучения и оценки нашей модели мы использовали небольшой набор данных, содержащий около 5000 изображений из Google изображений. Чтобы справиться с проблемой переобучения, мы использовали некоторые методы увеличения объема данных, чтобы помочь нашей модели успешно обобщить новые данные. И, наконец, мы обучили нашу модель на 50 эпохах.

Ниже, результаты наших экспериментов

Таблица 1 – Полученные результаты

Classes	**Fire**	**NonFire**
Precision	0.87	0.86
Recall	0.86	0.88
F1-score	0.86	0.87
Support	611	639

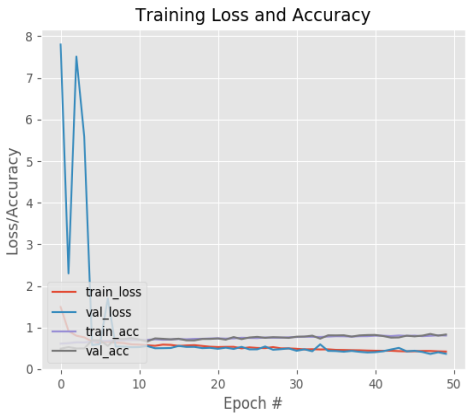


Рисунок 3 – Обучение нашей модели.

УДК 004.89

Ж. В. Федорова (2 курс магистратуры)
С. А. Молодяков, д.т.н., профессор

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ РЕКОМЕНДАЦИЙ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ С ПРИМЕНЕНИЕМ НЕЙРОННЫХ СЕТЕЙ

Современному человеку очень сложно найти нужную информацию в непрерывном потоке меняющихся данных. Применение классических средств поиска и систематизации не может полностью удовлетворить потребности пользователя: невозможно просмотреть все материалы, чтобы выбрать для себя подходящие. В связи с этим стало появляться все больше рекомендательных систем, которые ориентированы на нахождение информации, наиболее полно удовлетворяющей интересы пользователя.

Рекомендательные системы - это программные средства, которые пытаются предсказать какие объекты (фильмы, музыка, книги, новости, веб-сайты и т. д.) будут интересны пользователю, если имеется определенная информация о его предпочтениях. Существуют две

основных стратегии рекомендательных систем: коллаборативная фильтрация и фильтрация содержимого.

Коллаборативная фильтрация строит модель, основываясь на прошлом поведении пользователя (выбор контента или оценивание) и мнении других пользователей, которые в прошлом делали такой же выбор. Фильтрация содержимого, основываясь на предварительно помеченных дискретных характеристиках, делает подбор элементов, со схожими характеристиками. Современные системы обычно объединяют один или несколько подходов в гибридную систему.

В настоящее время существует множество музыкальных рекомендационных систем, в основном они встроены в более крупные системы стриминга: Spotify, YouTube Music, Pandora, iTunes, Яндекс Музыка. Они построены преимущественно на коллаборационном или гибридном подходе. К сожалению, в таких системах нет возможности управлять расчетом рейтинга и выбирать пользователей, на основе которых система могла бы делать свои рекомендации. Также нет возможности предварительно выбирать ограниченный список претендетов, среди которых система делает свой поиск. Более того, некоторые из наиболее развитых в этой области сервисов не доступны в России.

Целью данной работы является разработка системы рекомендаций музыкальных композиций, которая может на основе предпочтений определенной группы пользователей и характеристик музыкальных композиций определить наиболее оптимальные варианты по заданным критериям.

Для построения такой системы требуется разработать следующие модули: модуль пользовательского интерфейса, модуль бизнес логики, модуль анализа музыкальной композиции.

Задача модуля анализа - строить предположения о характеристиках данного ему аудиофайла. В рамках данной работы были выбраны следующие характеристики:

–Danceability: танцевальная способность, описывает, насколько дорожка подходит для танцев на основе сочетания музыкальных элементов, включая темп, стабильность ритма, силу удара и общую регулярность. Значение 0.0 является наименее танцевальным, а 1.0 - наиболее танцевальным.

–Energy: Энергия - мера от 0.0 до 1.0, представляет собой меру восприятия интенсивности и активности. Как правило, энергичные треки кажутся быстрыми, громкими и шумными.

–Valence: мера от 0.0 до 1.0, описывающая музыкальную позитивность, переданную треком.

В контексте машинного обучения в первую очередь требуется сформировать унифицированную выборку, на основе которой будет проходить обучение. Существует ограниченный список ресурсов с музыкальными данными для машинного обучения, наиболее широко используемые это GTZan и Million Songs dataset. Они предоставляют предразмеченные по жанрам аудио ресуры. Так-как, в решаемой задаче конечным результатом модуля анализа должен быть набор характеристик аудио, было принято решение создать датасет самостоятельно с использованием ресурсов сайта о машинном обучении Kaggle.

Следующий шаг в разработке модуля анализа – трансформация цифровых представлений музыки в частотную спектограмму, которая представляет собой визуальное представление спектра частот во времени. Преобразование звукового сигнала в спектограмму основано на преобразовании Фурье. Наиболее важными параметрами, используемыми в преобразовании, являются - длина окна, которая определяет окно времени для выполнения преобразования Фурье, и длину скачка, которая представляет собой число выборок между последовательными кадрами. Типичная длина окна для этого преобразования составляет 2048, что составляет около 10 мс, самый короткий разумный период, который может различить человеческое ухо.

Следующий этап в построении модуля анализа – разработка архитектуры нейронной сети, подбор оптимальных параметров для увеличения точности оценки, тестирование и

анализ результата. Планируется произвести анализ рекуррентных и сверточных архитектур. Выбор сверточных архитектур обусловлен тем, что спектограмма является изображением, разные виды музыки существенно отличаются на спектограмме. Рекуррентные архитектуры же хорошо работают на задачах предсказания временных рядов, когда текущее значение зависит от предыдущего значения. Спектограммы имеют временную составляющую, и рекуррентные архитектуры могут намного лучше идентифицировать краткосрочные и долгосрочные временные особенности аудиоряда. Для общения между модулями выбран архитектурный стиль REST (Representational State Transfer), т.к. это наиболее простой и легковесный протокол взаимодействия web-приложений.

В докладе представлены не только архитектура и алгоритм взаимодействия модулей, но результаты применения системы.

ЛИТЕРАТУРА

1. Dwivedi, P. (2020, 03 18). Using CNNs and RNNs for Music Genre Recognition. Retrieved from medium: https://towardsdatascience.com/using-cnns-and-rnns-for-music-genre-recognition-2435fb2ed6af
2. GTZAN Genre Collection. (2020, 03 18). Retrieved from marsyas: http://marsyas.info/downloads/datasets.html
3. kaggle. (2020, 03 12). Retrieved from Spotify Audio Features: https://www.kaggle.com/tomigelo/spotify-audio-features
4. Kaitila, J. (2017). A content-based music recommender system.
5. Million song dataset. (2020, 03 18). Retrieved from millionsongdataset: http://millionsongdataset.com/
6. Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook

УДК 004.414.22

К. С. Фигловский (4 курс бакалавриата)
Т. А. Вишневская, ст. преподаватель

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ОРГАНИЗАЦИИ УПРАВЛЕНИЯ ВЗАИМООТНОШЕНИЯМИ С КЛИЕНТАМИ ПОД ОПЕРАЦИОННУЮ СИСТЕМУ ANDROID

Не секрет, что в настоящее время одной из самых популярных является сфера услуг, в которой по некоторым источникам (https://utmagazine.ru/posts/10567-ekonomika-rossii-cifry-i-fakty-chast-14-sfera-uslug) работает порядка 65% занятых в экономике нашей страны граждан. Одна из важных задач этой сферы - работа с клиентами, требующая записи и использовании информации о взаимодействии с ними. Кто-то организует процесс ведения записей по старинке - через блокнот или ежедневник, но в век повсеместной распространённости смартфонов все большую популярность набирают мобильные решения.

Целью работы является разработка мобильного приложения под операционную систему Android для организации работы с записью клиентов. Приложение должно предоставлять следующие возможности: создание одинарных и повторяемых событий; добавление дополнительной информации различного формата к событиям (текст, изображение, контакт и т.д.); просмотр событий на интерактивном календаре; импорт данных из сторонних приложений; получение уведомлений о приближающихся событиях; обработка входящих звонков, с возможностью записи разговора, работа без доступа к сети

Сравнение с аналогами

По запросу «запись клиентов» в Google Play, официальном магазине Android приложений, самыми популярными решениям с похожей целью являются Bumpix и ГномГуру (по 100000+ установок). В таблице 1 приведено сравнение между указанными приложениями и разрабатываемым

Таблица 1 – Сравнение с аналогичными решениями

Название \ Характеристика	Цена	Требуется регистрация	Оффлайн режим	Обработка звонков	Рассылка уведомлений клиентам
Разрабатываемое приложение	Бесплатно	Нет	Да	Да	Да
ГномГуру	Платное	Да	Да	Да	Да
Bumpix	Платное	Да	Да	Да	Да

Структура приложения

Для реализации будет использован один из популярных архитектурных подходов – «Clean Architecture». В Clean Architecture код разделен на несколько уровней, с единственным правилом зависимости: внутренний уровень должен не зависеть от внешних уровней. Это значит, что зависимости должны указываться внутри каждого уровня, чтобы не было зависимостей между уровнями (слоями).

На рис. 1 изображена схема вышесказанного [8]

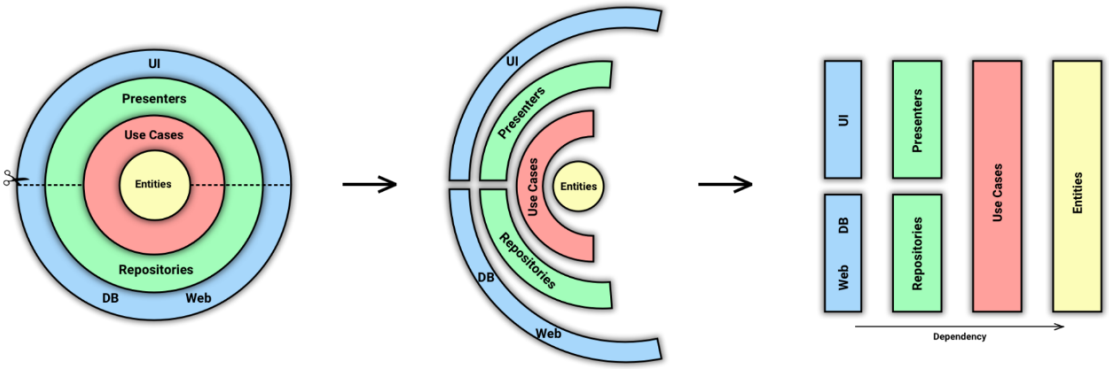


Рисунок 1 – Схема Clean Architecture

В соответствии с выбранным подходом, на рис. 2 получим следующую структуру приложения

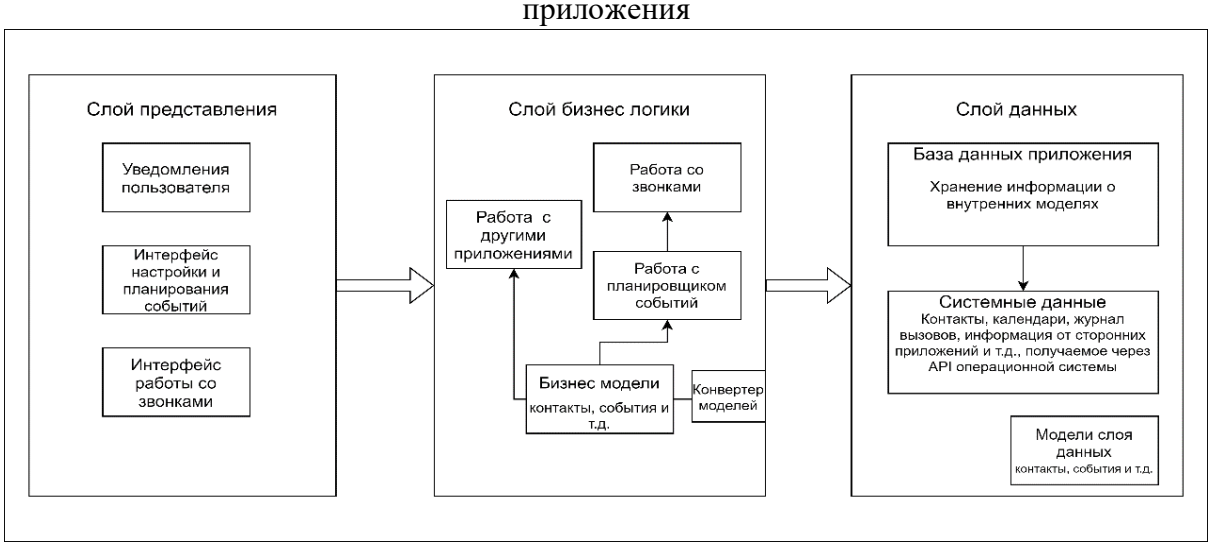


Рисунок 2 – Структура приложения

Используемые технологии

Разработка будет вестись на Kotlin - статически типизированном языке программирования, работающем поверх Java Virtual Machine [5]. Для построения пользовательского интерфейса будет использован стандартный Android Java Framework [1] и вспомогательные библиотеки Android Jetpak [2]. В качестве СУБД используется SQLite [4]

через Room Persistence Library [3]. Для тестирования будут использованы Junit5, Mockito [6] и Kaspresso [7]

ЛИТЕРАТУРА

1. Android User Interface & Navigation. URL: https://developer.android.com/guide/topics/ui
2. Android Jetpack. URL: https://developer.android.com/jetpack
3. Room Persistence Library. URL: https://developer.android.com/topic/libraries/architecture/room
4. SQLite. URL: https://www.sqlite.org/index.html
5. Kotlin. URL: https://kotlinlang.org/
6. Fundamentals of Testing. URL https://developer.android.com/training/testing/fundamentals
7. Kaspresso is a great UiTest framework. URL: https://github.com/KasperskyLab/Kaspresso
8. Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series)

УДК 004.054

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

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ СИСТЕМЫ СОЗДАНИЯ И ОБРАБОТКИ ТЕХНОЛОГИЧЕСКИХ МАРШРУТОВ МЕЛКОСЕРИЙНОГО ПРОИЗВОДСТВА

В настоящий момент тестирование является неотъемлемой частью процесса разработки программного обеспечения. Одним из аспектов является проверка соответствия между ожидаемым и реальным поведением системы. В рамках данной работы будет рассмотрена разработка серии автоматических тестовых сценариев для приложения TechnologARM [4].

Тестируемое приложение написано на языке JavaScript с использованием фреймворка ReactJS. TechnologARM нацелен на частичную автоматизацию мелкосерийного производства. Основным функционалом приложения является создание технологических маршрутов обработки, от заготовки до технологических операций [2]. В связи с большим количеством возможностей для редактирования, предоставляемых пользователю, была поставлена задача создания набора тестовых сценариев, покрывающих взаимодействие пользователя с интерфейсом программы.

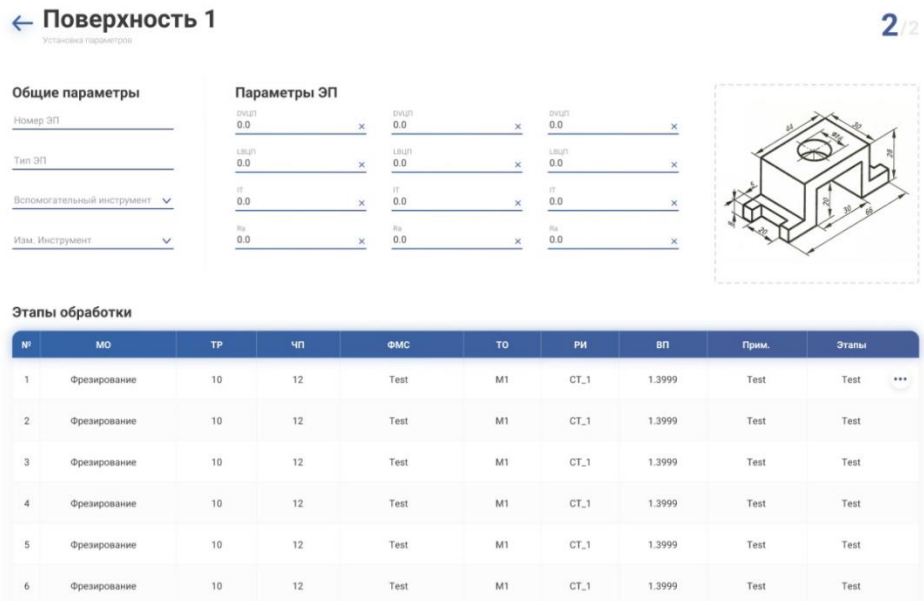


Рисунок 1 – Пример страницы TechnologARM

Первой частью работы стал выбор подходящего фреймворка. Основным требованием стала возможность записывать сценарии в реальном времени, исходя из этого выбор делался из трёх возможных фреймворков: Jest [5], React Test Utils [6], и Selenium Web Driver [3]. В ходе сравнения функционала фреймворков было выяснено, что Jest, работающий через механизм снимков, некорректно показывает себя в тестах. React Test Utils основан на Jest, поэтому от него также пришлось отказаться. В результате для разработки был выбран Selenium Web Driver, зарекомендовавший себя как оптимальный фреймворк для тестирования комплексных web-приложений.

Как известно, Selenium[1] – инструмент, специально разработанный для проведения автоматизированного тестирования веб-приложений в различных браузерах и на различных платформах. Следует заметить, что не только веб-приложения, а любые рутинные действия, выполняемые в браузере, могут быть автоматически протестированы с помощью Selenium. Одним из недостатков Selenium'а является высокая чувствительность к положению курсора мыши: при записи сценария и клики на крайние границы элементов возможны случаи падения тестов. Поэтому все сценарии записывались с учетом этой проблемы – для её решения использовалась центровка кликов по элементам. Всего было написано шестнадцать автоматических сценариев, каждый из которых приводит пользователя к конечному результату. Автоматическими сценариями было покрыто все React приложение, начиная со страницы авторизации, и заканчивая выбором и созданием новых деталей. При единовременном запуске всех тестов некоторые тесты не проходили проверку, из-за багов, которые позже были успешно устранены.

Следует обратить внимание, что Selenium позволяет создавать полноценные коллекции тестов, а также поддерживает группировку и перемещение тестов между группами, что позволяет создавать наборы тестов, направленные на тестирование конкретной функциональности.

Также отдельно можно отметить простоту модификации существующих тестов, а именно возможность ручного редактирования записанного сценария при необходимости, что значительно упрощает обновление существующих сценариев в случае изменения программы.

Таким образом, в процессе работы была создана полноценная группу автоматических сценариев проверяющих основную функциональность существующего React-приложения, что повышает надёжность системы и позволяет избежать ошибок при дальнейшей разработке.

ЛИТЕРАТУРА

1. Краткое описание Selenium // — URL: https://habr.com/ru/post/152653/
2. Хрусталева И.Н. Модель технологической подготовки производства в условиях единичного и мелкосерийного типов производства. // Неделя науки СПбПУ: материалы научной конференции с международным участием. Институт металлургии, машиностроения и транспорта. Ч.2. – Спб.: Изд-во Политех. ун-та. 2017. С.312-314.
3. Справочник по Selenium Web Driver// [официальная документация]/— URL: https://www.selenium.dev/selenium-ide/ (дата обращения:15.12.2019).
4. Тестируемое React приложение // — URL: https://gitlab.com/Vetkin/react-technolog/
5. Официальная документация Jest // — URL: https://jestjs.io/docs/en/getting-started
6. Официальная документация React Test Utils // — URL: https://ru.reactjs.org/docs/test-utils.html

УДК 519.6:004.94

С. В. Шульц (1 курс магистратуры)
О. Н. Петров, к. т. н., доцент

РАЗРАБОТКА СИСТЕМЫ ТРЕХМЕРНОГО МОДЕЛИРОВАНИЯ МОРСКОГО ВОЛНЕНИЯ И КАЧКИ КОРАБЛЯ

Целью работы является разработка Web-3D-приложения, для визуального моделирования реального морского волнения по заданному пользователем частотному (климатическому) спектру, и качки корабля на нем. Приложение предназначено для визуальной оценки параметров волнового процесса (таких как угол волнового склона, влияние ветра на геометрию волновой поверхности и т. п.), параметров качки корабля заданного проекта, его остойчивости и может использоваться как с целью оценки и нормирования остойчивости (т. е. обеспечения безопасности мореплавания), так и в тренажерных системах.

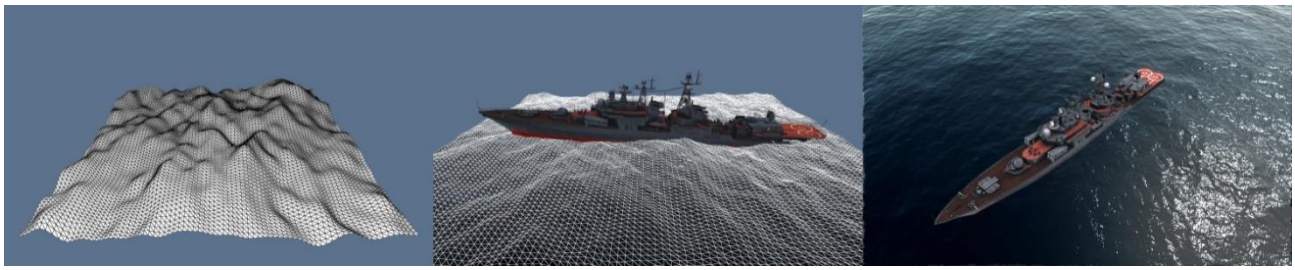


Рисунок 1 – Система моделирования в работе: режим моделирования волнения на примере спектра Неймана; режим моделирования качки на примере БПК проекта 1155.1; тренажерный режим.

Моделирование пространственно-временного поля морского волнения $y(\omega,x,z,t)$ по заданному энергетическому спектру процесса волнения $S(\omega)$ осуществлено с помощью модифицированной модели Лонге-Хиггинса:

$$y(\omega,x,z,t)=\sum_{i=1}^{N}\sum_{j=1}^{M} r_{ij}\cdot\cos\left(k_x\cdot x\cdot\cos\alpha_j+k_y\cdot\sin\alpha_j\cdot z-\omega_i\cdot t+\varepsilon_{ij}\right), \quad (1)$$

где r_{ij} – амплитуды гармоник, вычисляемые по энергетическому спектру $S(\omega)$;

ε_{ij} – случайные фазы, равномерно распределенные по интервалу $[0,2\pi]$;

(k_x,k_y) – волновые числа, связанные между собой через коэффициент трехмерности $\gamma_t=\dfrac{k_x}{k_y}$.

Наиболее важными с точки зрения безопасности мореплавания являются бортовая, килевая и вертикальная качка. Динамика корабля по этим видам качки на нерегулярном волнении задается системой нелинейных дифференциальных уравнений:

$$\begin{cases}\zeta''+2\mu_\zeta\cdot\zeta'+n_\zeta^2=\chi_\zeta\cdot n_\zeta^2\cdot y_0\\ \theta''+2\mu_\theta\cdot\theta'+n_\theta^2=\chi_\theta\cdot n_\theta^2\cdot y_x'\\ \psi''+2\mu_\psi\cdot\psi'+n_\psi^2=\chi_\psi\cdot n_\psi^2\cdot y_z'\end{cases}, \quad (2)$$

где $\zeta(t)$, $\theta(t)$, $\psi(t)$ – вертикальное смещение корабля, угол бортовой качки (крена) и угол килевой качки (дифферента) в момент времени t;

2μ, n^2, χ – коэффициенты демпфирования, квадраты частот собственных колебаний**,** поправочные коэффициенты для каждого вида качки (уникальны для проекта корабля).

Приложение разработано на базе самых современных на данный момент web-технологий и языков: WebGL и GLSL, HTML5 и CSS3, TypeScript. Приложение кроссплатформенно, может работать без интернет-соединения (PWA), а кроме того, оптимизировано для работы на маломощных устройствах, в том числе и на мобильных: ноутбуках и планшетах.

Моделирование геометрии водной поверхности производится на видеоускорителе в вершинном шейдере, поскольку эта задача прекрасно распараллеливается. Моделирование волнения начинается с создания плоской сетки, разбитой на заданное количество треугольников. Затем каждая из точек сетки (вершин) независимо и параллельно с другими проходит цикл вычислений по модели Лонге-Хиггинса. С целью увеличения производительности в тренажерном режиме реализованы уровни детализации (LOD): чем дальше от наблюдателя, тем сильнее упрощаются вычисления геометрии поверхности. На точность моделирования это не влияет, но при ряде условий позволило увеличить производительность в 1,5-2 раза. Окрашивание поверхности происходит во фрагментном шейдере по модели Фонга с учетом отражений атмосферы и других объектов. Мелкие возмущения водной глади реализованы с помощью карты нормалей. Для улучшения качества изображения применяется тональная компрессия с использованием HDR. Наложение сетки на окрашенную поверхность реализовано с помощью барицентрических координат.

Моделирование качки выполняется на центральном процессоре с целью балансировки нагрузки между GPU и CPU. В первую очередь вычисляются все коэффициенты, входящие в уравнения качки, и зависящие от процесса волнения. Затем последовательно производится численное интегрирование уравнений качки классическим методом Рунге – Кутты. Полученные углы и смещения применяются к 3D-модели корабля.

Архитектура приложения состоит из двух частей: серверной и клиентской. Серверная часть представлена набором статических файлов, которые хранят описания предустановленных спектров, ракурсов камеры, …, 3D-моделей (JSON). А также сами 3D-модели (glTF) и шейдеры моделирования волнения (GLSL). Клиентская часть состоит из трех подсистем: моделирования волнения и качки, работы со спектрами и графического интерфейса пользователя. Пользовательский интерфейс разработан на HTML5 и CSS3 без использования каких-либо библиотек; он позволяет изменять параметры моделирования такие как: моделируемый спектр, скорость ветра, коэффициент трехмерности, проект корабля, курсовой угол, детализация океана, режим отображения сетки, атмосферы и многое другое. При необходимости панель параметров можно свернуть (см. рисунок 1). Вся логика на клиенте написана на языке TypeScript. Подсистема для работы со спектрами содержит средства для «разбора» спектров, заданных таблично и аналитически. Тип спектра определяется автоматически по переданной строке, содержащей его данные. Парсингом каждого вида спектров занимается отдельный класс, поэтому система легко масштабируется: достаточно разработать класс для соответствующего способа задания спектра.

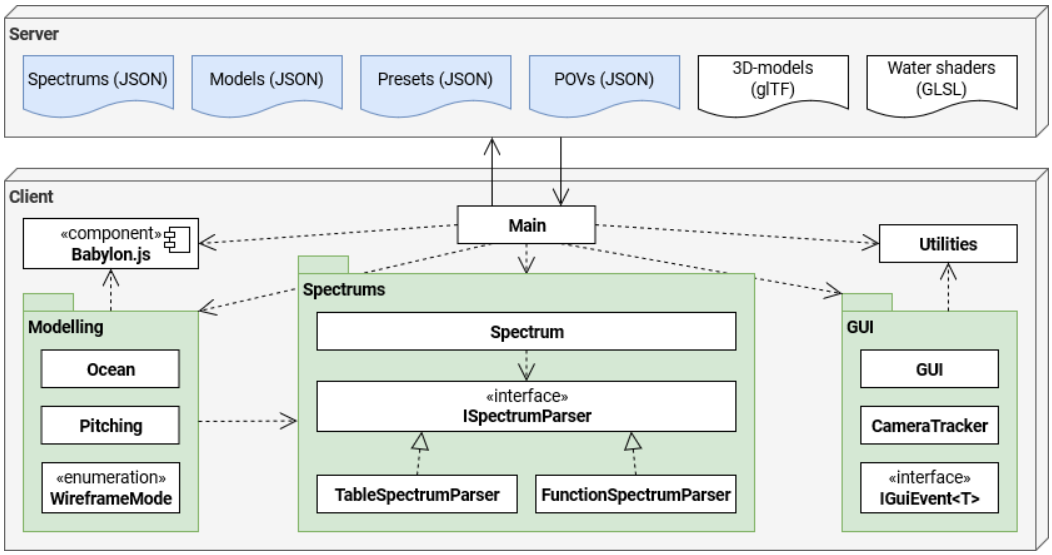


Рисунок 2 – Примерный вид архитектуры приложения с отображением разбиения на подсистемы.

Примененные технологии показали себя с наилучшей стороны, поскольку даже на устройствах без дискретной графики показали достойную производительность в 50-60 FPS.

Секция «Программная инженерия: инструментальные средства и технологии проектирования и разработки»

УДК 681.3.06

А. С. Воскресенский (3 курс бакалавриата)
В. А. Семёнова-Тян-Шанская, к.т.н., доцент

СОВРЕМЕННЫЙ ХАКИНГ: ПАРСИНГ САЙТОВ

Цель данной работы - рассмотрение существующих методов парсинга сайтов и защиты от него. Под сайтом здесь подразумевается некий интернет-ресурс, размещающий какую-либо информацию в открытом доступе. Парсеры — это боты, запрограммированные на парсинг сайтов и извлечение из них информации. Чаще всего их целью становятся сайты, содержащие интересную с точки зрения бизнеса информацию (например, отслеживание цен и ассортимента товаров-конкурентов или проведение маркетинговых исследований для мониторинга цен конкурентов для работы с ценообразованием своих товаров). Известно, что парсеры настраиваются для: распознавания уникального HTML; извлечения и преобразования контента; хранения очищенных данных; извлечения из API.

Для этого, со стороны сервера на клиент должна приходить HTML-разметка, которую парсер преобразует в объектную модель документа (DOM), используя некоторые средства (например, для языка C# - библиотека AngleSharp, для языка Python – библиотека Scrapy). Из DOM он «вытаскивает» интересующую информацию при помощи, к примеру, CSS-селекторов, выражений XPath или стандартных средств языка, на котором написан парсер.

Иногда сервер может присылать неполную разметку или вообще её не присылать. Так может произойти в следующих случаях:

- Архитектура фронтенда сайта подразумевает генерацию разметки прямо на клиенте при помощи исполняемых скриптов
- Динамическое именование CSS-классоа интересующих элементов вёрстки (может существенно замедлить/сделать невозможным подбор корректного селектора).

В первом случае, на помощь может прийти веб-драйвер (например, ChromeDriver или GeckoDriver для Firefox), который физически откроет окно браузера и пошлет запрос. Получив ответ, он исполнит присланные скрипты и, скорее всего, сгенерирует HTML-код, понимаемый парсером.

Во втором же случае, помогает API сайта, который можно найти, отследив запросы и ответы между клиентом и сервером (например, при помощи стандартных средств разработчика Google Chrome или другого современного браузера). Тогда информацию с сервера можно будет получать напрямую возвращаемыми по API JSON-файлами. Для нахождения и изъятия нужной информации из JSON-файла, чаще всего используют библиотеку Newtonsoft, которая есть для большинства современных языков (в некоторых языках, она даже включена в стандартную библиотеку). Десериализуется JSON в объект, доступ к которому осуществляется как к обычному объекту данного языка.

Однако сайт могут нарочно защищать от несанкционированного доступа к информации. Например, для доступа к API может быть необходим некий токен из генерируемой клиентом разметки. В таком случае, перед тем, как посылать запросы, придётся инициировать сессию при помощи веб-драйвера, достать из разметки этот токен и сохранить его для формирования запросов. Некоторые токены имеют короткую «длину жизни», что при длительном парсинге может вызвать необходимость получать этот токен заново (например, при написании сайта на

ASP.NET, при условии использования CSRF token, существует полуавтоматическая возможность задания времени жизни в cookie - MaxAge). Также, для доступа к желаемой информации может потребоваться регистрация и залогинивание (тоже с ограниченным сроком сессии).

Чаще всего, парсинг одного сайта производится с одного компьютера и, следовательно, запросы идут с одного адреса. На серверной части это легко отследить и перестать на некоторое время отвечать на запросы с данного IP-адреса – “забанить”. “Бан” можно получить за слишком частые запросы, слишком большое их количество или за другой вид подозрительной активности.

Нередко для отслеживания активности (посещений сайта, запросов) разработчики пользуются сервисами metrica.yandex, которые предоставляют возможности «интеллектуального» отслеживания ботов. Помимо отчетов по активности на сайте, среди них есть: вебвизор, карта скроллинга, аналитика форм, подробные записи действий посетителей на сайте: движения мышью, прокручивание страницы и клики. В работе рассмотрены способы защиты от “бана”: выставляя задержки – «таймауты» – перед отправкой каждого запроса или используя proxy-сервисы для обмана анализаторов активности. Для «обмана» метрики, нередко приходится имитировать действия человека.

Также, анализаторы пользовательской деятельности могут присутствовать и на клиенте. Если есть необходимость использования веб-драйвера, то может появиться необходимость обманывать такие анализаторы, добавляя скрипту веб-драйвера, помимо запросов, ненужные клики мышью, избыточные прокрутки страниц и ошибки при вводе.

Нередко можно столкнуться с тем, что разработчики сайтов, руководствуясь экономическими соображениями, не хотят сразу банить пользователей из-за подозрительной активности. Уже не нова технология отсеивания ботов - капча, рекапча, аудиофрагменты… И почти на каждую такую технологию существует интернет-сервис – чаще всего, платный, – помогающий с их обходом или решением.

Парсинг сайтов - важная часть современного экономического мира. Например, без мониторинга цен конкурентов, предпринимателям приходилось бы, как и годами ранее, извлекать всю необходимую информацию вручную. Технологии не стоят на месте и находят свое отражение даже в такой специфической области, как парсинг, затрагивая не только его, но и его антагониста - защиту. А симметричное развития миров технологий и бизнеса - залог экономического и, как следствие, технологического роста.

УДК 004.051

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

ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛЕНИЯ МЕТРИКИ COLLECTWGS ПАКЕТА PICARD TOOLS

Исследование генома — одна из наиболее перспективных и быстро развивающихся отраслей науки в наше время. Основополагающей частью любого исследования является анализ и интерпретация данных, полученных в ходе эксперимента, и использование эффективных вычислительных алгоритмов в данной области играет существенную роль [1]. Picard Tools является одним из современных наборов инструментов, реализующих эти алгоритмы. Данная работа посвящена его усовершенствованию.

Инструмент Collect WGS Metrics используется для оценки результатов ранее произведенной обработки данных – проценте ранее выровненных на геном ридов, прошедших базовые фильтры качества [2]. Эта метрика является важной для понимания пригодности полученного промежуточного результата для дальнейшей работы и вероятности получения в итоге достоверного и однозначного результата всего эксперимента [3].

В ходе изучения текущего алгоритма были выявлены следующие особенности:
Алгоритм считывает данные с диска и обрабатывает их частями. Время работы чтения можно представить как t_r, время обработки - t_p , число фрагментов – n. Тогда общее время работы можно представить как $t = (t_r + t_p) * n$. Код выполняет работу в одном потоке (рис. 1). Независимо от числа ядер на используемой машине, скорость его работы изменяться не будет, что не дает использовать потенциал современных компьютеров.

Рисунок 1 – Схема работы исходного алгоритма

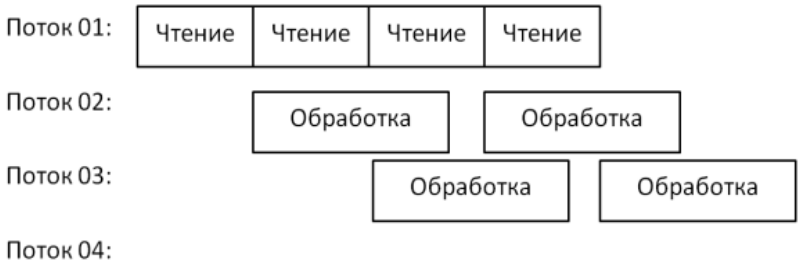


Рисунок 2 – Схема работы усовершенствованного алгоритма

Так как время обработки существенно выше времени чтения, было решено сконцентрироваться на ускорении его работы. В этом случае схема имела бы вид как на рис. 2, а время выполнения сократилось бы до $t = t_r * n + \frac{t_p * n}{M}$, где M – число задействованных потоков [4].

Алгоритм итератора (чтения):

- Чтение объектов класса SamRecord вложенным SamIterator
- Создание объекта LocusInfo для каждой позиции в Reference
- В LocusInfo скапливаются объекты RecordAndOffset для каждого выровненного на позицию SamRecord рида, создаётся новый объект RecordAndOffset.
- Если начало выравнивания следующего считанного SamRecord рида больше, чем позиция первой собранной LocusInfo, позиция считается законченной и передаётся следующему методу.

Алгоритм обработки данных (метод addInfo(LocusInfo info)):

- Одна позиция (LocusInfo) обрабатывается один раз
- Проход по собранным объектам RecordAndOffset, ища эту позицию.
- Для каждого RecordAndOffset мы получаем baseQuality на текущее смещение в риде.
- Проверка на наложение.
- Результаты работы метрики CollectWgsMetrics складываются в объект WgsMetricsCollector.

В ходе тестирования и просмотра логов было замечено, что основная часть выполняется в методе doWork() последовательно для каждого рида в цикле while (iterator.hasNext()). Теперь в этом же цикле последовательно считывается пачка ридов и отправляется на обработку в отдельный поток.

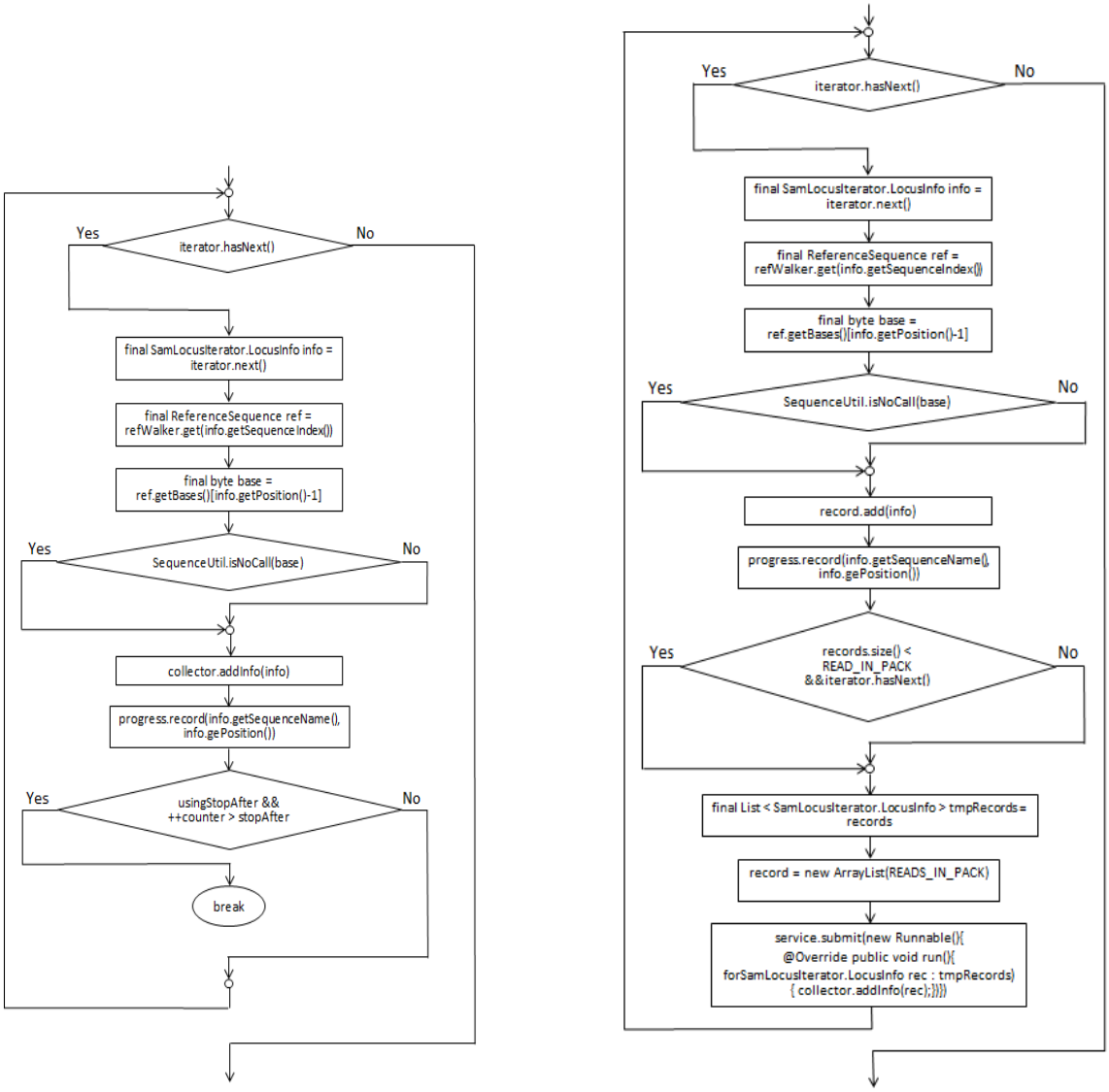


Рисунок 3 – Алгоритм работы метрики до и после распараллеливания

В ходе тестирования было достигнуто уменьшение времени работы над тестовым файлом в 250Гб с 12 до 5 часов (с 16 ядрами) и 4 часов (с 48 ядрами). Т. е. его удалось сократить минимум в 2,5 раза.

Наиболее перспективными направлениями усовершенствования рассматриваются:

- Улучшение алгоритма для использования на одном узле.
- Добавление автоматического выбора программой оптимальных параметров запуска.
- Добавление настроек, позволяющих пользователю подбирать параметры запуска для конкретной задачи.

ЛИТЕРАТУРА

1. Michael Janitz Next-Generation Genome Sequencing: Towards Personal Medicine, – Hoboken : Wiley-Blackwell, 2008. – 282p.
2. Broad Institute, Picard Toolkit documentation, https://broadinstitute.github.io/picard/, 2019.
3. Р. Дурбин, Ш. Эдди, А. Крог, Г. Митчисон Анализ биологических последовательностей, – М.: Регулярная и хаотическая динамика, 2006. – 480с.
4. Brian Goetz, Java Concurrency in Practice, – Boston : Addison-Wesley Professional, 2006. – 424с.

УДК 004.054

А. А. Игнатьев (4 курс бакалавриата)
Т. В. Коликова, ст. преподаватель

СРАВНЕНИЕ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ НА ПРИМЕРЕ РАЗРАБОТКИ КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ

Целью работы является сравнение паттернов проектирования для разработки кроссплатформенных приложений на примере Clean Architecture и Пользовательской архитектуры.

На сегодняшний день для создания качественного приложения, нужно использовать паттерны проектирования, ведь используя их, вы уменьшаете количество проблем на стадии создания архитектуры, и повышаете читаемость кода, что уменьшит затраты в будущем. В данной работе решено наглядно проверить, как архитектура влияет на сроки разработки, масштабируемость проекта и создания тестов. На примере интернет-магазина “Офелия”[3] сравним пользовательскую архитектуру и clean architecture[1].

Первоначально приложение имело пользовательскую архитектуру. Она имела следующую структуру папок:

- bloc - логика работы состояния приложения
- models - модель данных
- providers - интерфейс для соединения с API
- screens - корневые экраны приложения
- themes - интерфейс для быстрого изменения темы
- widgets - основные компоненты приложения, используемые на экранах

Такая структура хорошо подходит для небольшого проекта, однако, написание тестов и масштабируемость такой архитектуры ограничивается. По окончанию разработки первой версии приложения стало понятно, что работа с данной архитектурой сильно замедляет разработку приложения, по причине того, что архитектура недостаточно понятна для новых сотрудников в команде, и появляются сложности с тестами.

В качестве альтернативы, приложение было полностью переписано с использованием так называемой “Чистой архитектуры”. Clean Architecture объединила в себе идеи нескольких других архитектурных подходов, которые сходятся в том, что архитектура должна:

- быть тестируемой;
- не зависеть от UI;
- не зависеть от БД, внешних фреймворков и библиотек.

Это достигается разделением на слои и следованием Dependency Rule (правилу зависимостей).

Данная архитектура сняла ограничения на масштабируемость приложения, за счет более понятной структуры проекта, особенно для новых членов команды:

- Core - глобальные переиспользуемые компоненты
  - bloc
  - client
  - error
  - network
  - pages
  - themes
  - usecases
  - util
  - widgets

- features - локальные используемые в пределах одной страницы

[pages…]
data
datasources
models
repositories
domain
entities
repositories
usecases
presentation
bloc
pages
widgets

Таким образом, по окончанию разработки приложения на новой архитектуре, получилось добиться существенного повышения производительности команды, неограниченную масштабируемость проекта и полное покрытие тестами всех частей приложения, чего нельзя было добиться технически на предыдущей архитектуре, и стандартизацию кода.

ЛИТЕРАТУРА

1. Официальный блог Роберта Мартина(Uncle bob) // [официальная документация]/— URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (дата обращения:16.03.2020).
2. Habr// [научные статьи]/— URL: https://habr.com/ (дата обращения:16.03.2020).
3. Мобильное приложение Интернет-магазин “Офелия” // [ссылка на приложение React Native]/— URL: https://apps.apple.com/ru/app/%D0%BE%D1%84%D0%B5%D0%BB%D0%B8%D1%8F-%D0%BE%D0%B4%D0%B5%D0%B6%D0%B4%D0%B0-%D0%B4%D0%BB%D1%8F-%D1%81%D0%BE%D0%B1%D0%B0%D0%BA/id1464103988 (дата обращения: 16.02.2020).

УДК 004.042, 004.054

Н. В. Макаров (4 курс бакалавриата)
Т. В. Коликова, ст. преподаватель

РАЗРАБОТКА И ТЕСТИРОВАНИЕ КРОСПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ DART

Целью работы является исследование возможностей программной платформы языка программирования Dart [1] для создания мобильных приложений под разные платформы с единой кодовой базой и тестирование полученного программного продукта.

В данной работе было реализовано мобильное приложение для интернет-магазина одежды для собак Офелия. Приложение было разработано всего за 1.5 месяца командной всего из 2 человек и опубликовано в Google Play и AppStore. Приложение позволило значительно расширить клиентскую базу магазина и захватить мобильный рынок в данной сфере продаж.

Объём рынка мобильных приложений растёт с каждым годом, сегодня мобильная реальность стала для пользователей и бизнеса насущной необходимостью. Существует всего две основные мобильные операционные системы: Android и iOS. В большинстве случаев, для этих двух платформ требуются разные программные продукты, а, следовательно, и разный персонал для его создания. Ресурсы на поддержание двух отдельных команд разработки есть далеко не у каждого бизнеса, а малый бизнес вообще не сможет позволить себе такое.

Выходом из данного положения послужила технология Flutter, язык программирования Dart. С помощью данной технологии разработчикам нужно создать и поддерживать только одну кодовую базу для двух разных мобильных платформ. Таким образом для бизнеса открываются более дешевые варианты разработки мобильных приложений.

Важным этапом в разработки мобильного приложения является его тестирование. В выбранной технологии есть все инструменты для этого: от юнит тестов, до полной автоматизации процесса сборки и тестирования приложения.

Хочется отметить, что данная технология не является единственной на рынке для создания единой кодовой базы для двух платформ, но по ряду причин является единственным эффективным вариантом, потому что: поддерживается и разрабатывается напрямую компанией Google, что обеспечивает долгий жизненный цикл и хорошую маркетинговую компанию, в тоже время данная технология является полной open-source, что позволяют любому желающему предлагать и вносить изменения, а так же имеет большую поддержу в сообществе разработчиков.

Использование фреймворка Flutter позволило значительно сократить время разработки, так как мы получили единую кодовую базу для двух мобильных платформ сразу. Как особенность данного фреймворка – свой собственный графический движок, удалось создать единый дизайн одновременно на двух платформах. Так же, в самом языке Dart есть встроенная система тестирования, которая позволяет намного быстрее писать тесты одновременно с основным рабочим кодом. Следует заметить, что данный фреймворк активно развивается и уже есть возможность с минимальными изменения готовой кодовой базы сделать desktop и веб приложение.

Хочется заметить, что в ходе выполнения данной работы несколько раз возникала необходимость на внесение изменений в исходных код фреймворка Flutter. За счёт структуры фреймворка внесение изменений оказалось очень простым и работало сразу на двух платформах. Основной рабочий блок в данном фреймворке – это виджет. Создание собственных виджетов не вызывает никаких трудностей, даже в случае сложных динамических компонентов. Во много это достигается из-за единой кодовой базы и открытой программной платформы.

ЛИТЕРАТУРА

1. Официальный сайт Dart // [официальная документация]/— URL: https://dart.dev/ (дата обращения:16.03.2020).
2. Официальный сайт Flutter Framework // [официальная документация]/— URL: https://flutter.dev/ (дата обращения:16.03.2020).

УДК 004.4'41

Н. А. Манаков (2 курс магистратуры)
И. В. Никифоров, к.т.н., доцент

МЕТОДИКА ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ РАБОТЫ ВИРТУАЛЬНЫХ JAVA-МАШИН ПРИ ИСПОЛЬЗОВАНИИ JIT КОМПИЛЯЦИИ

Высокоуровневый язык программирования Java используется во многих сферах разработки: приложения для персональных компьютеров, мобильная разработка под Android-платформу, веб- и серверные программы, инструменты для разработки. Язык Java является платформо-независимым языком, так как программный код транслируется в промежуточное представление, выполнение которого осуществляется на специализированной виртуальной машине — Java Virtual Machine (далее, как JVM). Процесс выглядит следующим образом:

1. исходный текст программы на языке Java поступает на вход компилятору, транслирующий текст в байт-код — последовательность байтов;
2. на старте приложения происходит загрузка всех классов в память JVM;

3. начинается интерпретация байт-кода JVM-интерпретатором, то есть трансляция байт-кода в машинные инструкции, исполняемые на процессоре;
4. также параллельной с интепретацией происходит компиляция байт-кода JIT-компилятором (Just-in-time) — компилятором, компилирующий некоторые методы в машинный код при определенных условиях.

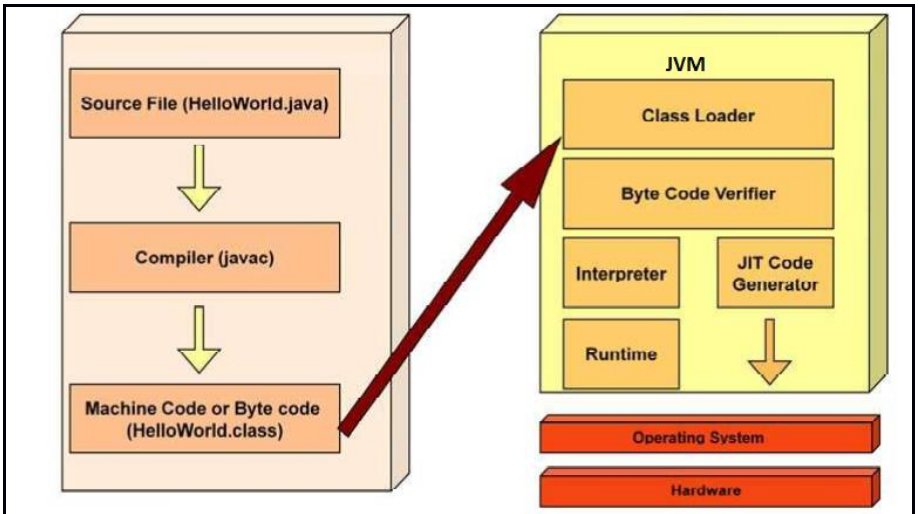


Рисунок 1 – Процесс исполнения программы на языке Java

Так как процесс интепретации и JIT-компиляции происходит во время выполнения программы и трудно предсказуем, а также тот факт, что JVM совершает различные оптимизации и деоптимизации с байт-кодом для ускорения его исполнения, то нельзя производить измерение производительности JVM стандартными способами.

Целью данной работы является создание методики оценки производительности работы виртуальных Java-машин для различных сценариев использования и при различных условиях запуска.

Имплементация JVM может выполнять с байт-кодом следующие оптмизаци:

- компиляция байт-кода в машинный код после достижения порогового значения при интерпретации — если ветвь байт-кода была интерпретирована достаточное количество раз без изменений в последовательности и содержимого инструкций, то JIT-компилятор компилирует данный байт-код в машинные инструкции и сохраняет их в кэше JVM, чтобы в дальнейшем использовать готовый нативный код;
- on-stack replacement — переход к использованию машинного кода в цикле во время процесса интерпретации байт-кода, а не во время повторного вхождения в точку входа в цикл;
- устранение неиспользованного кода, то есть не выполнение такого кода, результат которого не оказывает влияние на результат программы;
- method inlining — подстановка тела метода в стек исполнения доступа к методу через адрес в памяти;
- мономорфное преобразование вызова — конвертирование виртуального вызова метода в непосредственный вызов метода, используется для уменьшения накладных расходов по поиску объекта, метод которого должен быть вызван;
- развертывание циклов — выполнение инструкций в цикле как последовательность повторяющихся блоков инструкций, используется для уменьшения операций ветвления при выполнении команд на процессоре;
- escape analysis — динамическое определение области видимости указателей, их расположение в регистрах процессора или в стеке;

Эти и другие оптимизации тратят время при выполнении программы, то есть при замере скорости работы JVM необходимо быть осведомленным о данных оптимизациях для корректного замера производительности.

Также комбинация вышеуказанных оптимизаций может привести к деоптимизации — повторной комплияции байт-кода на основе новой информации, поступившей JVM. Это также вносит время в исполнение программы.

Типичные случаи использования программ на Java позволяют вделать следующие режимы, для которых имеет смысл замерять производительность виртуальных машин:

- скорость старта приложения — время, затраченное приложением на переход в состояние готовности работы;
- скорость выполнения программы с прогревом — предварительный запуск программы е количество раз для выполнения всех JVM-оптимизаций, измерение производительности осуществляется после прогрева, для серверных приложений, выполняющихся неопределенно долго;
- скорость выполнения программы без прогрева — выполнение программы «как есть», подходит для программ, выполняющие какую-либо операцию и завершающие свое выполнение после;

Многие реализации JVM спецификации поддерживают использование кэша с классами — общие классы, которые могут быть использованы несколькими экземплярами JVM при работе.

Одна не маловажная особенность JVM — сборщик мусора. Это программа, удаляющая из памяти ссылки на участки памяти, по которым никогда не будет произведено обращение из программы.

Для измерения производительности необходимо запускать выполнение какой-то определенной программы, которая могла бы считаться типичной, эталонной. В такой программе должны присутствовать следующие операции: операции ввода/вывода, синхронизация и одновременной выполнение потоков, работа с памятью, передача данных между потоками, системные вызовы, использование нативных методов. Программа должна быть написана так, чтобы не произошла излишняя оптимизация ее байт-кода с помощью JVM.

Таким образом, методика оценки производительности виртуальных Java машин сводится к сравнению выполнения типичных программ со следующими комбинациями условий:

- различные режимы компиляции байт-кода;
- режим с прогревом или без;
- наличие или отсутствие кэша классов;
- дополнительное выделение памяти для JVM во время исполнения программы;
- различные сборщики мусора;

После всех запусков высчитывается коэффициент производительности в процентном соотношении для различных реализаций JVM.

УДК 004.4

Мулонде Филипе Педро Ду Нашсименту (4 курс бакалавриата)
С. А. Фёдоров, ст. преподаватель

ИСПОЛЬЗОВАНИЕ ARDUINO И ЯЗЫКА ПРОГРАММИРОВАНИЯ С ДЛЯ УМЕНЬШЕНИЯ ОБЪЕМА АППАРАТНОГО ОБЕСПЕЧЕНИЯ.

Целью работы является создание программатора Arduino EEPROM, аппаратного функционального блока, который можно использовать для замены больших сложных аппаратных схем на EEPROM.

Я собираю программируемый 8-битный компьютер с нуля на макетах, используя только простые логические элементы, один из модулей моего компьютера, это 8-битный десятичный дисплей, каждый десятичный дисплей требует аппаратного обеспечения, как на рисунке 1, для

моего 8-битного компьютера чтобы показывать числа от 0 до 255, мне понадобится 4 дисплея, затем мне понадобится эта схема ниже (рис. 1) 4 раза.

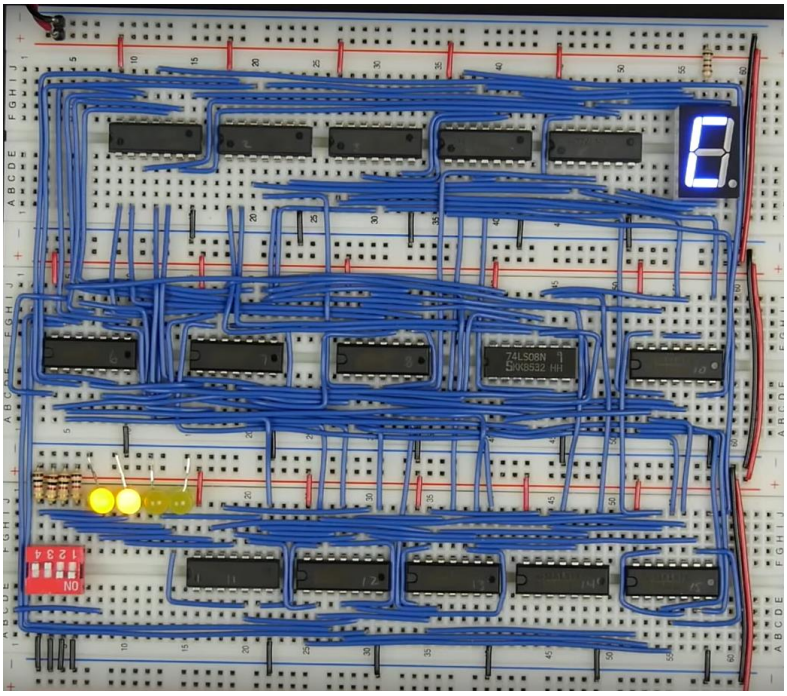


Рисунок 1 – Схема для одного сегмента дисплея

Но есть лучшее решение этой проблемы, которое может уменьшить объем работы, стоимость компьютера и его размер. Это оптимальное решение состоит в том, чтобы использовать Arduino и язык программирования C для программирования EEPROM, запрограммированной схемы EEPROM на рисунке 2 достаточно для замены схемы на рисунке 1.

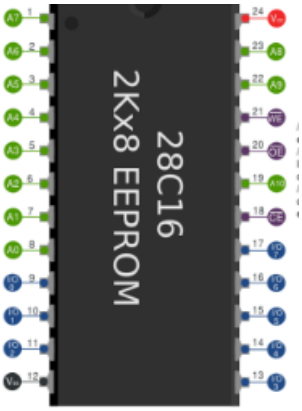


Рисунок 2 – EEPROM запрограммирован

EEPROM programmer

Это простая схема для программирования 28C16, 28C64, 28C256 и аналогичных параллельных EEPROM с использованием Arduino. Поскольку Arduino не имеет достаточно контактов для непосредственного управления всеми адресами, данными и линиями управления EEPROM, два сдвиговых регистра 74HC595 используются для 11 адресных линий (15 для 28C256) и линии управления разрешением выхода.

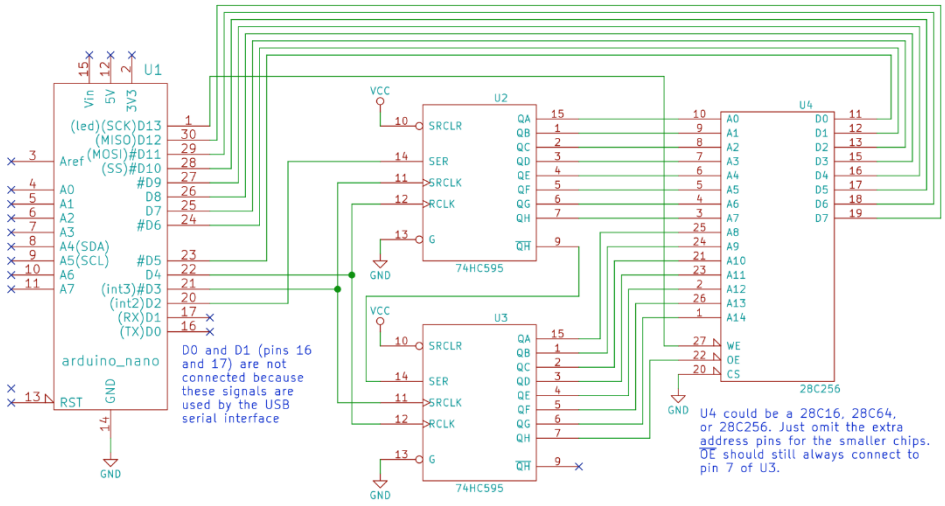


Рисунок 3 – Программатор EEPROM схема.

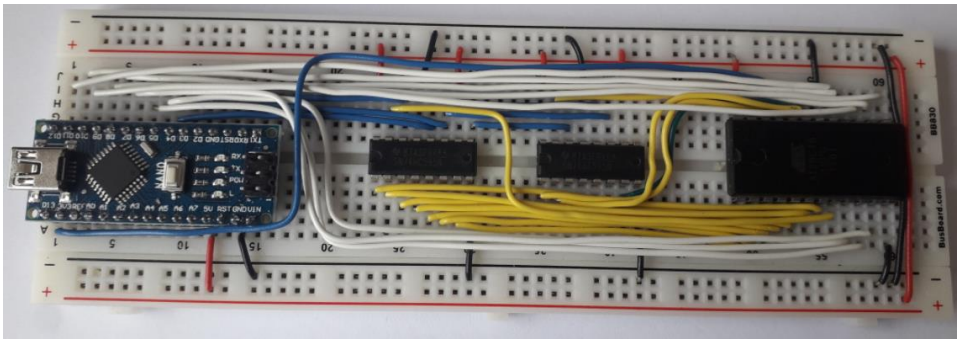


Рисунок 4 – реализация схемы программатор EEPROM.

Arduino Integrated Development Environment (IDE) - это кроссплатформенное приложение (для Windows, macOS, Linux), написанное на функциях из C и C ++ . Он используется для разработки и загрузки программ на Arduino-совместимые платы, а также, с помощью ядер сторонних производителей, на платы разработки других производителей.

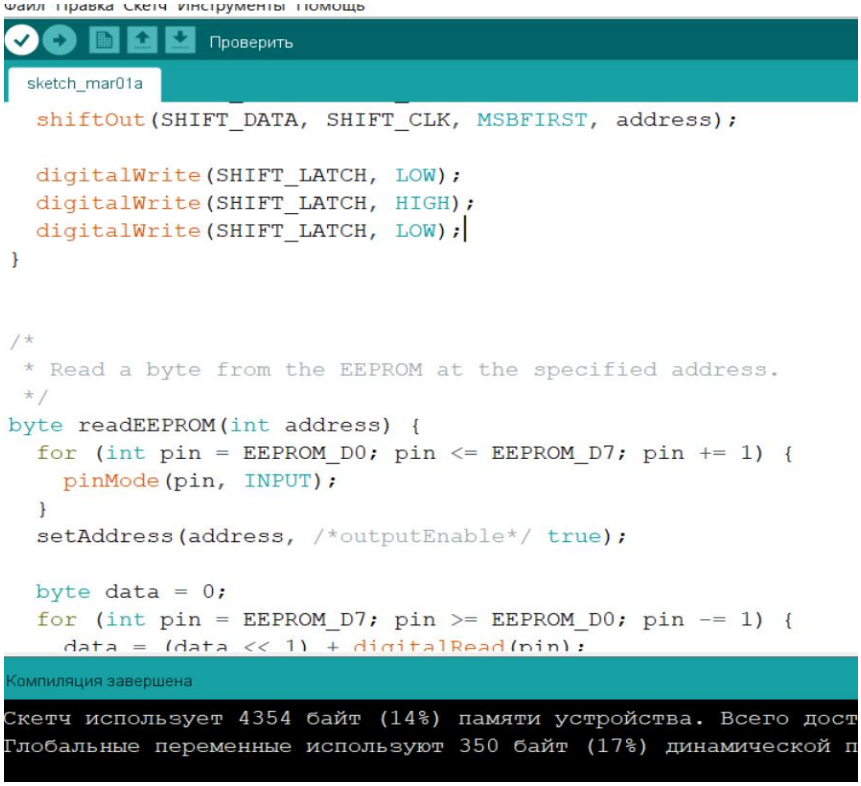


Рисунок 5 – Arduino IDE

Программатор EEPROM с Arduino и другими микросхемами, такими как 74HC595 и EEPROM 28C16, и используйте Arduino IDE для программирования EEPROM. Этого достаточно для замены большого количества сложных аппаратных схем, что дает преимущества низких затрат, меньших размеров схем, меньшего количества сложность образуют некоторые схемы.

ЛИТЕРАТУРА

1. https://en.wikipedia.org/wiki/Arduino_IDE
2. Leo Louis WORKING PRINCIPLE OF ARDUINO AND USING IT AS A TOOL FOR STUDY AND RESEARCH
3. https://store.arduino.cc/usa/arduino-nano
4. ATMEL- AT28C16 16K (2K x 8) Parallel EEPROMs

УДК 004.45

А. Т. Нижарадзе (4 курс бакалавриата)
А. Н. Терехов, д. ф.м. н., профессор

ПОРТИРОВАНИЕ ОСРВ EMBOX НА ОТКРЫТУЮ АРХИТЕКТУРУ RISC-V

Целью данной работы является реализация архитектурно-зависимой части ОСРВ Embox для RISC-V.

Embox — это свободная операционная система реального времени, разрабатываемая для встраиваемых систем. Ее особенность в том, что ее исходные коды разбиты на модули, что позволяет включать в сборку только те части системы, которые нацелены на решение конкретной задачи. Также конфигурируемость позволяет легко включать в сборку только необходимые в данный момент модули, что позволяет легко запускать систему на платформах с большими ограничениями по памяти.

RISC-V — это открытый и свободный проект, запущенный в 2010 году в Калифорнийском университете в Беркли, в последнее время становится все более популярным. ISA RISC-V, по сравнению с другими популярными не RISC архитектурами, такими как x86, обладает следующими преимуществами: сокращённый набор команд, открытость исходных кодов реализаций процессоров, свобода использования и увеличенное быстродействие.

Портирование операционной системы на новую архитектуру можно разделить на несколько этапов по некоторым логическим модулям: карта памяти, стартовый код, отладочный вывод, исключения, системный таймер, переключения контекстов, и т.д.

Карта памяти (memory map) определяет положение регионов памяти, т.е. расположение RAM, ROM, I/O и т.д. Информацию о RAM и ROM, а также о том, какие сегменты в каких из этих регионов будут располагаться, была добавлена в Linker Script File.

Стартовый код — код, размещающийся по стартовому адресу, и производящий первичную инициализацию системы, такую, как инициализация регистров, инициализация режима процессора, отключение прерываний и т.д. В конце стартовый код передает управление функции инициализации ядра ОС.

Основная функция отладочного вывода — возможность посимвольного вывода на доступное для разработчика устройство вывода. Самым базовым устройством является UART. Были реализованы стандартные функции передачи символов serial порта.

В архитектуре RISC-V исключения разделены на 3 группы: программные, внешние и исключение таймера. Внешние исключения поддерживает контроллер прерываний на уровне платформы (PLIC). Если в регистре mstatus выставлен бит MIE, значит прерывания включены глобально. Также если в регистре mie выставлены биты MSIE, MTIE, MEIE, то значит

включены программные, таймера и внешние прерывания соответственно. Если при этом срабатывает какое-либо прерывание, то выставляется соответствующий бит в регистре обрабатываемого прерывания mip, а также управление передается по указателю регистра mtvec — вектора прерываний.

В RISC-V есть 2 регистра для работы таймера: mtime и mtimecmp. Во 2ой устанавливается значение, а первый инкрементирует свое значение, начиная с 0, с постоянной частотой. Как только значение регистра mtime становится больше либо равно значению в регистре mtimecmp, происходит прерывание таймера.

Контекст процессора — это такая структура данных, которая хранит состояние процессорного ядра. Сохранять контекст необходимо, если мы хотим иметь возможность исполнения на нескольких потоках. Сохранение контекста — это сохранение значений всех доступных регистров в некоторой структуре, а переключение — это восстановление старого контекста другого процесса, то есть возвращение состояния регистров из значений регистров структуры контекста.

Для выполнения данной работы для эмуляции RISC-V я использовала qemu-system-riscv32, версия qemu-4.2.0; для сборки и линковки Embox — riscv64-unknown-elf-gcc, для отладки — riscv64-unknown-elf-gdb.

УДК 004.414.22

Б. Д. Полянок (4 курс бакалавриата)
Т. А. Вишневская, ст. преподаватель

РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА С ИСПОЛЬЗОВАНИЕМ РАЗЛИЧНЫХ АРХИТЕКТУРНЫХ ПРИНЦИПОВ ПРОЕКТИРОВАНИЯ ПО

Введение.

Сегодня сложно представить наш быт без использования приложений и взаимодействия с электронными устройствами во всех сферах нашей жизни. Для удовлетворения потребностей конечного пользователя, разработчикам программных решений необходимо постоянно выпускать обновления своего продукта и дополнять его новой функциональностью, не теряя в качестве.

Организация программы, ее структура играет важную роль в разработке любого проекта. От выбранной архитектуры зависит дальнейший процесс разработки и поддержки: насколько сложно будет разрабатывать продукт? Как дорого будет обходиться поддержка продукта? Сможем ли мы добавить новую функциональность в уже существующую?

Большое заблуждение считать, что продуманная архитектура необходима крупным AAA проектам. Неправильно выбранная архитектура может привести к «смерти» проекта из-за невозможности его дальнейшей поддержки (этот процесс становится слишком затратным и сложным). Напротив, хорошая архитектура может сэкономить очень много сил, времени и денег.

Что же такое процесс создания архитектуры программного продукта? Какие архитектурные решения применяются сегодня и какие проблемы при этом решаются? Как оценить правильность выбранной архитектуры, какие метрики использовать?

Выбор архитектуры для исследования.

Сегодня все чаще программные продукты разрабатываются с использованием микросервисной парадигмы проектирования ПО, когда приложения состоит из множества модулей, слабо связанных между собой и отвечающих за один тип задач. Микросервисы пришли на смену монолиту - совершенно противоположному принципу, когда приложения представляет собой один связный модуль, где все компоненты спроектированы так чтобы работать вместе с друг другом.

Микросервисная архитектура обладает большей гибкостью, ее проще тестировать, а обособленные микросервисы можно легко переиспользовать, но и у этого решения имеются недостатки не позволяющие полностью отказаться от использования монолита.

В ходе работы планируется сравнить эти два фундаментальных подхода в разработке ПО, выделить их преимущества и недостатки, а также провести исследование на реальном проекте, сравнив метрики качества получившегося программного продукта.

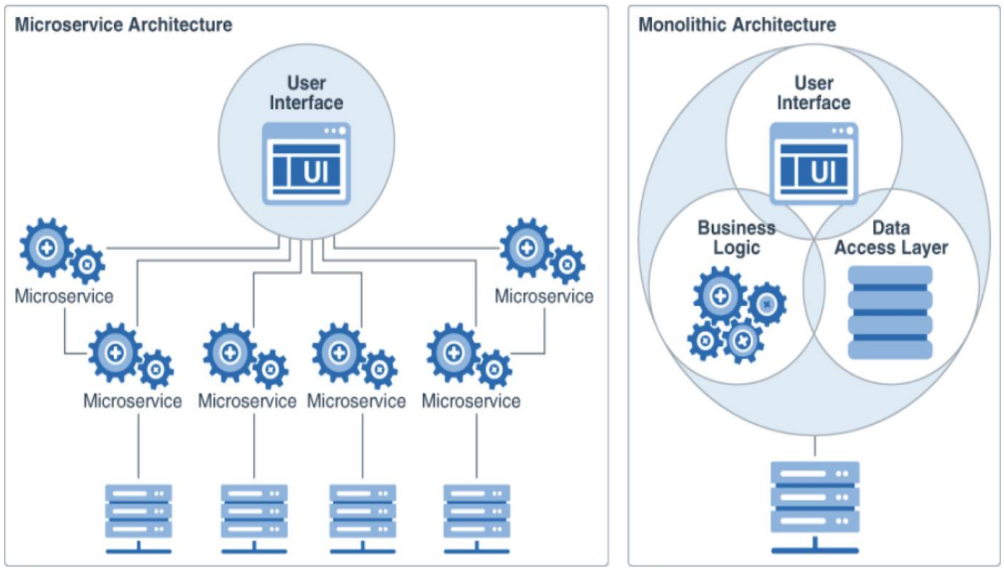


Рисунок 1 – схема приложения с использованием монолитной и микросервисной

Используемые технологии.

Разработка программной системы будет вестить на языке программирования Python с использованием фреймворка Flask [3]. В качестве СУБД используется PostgreSQL [4]. Для автоматизации развёртывания и управления приложением воспользуемся Docker [5]. Для тестирования приложения будут использован фреймворк для тестирования - PyTest [6].

ЛИТЕРАТУРА

1. *Paul Clements; Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; Judith Stafford.* Documenting Software Architectures: Views and Beyond. — Second Edition. — Addison-Wesley Professional, 2010.
2. Ньюмен С. Создание микросервисов — СПб.: Питер, 2016.
3. Flask. URL: https://palletsprojects.com/p/flask/
4. PostgreSQL. URL: https://www.postgresql.org
5. Docker. URL: https://www.docker.com
6. PyTest. URL: https://docs.pytest.org/en/latest/

УДК 004.89

М. С. Ромицына (4 курс бакалавриата)
О. Н. Петров, к.т.н., доцент

РАЗРАБОТКА МОДУЛЯ ПРОГРАММЫ ТЕСТИРОВАНИЯ ДЛЯ ПРОВЕРКИ РЕЗУЛЬТАТОВ МЕТОДАМИ НЕЧЕТКОЙ ЛОГИКИ

Мультимедийная система-тренажер содержит учебный материал, разбитый на модули и уроки, каждый из которых может включать видеоклипы, практические тесты и теоретические тесты. Теоретический тест – это набор произвольного количества вопросов с вариантами ответов, из которых студент будет отмечать правильные. На настоящий момент реализованы

следующие типы вопросов: один из нескольких (студент выбирает один правильный ответ из 4-х предложенных), несколько из нескольких (студент выбирает от нуля до 4-х правильных ответов из 4-х предложенных), выбор из списка (студент выбирает один из 4-х предложенных из выпадающего списка), набор ответа (студент пишет ответ в поле ввода).

Целью данной работы является разработка модуля, обеспечивающего оценку правильности ответа на вопрос с использованием методов нечеткой логики. Для этого реализуются два дополнительных типа вопросов теоретического теста: один из нескольких с заданной степенью истинности (студент выбирает один ответ из 4-х предложенных, каждый ответ имеет заданную преподавателем степень истинности, правильности), набор ответа с оценкой степени истинности (студент пишет ответ в поле ввода, после чего вычисляется степень истинности, правильности этого ответа).

В качестве вариантов реализации нечеткого логического вывода рассмотрены 2 модели: Мамдани и Сугено.

Модель Мамдани предполагает выполнение следующих этапов:

- формирование нечёткой базы знаний на основе нечётких правил;
- введение нечёткости (фаззификация) для поступившего чёткого значения;
- нечёткий вывод по каждому правилу (усечённые функции принадлежности для переменной вывода);
- композиция (объединение результатов предыдущего этапа операцией max; результат – итоговое нечёткое подмножество для переменной вывода с функцией принадлежности);
- приведение к чёткости (дефаззификация): четкое значение определяется как центр тяжести кривой.

Этапы модели Сугено:

- формирование нечёткой базы знаний на основе нечётких правил;
- введение нечёткости (фаззификация) для поступившего чёткого значения;
- приведение к чёткости (дефаззификация).

В результате анализа постановки задачи и предложенных примеров вопросов в соответствии с новыми типами вопросов теоретического теста выбор сделан в пользу модели Сугено.

В качестве иллюстративного примера рассмотрим теоретический вопрос, относящийся к типу «набор ответа с оценкой степени истинности». Подсказка студенту формулируется следующим образом: «Укажите эмпирическую вероятность скрещивания, обеспечивающую быстрое схождение к глобальному экстремуму (число от 0 до 1)». Студенту предоставляется поле ввода, в которое ему следует ввести число от 0 до 1.

Оценка правильности ответа осуществляется на основе функции принадлежности, заданной преподавателем (автором теоретического теста), трапециевидной формы с максимальным значением 1 около 0.55 – 0.65 и минимальным значением 0 около 0.45 и 0.75. Конкретный пример такой функции принадлежности представлен на рис. 1.

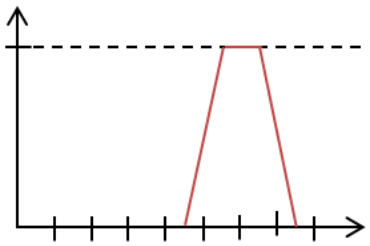


Рисунок 1 – "Функция принадлежности"

Значение степени истинности ответа вычисляется по заданной функции принадлежности (окружности на Рис.2). Это значение – балл за данный вопрос. При необходимости автор теста может дополнительно указать пороговое значение балла, при превышении которого ответ засчитывается.

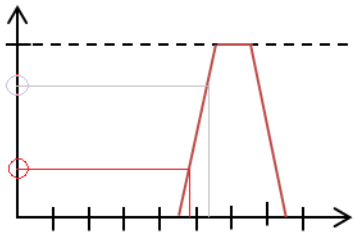


Рисунок 2 – "Определение значения функции принадлежности"

При обработке вопроса типа «один из нескольких с заданной степенью истинности» этап фаззификации осуществляется автоматически, поскольку автор теста заранее указывает степень истинности (правильности) каждого варианта ответа. При необходимости данный тип может быть обобщён до «несколько из нескольких с заданной степенью истинности», в этом случае итоговая степень истинности ответа определяется с помощью операции нечеткого пересечения.

ЛИТЕРАТУРА

1. Горавнёва Т. С., Петров О. Н. Руководство пользователя мультимедийной системой-тренажером: учебное пособие / Т.С. Горавнева, О.Н. Петров. – СПб: Изд-во СПбГМТУ, 2016. – 79 с.
2. Горбаченко, В. И. Интеллектуальные системы: нечеткие системы и сети : учебное пособие для вузов / В. И. Горбаченко, Б. С. Ахметов, О. Ю. Кузнецова. – 2-е изд., испр. и доп. – М.: Издательство Юрайт, 2018. – 103 с. – (Серия : Университеты России). — ISBN 978-5-534-03678-7. – Режим доступа: www.biblio-online.ru/book/7F3CBB90-F2E4-4A1A-80C6-705B143D0E27.

УДК 681.3.06

А. Е. Семенцов (4 курс бакалавриата)
Ю. Б. Сениченков, д.т.н., профессор

ИСПОЛЬЗОВАНИЕ «ФИЗИЧЕСКОГО МОДЕЛИРОВАНИЯ» НА ПРИМЕРЕ ПРОЕКТА «УМНЫЙ ДОМ»

Современные квартиры в городах и загородные дома представляют собой с точки зрения моделирования иерархические многокомпонентные событийно-управляемые системы. Эти системы нужно проектировать, поддерживать в рабочем состоянии, управлять ими, в том числе и дистанционно, и стараться минимизировать расходы, связанные с их эксплуатацией.

Они используются в разных режимах (постоянное проживание, дача, «жизнь на два дома» - квартира в городе и дача в течении года»), у них разные по количеству и запросам пользователи.

Квартиры в городе, это элементарные ячейки многокомпонентных систем – многоквартирных домов, являющихся в свою очередь элементарными ячейками районов города. С точки зрения систем обеспечения домов водой, газом, электричеством – это сложно устроенная сеть, поведение которой зависит от времени (дневной и ночной режим, рабочие и выходные дни, время года).

Расходы, связанные с проживанием в квартирах и домах, и обеспечением квартир и домов жизненно необходимыми энергетическими ресурсами, интересуют как пользователей, так и тех, кто управляет и поддерживает жизнь регионов.

В статье предлагается создать библиотеку компонентов «Умный дом» и реализовать ее в среде визуального моделирования Rand Model Designer. Пользователь библиотеки (хозяин дома, агент по продаже) может собрать из готовых компонентов модель загородного дома, продемонстрировать возможности системы управления домом (или использовать модель как тренажер, «цифровой двойник», для обучения управлением систем дома) оценить стоимость оборудования и эксплуатации, анализируя результаты эксплуатации модели в выбранном режиме и заданном количестве пользователей.

На основе проведённого сравнительного анализа коммерческих систем «Умный дом», для демонстрации возможностей разработанной библиотеки, были отобраны и реализованы в среде RMD подсистемы предотвращения протечки воды (от компании WISOL) и подсистема управления освещением (от компании EasyHome).

Благодаря возможности Rand Model Designer, создавать модели в виде dll, был реализован WPF-проект на языке C#, который демонстрирует взаимодействие с моделью вне среды RMD. В данном программном решении можно будет следить за показателями потребления воды, электричества, а также за текущей температурой, и управлять этой системой.

УДК 62.529

А. А. Слюсарев (4 курс бакалавриата)
И. Г. Черноруцкий, д.т.н., профессор

АНАЛИЗ И УЛУЧШЕНИЕ ОРГАНИЗАЦИИ ТРАФИКА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

Крупные города уже давно столкнулись с такой проблемой, как пробки. Источники пробок – это различные явления на дорогах, одно из которых может быть задержка на светофоре. Из-за пробок тратится много времени, ухудшается качество жизни горожан и экология.

Есть несколько способов решить эту проблему. Например, централизованное управление светофорами, которое, однако, требует значительного улучшения инфраструктуры и, следовательно, высоких затрат. Но такая система можно сделать распределенной.

В данной статье рассматривается улучшение работы светофоров при помощи машинного обучения. Благодаря современным алгоритмам машинного обучения стало возможным оборудовать каждый светофор независимой самообучающейся системой, которая позволяет ему автоматически адаптироваться к быстро меняющимся условиям транспортного потока. «Умный светофор» определяет, какой стороне сколько времени надо уделять, чтобы движение было оптимальным: на каждом перекрестке наша программа делает оценку, где машин больше, а где – меньше. Сейчас создана симуляция перекрестка и машин, в реальном мире предполагается делать осуществлять это при помощи камер и технологии распознавания образов. Также с помощью этой технологии можно вести подсчет количества машин, пешеходов, оценивать нагрузку – все эти данные подаются на вход нейронной сети. Она самообучается и выбирает время, через которое нужно переключить светофор, и так далее. Имеется ограничение на верхний предел времени, по истечении которого умный светофор гарантировано переключит сигнал.

Также в процессе обучения светофоры учитывают прошлый опыт работы. На основе получаемого от среды вознаграждения (мера оптимальности решения) встроенная программная система формирует функцию полезности Q, которая впоследствии дает ему возможность не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой.

Следующим этапом развития является объединение светофоров в специальную сеть. Каждый светофор будет принимать не только оптимальные решения для своего перекрестка, но и общаться с остальными светофорами, все локальные решения будут согласованы, что приведет к оптимальному глобальному решению – снижению застоев автомобилей во всем городе и ускорению транспортных потоков. Как уже было описано выше, создание единой централизованной сети достаточно громоздкое и сложное решение, поэтому предлагается следующая конфигурация, где каждый регулируемый перекрёсток имеет представление о соседних. Например, светофорам на перекрёстке *A* известно о периодах переключения сигналов, количестве проезжающих машин и так далее на перекрёстке *B*. Таким образом, в

теории данная конфигурация должна быть проще в развёртывании, поддержании, создании и расширении. При этом преимущества остаются прежними, так как даже такая организация позволяет грамотно и полно оценить ситуацию, а также синхронизировать узлы между собой.

Важным нюансом является рассмотрение спорных ситуаций в сети, когда у двух узлов оптимальные локальные решения противоречат или недостаточно оптимальны для эффективности общей сети. Для решения данной проблемы можно пойти двумя очевидными путями. Первый: рассчитывать функцию полезности для узлов и учитывать её значения в окончательном принятии решения. Другой способ менее элегантный и эффективный: это выбор опорного узла, таким образом соседние узлы просто должны находить оптимальное решение из заданной ситуации. У данного способа есть только один плюс – простота реализации, на ранних этапах разработки можно использовать данную стратегию, а в дальнейшем результаты работы системы можно использовать в качестве сравнения.

Искусственный интеллект дает нам возможность взглянуть на процессы иначе и повысить их производительность в разы, что нам и удалось сделать, применяя различные алгоритмы машинного обучения для решения задачи дорожного регулирования.

Преимуществами нашего продукта перед уже имеющимися являются:

1) новизна (только в “умных городах” со взаимосвязанной системой коммуникативных и информационных технологий с интернетом вещей (Internet of Things) возможно реализован такой проект, но наш проект можно внедрять не только в “умные города”, но и в “обычные”);

2) автономность и децентрализация (все светофоры, по сути, работают независимо от остальных, хотя и общаются друг с другом. Т.е. наш светофор можно внедрить как отдельную вещь – не требуется создавать единое сложное централизованное управление. Также благодаря этому подключить новый “умный светофор” к сети уже таких же имеющихся (умных) светофоров тоже очень просто.);

3) ”умный светофор” не работает по заранее известному графику, а по ходу учится и, как вывод, распределяет время намного эффективнее, чем обычный светофор;

4) объединённые в децентрализованную сеть светофоры позволяют наиболее эффективно организовать движение, с учётом контекста работы ближайших светофоров;

5) система способна помимо прочего динамически подстраиваться под меняющиеся условия среды;

6) с учётом того, что для каждого перекрёстка требуется знать только о соседях, сеть легко расширить и масштабировать, так как количество затрагиваемых узлов относительно общего числа невелико.

УДК 004.43:004.65

А. А. Худяков (2 курс бакалавриата)
В. М. Филиповский, к.т.н., доцент

СОЗДАНИЕ БАЗЫ ДАННЫХ НА ЯЗЫКЕ C++ В СРЕДЕ MICROSOFT VISUAL STUDIO

С каждым днем количество информационных ресурсов стремительно увеличивается. Данные необходимо накапливать, упорядочивать и рационально использовать. Для работы с ними существуют системы управления базами данных, или сокращенно СУБД. При разработке БД чаще всего предпочтение отдают реляционным базам данных. Под них построено большинство современных СУБД (MS SQL Server, MS Access, InterBase, FoxPro, PostgreSQL, Paradox и другие) [1]. Большая часть из существующих СУБД требует дополнительно разработки интерфейса, чтобы пользователь мог взаимодействовать с базой данных.

Данная работа посвящена созданию реляционной базы данных на языке программирования C++ в среде Microsoft Visual Studio. Цель работы: показать, что

современные технологии позволяют создать БД и интерфейс к ней в одной удобной программисту среде, например, на языке C++.

В качестве демонстрации будет представлено создание базы данных об автомобилях и их владельцах. К основным элементам БД можно отнести сведения об автомобилях (ID, марка, модель, год выпуска, гос. номер, цвет, мощность двигателя, текущий владелец), сведения о владельцах (ID, фамилия, имя, отчество, пол, дата рождения, адрес регистрации), смена владельца у автомобиля, штрафы.

БД создана по принципу реляционной базы, где информация располагается в таблицах, которые имеют горизонтальные связи между собой (связи организуются либо по первичному ключу, либо по внешнему) [2].

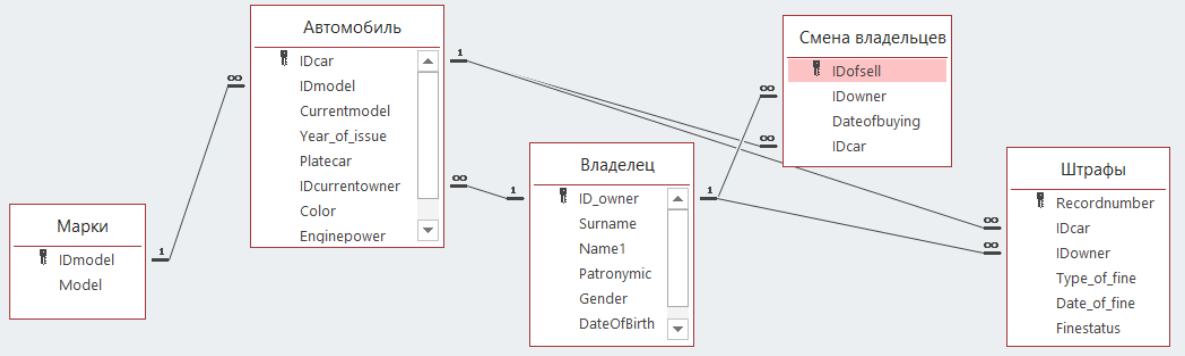


Рисунок 1 – Схема данных

Вместо классических таблиц используются списки, так как вышеперечисленные связи в них можно организовать с помощью указателей на элементы другого списка [3].

C++ как язык ООП дает возможность упростить и сократить время написания БД благодаря использованию шаблонов [4]. Шаблонный класс состоит из поля с информацией, указателей на соседние элементы, а также ряда методов – шаблонов, среди которых функции чтения/записи информации из файла/в файл, поиск элемента по его id (данная функция реализует необходимое ограничение foreign key), вставка элемента в список, удаление элемента из списка, которое в основном используется для отсеивания ненужных записей.

Создание списков, соответствующих конкретным сущностям, осуществляется путем перегрузки шаблонного списка. Для реализации связи объявляется поле с помощью которого строится связь с другим списком как указатель на элемент из родительского списка. При переносе информации из списка в файл по данным указателям (адресам) находятся идентификаторы соответствующих строк родительского списка, которые и записываются в этот файл в качестве конкретных значений. Данная функция обеспечивает столь необходимую в БД целостность данных.

```
template <typename T, typename V>
class table
{
public:
        T data;
        V *next;
        V *prev;
        table();
        V *inputfile(V*, const char*);
        void outputfile(V*, const char*);
        V *push(V*, V*);
        V* search_id(V *, int);
        V* request_comboBox(V*, int);
        V* del(V*, V*);
};
```

```
typedef struct Models_
{
        int id;
        char Model[20];//название марки
}MODELS;
class Models :public table <MODELS, Models>
{
public:
        Models()
        {
                this->next = NULL;
                this->prev = NULL;
        };
        ~Models()
        {};
};
```

Листинг 1 – Шаблонный и порожденный списки

Для полного функционирования программы шаблонных методов недостаточно, поэтому было разработано несколько дополнительных функций. Это функции чтения из файла, в которых часть информации берется из существующих списков, и функции отбора нужной информации для формирования отчетов.

После того как ключевые этапы разработки БД пройдены, переходят к проектированию визуальной составляющей, интерфейса. Интерфейс служит для организации ввода/вывода информации, позволяет просматривать отчеты и в целом автоматизирует взаимодействие с БД. Его разработка происходит в той же самой среде, где ранее были созданы списки и методы работы с ними. Часть форм используют поля с подстановкой, так как вместо кодов Id здесь отображаются более информативные значения. Кроме того, эти поля дополнительно обеспечивают целостность данных.

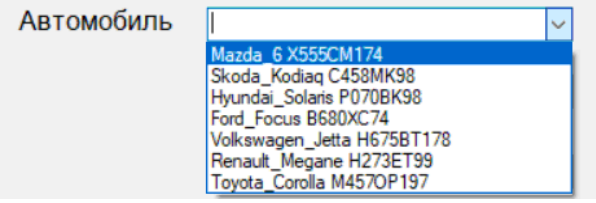


Рисунок 2 – Демонстрация полей с подстановкой (фрагмент формы «Новый штраф»)

Заключительным этапом является защита информации. Среда разработки дает возможность гибко настраивать свойства форм. На формах, где пользователь просматривает информацию, активируем свойство, запрещающее изменение информации. Для ввода информации создаются отдельные формы.

На этом процесс создания БД можно считать оконченным. В результате проделанной работы была создана БД, а также разработан для нее интерфейс. Что подтверждает возможность и удобство одновременного создания как самой БД, так и интерфейса к ней в рамках одной среды разработки Microsoft Visual Studio на языке C++.

ЛИТЕРАТУРА

1. Хомоненко А. Д. Базы данных. Учебник для высших учебных заведений / Хомоненко А. Д., Мальцев М. Г., Цыганков В. М; под редакцией А. Д. Хомоненко. – 6-е изд., доп. – СПб.: КОРОНА – Век, 2009. – 736 с.
2. Д. Крёнке [D. Kroenke] Теория и практика построения баз данных / Д. Крёнке. – СПб.: Питер, 2003. – 800 с.
3. Р. Лафоре [R. Lafore] Объектно – ориентированное программирование в C++ / Р. Лафоре. – 4-е изд. - СПб.: Питер, 2004. – 923 с.
4. Д. Вандевурд, Н. М. Джосаттис, Д. Грегор [D. Vandevoorde, N. M. Josuttis, D. Gregor] Шаблоны C++. Справочник разработчика, 2-е изд.: Пер. с англ. – СПб.: ООО “Альфа-книга”, 2018. — 848 с

УДК 004.056.53

Д. А. Чернов (4 курс бакалавриата)
А. П. Маслаков, ст. преподаватель

РАЗРАБОТКА ПРИЛОЖЕНИЙ НА ОСНОВЕ ТЕХНОЛОГИИ JAVA CARD™

Целью работы является исследование возможностей смарт-карт основанных на технологии JavaCard и проектирование апплета на основе технологии JavaCard. Апплет должен выступать в роли части системы идентификации личности. Он должен хранить некоторые данные о пользователе, уметь их шифровать с помощью актуального алгоритма шифрования и передавать на приёмное устройство.

Технология JavaCard – это сочетание подмножества языка программирования Java и среды времени исполнения, оптимизированных для использования в смарт-картах и других

вычислительных устройствах с ограниченным объемом памяти. Широкое распространение технологии смарт-карт в настоящее время обусловлено рядом её преимуществ. Например, процессор, память и интерфейсы ввода/вывода реализуются в одной интегральной схеме (integrated circuit card, *ICC*) небольших размеров [3]. Смарт-карты не использует потенциально незащищенные внешние ресурсы, а для получения информации со смарт-карты требуется наличие определенного оборудования, знаний аппаратного и программного обеспечения смарт-карты и непосредственный доступ к самой карте. Данные, находящиеся в памяти смарт-карты, и данные, которые передаются на карту и с неё могут быть зашифрованы и использовать электронную подпись, т.е. активно используются криптографические алгоритмы для повышения безопасности карты.

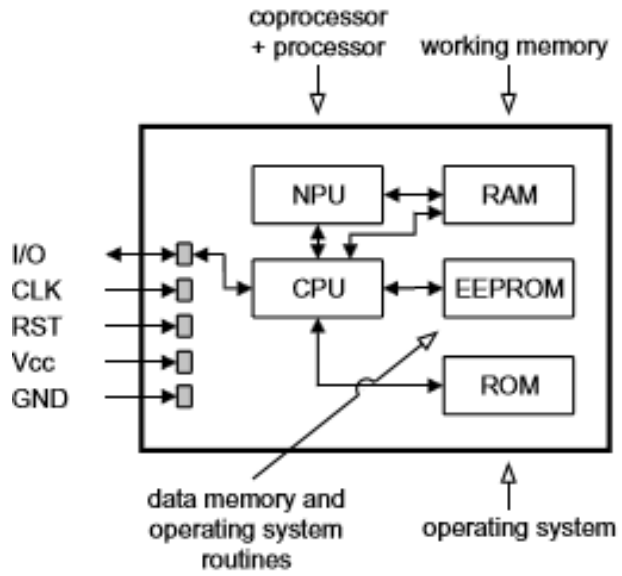


Рисунок 1 – Архитектура контактной смарт-карты с микропроцессором и сопроцессором

Обмен данными между смарт-картами и другими вычислительными устройствами происходит в форме пакетов или блоков данных специального формата, которые называются блоками данных прикладного протокола application protocol data unit (APDU) [1]. Блок APDU содержит или команду, или ответное сообщение. В мире смарт-карт используется модель взаимодействия «главный – подчиненный». Смарт-карта всегда выполняет пассивную роль подчиненного устройства. Она ожидает поступления команды APDU от хост-приложения, затем выполняет инструкцию, которая содержится в команде и отправляет блок APDU с ответным сообщением обратно.

Таблица 1 – Структура команды APDU

Обязательный заголовок				Необязательный заголовок		
CLA	INS	P1	P1	LC	DATA	LE

Таблица 2 – Структура ответа APDU

Необязательный заголовок	Обязательная часть	
Поле данных	SW1	SW2

Заголовок команды APDU состоит из 4 байтов: CLA – класс инструкции, INS – код инструкции, P1 и P2 – параметры для дополнительного уточнения инструкции. После заголовка команды APDU может передаваться необязательный блок переменной длины. Поле Lc в этом блоке определяет длину поля данных в байтах, поле Le определяет длину в байтах сообщения, ожидаемого хост-приложением в ответ на данную команду. Ответное сообщение APDU, соответствующее полученной команде APDU, содержит необязательный блок,

который включает поле данных, длина которого определена в поле Le соответствующей команды APDU. Заключительная обязательная часть состоит из двух полей SW1 и SW2. Эти поля называются словом состояния, которое определяет состояние карты после выполнения команды APDU.

Безопасность данных на смарт-картах была одним из основных приоритетов при разработке JavaCard. Чтобы обеспечить нужный уровень безопасности при передаче данных, в JavaCard поддерживаются популярные алгоритмы шифрования, такие, как DES, 3DES, AES, RSA [2]. Также поддерживаются другие криптографические сервисы: цифровые подписи, генерирование электронных ключей и обмен ими.

В нашем апплете успешно реализованы методы хранения и передачи данных о пользователе. Для хранения данных в апплете выделены специальные участки памяти, которые могут быть записаны только один раз при установке апплета на карту. Для передачи данных в апплете реализованы функции обработки определённых APDU команд, которые так же возвращают APDU ответ с запрашиваемыми данными. Чтобы обеспечить безопасность нашей системы, перед передачей данные шифруются с помощью криптографической подписи. Таким образом, мы можем гарантировать подлинность передаваемых данных.

ЛИТЕРАТУРА

1. Жикун Чен Технология Java Card™ для смарт-карт. архитектура и руководство программиста, Москва: Техносфера, 2008. -344с.
2. Статья с сайта Хабр «Использование Java смарт-карт для защиты ПО.» [Электронный ресурс] –URL: https://habr.com/ru/post/255529/
3. Намиот Д.Е., Сиховец Л.Б. Введение в Java Card. Часть 1. 2002 г.

УДК 004.042

Ф. О. Шарай, (4 курс бакалавриата)
Т. В. Коликова, ст. преподаватель

СРАВНЕНИЕ ФРЕЙМВОРКОВ ДЛЯ РАЗРАБОТКИ КРОССПЛАФТОРМЕННЫХ НАТИВНЫХ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

На сегодняшней день существует проблема разработки качественных мобильных приложений без необходимости использования множественных человеческих ресурсов. Многие компании малого бизнеса не в состоянии позволить себе нанять две команды разработчиков для разработки собственного мобильного приложения для двух основных платформ: Android и iOS. В поисках решения данной проблемы было найдено прогрессивное решение: разработка кроссплатформенного мобильного приложения. Такое решение всё ещё применяется достаточно редко, однако сегодня появились решения, которые позволяют добиться качества продукта, сопоставимого с нативным аналогом, и даже лучше.

В данной работе предлагаю рассмотреть и сравнить предлагаемые решения для разработки кроссплатформенных мобильных приложений. Для сравнения возьмём три популярных фреймворка: Ionic, Flutter и React Native. Сравнивать их будем при помощи трёх мобильных приложений, два из которых являются полной копией, разработанных с использованием разных технологий: два мобильных приложения Интернет-магазина “Офелия” [2], разработанной командой разработчиков на Flutter и React Native, мобильное приложение компании UNICLO [1] разработанное с использованием Ionic.

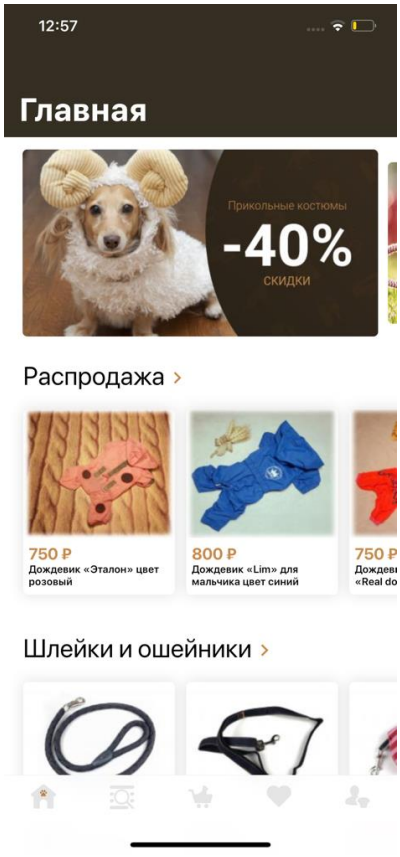


Рисунок 1 – Flutter

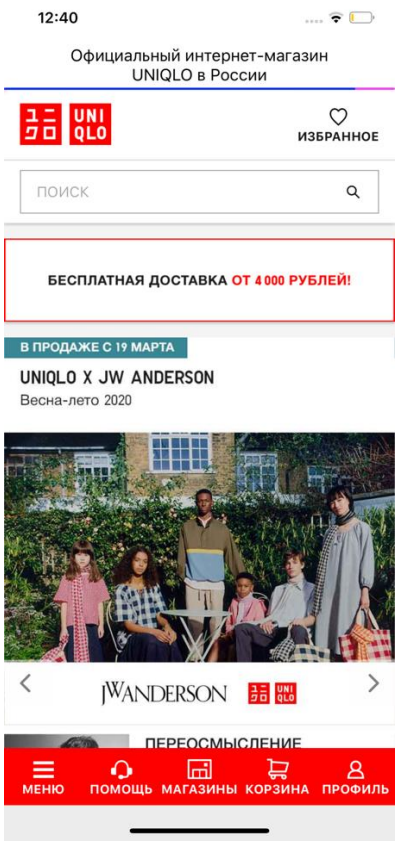


Рисунок 2 – Ionic

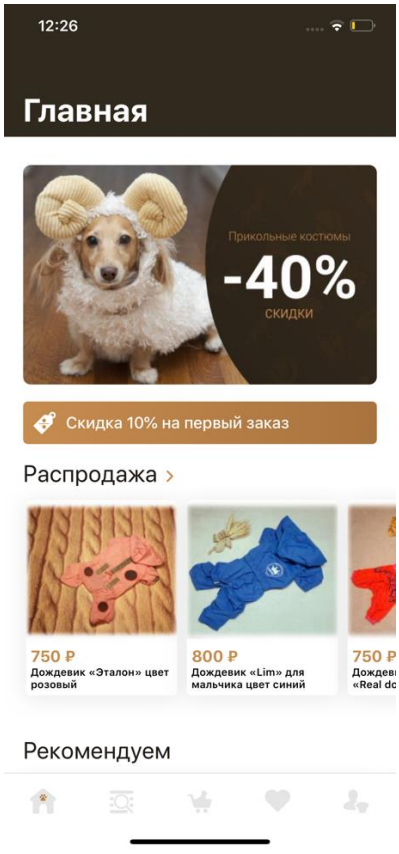


Рисунок 3 – React Native

Первоначально приложение для Интернет-магазина “Офелия” было разработано с использованием React Native, однако в качестве альтернативы в дальнейшем было решено попробовать написать то же самое на Flutter. Использование Ionic рассматривалось, однако с самого начало было решено от него отказаться.

Ionic является решением, используемое WebView в качестве отображения UI элементов, что сильно влияет на производительность, а также сильно ограничивает функциональные возможности приложения. Этот фреймворк является наиболее простым в использовании и поддержке, что, несомненно, является плюсом, однако при погружении в этот фреймворк становится понятно, что его профессиональное использование достаточно сложно и добиться качественных результатов в нём достаточно сложно. Примером тому может послужить как раз-таки рассматриваемое приложение от компании UNICLO, где имеются проблемы с отображением на разных типах устройств, низкая производительность, а также низкая пользовательская оценка.

React Native является решением, используемым для отображения нативные компоненты, а для взаимодействия используется так называемый “Мост” для контроля над каждым нативным компонентом. Это является своего рода “bottleneck”[3] (Бутылочным горлышком), которое замедляет работку приложения при использовании больших списков информации. Это решение является значительно лучше и быстрее, чем Ionic, однако сильно зависимо от сторонних разработчиков, особенно по части навигации приложения, а проблем сторонних библиотек требует знания нативных языков программирования (Obj-c, Swift, Kotlin, Java), что может повлечь за собой увеличение команды разработчиков и времени разработки приложения.

Flutter является решением, используемое для отображения графический движок Skia. Таким образом все элементы, отображаемые на экране, являются “копией” настоящих нативных элементов, от чего они могут выглядеть не совсем так, как если бы приложение было написано на нативном языке программирования платформы. Производительность выше, чем

у Ionic и сопоставима с React Native. Данное решение значительно ниже зависит от сторонних разработчиков и требует меньше необходимости обращения к нативному коду.

Таким образом, в ходе разработки стало понятно, что Flutter является наилучшим решением в разработке кроссплатформенных нативных приложений. Его особенность заключается в том, что он использует один язык программирования – Dart, активно развивается, не ограничивает в своих возможностях и производительности, не требует большого количества людей для реализации, вследствие чего уменьшаются затраты на разработку и поддержку приложения. Ionic и React Native, безусловно, являются хорошими решениями, однако в сравнении с Flutter имеют большое количество ограничений и проблем, которые достаточно сложно исправить небольшой команде разработчиков. На сегодняшний день этот фреймворк является наилучшем решением для создания кроссплатформенных приложений и в перспективе может заменить классическую нативную разработку в большинстве случаев.

ЛИТЕРАТУРА

1. Мобильное приложение UNICLO // [ссылка на приложение]/— URL: https://apps.apple.com/ru/app/uniqlo/id1475195340 (дата обращения: 16.02.2020).
2. Мобильное приложение Интернет-магазин “Офелия” // [ссылка на приложение React Native]/— URL: https://apps.apple.com/ru/app/офелия-одежда-для-собак/id1464103988 (дата обращения: 16.02.2020).
3. Bottleneck (production)// [From Wikipedia, the free encyclopedia]/— URL: https://en.wikipedia.org/wiki/Bottleneck_(production) (дата обращения: 16.02.2020).

Секция «Программная инженерия: методы и алгоритмы теории программирования»

УДК 621.319

П. И. Алейников, А. С. Погребной (4 курс бакалавриата)
С. А. Фёдоров, ст. преподаватель

РЕАЛИЗАЦИЯ СИСТЕМЫ ПРАВ ДОСТУПА ДЛЯ ВЕБ-СЕРВИСА ПО АВТОМАТИЗАЦИИ ВЕДЕНИЯ ДОГОВОРОВ

Целью работы является проектирование и реализация системы прав доступа для веб-сервиса по автоматизации ведения договоров. Система прав призвана разграничить доступ к договорам в веб-сервисе с иерархической структурой пользователей. Серверная часть сервиса реализована на универсальном фреймворке Java Spring Boot 2.1 и РСУБД PostgreSQL 11. Клиентская часть реализована на фронтенд-фреймворке Angular 8.1. Коммуникация между сервером и клиентом осуществляется посредством REST API. Разрабатываемая система прав должна хранить CRUD права пользователей на договоры, обеспечивать корректную модификацию прав и быстрый доступ. Для получения прав конкретного пользователя система принимает на вход уникальный идентификатор пользователя, например, логин и отдаёт права в виде массива структур, содержащих уникальный идентификатор договора и права на этот договор как список булевых значений.

Так как в разрабатываемом веб сервисе для хранения данных серверная часть использовала РСУБД PostgreSQL, то было решено хранить права пользователей так же в этой РСУБД. Для хранения прав в РСУБД обычно создают таблицу, которая связывает пользователя, объект на который назначаются права и значения этих прав. Так же если в системе большое число пользователей должны иметь равные права на одни и те же объекты, то пользователей объединяют в группы и создают аналогичную таблицу для хранения прав группы. Однако в нашем случае веб сервис рассчитан на хранение договоров крупных компаний сотрудники которых объединены в иерархическую структуру.

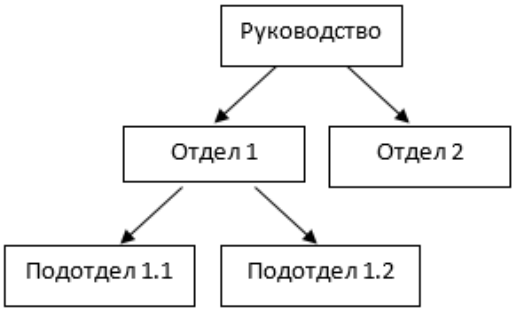


Рисунок 1 – Иерархическая структура компании.

Для такой структуры характерно то, что вышестоящие группы сотрудников, всегда должны иметь доступ к договорам, которые ведут нижестоящие сотрудники. В таком случае, если группа “Подотдел 1.1” получает права на какой-то договор, то права на этот договор должны получить группы “Отдел 1” и “Руководство”. Для такой иерархический структуры, способ хранения прав, описанный выше может потребовать большой объём памяти и более

того он не гарантирует корректность назначенных прав. То есть не позволяет убедиться в том, что назначенные права соответствуют требованиям иерархической структуры.

Ввиду описанной выше специфики веб сервиса, была разработана древовидная структура прав, которая позволяет экономнее хранить данные о назначенных правах. Так же был разработан рекурсивный алгоритм модификации прав, при помощи которого обеспечивается корректное изменение структуры прав пользователей во всей иерархии. Разработанная система использует в своей основе иерархическую (древовидную) структуру данных, для хранения которой к PostgreSQL было подключено расширение ltree. Это расширение добавляет необходимые типы данных в РСУБД и позволяет манипулировать иерархическими структурами в SQL запросах. В свою очередь возможность манипулировать со структурой прав в SQL запросах, положительно повлияло на производительность системы. Для оценки производительности было проведено нагрузочное тестирование, которое показало, что реализованная система увеличила время обработки REST запросов взаимодействия с договорами на 30% (100 мс.), что удовлетворяет требованиям заказчика.

В рамках этой работы так же был разработан пользовательский интерфейс, построенный на фронтенд-фреймворке Angular. Данный интерфейс позволяет просматривать и корректно редактировать права на определённый договор. При этом структура прав представляется в понятном для пользователя виде.

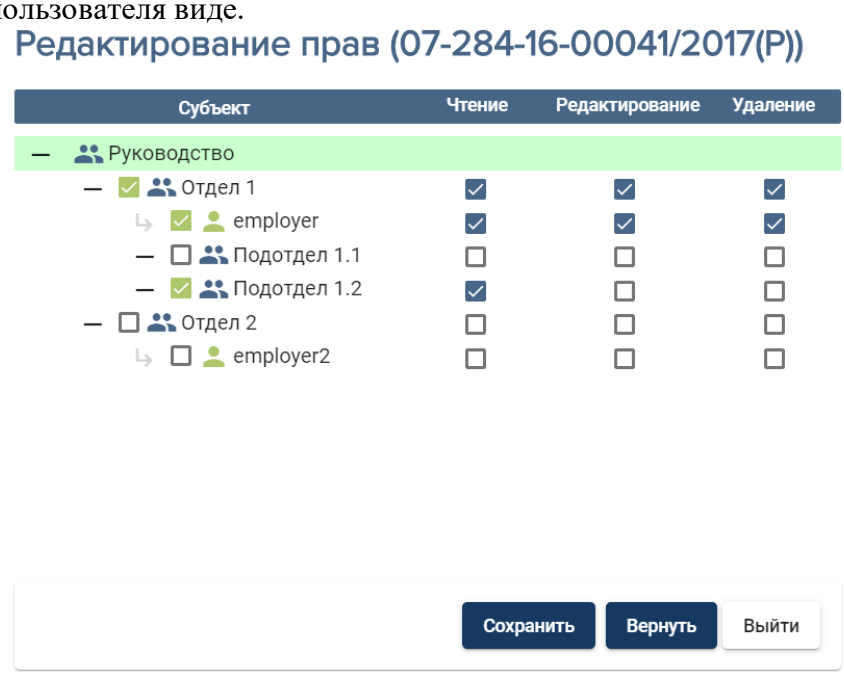


Рисунок 2 – Иллюстрация разработанного графического интерфейса.

УДК 004.942

А. П. Белянин (4 курс бакалавриата)
Н. Б. Ампилова, к. ф.-м. н., доцент

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ АГРЕГАЦИИ ОГРАНИЧЕННОЙ ДИФФУЗИЕЙ ДЛЯ ТРИАНГУЛИРУЕМОЙ ПОВЕРХНОСТИ

Модель Diffusion-limited aggregation (DLA) иллюстрирует процесс, в ходе которого частицы, подвергающиеся случайному блужданию в результате броуновского движения, группируются вместе с образованием агрегатов. Такой подход применим к моделированию агрегации в любой системе, где диффузия является основным средством транспорта в системе. DLA можно наблюдать во многих системах, таких как рост минеральных кристаллов, осаждение металла при электролизе и диэлектрической пробой. DLA применяется в физике, химии и медицине.

Целью работы является создание приложения для моделирования процессов агрегации ограниченной диффузией для триангулируемой поверхности. Это приложение должно иметь интерфейс для ввода данных, определяющих поверхность с ограничениями, а также триангулировать заданную поверхность, запускать алгоритм моделирования на полученной сетке, визуализировать полученные на различных стадиях результаты.

Проект полностью написан на C#. Для создания интерфейса и визуализации триангулированной поверхности используются стандартные методы WPF. Для триангуляции был выбран шагающий алгоритм, предложенный в [1]. Он заключается в том, что выбирается начальная точка на поверхности, вокруг которой строится 6 треугольников. После этого выбирается получившийся шестиугольный контур. От вершины с наименьшим внешним углом в данном контуре отстраивается подходящее количество новых треугольников. Выбор вершины в новом контуре и строительство треугольников вокруг нее продолжается до тех пор, пока вся поверхность не будет триангулирована. Алгоритм позволяет триангулировать поверхности без граничных условий, например, замкнутые, строит почти равносторонние треугольники, что предпочтительно для визуализации и для моделирования DLA.

В алгоритме DLA процесс формирования агрегата начинается с одного выбранного треугольника. Затем на сетке случайно выбирается треугольник, который начинает блуждать. Он перейдет из данного треугольника со сторонами (a, b, c) в соседний треугольник с общей стороной a с вероятностью $\frac{\frac{1}{a}}{\frac{1}{a}+\frac{1}{b}+\frac{1}{c}}$. Когда он дойдет до треугольника, соседствующего с агрегатом, блуждание остановится, и этот треугольник присоединится к агрегату. Процесс бросания треугольника на сетку с последующим блужданием продолжается заданное число раз.

ЛИТЕРАТУРА

1. Erich Hartmann, Geometry and Algorithms for COMPUTER AIDED DESIGN, Department of Mathematics Darmstadt University of Technology, 2003

УДК 004.021

Д. В. Боженко (4 курс бакалавриата)
И. Г. Черноруцкий, д.т.н., профессор

ИССЛЕДОВАНИЕ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ ОПТИМИЗАЦИИ ГРАДИЕНТНОГО СПУСКА В МОДЕЛЯХ ГЛУБОКОГО ОБУЧЕНИЯ

Градиентный спуск — алгоритм оптимизации, заключающийся в методе нахождения минимального значения функции потерь. Минимизация любой функции означает поиск самой глубокой впадины в этой функции. Функция используется, чтобы контролировать ошибку в прогнозах модели машинного обучения. Поиск минимума означает получение наименьшей возможной ошибки или повышение точности модели. Мы увеличиваем точность, перебирая набор учебных данных при настройке параметров нашей модели (весов и смещений). Суть алгоритма – процесс получения наименьшего значения ошибки.

Целью этой работы является выявление наилучших алгоритмов оптимизации градиентного спуска. Так как градиентный спуск - один из самых популярных способов оптимизации нейронных сетей, данное исследование является очень актуальным.

Алгоритмы, основанные на импульсе:

– Градиентный спуск. Есть преимущество ускорения обучения градиентного спуска вдоль пространств, где градиент остается относительно постоянным, и замедления обучения вдоль нестабильных пространств, где градиент значительно колеблется.

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1}) ;\ \theta_t \leftarrow \theta_{t-1} - \eta g_t$$

– Классический импульс. Добавляет часть направления предыдущего шага к текущему шагу. Это обеспечивает усиление скорости в правильном направлении и смягчает колебания в неправильных направлениях. Эта доля обычно находится в диапазоне (0, 1).

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1});\ m_t \leftarrow \mu m_{t-1} + g_t;\ \theta_t \leftarrow \theta_{t-1} - \eta m_t$$

– Импульс Нестерова. В импульсе вычисляем градиент, а затем делаем скачок в этом направлении, усиленный тем импульсом, который мы имели ранее.

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1} - \eta\mu m_{t-1});\ m_t \leftarrow \mu m_{t-1} + g_t;\ \theta_t \leftarrow \theta_{t-1} - \eta m_t$$

Алгоритмы L2, основанные на нормах:

– Адаптивный субградиентный спуск (AdaGrad). Выполняет большие обновления для нечастых параметров и меньшие обновления для частых. Из-за этого он хорошо подходит для разреженных данных. Еще одним преимуществом является то, что он в основном устраняет необходимость настраивать скорость обучения.

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1});\ n_t \leftarrow n_{t-1} + g_t^2;\ \theta_t \leftarrow \theta_{t-1} - \eta \frac{g_t}{\sqrt{n_t} + \varepsilon}$$

– RMSProp. На каждом этапе вы добавляете еще один квадратный корень к сумме, что приводит к постоянному увеличению знаменателя. В AdaDelta вместо суммирования всех прошлых квадратных корней используется скользящее окно, которое позволяет уменьшить сумму.

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1});\ n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2;\ \theta_t \leftarrow \theta_{t-1} - \eta \frac{g_t}{\sqrt{n_t} + \varepsilon}$$

– Adam, объединяющий классический импульс с RMSProp. Аналогичен AdaDelta. Но в дополнение к хранению скорости обучения для каждого из параметров он также хранит изменения импульса для каждого из них отдельно.

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1});\ m_t \leftarrow \mu m_{t-1} + (1-\mu) g_t;\ \hat{m}_t \leftarrow \frac{m_t}{1-\mu^t};\ n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2;$$

$$\hat{n}_t \leftarrow \frac{n_t}{1-\nu^t};\ \theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \varepsilon}$$

–NAG - импульс Нестерова. Сначала мы делаем большой скачок на основе нашей сохраненной информации, а затем вычисляем градиент и делаем небольшую коррекцию

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1});\ m_t \leftarrow \mu_t m_{t-1} + g_t;\ \overline{m}_t \leftarrow g_t + \mu_{t+1} m_t;\ \theta_t \leftarrow \theta_{t-1} - \eta \overline{m}_t.$$

–Nadam. Импульс Нестерова, включенный в Adam.

$$g_t \leftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1});\ \hat{g} \leftarrow \frac{g_t}{1 - \prod_{i=1}^{t} \mu_i};\ m_t \leftarrow \mu m_{t-1} + (1-\mu) g_t;\ \hat{m}_t \leftarrow \frac{m_t}{1 - \prod_{i=1}^{t+1} \mu_i};$$

$$n_t \leftarrow \nu n_{t-1} + (1-\nu) g_t^2;\ \hat{n}_t \leftarrow \frac{n_t}{1-\nu^t};\ \overline{m}_t \leftarrow (1-\mu_t)\hat{g}_t + \mu_{t+1}\hat{m}_t;\ \theta_t \leftarrow \theta_{t-1} - \eta \frac{\overline{m}_t}{\sqrt{\hat{n}_t} + \varepsilon}$$

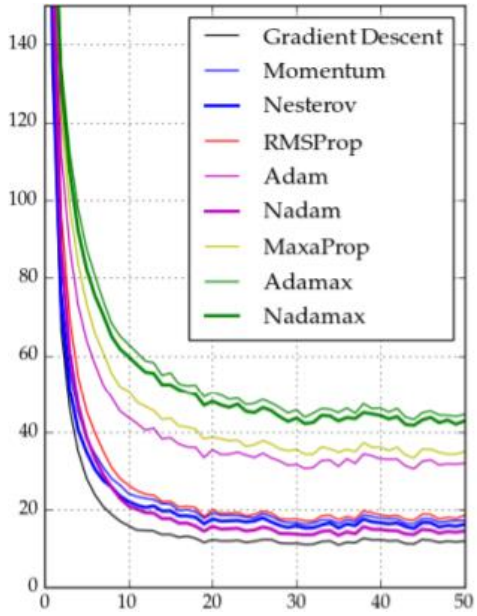


Рисунок 1 - Сравнение производительности алгоритмов.

Методы, использующие RMSProp, производят вектора слов, представляющих взаимоотношения между словами, значительно лучше, чем остальные методы, но RMSProp с импульсом Нестерова (Nadam) работает гораздо лучше, чем RMSProp без импульса и с классическим импульсом (Adam). Также можно заметить, что алгоритмы с NAG работают значительно лучше, чем алгоритмы с классическим импульсом.

УДК 621.319

В. А. Васильева (4 курс бакалавриата)
И. Г. Чернорyцкий, д.т.н., профессор

МУРОВЬИНЫЙ АЛГОРИТМ. ANT COLONY OPTIMIZATION

Тема доклада - Ant Colony Optimizaton, подалгоритм Генетического алгоритма, использующийся для решения задачи коммивояжера.

Идея алгоритма исходит от наблюдения за муравьями в процессе поиска пищи, состоящего из этапов:

1. Первый муравей находит источник пищи (F) любым способом (a), а затем возвращается к гнезду (N), оставив за собой тропу из феромонов (b).
2. Затем муравьи выбирают один из четырёх возможных путей, затем укрепляют его и делают привлекательным.
3. Муравьи выбирают кратчайший маршрут, так как феромоны с более длинных путей быстрее испаряются.

Модель такого поведения заключается в следующем:

- Муравей (так называемый «Блиц») проходит случайным образом от колонии;
- Если он находит источник пищи, то возвращается в гнездо, оставляя за собой след из феромона;
- Эти феромоны привлекают других муравьёв, находящихся вблизи, которые вероятнее всего пойдут по этому маршруту;
- Вернувшись в гнездо они укрепят феромонную тропу;
- Если существует 2 маршрута, то по более короткому, за то же время, успеют пройти больше муравьёв, чем по длинному;

- Короткий маршрут станет более привлекательным;
- Длинные пути, в конечном итоге, исчезнут из-за испарения феромонов.

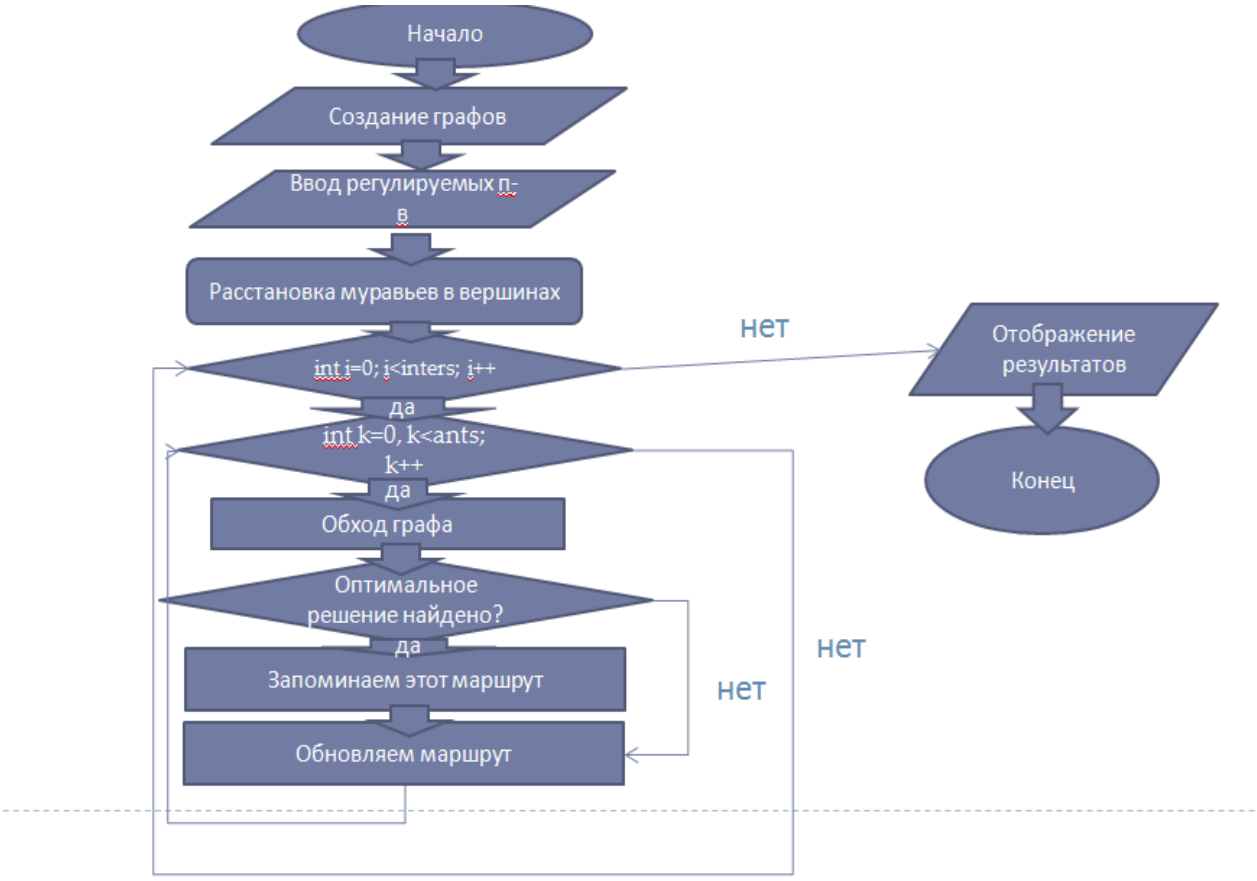


Рисунок 1 – Блок-схема реализации АСО.

УДК 004.896

И. В. Гнатюк (2 курс магистратуры)
С. А. Молодяков, д.т.н., профессор

РАЗРАБОТКА И АВТОМАТИЗАЦИЯ МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРЕДСКАЗАНИЯ ВЫЖИВАЕМОСТИ КЛИЕНТА КОММЕРЧЕСКОГО ПРЕДПРИЯТИЯ

Для многих компаний страхового и банковского сектора необходима постоянная оценка финансовой деятельности клиентов, в частности выживаемости при различных внешних и внутренних воздействующих факторов. Известные публикации по этой теме носят поверхностный характер и не позволяют разработать систему. Целью работы является разработка модели машинного обучения для прогнозирования оттока клиентов. Необходимо разработать модель машинного обучения, которая способна определить “выживаемость” клиентов с точки зрения долгосрочного сотрудничества с коммерческим предприятием. Для того, чтобы сделать предсказание того, продолжит ли конкретный клиент работу с конкретным коммерческим предприятием необходимо отталкиваться от данных, которые есть о клиенте. Сама задача представляет из себя бинарную классификацию, следовательно, критерием выживаемости клиента будут являться метки:

– выжил, в разметке 1;
– не выжил, в разметке 0.

Критерий выживаемости рассчитывается для выборки клиентов, попавших под риск, и, соответственно, для выборки клиентов без рисков.

Необходимо произвести анализ имеющихся о клиенте данных, сконструировать новые признаки, способные улучшить качество модели и очистить данные. В качестве обучающей выборки, были взяты и размечены в соответствии с задачей, клиенты за 2016, 2017 и 2018 годы, а в качестве тестовой, клиенты за 2019. Необходимость нормализации выборок данных обусловлена природой используемых алгоритмов и моделей Machine Learning. Исходные значения признаков могут изменяться в очень большом диапазоне и отличаться друг от друга на несколько порядков, поэтому, завершающим этапом предобработки будет нормализация.

Необходимо выделить признаки, которые можно рассматривать не как отдельные, а как некоторую совокупность, которая может нести в себе шаблоны и, потенциально, способны улучшить качество модели. Для этого необходимо обучить так называемые, эмбеддинги (от англ. *embedding* – вложение). Самобучающийся алгоритм для получения эмбеддингов (в данном случае, использовались SDNE[1], Struct2Vec[2], Node2Vec[3]) выдает, матрицу с эквивалентами динамичным признакам в векторном пространстве. Матрица сохраняется в Hadoop Distributed File System (HDFS).

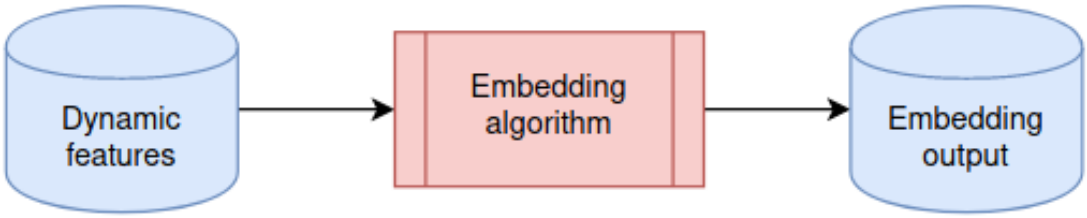


Рисунок 1 – Этап обучения эмбеддингов.

Далее, на эмбеддингах обучается рекурретная нейронная сеть Long short-term memory (LSTM) и выдаёт на выходе совокупное представление всех эмбеддингов. Это делается из-за того, что какой-либо классический алгоритм машинного обучения может не найти сложные закономерности эмбеддингов. Матрицы представлений и статических признаков соединяются в одну большую матрицу признаков.

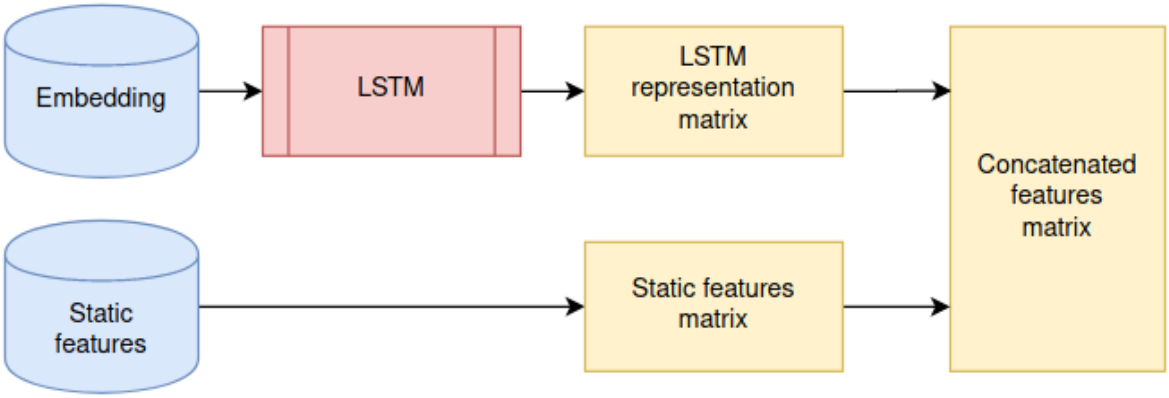


Рисунок 2 – Построения конечной матрицы признаков.

Затем на соединенной матрице обучается итоговый алгоритм принятия решений (в данном случае, логистическая регрессия), который выдаёт предсказанные метки для каждого вектора признаков. Предсказания и обученная модель сохраняются в HDFS, а затем тестируется качество предсказаний модели.

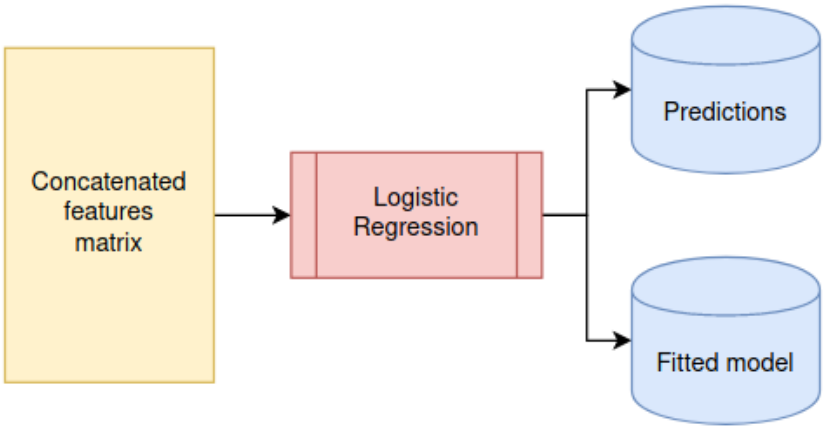


Рисунок 3 –Обучение логистической регрессии.

Обученная модель собирается в production-проект с использованием фреймворка для обработки больших объёмов данных Apache Spark. Затем расчёты запускаются на вычислительном hadoop-кластере с использованием менеджера ресурсов YARN. Модель копируются на каждый worker-узел Spark, чтобы подсчитывать предсказания. В качестве оркестратора используется Apache Airflow, чтобы автоматически производить расчёты в нужные периоды времени. Для быстрого доступа к большим данным можно использовать систему с расширенным кэшированием [4].

Таким образом, мы получили обученную и автоматизированную модель машинного обучения, способную систематически делать точные прогнозы относительно сотрудничества с клиентами.

ЛИТЕРАТУРА

1. Structural Deep Network Embedding / Daixin Wang, Peng Cui, Wenwu Zhu // Department of Computer Science and Technology 2016
2. struc2vec: Learning Node Representations from Structural Identity / Leonardo F. R. Ribeiro, Pedro H. P. Saverese // Federal University of Rio de Janeiro Systems Eng. 2016
3. node2vec: Scalable Feature Learning for Networks / Aditya Grover // Stanford University 2016
4. Ermakov N.V., Molodyakov S.A. Development and implementation of accelerated methods of data access // Journal of Physics: Conference Series 1326(1),012025 2019

УДК 004.67

Д. Е. Дулетов (2 курс бакалавриата)
Е. К. Куликов, аспирант

ИССЛЕДОВАНИЕ АППРОКСИМАЦИИ СИГНАЛОВ ПРИ ПОМОЩИ МИНИМАЛЬНЫХ B-СПЛАЙНОВ МЕТОДОМ ORTHOGONAL MATCHING PURSUIT

Математические модели сигналов, точно описывающие определенные физические процессы, могут быть сложными и малопригодными для использования в прикладных задачах, основанных на математическом моделировании. Кроме того, практическая регистрация сигналов выполняется, как правило, с определенной погрешностью или с определенным уровнем шумов, которые по своим значениям могут быть выше теоретической погрешности прогнозирования сигналов при расчетах по даже очень точным формулам. Поэтому возникает задача аппроксимации – представления произвольных сложных функций простыми и удобными для практического использования таким образом, чтобы

отклонение исходной функции от её приближения в области ее задания было наименьшим по определенному критерию приближения.

Один из наиболее известных подходов к построению приближений связан с использованием сплайн-функций. Аппроксимации, основанные на использовании B-сплайнов, являются де-факто стандартом при построении кривых и поверхностей для различных CAGD-систем. Однако используемые базисы не могут точно представить трансцендентные кривые, часто используемые в прикладном проектировании, поэтому активно исследуются различные подходы построения сплайнов, обладающих свойствами B-сплайнов. Среди них отметим так называемые минимальные сплайны, выводимые из системы тождеств, называемой аппроксимационными соотношениями. Они позволяют использовать неполиномиальные базисы, а также строить сплайны на неравномерных сетках [1].

Стандартный метод построения приближения связан с выбором ряда базисных сплайн-функций и представления исходной функции как линейной комбинации базисных с некоторыми коэффициентами, вычисленными, как правило в соответствии с некоторой локальной схемой. В последнее десятилетие начинает активно развиваться новое направление – разреженная аппроксимация [2]. Сигнал представляется в виде конечной линейной комбинации элементарных функций, выбранных из некоторого большого, в общем случае линейно зависимого набора функций. В контексте аппроксимации сплайнами такой набор называют сплайновым словарём. Отличие от простой аппроксимации состоит в том, что набор, участвующий в разложении линейно зависим. При этом выбор наиболее важных функций осуществляется по некоторому, как правило жадному, алгоритму [3]. Разреженная аппроксимация является одним из наиболее динамично развивающихся и перспективных методов представления сигналов, имеющих малую временную протяженность, поскольку не зависит от качества частотно-временного разрешения.

Алгоритм OMP (Orthogonal Matching Pursuit) является одним из методов разреженной аппроксимации и на данный момент имеет доступную и работающую реализацию только на языке MATLAB.

Целью работы является экспериментальное исследование возможностей улучшения алгоритма OMP при помощи минимальных сплайнов и портирование данного алгоритма на язык C++.

При переносе алгоритма на C++ ожидается повышение мобильности и кроссплатформенности кода, так как разработок под C++ сильно больше и возможно, например, использование на платформе x32. Также будет убрана необходимость в функциях ядра MATLAB. Вторым результатом исследования является возможность улучшения качества разреженной аппроксимации за счёт использования словарей, основанных на минимальных сплайнах, возможности которых ещё не были опробованы на практике. В качестве параметров оценивания будут использоваться скорость вычислений и точность аппроксимации.

ЛИТЕРАТУРА

1. Е.К. Куликов, А.А. Макаров «Об аппроксимации гиперболическими сплайнами», Зап. научн. сем. ПОМИ, выпуск 472 стр. 179-194 (2018)
2. О.О. Луковенкова «Моделирование и обработка импульсных сигналов геоакустической эмиссии на базе разреженной аппроксимации», диссертация на соискание учёной степени ктн (2016)
3. L. Rebollo-Neira «A dedicated greedy pursuit algorithm for sparse spectral representation of music sound», Journal of the Acoustical Society of America, vol. 140, no. 4, p. 2933-2943 (2016)

УДК 004.90

Д. А. Иванов (4 курс бакалавриата)
С. А. Молодяков, д.т.н., профессор

РАЗРАБОТКА НЕЙРО-АЛГОРИТМА ДЛЯ РАСПОЗНАВАНИЯ АККОРДОВ В МУЗЫКАЛЬНЫХ ФРАГМЕНТАХ

В сфере музыкального творчества много высококвалифицированных специалистов и любителей каждый день сталкиваются с задачей определения услышанных ими нот или аккордов. С развитием технологий и алгоритмов цифровой обработки сигналов появилась возможность автоматизировать эту задачу. Автоматическое распознавание высоты звуков – это довольно трудоёмкий процесс цифровой обработки звуковых сигналов с последующим определением высотности нот, звучащих в каждый момент музыкального фрагмента. Трудоёмкость этого процесса определяется рядом факторов, которые нужно учесть при непосредственно самом распознавании. Факторами этими является ритм, громкость, тембр звука и специфика инструмента, с помощью которого воспроизводится музыкальный фрагмент. Также осложняют задачу и некоторые составляющие процесса распознавания: разделение музыкального фрагмента на одинаковые части (фреймы), в пределах которых не подразумевается смена высотности ноты или аккорда; применение оконной функции к каждому фрейму записи; фильтрация шумов исходной записи.

Целью работы является разработка эффективного нейро-алгоритма для распознавания аккордовых последовательностей в музыкальных фрагментах.

В настоящее время алгоритмы по распознаванию одной ноты за один промежуток времени (одноголосной мелодии) достигают в своих результатах высокой точности определения частот звуков, хотя и часто не учитывают специфику тембров инструментов и игры на них. К сожалению, качество распознавания аккордовых (многоголосных) мелодий и вовсе оставляет желать лучшего: эффективные в своей области алгоритмы распознавания одноголосных мелодий в контексте распознавания многоголосных являются непригодными [4]. Также одноголосной музыки в мире намного меньше, и определить порядок звучащих в ней нот человеку проще, и именно поэтому проблема распознавания аккордов стоит куда острее и представляется более серьёзной и сложной задачей [3].

Задача распознавания многоголосных мелодий и аккордов в контексте машинного обучения в первую очередь требует сформированной унифицированной выборки, на основе которой будет проходить обучение. Было принято решение создать данную выборку, состоящую из двухсекундных wav-файлов, в каждом из которых будет проигрываться определенный аккорд. От каждой ноты в пределах пяти октав от большой до третьей записано по 36 различных наиболее часто встречающихся аккордов, каждый из которых имеет свое наименование (большие и малые минорные и мажорные трезвучия, секстаккорды, квартсекстаккорды, септаккорды, доминантсептаккорды и т. д. [1]). Запись каждого аккорда будет производиться дважды для достижения более высокой надёжности. Таким образом, обучающая выборка будет вмещать в себя 4320 двухсекундных wav-файлов. Классов для классификации у алгоритма будет 432 (по 36 аккордов от каждой ноты в пределах одной октавы).

Следующий задачей, подлежащей решению, является задача трансформирования цифровых представлений звуковой волны в результирующие векторы, которые будут подаваться на вход алгоритма. Файлы wav-формата подразумевают дискретно фиксирование амплитуды звуковой волны через очень малый фиксированный промежуток времени. Таким образом, мы можем также фиксировать амплитуду нашего звукового сигнала через некоторые равные промежутки времени для получения результирующих векторов для каждого wav-файла. Наглядно процедура трансформации цифрового представления в векторное представлена на рис. 1. При этом важно выбрать оптимальный промежуток времени, через который берётся очередное значение амплитуды. Если выбрать слишком большой

промежуток, то форма изначального сигнала может заметно искажаться, что в последствии приведёт к ошибкам распознавания. Если же взять слишком маленький шаг, то, во-первых, это приведёт к заметному увеличению требований к вычислительным мощностям, и, во-вторых, алгоритм рискует достичь эффекта переобучения, который проявляется в том, что разработанная модель будет эффективна при распознавании с обучающей выборки, но при этом будет относительно плохо классифицировать аккорды из тестовой выборки.

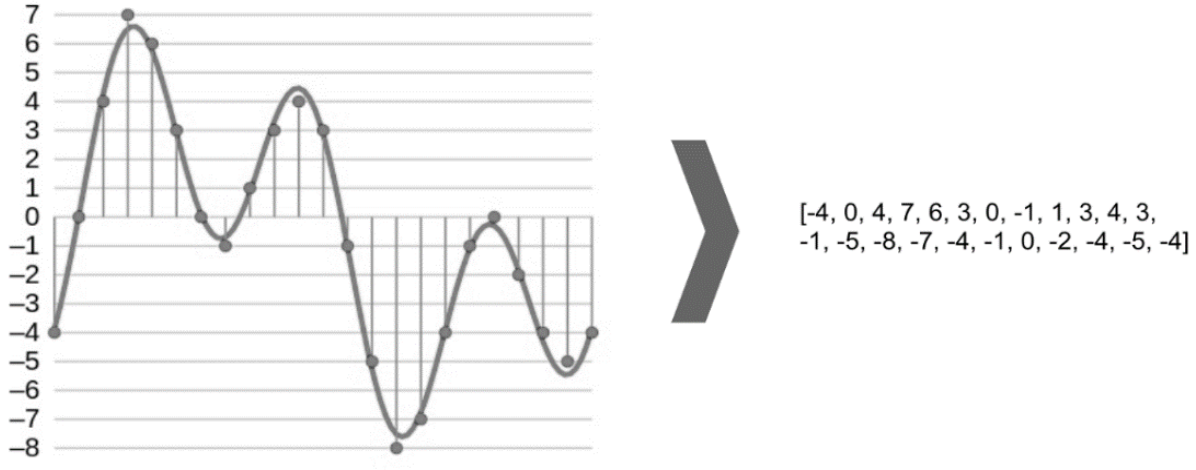


Рисунок 1 –. Трансформация цифрового представления звуковой волны в векторное [2]

Разрабатываемая модель будет содержать в себе модуль для фильтрации шумов в музыкальном фрагменте, подлежащем распознаванию; модуль для трансформации цифрового представления звуковой волны в векторное, а также самую главную часть, включающую в себя процесс машинного обучения на сформированной выборке.

Хочется отметить, что модель разрабатывается на выборке звуковых файлов с аккордами, ноты которых располагаются лишь в пределах одной октавы. Однако в музыкальных произведениях часто распространена практика переноса одной или нескольких нот в другую октаву. Аккордов такого формата в обучающей выборке нет, поэтому при работе с такими аккордами ожидается потеря точности распознавания модели, однако дать количественную оценку данной величины пока что не представляется возможным. Добавление таких нестандартных широких аккордов к обучающей выборке не рассматривается, так как в противном случае придётся по нескольку раз модифицировать 432 аккорда путём перестановок каждой ноты из трёх или четырёх на одну или две октавы вверх или вниз, что может увеличить обучающую выборку в сотни раз.

Также в разрабатываемой модели необходимо учесть возможные искажения от тембров инструментов. К примеру, национальные инструменты азиатских стран имеют характерный гнусавый отзвук, а ноты на некоторых духовых инструментах в первое мгновения игры могут слышаться на октаву ниже. Если первую проблему можно решить с помощью выбора оптимального фильтра, то вторая связана с нетривиальным выбором длины фрейма и оконной функции. Все эти факты и сложности наталкивают на рассмотрение варианта модели, где пользователь при взаимодействии с ней мог бы указывать инструмент, на котором исполняется музыкальный фрагмент, подлежащий распознаванию.

ЛИТЕРАТУРА

1. Вахромеев В. Элементарная теория музыки. – М.: 1961. с. 241.
2. Smales M. Sound Classification using Deep Learning. 2019 [Электронный ресурс]. URL: https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7 (дата обращения: 12.03.2020)
3. Nakamura, Eita & Benetos, Emmanouil & Yoshii, Kazuyoshi & Dixon, Simon. (2018). Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization. 101-105. 10.1109/ICASSP.2018.8461914.
4. Quenneville, D.: Automatic Music Transcription. Ph.D. thesis, Middlebury College (2018)

УДК 004.75

К. С. Кобышев (2 курс магистратуры)
И. В. Никифоров, к.т.н., доцент

АЛГОРИТМ ПОДБОРА ИЗОБРАЖЕНИЙ, АНАЛИЗИРУЮЩИЙ ИНФОРМАЦИЮ О ВИЗУАЛЬНЫХ ПРИЗНАКАХ

Рекомендательные системы – это программные средства, используемые для подбора пользователю Интернета материалов: книг, фильмов, статей, товаров и так далее. Разработка подобных систем проводится с применением методов математической статистики, машинного обучения, теории оптимизации, дискретной математики.

Один из видов рекомендаций связан с подбором изображений, которая применяется в веб-приложениях, содержащих графические материалы. На данный момент существует три подхода для рекомендации изображений: использовать для рекомендаций информацию о метаданных изображений [1], использовать информацию о визуальных признаках изображений [2][3] и использовать информацию о гибридных признаках [4], сгенерированную из метаданных и визуальных признаков. Решения, основанные на данных подходах, не позволяют одновременно учесть интересы пользователя и избежать ручного заполнения метаданных изображений.

В данной исследовательской работе был предложен новый алгоритм для рекомендации изображений, согласно которому взаимосвязи между пользователями и изображениями представлены в графовой структуре. Предложенный алгоритм основан на подходе, связанном с анализом визуальных признаков изображений. Рекомендательная система извлекает из каждого из изображений список образов и помещает списки образов в графовую БД. Изображения, которые заинтересовали пользователя также преобразуются в список образов, после чего формируется список интересов пользователя, который тоже помещается в графовую БД (Рис. 1).

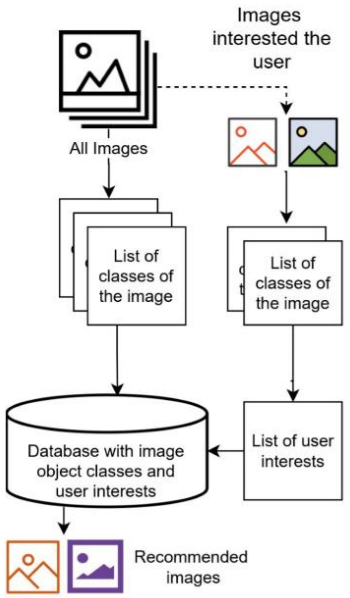


Рисунок 1 – Концепт предлагаемого решения

На рис. 2 представлен пример графовой базы данных. Графовая база данных состоит из ребер и узлов трех типов: пользователи, тематики изображения. Пользователи связаны с тематиками, которые представляют собой интересы пользователя.

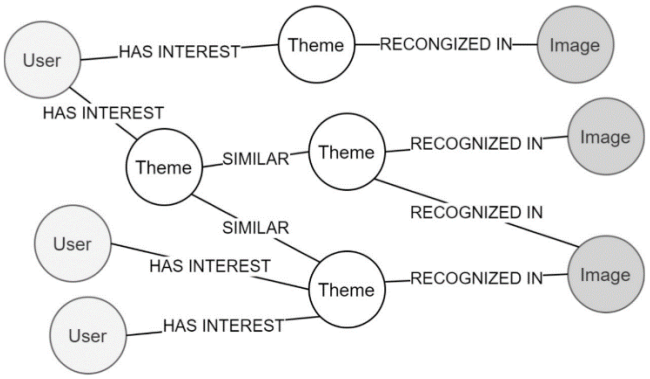


Рисунок 2 – Пример графовой базы данных

Вес ребра между пользователем и тематикой пропорционален степени заинтересованности пользователя в данной тематике и вычисляется согласно формуле (1), где M_{AB}- степень заинтересованности пользователя A в тематике B, M_{Ai}- степень заинтересованности пользователя A в тематике i.

$$weight(E_{U_A T_B}) = \frac{M_{AB}}{\sum M_{Ai}} \qquad (1)$$

Степень заинтересованности пользователя в какой-либо из тематик определяется образами изображений, которые заинтересовали его прежде, и определяется по формуле (2), где $classes(I_m)$ – список образов изображения I_m, $coef(T_m)$- коэффициент типа действия пользователя над изображением (каждому типу действия пользователя соответствует свой заданный коэффициент).

$$M_{Ai} = \sum P(i \in classes(I_m)\ x\ coef(T_m) \qquad (2)$$

Изображения связаны ребрами с тематиками, которые являются образами для изображений. Вес ребра между изображением и тематикой равен вероятности того, что данная тематика является образом для изображения.

Подграф из тематик является постоянным и формируется из модели Word2Vec. Преобразование модели Word2Vec в подграф тематик описан в [5] и представляет собой алгоритм иерархической кластеризации. Сначала определяются кластера 1 уровня, для каждого из кластеров определяются центральные слова. На следующей итерации определяются кластеры 2 уровня, объединяющие центральные слова кластеров 1 уровня. Данная процедура повторяется до того момента, пока не будут определены кластеры заданного N-го уровня.

В результате исследования был разработан прототип рекомендательной системы, где Neo4J был использован в качестве графовой БД. Был проведен подбор параметров алгоритма, в результате которого определена максимальная F1-мера, которую возможно достичь в данном алгоритме, равная ~0.59.

ЛИТЕРАТУРА

1. Fuhu Deng, Panlong Ren, Zhen Qin, Gu Huang, Zhiguang Qin, Leveraging Image Visual Features in Content-Based Recommender System, Hindawi, April 2018.
2. Kemal Ozkan, Zuhal Kurt, Erol Seke, Image-based recommender system based on feature extraction techniques, International Conference on Computer Science and Engineering (UBMK), October 2017.
3. Hessel Tuinhof, Clemens Pirker, Markus Haltmeier, Image Based Fashion Product Recommendation with Deep Learning, Volterra, Italy: International Conference on Machine Learning, Optimization, and Data Science Machine Learning, Optimization, and Data Science 4th International Conference, July 2018.
4 Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, Graph Convolutional Neural Networks for Web-Scale Recommender Systems, arXiv, June 2018.

УДК 004.855.5

В. С. Ларионов, Д. К. Курсанин (4 курс бакалавриата)
И. Г. Черноруцкий, д.т.н., профессор

ОПТИМИЗАЦИЯ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ МЕТОДА РОЯ ЧАСТИЦ

Бурное развитие искусственного интеллекта в последнее десятилетие выдвинуло на передний план тему нейронных сетей. Нейронная сеть — это последовательность нейронов, соединенных между собой синапсами. Структура нейронной сети пришла в мир программирования прямиком из биологии. Благодаря такой структуре, машина обретает способность анализировать и даже запоминать различную информацию. Сфера применения нейронных сетей включает в себя такие задачи, как: распознавание образов и классификация, принятие решений и управление, кластеризация, прогнозирование и многие другие.

Существуют множество методов, пригодных для обучения нейронных сетей. Обычно для этой цели используют один из стандартных методов – Метод обратного распространения. Он используется, т. к. он обеспечивает высокую скорость и приемлемую точность результатов. Однако, существуют ситуации, при которых важнее точность метода, а не его скорость. В таких случаях обычно ищут альтернативные методы для обучения нейронной сети, которые могут обеспечить результат, не уступающий методу обратного распространения по точности при любом наборе входных данных.

Описание цели работы

В данной работе рассматривается возможность использования альтернативных методов обучения для нейронной сети, таких, как метод роя частиц и его возможные модификации, с целью прогнозирования результатов по входным данным. На основе полученных результатов производится сравнение скорости и точности с результатами стандартного метода (обратного распространения).

Описание исследования

Начальным этапом исследования является реализация нейронной сети и метода обратного распространения для подсчета весов и обучения сети. Производится оценка точности предсказания и времени работы программы.

Следующим этапом является реализация метода роя частиц, и его интеграция в нейронную сеть в качестве метода обучения для подсчета весов. Производится оценка точности предсказания полученных результатов (сравнение с известными результатами на определенном наборе данных) и скорости работы программы по отношению к полученным ранее с помощью алгоритма обратного распространения. Делаются предварительные выводы о целесообразности замены в нейронной сети обучающего метода и его сравнение со стандартным на предмет зависимости его от выбранных параметров в ходе реализации.

Конечным этапом является реализация оставшихся методов PSO семейства – метода комплексного обучения роя частиц (CLPSO), метода комплексного обучения роя частиц с адаптивными параметрами (CLPSO-ap), улучшенная комплексная оптимизация роя частиц (ECLPSO), их интеграция в нейронную сеть в качестве метода обучения для подсчета весов. Производится оценка точности предсказания полученных результатов и скорости работы в сравнении с предыдущими этапами. Делаются выводы о семействе PSO методов как о возможных альтернативных методах обучения нейронных сетей.

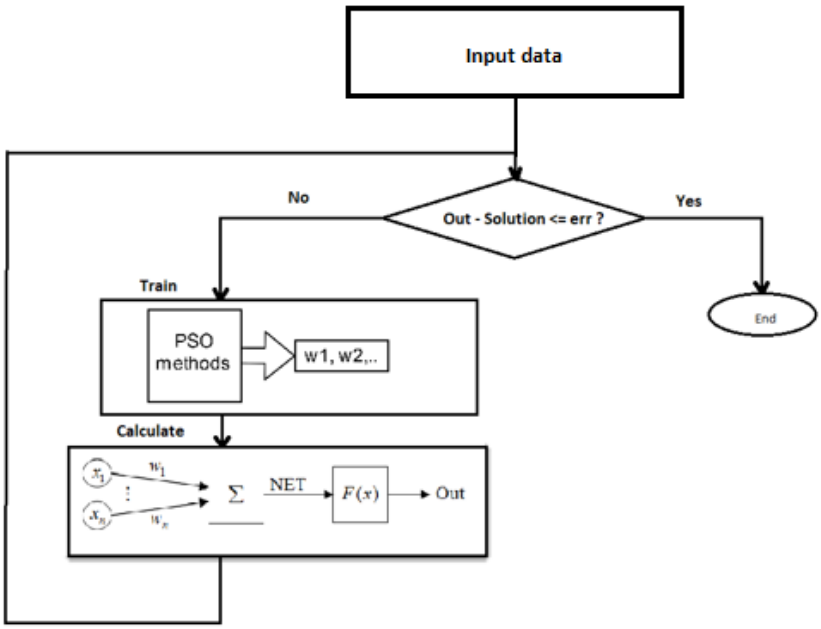


Рисунок 1 – Схема обучения нейронной сети семейством PSO-методов

Подведение итогов, оценка достигнутых результатов

В результате проделанной работы было установлено, что, как и сам метод роя частиц, так и разработанные на его основе модификации, показали себя не хуже алгоритма обратного распространения, с точки зрения точности получаемых результатов.

В ходе исследования рассматриваемое семейство методов показало меньшую скорость обучения нейронной сети на основе входных данных, т. к. в этих методах данное обстоятельство зависит от границ, которые задаются для высчитывания скорости сходимости, влияющей на быстроту приближения к оптимальному значению весов.

В большинстве случаев использование PSO-методов для обучения нейронной сети занимает немного больше времени, чем обратное распространение. В ситуациях, когда важно время обучения нейронной сети, рекомендуется использование алгоритма обратного распространения. В случаях, когда для решения проблемы не важны временные затраты, рекомендуется использование PSO-методов.

УДК 004.023

Д. П. Меркурьева, С. И. Дамирова (4 курс бакалавриата)
И. Г. Черноруцкий, к.т.н., профессор

ИСПОЛЬЗОВАНИЕ АЛГОРИТМОВ МЯГКИХ ВЫЧИСЛЕНИЙ ДЛЯ ОПТИМИЗАЦИИ РАБОТЫ АЛГОРИТМА КЛАСТЕРИЗАЦИИ K-MEANS

Целью работы является исследование одного из способов оптимизации классического алгоритма кластеризации K-means (KM), разработка оптимизированного алгоритма кластеризации на основе метода KM и анализ его эффективности.

Метод KM реализуется сначала путем случайного выбора *k* начальных центроидов, а затем эвристически минимизирует сумму квадратов расстояний, таких как Евклидово расстояние или расстояние Манхэттена, между каждой точкой данных до центроидов.

KM является эвристическим алгоритмом и имеет два недостатка:

1. Его результат чувствителен к выбору начальных центроидов.
2. Алгоритм может остановиться в локальном оптимуме.

Для устранения чувствительности к выбранным начальным центроидам будем использовать модификацию метода КМ - алгоритм k-harmonic means (KHM). В KHM данные группируются таким образом, чтобы сумма средних гармонических расстояний между каждым объектом и всеми центроидами кластеров будет минимизирована. Для устранения проблемы возможной остановки метода в локальном оптимуме будем использовать механизм обновления из упрощенного метода роя частиц (УМРЧ) для изменения значений центроидов.

Алгоритм достаточно прост: создается N решений, каждое решение представляет собой набор центроидов. Каждое решение улучшается с помощью механизма обновления. Механизм обновления сочетает в себе локальный поиск, глобальный поиск и переход к случайно сгенерированному допустимому значению, который также является глобальным поиском для повышения способности алгоритма отойти от локального оптимума.

Классический механизм обновления может нарушить тенденцию и устойчивость сходимости резкими переходами центроида к случайно сгенерированному допустимому значению.

В алгоритме улучшенного УМРЧ новый сгенерированный центроид будет основан на значении предыдущего центроида с вносимыми механизмом обновления изменениями.

Механизм обновления улучшенного УМРЧ представлен формулой:

$$c_{j,k} = c_{j,k} + \rho_1 - \rho_2 * \begin{cases} (c_{gBest,k} - c_{j,k}), & \text{если } \rho_c \in [0, C_g) \\ (c_{gBest,k} - c_{y,k}), & \text{если } \rho_c \in [C_g, C_w) \\ (c_{x,k} - c_{j,k}), & \text{если } \rho_c \text{ принимает иные значения} \end{cases}$$

Где $j = 1, 2, \ldots, Nsol$, где $Nsol$ – количество решений;

- $k = 1, 2, \ldots, K$ – число центроидов;
- ρ_1, ρ_2 и ρ_c - случайные числа из равномерного распределения в пределах [0,1];
- $c_{j,k}$ – выбранный центроид;
- C_w и C_g – заданные константы;
- $c_{gBest,k}$ – k-й центроид решения с лучшем значением функции приспособленности;
- x, y – случайные значения из $\{1, 2, \ldots, K\}$.

В качестве единого показателя близости решения к оптимальному решению будем использовать функцию приспособленности, которая будет иметь следующий вид:

$$F(c_1, c_2, \ldots, c_k) = \sum_{i=1}^{N} \frac{K}{\sum_{k=1}^{K} \frac{1}{||X_i - c_k||^p}}$$

Где $F(c_1, c_2, \ldots, c_k)$ – значение функции приспособленности для одного решения с;

- $k = 1, 2, \ldots, K$ – число центроидов;
- N – количество точек;
- c_k – k-й центроид решения с;
- X_i – i-я точка данных;

В результате были реализованы 2 алгоритма K-harmonic means и улучшенный упрощенный метод роя частиц для оптимизации K-harmonic means. Для сравнения реализованных методов с классическим КМ будет использоваться алгоритм K-means библиотеки sklearn. Алгоритмы и программное обеспечение для их использования были реализованы на языке Python 3.6.

Рассмотрим пример. На вход 3 программам подадим одинаковые данные, сгенерированные с помощью библиотеки sklearn.datasets с помощью функции make_blobs. Данные приставляют из себя набор из 600 точек, формирующий 6 кластеров. Однако два кластера данных будут находиться достаточно близко друг к другу.

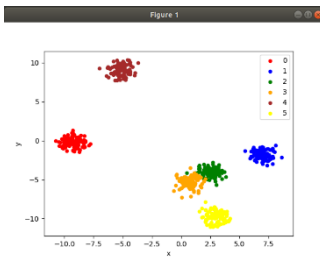


Рис. 1 – Исходные кластеры

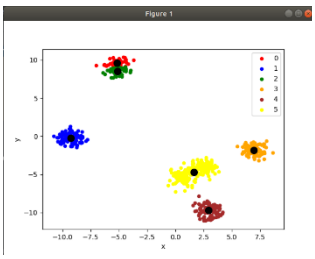


Рис. 2 – Результат KM

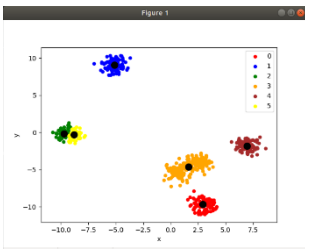


Рис. 3 – Результат KHM

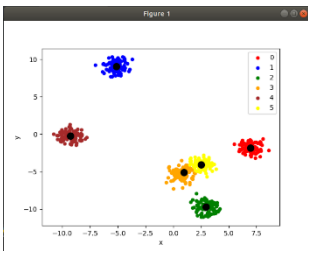


Рис. 4 – Результат оптимизированного KHM

На рис. 2 и рис. 3 видно, что алгоритм KM, как и KHM, не верно определил кластеры. Значение функции приспособленности для метода KHM:

$$F = 1002231.4611013364$$

Несмотря на высокое значение функции приспособленности алгоритм не справился с задачей и принял близко расположенные друг к другу кластеры за один. На рис. 4 видно, что метод оптимизированного KHM единственный верно определил кластеры. Значение функции приспособленности выбранного оптимизированным алгоритмом KHM решения:

$$F = 1016481.6313296341$$

Полученный оптимизированный алгоритм KHM будет слабо чувствителен к начальной инициализации центроидов, благодаря использованию суммы средних гармонических расстояний при подсчете расстояния от точек кластера до центроидов. Благодаря модификации посредством использования механизма обновления из метода роя частиц алгоритм будет менее восприимчив к проблеме попадания в локальный оптимум, что будет давать лучшие результаты при кластеризации данных.

ЛИТЕРАТУРА

1. Yeh W-C., Jiang Y, Chen Y-F., Chen Z. (2016) New Soft Computing Method for K-Harmonic Means Clustering - [Электронный ресурс] URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5112810/pdf/pone.0164754.pdf
2. Bin Zhang, Meichun Hsu, Umeshwar Dayal K-Harmonic Means - A Data Clustering Algorithm - URL: https://www.hpl.hp.com/techreports/1999/HPL-1999-124.pdf Software Technology Laboratory HP Laboratories Palo Alto HPL-1999-124 October, 1999. С. 1 – 4
3. Шамин Р.В. Практическое руководство по машинному обучению. — М.: Научный канал Lector.ru, 2019. С. 88 - 92.
4. Leong KH, Abdul-Rahman H, Wang C, Onn CC, Loo SC (2016) Bee Inspired Novel Optimization Algorithm and Mathematical Model for Effective and Efficient Route Planning in Railway System - [Электронный ресурс] URL: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0166064&type=printable

УДК 004.932.2

Г. Р. Мирзазянов (2 курс бакалавриата)
Е. К. Куликов (1 курс аспирантуры)

РЕАЛИЗАЦИЯ АЛГОРИТМА ВЫЧИСЛЕНИЯ МУЛЬТИФРАКТАЛЬНОГО СПЕКТРА С ПОМОЩЬЮ ФУНКЦИИ ПЛОТНОСТИ

Работа посвящена решению одной из актуальных задач нахождения классификационных признаков цифровых изображений, обладающих сложной структурой. Такие изображения возникают в самых разных предметных областях, таких как геология (изображения земных ландшафтов), биология (изображения процессов, происходящих в живых клетках), медицина (снимки всевозможных тканей), техника (изображения срезов металлов) и часто представляют собой мультифракталы --- множества, состоящие из нескольких сложно расположенных фракталов, каждый из которых имеет свою фрактальную размерность. Набор значений этих

размерностей называется мультифрактальным спектром и может использоваться как классификационный признак.

Мы используем метод, предложенный в [1] и основанный на вычислении так называемой локальной функции плотности. Для каждой точки $x \in R^2$ обозначим за $B(x,r)$ квадрат с центром в x и "радиусом" r . Под радиусом в данном случае будем понимать половину длины стороны. В качестве меры квадрата $\mu\big(B(x,r)\big)$ мы используем сумму интенсивностей составляющих его пикселей. Предположим, как обычно в случае анализа фрактальных множеств, что $\mu\big(B(x,r)\big) = kr^{d(x)}(x)$, где $d(x)$—локальная функция плотности, k — некоторая константа. Функция плотности в точке x определяется следующей формулой:

$$d(x) = \lim_{r\to 0} \frac{\log \mu(B(x,r))}{\log r}.$$

Эта величина характеризует степень неоднородности распределения интенсивности в окрестности точки x. Точки x, для которых функция плотности равна α, образуют множество уровня $E_\alpha = \{x \in R^2 : d(x) = \alpha\}$. На практике обычно рассматривают множества $E(\alpha,\varepsilon) = \{x \in R^2 : d(x) \in [\alpha, \alpha+\varepsilon)\}$. Множества уровня представляют собой бинарные изображения, поэтому для каждого из них вычисляется емкостная размерность. Совокупность этих размерностей образует мультифрактальный спектр $f(\alpha)$ исследуемого изображения.

Визуализация множеств уровня позволяет лучше представить разбиение изображения на составляющие его подмножества. Важной особенностью данной реализации является представление изображения в 3D, которое является послойным изображением множеств уровня.

Результат работы алгоритма показан на примере трех изображений, представляющих собой результаты тестов кристаллизации для трех различных видов молока. (Метод кристаллизации заключается в добавлении малой дозы тестируемого вещества к раствору хлорида меди. Структура полученного кристалла позволяет судить о свойствах вещества.) На первом изображении показан результат кристаллизации для молока, которое не подвергалось дополнительной обработке (biodynamically produced raw milk, BD milk), на графиках ему соответствует обозначение BD, на второй результат для пастеризованного молока (microfiltrate homogenized and pasteurized, MHP milk, обозначение MHP), на третьей для ультрапастеризованного (ultra-heat treated milk, UHT milk), обозначение UHT соответственно. Все изображения имеют одинаковый размер 2016x1512 пикселей и сохранены в цветовой модели RGB.

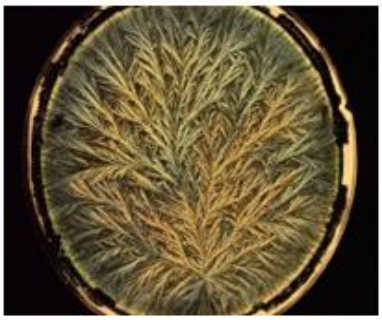


Рис. 1 BD milk

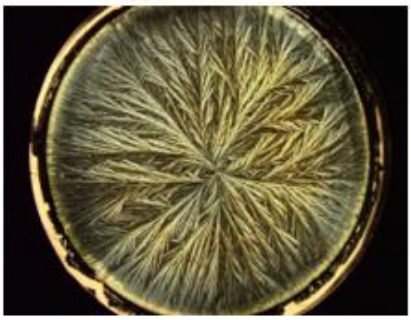


Рис. 2 MHP milk

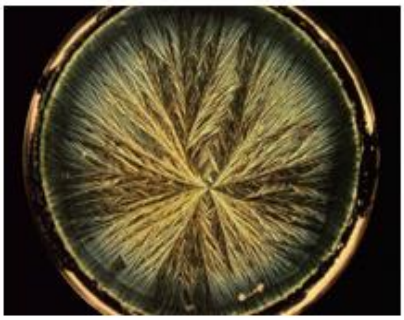


Рис. 3 UHT milk

Для проведения экспериментов изображения были преобразованы в монохромные путём усреднения значений компонент.

На рис. 4 показаны мультифрактальные спектры для каждого из анализируемых изображений. По оси Ox отмечены значения функции плотности, соответствующие началу интервала $[\alpha, \alpha+\varepsilon)$, а по оси Oy — фрактальные размерности множеств $E(\alpha,\varepsilon)$. Шаг между значениями функции плотности (ε) был взят равным 0,4. Видно, что, третий график несколько

отличается от первых двух. Таким образом, рассматриваемый метод позволил отличить структуру изображения для ультрапастеризованного молока от остальных.

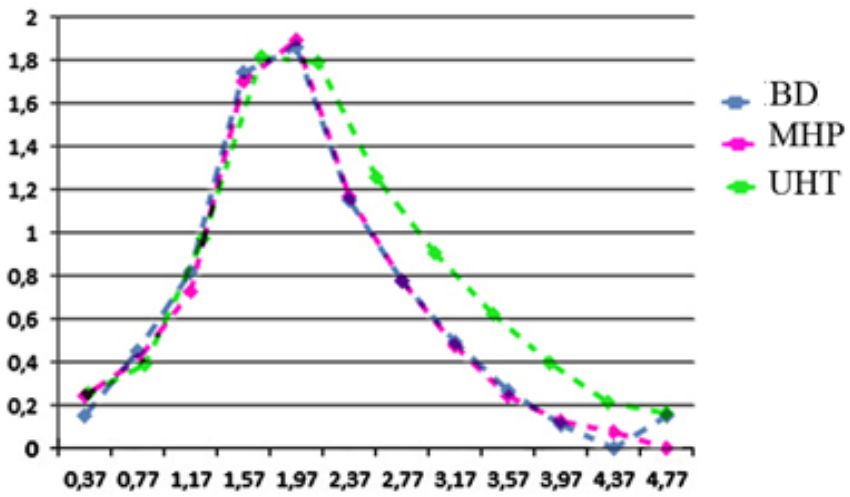


Рис. 4 –Мультифрактальные спектры для изображений молока в полутоновой палитре

Инструментарий реализован на языке Python. Ввиду того, что картинка представляет двумерный массив, была использована библиотека NumPy для оптимизации программы, так как стандартные структуры данных сильно проигрывают по скорости работы. Возможности работы OpenCV с изображениями позволили упростить архитектуру программы и производить мгновенную конвертацию в нужный тип.

ЛИТЕРАТУРА

1. Xu Y., Ji H., Fermüller C. Viewpoint invariant texture description using fractal analysis. — International Journal of Computer Vision, no. 83, 2009, p. 85–100, (2009)

УДК 621.319

А. О. Плеханов (4 курс бакалавриата)
И. Г. Черноруцкий, д.т.н., профессор

АЛГОРИТМ ОПТИМИЗАЦИИ, ОСНОВАННЫЙ НА ПОВЕДЕНИИ МОРСКИХ ЛЬВОВ

Целью работы является рассмотрение алгоритма, основанного на поведении морских львов во время охоты - See Lion Optimization Algorithm. Этот алгоритм решает задачу минимизации функций.

Данный алгоритм дает хорошие результаты по сравнению с другими эволюционными алгоритмами. Это обусловлено тем, что он очень точно описывает охотничье поведение морских львов, которые за миллионы лет эволюции довели свою технику охоты до совершенства.

Алгоритм SLnO отличается от других методов (таких, как градиентный спуск) тем, что он не требует вычисления производных функции, а непосредственно вычисляет значение функции в разных точках. При этом вычисления не зависят друг от друга, что позволяет выполнять их параллельно, тем самым увеличив производительность.

Данный алгоритм условно можно разделить на четыре фазы:

1) Фаза обнаружения и отслеживания добычи:

Один морской лев может обнаружить добычу и подать сигнал другим участникам охоты. Этот лев условно считается лидером охоты, другие же участники охоты должны обновлять свои позиции по отношению к цели охоты. Целью является лучшее текущее решение или решение, наиболее близкое к оптимальному.

2) Фаза вокализации:

Морские львы переговариваются друг с другом во время охоты. Причем они подают сигналы не только непосредственным участникам охоты, но и тем, кто остался на берегу. Поэтому львам удобнее гнать добычу к поверхности океана. Когда лев обнаруживает добычу, он подает сигнал об атаке другим участникам.

3) Фаза атаки (фаза эксплуатации):

Лидер охоты является лучшим поисковым агентом. Агенты распознают целевую жертву, которая считается кандидатом на лучшее решение. Но разные поисковые агенты могут находить новые, лучшие решения.

4) Поиск добычи (фаза исследования).

Данный алгоритм хорошо проявляет себя для сложных функций, у которых тяжело вычислить производные. Однако, его нельзя использовать для обучения нейронных сетей, поскольку требуется варьировать большое количество параметров.

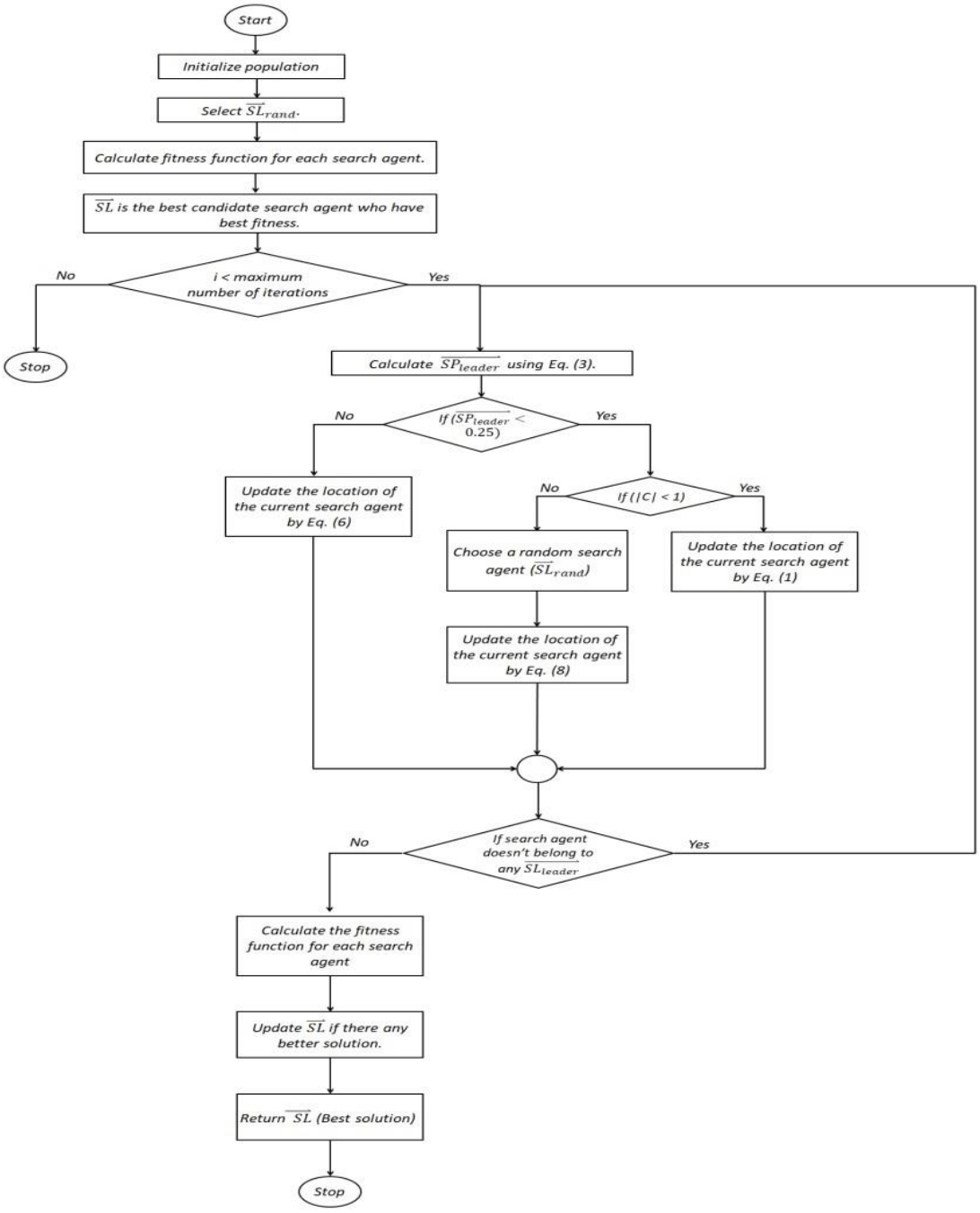


Рисунок 1 – Блок-схема алгоритма SLnO.

УДК 004.832

С. А. Черноусова (4 курс бакалавриата)
И. В. Шошмина, к.т.н, доцент

РАЗРАБОТКА ЛОГИЧЕСКОГО АГЕНТА ДЛЯ ОБУЧАЮЩЕГО ТРЕНАЖЕРА “МИР ВАМПУСА”

Настоящая работа посвящена разработке и реализации агента, решающего задачу поиска клада в мире Вампуса с помощью логических ограничений. В дальнейшем предполагается использовать этого логического агента как модуль обучающего тренажера в дистанционном курсе «Математическая логика» профессора Ю. Г. Карпова [1].

В курсе «Математическая логика» темы «Бинарные решающие ограничения», «Символьные вычисления» позволяют получить обучающимся навыки практического применения логики для решения задач. Сейчас для раскрытия этих тем используются несложные аналитические задачи, а программистские навыки отрабатываются на статических логических задачах, в частности, на варианте задачи Эйнштейна [2]. Однако, как показывает практика, освоение этих тем, по-прежнему, вызывает у учащихся значительные затруднения. Чтобы помочь студентам справиться с материалом, хотелось бы повысить их вовлеченность в процесс решения задач. Для этого предлагается повысить интерактивность процесса решения задач, увеличить количество задач, сохраняя единообразие и поддерживая вариативность.

Для достижения этой цели планируется разработать обучающий тренажер задачи «Мир Вампуса». «Мир Вампуса» - компьютерная игра, придуманная Грегори Йобом. Несмотря на довольно скромный игровой процесс «Мир Вампуса» является отличной средой для испытания интеллектуальных агентов, что заметил Майкл Генезерет, предложивший использовать эту игру для обучения студентов логике [3]. Охотник «ходит» по пещере, расчерченной на клетки, пытается не упасть в яму и избежать Вампуса. Игра выигрывается охотником, если он находит клад и успешно выбирается из пещеры. Эта игра может быть реализована с помощью логического агента, который осуществляет перемещение по пещере, а для поиска новой безопасной непосещенной клетки запрашивает базу знаний. База знаний описывается с помощью формул математической логики. Существует много реализаций этой игры (например, написанная на языке IndiGolog [4] или на основе платформенно-независимой системы SNePS [5]), отличительной особенностью нашего решения является разработка тренажера на основе этой игры, а также то, что все решения в базе знаний осуществляются с применением только формул математической логики. Планируемый интерфейс пользователя нашего обучающего тренажера представлен на рис. 1: есть реальная карта мира Вампуса, которая отображается для студента, карта, по которой движется агент, и окно с логическими выражениями, которые должен добавить обучающийся, чтобы добавить к базе знаний, в соответствии со сведениями, имеющимися у агента.

Карта мира Вампуса, известная агенту — Реальная карта мира Вампуса — Логические ограничения

Рисунок 1 – Планируемый интерфейс пользователя нашего обучающего тренажера. В изображении интерфейса использованы рисунки из [3].

Ключевым модулем обучающего тренажера «Мир Вампуса» является логический агент с базой знаний. Разработка и реализация этого агента и является основной задачей данной работы.

В настоящее время существует несколько реализаций логических агентов, “живущих” в мире Вампуса. Так, при помощи метода генетического программирования на основе логических схем был сформирован реактивный агент, способный эффективно «проходить» игру «Мир Вампуса» [6].

Также существует реализация логического агента на основе платформенно-независимой системы SNePS, написанной на языке Common Lisp [5]. Агент на основе SNePS имеет многоуровневую архитектуру с интегрированными рассуждениями, вдохновленную человеческой психологией и биологией.

Основная причина, по которой существующие реализации логических агентов мира Вампуса не могут быть применены для настоящего тренажера – невозможность использовать и редактировать их код.

Логический агент состоит из базы знаний, собственно агента и карты, по которой агент передвигается.

База знаний, в свою очередь, содержит правила мира Вампуса, модули выбора пункта назначения и маршрута. Для представления логических ограничений используются BDD, обеспечивающие компактное и каноническое представление функций и множеств, а для выбора маршрута агента решается задача анализа достижимости вершины на графе.

Карта включает в себя модуль движения логического агента, его физическое перемещение по карте.

Логический агент принимает решения, основываясь на информации, поступающей через его датчики от карты и действий, переданных ему базой знаний в качестве отклика на эту информацию.

Таким образом, на данный момент времени формализованы правила мира Вампуса, а также реализована база знаний агента и работа с ней. В дальнейшем планируется закончить работу над модулем выбора маршрута и движением агента по карте.

ЛИТЕРАТУРА

1. Карпов Ю. Г., Шошмина И. В. Курс «Математическая логика» ссылка https://openedu.ru/course/spbstu/MATLOG/ дата доступа 18.03.2020
2. Zebra Puzzle ссылка https://en.wikipedia.org/wiki/Zebra_Puzzle дата доступа 18.03.2020
3. Рассел С., Норвиг П. Искусственный интеллект. Современный подход. -Москва: Вильямс, 2006. - 1409 с.
4. Sardina S., Vassos S. The Wumpus World in IndiGolog: A Preliminary Report // Proceedings of the Workshop on Non-monitonic Reasoning, Action and Change in IJCAI. -2005. -№05. С-90-95.
5. Shapiro S., Kandefer M. A SNePS Approach to The Wumpus World Agent or Cassie Meets the Wumpus // Proceedings of the Workshop on Non-monitonic Reasoning, Action and Change in IJCAI. -2005. -№05. С-96-103.
6. Ю. В. Казанцева, Л. В. Липинский Автоматизированное формирование реактивных агентов с помощью генетического программирования // Актуальные проблемы авиации и космонавтики. - 2016. -№12. -С.35-36.

УДК 004.42

А. Б. Чикишева (2 курс магистратуры)
С. П. Воскобойников, к.ф.м.н., доцент
Е. Н. Попов, к.ф.м.н., доцент

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ МОДЕЛИРОВАНИЯ ДИФФУЗИИ ЯДЕРНОЙ НАМАГНИЧЕННОСТИ КСЕНОНА

Изучение механизмов релаксации ядерной намагниченности ксенона является важной прикладной задачей в магнитометрии и навигации. Газовые ячейки, содержащие смесь двух изотопов ксенона с разными ядерными гиромагнитными отношениями, используются в качестве чувствительных элементов гироскопов. Преимущество ксенона в качестве рабочего вещества состоит в невосприимчивости прибора к линейным ускорениям, ударам и вибрации, в результате чего ядерная намагниченность не разрушается при столкновениях между атомами.

Одним из наиболее важных негативных факторов, которые влияют на скорость релаксации ядерной намагниченности, является расфазировка прецессии ядерных спинов около стенок ячейки. Она происходит из-за существования неоднородного электрического поля у границы раздела между объёмом ячейки и её стеклянной границей. Для уменьшения влияния эффекта разрушения ядерной намагниченности на стенках ячейки модель должна учитывать механизм диффузии атомов со смещённой фазой внутрь объёма ячейки.

Целью данной работы является разработка программы, позволяющей проводить моделирования диффузии ядерной намагниченности ксенона в условиях её разрушения. Полученная модель должна помочь определить оптимальные размеры ячейки для получения максимально узкой линии магнитного резонанса при постоянном объёме ячейки, температуре и концентрации ксенона внутри, а также пронаблюдать качественное различие процесса выстраивания ядерных спинов вдоль лазерной накачки в зависимости от механизма релаксации.

Математическая модель диффузии ядерной намагниченности ксенона описывается системой трёх нестационарных дифференциальных уравнений в частных производных второго порядка в трёхмерном пространстве следующего вида:

$$\frac{\partial M_\alpha}{\partial t} = D\left(\frac{\partial^2 M_\alpha}{\partial x^2} + \frac{\partial^2 M_\alpha}{\partial y^2} + \frac{\partial^2 M_\alpha}{\partial z^2}\right) + \gamma \sum_{\beta,\lambda} \varepsilon_{\alpha\beta\lambda} M_\beta B_\lambda - G\left(M_\alpha - e_\alpha\right), \qquad \alpha = x, y, z \tag{1}$$

$$x \in \left[0, L_x\right], \quad y \in \left[0, L_y\right], \quad z \in \left[0, L_z\right], \tag{2}$$

где M_x, M_y, M_z – это три компоненты вектора ядерной намагниченности, которые выступают в роли неизвестных функций четырёх переменных с областью определения (2), D – коэффициент диффузии, B_x, B_y, B_z – компоненты вектора индукции магнитного поля, e_x, e_y, e_z – компоненты электрического поля, γ – гиромагнитное отношение каждого изотопа ксенона, G – скорость намагничивания под действием столкновений с поляризованными щелочными атомами в ячейке, $\varepsilon_{\alpha\beta\lambda}$ – антисимметричный тензор третьего ранга. Векторы **B** и **e** являются известными функциями трёх пространственных координат и времени.

В модели учтены несколько механизмов релаксации на стенках ячейки: они описаны с помощью 18-ти феноменологических констант A_{ia} в граничных условиях системы дифференциальных уравнений (1):

$$\left.\frac{\partial M_a}{\partial x}\right|_{x=0} = A_{1a} M_\alpha\big|_{x=0}, \qquad -\left.\frac{\partial M_a}{\partial x}\right|_{x=L_x} = A_{2a} M_\alpha\big|_{x=L_x}, \qquad a = \{x, y, z\} \tag{3}$$

$$\left.\frac{\partial M_a}{\partial y}\right|_{y=0} = A_{3a} M_\alpha\big|_{y=0}, \qquad -\left.\frac{\partial M_a}{\partial y}\right|_{y=L_y} = A_{4a} M_\alpha\big|_{y=L_y}, \qquad a = \{x, y, z\} \tag{4}$$

$$\left.\frac{\partial M_a}{\partial z}\right|_{z=0} = A_{5a} M_\alpha\big|_{z=0}, \qquad -\left.\frac{\partial M_a}{\partial z}\right|_{z=L_z} = A_{6a} M_\alpha\big|_{z=L_z}, \qquad a = \{x, y, z\} \tag{5}$$

До действия накачки ксенон предполагается не намагниченным, поэтому начальные условия системы (1) имеют более простой вид:

$$M_x(x, y, z, 0) = 0, \quad M_y(x, y, z, 0) = 0, \quad M_z(x, y, z, 0) = 0 \tag{6}$$

Ожидаемым результатом моделирования является зависимость картины установившейся динамики ядерной намагниченности от частоты внешнего магнитного поля, полученная для разной геометрии ячейки. Критерием успешного решения поставленной задачи будет определение формы ячейки, соответствующей наиболее узкой резонансной линии зависимости амплитуды компоненты M_x для изотопов ксенона от частоты Ω внешнего переменного магнитного поля B_y.

С помощью интегро-интерполяционного метода [1] исходная система была сведена к системе обыкновенных дифференциальных уравнений размерности $N = (N_x + 1)(N_y + 1)(N_z + 1)$, где N_x, N_y, N_z – число разбиений отрезков L_x, L_y, L_z соответственно.

Полученная система обыкновенных дифференциальных уравнений имеет ряд особенностей, усложняющих процесс её решения:

– рассмотренная система является жесткой [2], то есть системой, численное решение которой явными методами является неудовлетворительным из-за ограничения на шаг интегрирования и резкого увеличения объёма вычислений. Поэтому для решения системы применяются неявные методы.

– системы имеет большую размерность N и поэтому следует ориентироваться на методы пригодные для решения больших систем алгебраических уравнений – итерационные и прямые методы для разреженных систем.

Эти особенности накладывают ограничения на программные средства, которые могут использоваться для решения данной системы. В работе приводятся результаты применения неявных методов с использованием метода сопряжённых градиентов и пакета ODEPACK [3] для моделирования диффузии ядерной намагниченности.

Разработка программы осуществлена на языке программирования Fortran, входящего в состав Intel Parallel Studio, которая интегрируется с Microsoft Visual Studio.

ЛИТЕРАТУРА

1. Самарский А.А. Теория разностных схем. – М.: Наука, 1983. 616 с.
2. Ракитский Ю.В., Устинов С.М., Черноруцкий И.Г. Численные методы решения жёстких систем. М.: Наука, 1979. 199 с.
3. Hindmarsh A.C. ODEPACK, a systematized collection of ode solvers. In Scientific Computing, R.S. Stepleman et al. (eds.), Nord-Holland, Amsterdam, pp.55-64. [V.5].

Секция «Технологии обработки и анализа больших массивов данных на основе работ ЕРАМ»

УДК 004.89

А. А. Алиев (4 курс бакалавриата)
С. А. Молодяков, д.т.н., профессор

СИСТЕМА ДЕТЕКТИРОВАНИЯ И РАСПОЗНАВАНИЯ НОМЕРНЫХ ЗНАКОВ ТРАНСПОРТНЫХ СРЕДСТВ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

В настоящее время вычислительная техника активно внедряется в большинство задач, связанных с автомобильным транспортом. Создаются и совершенствуются системы автопилотов, безопасности движения, учета транспортных средств и др. Одной из актуальных задач является задача распознавания номерных знаков автомобилей. Известны системы распознавания номерных знаков [1, 2]. В них используются ряд методов, позволяющих проводить распознавание знаков. Последовательность применения методов сводится к следующим этапам: улучшение качества потока поступающих с видеокамеры кадров, сегментирование области расположения номерного знака, распознавание с использованием свёрточной нейронной сети. Известен метод использования инфракрасной камеры с подсветкой. Он используется в ГИБДД. Благодаря известным алгоритмам компьютерного зрения, качественным видео и инфракрасным камерам задача распознавания номерных знаков решается достаточно успешно. Однако данные системы хорошо работают в нормальных условиях, но ряд факторов существенно ухудшают качество распознавания. К таким факторам следует отнести отражение солнечного света и яркого света автомобильных фар, движение автомобиля, загрязнение номера, низкое разрешение видеокамеры, погодные условия. Поэтому задача разработки эффективных программных алгоритмов распознавания номерных знаков автомобилей продолжает быть актуальной.

Целью работы является разработка системы детектирования и распознавания номерных знаков транспортных средств с применением нейронных сетей. Сейчас известно большое количество современных моделей нейронных сетей (U-Net, Mask-RCNN, Faster-RCNN...). В сочетании с другими методами машинного обучения можно ожидать существенно лучших результатов [3].

Разрабатываемый алгоритм и соответственно программная система включает следующие этапы: получение потока видеокадров, сегментирование, определение ключевых точек, выравнивание изображения, распознавание символов. Схема разрабатываемого алгоритма с кадрами реального изображения представлена на рис.1.

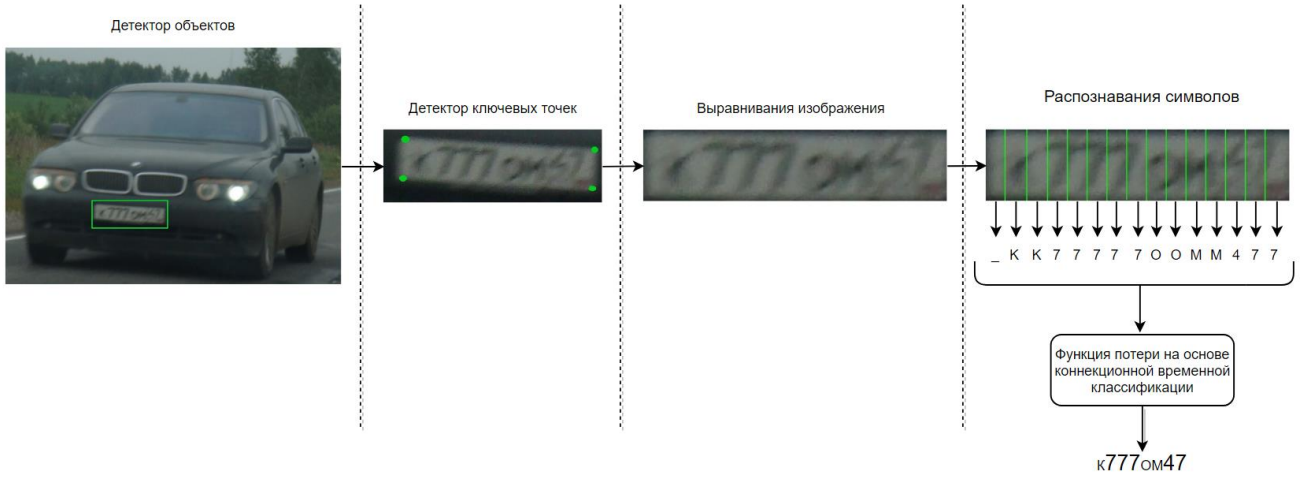


Рисунок 1 – Схема работы системы детектирования и распознавания номерных знаков.

В качестве детектора объектов используется свёрточная нейронная сеть Yolov3[4]. Затем изображения выравнивается с помощью детектора ключевых точек, который находит углы номерного знака. После этого изображение подается на свёрточную рекуррентную нейронную сеть[5]. Изображения сначала обрабатывается свёрточной нейронной сетью, а затем разделяется на N=15 частей, в зависимости от параметров обучения, затем эти данные со свёрточной нейронной сети подаются на рекуррентную нейронную сеть. В результате работы, мы получаем массив символов с размером N=15, но так как российские номерных знаки транспортных средств имеют 8 или 9 символов. Для того чтобы исправить это используется функция потери на основе коннекционной временной классификации [6], который на вход принимает массив символов N=15, а на выходе выдает массив размером N=8,9.

Разработанная система позволяет распознавать номерные знаки с точностью 99.7% и с полнотой 97% на тестовой выборке из 20 тысячи изображений. Предложенный алгоритм дает возможность получить большую точность, чем известные аппаратные средства для распознавания номерных знаков. Дальнейшее направление работы связано с исследованием работы алгоритма на сложных изображениях, использованием видеокадров с разрешением до 4К и с повышением скорости работы.

ЛИТЕРАТУРА

1. R. Laroca, E. Severo, L.A. Zanlorensi, L.S. Oliveira, G.R. Goncalves A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. International Joint Conference on Neural Networks (IJCNN), 2018. P. 1-10.
2. S.M. Silva, C.R. Jung License Plate Detection and Recognition in Unconstrained Scenarios. Computer Vision – ECCV 2018. Ch.36. DOI: 10.1007/978-3-030-01258-8_36
3. Дейлид И.А., Молодяков С.А Применение методов машинного обучения для определения препятствий с помощью стереозрения // Железнодорожный транспорт. 2019. № 12. С. 27-29.
4. J. Redmon, A. Farhadi You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. DOI: 10.1109/CVPR.2016.91
5. B. Shi, X. Bi, C. Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence PP(99) 2015. DOI: 10.1109/TPAMI.2016.2646371
6. A. Graves, S. Fernandez, F. Gomez, J. Schmidhuber Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. 2006. Proceedings of the 23rd international conference on Machine learning June 2006. P. 369–376 DOI: 10.1145/1143844.1143891

УДК 004.4

С. Р. Астафьев (1 курс магистратуры)
В. В. Шаляпин, к.т.н., доцент

СОЗДАНИЕ ОБЛАКОВ ТЕГОВ ДЛЯ ФИЛЬМОВ

Для того, чтобы построить хороший рекомендационный алгоритм, нужно иметь данные, на которых этот алгоритм будет работать. В данном докладе рассматривается способ создания системы, которая позволяет автоматически обрабатывать характеристики фильмов и создавать на их основе облака тегов. Тег – ключевой слово для категоризации в фолксономиях. Фолксономия – народная классификация, практика совместной категоризации информации посредством произвольно выбираемых меток, называемых тегами. Облако тегов - множество тегов, характеризующих определенный элемент. В начале необходимо выбрать характеристики фильмов, которые будут обрабатываться. У фильмов есть множество характеристик, такие как: список режиссеров, список продюсеров, список актеров и актрис, синопсис или краткий сюжет. В данном случае ключевой характеристикой фильма является трейлер, так как он содержит в себе много информации.

Есть множество способов получить теги, даже не обрабатывая характеристики фильмов. Например, взять готовый список тегов с известных Интернет-ресурсов по фильмам, таких как: Kinopoisk и Imdb. Но такой способ не надежный, ведь у малоизвестных фильмов будет малое количество тегов или они вовсе будут отсутствовать. Другие способы — это обработка трейлеров. Два главных направления — это покадровый анализ и анализ звуковой дорожки. Оба этих анализа можно проводить как автоматически, так и вручную. Но при ручном анализе уйдет слишком много человеко-часов, поэтому прибегнем к использованию алгоритмов анализа.

У алгоритмов анализа речи существуют огромный недостаток, из-за которого в данном проекте использовались алгоритмы покадрового анализа. Суть недостатка в том, что не для всех фильмов выходит трейлер на английском языке, поэтому пришлось бы настраивать алгоритмы по многу раз с учитыванием каждого языка. Исходя из этого, был выбран покадровый анализ. Большинство алгоритмов покадрового анализа основано на свёрточной нейронной сети.

Вся архитектура системы состоит из четырех компонентов: хранилище с характеристиками фильмов, канал передачи информации, приложение с нейронной сетью и хранилище для выходных данных. Ниже представлен рисунок архитектуры системы.

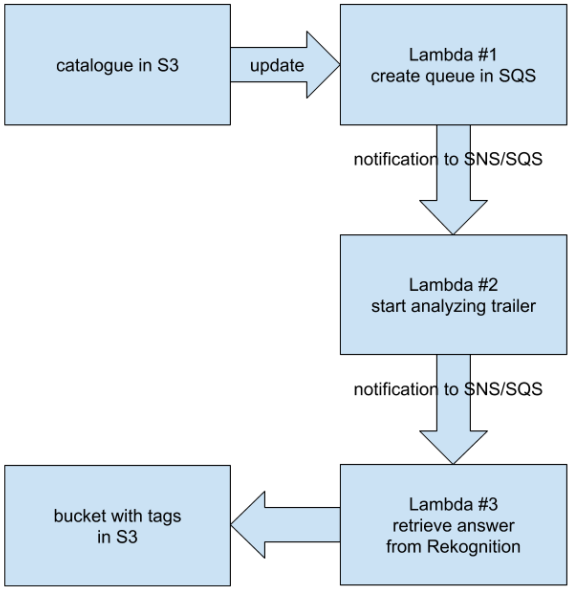


Рисунок 1 – Архитектура системы.

Все необходимые для данной системы функции присутствуют на стеке сервисов Amazon. Также у сервисов Amazon подробная документация и они просты в настройке. Поэтому в данном случае использовались сервисы от Amazon. За хранилище данных, как входных, так и выходных отвечает сервис Amazon Simple Storage Service, далее S3. За канал передачи информации отвечает два связанных сервиса Amazon Simple Queue Service и Amazon Simple Notification Service, далее SNS и SQS соответственно. За обработку трейлеров отвечает сервис Amazon Rekognition. За облачное выполнение кода функций Amazon Lambda. Для начала работы этой системы, надо создать модули создания очереди, анализа трейлера и сбора ответов.

Первый модуль отвечает за создание очереди из фильмов, трейлеры которых должны быть загружены в S3 и проанализированы сервисом Rekognition. Данная функция начинает свое выполнение, когда база данных, хранящаяся в S3 обновляется. Обновление базы данных происходит 3 раза в день, и в каждое обновление новые фильмы как могут быть добавлены, так и не могут. Для предотвращения повторного анализа уже проанализированных трейлеров, нам нужно сопоставлять список уже загруженных трейлеров и фильмов из базы данных по их индивидуальному id, заранее сгенерированному. Так как стандартная функция пакета botocore s3.list_objects_v2() не поддерживает возвращение полного списка объектов в bucket S3, то нужно создать собственную функцию. Наша функция использует параметр «ContinutationToken», что позволяет извлечь все объекты, находящиеся в нужном bucket, а не только первые 100 объектов. После того, как у нас есть полный список уже загруженных трейлеров, нам нужно обработать всю базу данных фильмов. База данных «дампится» в json файл и загружается в S3. Данный файл называется каталог, и мы будем использовать его как реплику базы данных, для уменьшения вычислительной нагрузки и сохранения конфиденциальности нашей базы данных с фильмами. Так как каталог создается в .json.gz формате, то для упрощенния навигации по нему, воспользуемся стандартными возможностями языка Python. Воспользуемся пакетами BytesIO и GzipFile, чтобы разархивировать каталог, используя функции BytesIO().read() и GzipFile().read().decode(). В итоге получаем обычный json. Чтобы обращаться к нему будем использовать стандартный пакет json, и функцию из него json.loads(). Проходя поэлементно по каталогу, мы найдем все фильмы, трейлеры которых еще не скачены и не проанализированы. После того как, такой фильм находится, используя функцию sns.publish() из пакета botocore, отправляем данные по этому фильму в нашу очередь. Из нее следующий модуль возьмет информацию по фильму и использует по назначению.

Второй модуль отвечает за загрузку трейлера фильма в S3 и отправление запроса на анализ этого трейлера. Событие из-за которого выполняется данная функция - появление в очереди сообщения с данными по трейлеру. А именно, это url трейлера с локального сервера и его id. Так как пакет botocore не имеет функции загрузки файла в S3 из url, нам придется написать ее самим. Используя стандартный пакет requests и функцию requests.get().raw.read(), мы можем извлечь трейлер фильма из url и загрузить во временную память. После чего загрузить в S3 используя функцию пакета botocore s3.put_object() в формате mp4. Название трейлера в S3 образуется из id фильма для упрощения нахождения уже загруженных трейлеров фильмов. После того как загрузка произошла, используя функцию пакета botocore recognition.start_label_detection(), отправляем сервису Rekognition запрос на анализ данного трейлера. Данная функция возвращает стандартный словарь Python, из которого мы можем извлечь JobID – индивидуальный номер запроса на анализ. Мы сохраняем связку из JobID и id фильма в отдельный файл, так как они понадобится нам для получения ответа от Rekognition и распознавания какой трейлер фильма был проанализирован. На этом работа второго модуля заканчивается.

Третий модуль ответственен за сбор ответа от Rekognition и преобразования его в нужный нам вид. Событие-уведомление, по которому выполняется данная функция, — это автоматическое сообщение о конце анализа от Rekognition, отправленное в ранее созданный нами SNS topic. Так как ответ от Rekognition не содержит имя проанализированного файла,

мы не знаем какому трейлеру данный ответ принадлежит. Чтобы решить данную проблему, мы заранее сохраняли id фильма и JobID анализа, и сопоставив JobID в сохраненом файле с JobID из ответа, мы сможем узнать id фильма. Ответ от Rekognition получаем используя функцию из пакета botocore recognition.get_label_detection(). Эта функция возвращает стандартный словарь Python, в котором находятся список объектов, которые смог распознать алгоритм Rekognition. Пройдя поэлементно по этому списку, мы найдем все не повторяющиеся теги, подсчитаем их общее количество и индивидуальное количество. Индивидуально количество тегов нам нужно для вычисления весов – нормализованного коэффициента количества данного тега относительно общего количества тегов. Также мы опять воспользуемся каталогом, так как нам нужны такие данные как id фильма, его оригинальное название и название в неформальной транслитерации. В конце сохраним все в файл в формате json. После этого наше приложение заканчивает работу.

По итогам работы системы, получается большой список особых структур, состоящий из: названия фильма, неформальной транслитерации названия фильма, уникального идентификатора, списка тегов и их весов.

УДК 004.457

К. М. Бойков, О. С. Командина (4 курс бакалавриата)
Н. В. Воинов, к.т.н., доцент

РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКУТРЫ ДЛЯ СЕРВИСА ПО УПРАВЛЕНИЮ ИНТЕРАКТИВНЫМИ ПОДПИСКАМИ

В настоящее время многие современные компании в сферах услуг и развлечений предоставляют использование контента на основе платных подписок. К таким компаниям относятся: онлайн-кинотеатры, дистанционные платные курсы, социальные сети и т.д.

Соответственно, было бы не лишним иметь под рукой сервис, в котором будут видны все продукты, на которые он подписан. Где есть возможность отписаться от существующих подписок и следить за общими тратами на них.

Данная задача, наблюдение за собственными подписками, является актуальной в наши дни.

Целью работы является создание сервиса, в котором можно будет следить за всеми подписками сразу и без переходов на сторонние сайты компаний, предоставляющих эти платные подписки. Цель выполнима и реалистична за счет выполнения следующих задач:

1) Провести подробный анализ существующих аналогов и, по возможности, узнать подробнее о стеке технологий, которые используются.
2) Реализовать веб-интерфейс на базе фреймворка React.js[1], который легко и быстро дает возможность создать одностраничный пользовательский интерфейс.
3) Создать подходящую базу данных, используя СУБД MongoDB[2], которая будет в себя включать все необходимые данные о пользователях и подписках.
4) Разработать серверную составляющую сервиса на основе платформы Node.js[3] и фреймворка Express.js[4]. А также использовать архитектурный стиль взаимодействия компонентов REST[5].
5) Предоставить полный отчет о сравнении предполагаемого сервиса и его аналогов.

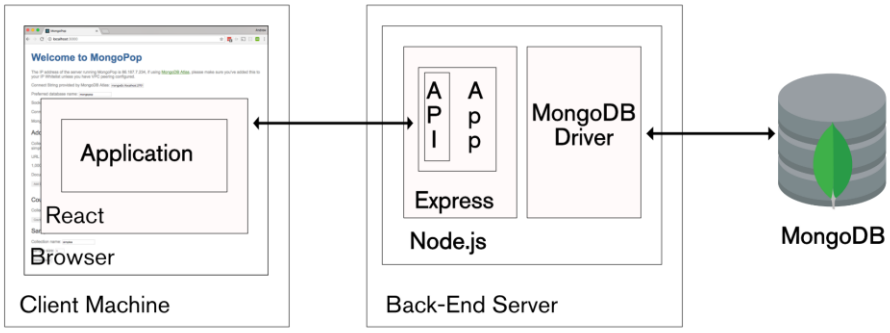


Рисунок 1 – Диаграмма взаимодействия инструментов для разработки сервиса.

ЛИТЕРАТУРА

1. Документация React.js [Электронный ресурс]. Режим доступа: https://ru.reactjs.org/docs/
2. Документация MongoDB [Электронный ресурс]. Режим доступа: https://docs.mongodb.com/
3. Документация Node.js [Электронный ресурс]. Режим доступа: https://nodejs.org/ru/docs/
4. Документация Express.js [Электронный ресурс]. Режим доступа: https://expressjs.com/ru/guide/
5. REST [Электронный ресурс]. Режим доступа: https://www.restapitutorial.com/

УДК 004.04

А. Д. Донцов (4 курс бакалавриата)
И. А. Селин, ст. преподаватель

АНАЛИЗ МЕДИЦИНСКИХ ДАННЫХ С ПОМОЩЬЮ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

В настоящее время в сфере медицины прикладное применение искусственного интеллекта и методов машинного обучения в частности становится всё более распространённым. В частности, данный подход позволяет эффективно искать закономерности в найденных данных, что в перспективе позволит предлагать новые схемы лечения и предсказывать диагнозы.

В наборе данных, подвергнутых рассмотрению, помимо результатов анализов, выраженных в количественной мере по соответствующим кодам-идентификаторам, были представлены:

- Пол.
- Возраст.
- История диагнозов пациента в процессе лечения
- Сопутствующие диагнозы (осложнения и сопутствующие заболевания).

Перед непосредственно анализом собранной информации, была проведена её предварительная обработка, которая заключалась в:

- Построении матрицы корреляции (Рис. 1)
- Выборе и удалении незначащих столбцов.
- Выборе и замене численных значений возраста пациента на матрицу возрастных групп.
- Выборе столбцов роста и веса с заменой на индекс массы тела.
- Выборе и кодировании столбца, содержащего пол пациента.

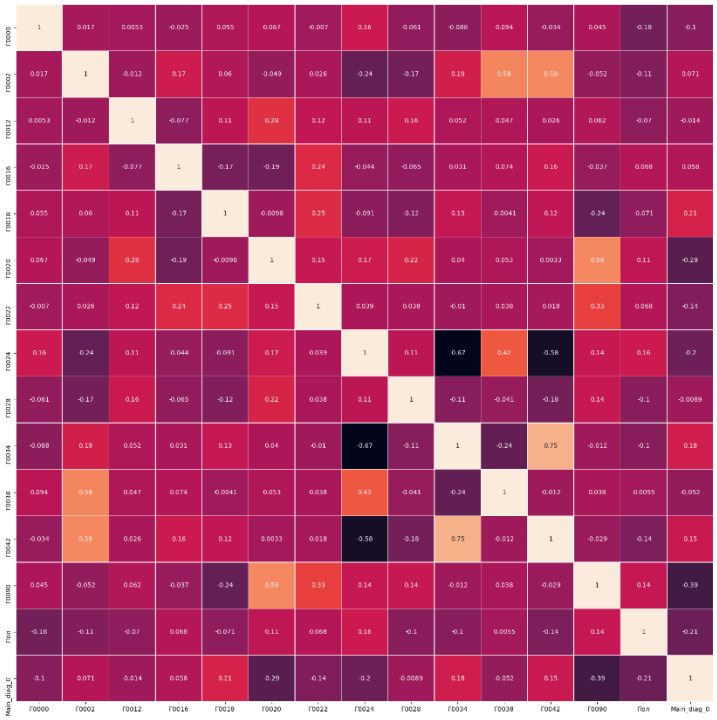


Рисунок 1 – корреляционная матрица для неразбитой выборки.

На основе этой информации был проведён кластерный анализ данных с использованием метода k-средних [1]. Для оценки качества разбиения использовались метрики силуэт [2], Калинского-Харабаза [3] и Дэвиса-Болдина [4]. Так же были построены графики в пространстве признаков с учётом проставленных меток кластеров (Рис. 2).

Рисунок 2 – График в пространстве признаков

Затем для выявления наиболее значимых признаков были использованы случайные леса [5] на разбитых данных и получены весовые коэффициенты. Для каждого признака в рамках каждого кластера была проведена проверка распределения на нормальность, построены

графики распределений для каждого признака, а также вычислены средние значения, граничные значения признаков.

Анализ был проведён с использованием языка Python 3.7 [6], с использованием следующих библиотек:

- Для загрузки и обработки данных использовалась библиотека pandas [7]
- Для методов кластеризации и классификации использовалась библиотека sklearn [8]
- Для визуализации использовались matplotlib [9] и seaborn [10]

Результаты метрик кластеризации говорят о значительном перекрытии кластеров (Табл1)

Таблица 1 – Значения метрик кластеризации

	Силуэт	Калинского-Харабаза	Дэвиса-Болдина
	0.3665	9101	0.881

График в пространстве признаков позволяет увидеть хорошо разбитые данные. Кроме того, результат анализа распределения в большинстве признаков позволил подтвердить гипотезу о том, что результаты распределены нормально, что является предметом дальнейшего исследования.

ЛИТЕРАТУРА

1. Forgy E., “Cluster analysis of multivariate data: efficiency versus interpretability of classifications.” Biometrics, vol. 21, no. 3, pp. 761–777, 1965.
2. Peter J. Rousseeuw . “Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis”, 1987.
3. T. Calinski and J. Harabasz. “A dendrite method for cluster analysis”, 1974.
4. Davies, David L.; Bouldin, Donald W., “A Cluster Separation Measure”, 1979.
5. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001.
6. "G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995."
7. W. McKinney, “Data Structures for Statistical Computing in Python”, Proceedings of the 9th Python in Science Conference, pp. 51-56, 2010.
8. Pedregosa et al, “Scikit-learn: Machine Learning in Python”, JMLR 12, pp. 2825-2830, 2011.
9. J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
10. Seaborn: statistical data visualization // Seaborn. URL: https://seaborn.pydata.org/ (дата обращения: 18.03.2020).

УДК 004.457

М. Г. Ковалев (4 курс бакалавриата)
А. Н. Терехов, к. ф.м. н., профессор

ОТСЛЕЖИВАНИЕ ПУТИ СЕТЕВЫХ ПАКЕТОВ В ЯДРЕ LINUX С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ EBPF

Целью работы является исследование возможностей технологии eBPF с целью её использования для создания инструментов, нацеленных на решение проблем, возникающих при разработке и эксплуатации больших и комплексных сетевых систем.

Количество программ, позволяющих решать те или иные проблемы, связанные с сетью, велико. Однако в больших и комплексных сетевых системах конфигурация настолько сложна, что для поиска источника проблемы обычными способами может уйти очень много времени и сил. В качестве крайней меры приходится прибегать к анализу работы сетевого стека ядра.

Список функций, которые пакеты исследуемого сетевого трафика посетили в ядре при обработке, сильно ускорит эту часть поиска источника проблемы.

В рамках данного исследования разрабатывается прикладная утилита, позволяющая получить информацию о функциях, которые обрабатывали структуру сетевого пакета в ядре Linux. Это позволит наглядно проследить путь сетевого пакета в ядре и увидеть, где конкретно может быть проблема. Пользователь поймёт, в каком месте пакет сбрасывается или где путь пакета отклоняется от предполагаемого. Таким образом станет понятно, как необходимо изменить конфигурацию, чтобы решить проблему.

eBPF (extended Berkeley Packet Filter) — технология ядра Linux, позволяющая безопасно и с высокой производительностью динамически исполнять инструкции практически в любом месте ядра. Это расширение технологии BPF, использовавшейся для фильтрации сетевого трафика.

Для работы с eBPF не нужно изменять исходный код ядра и пересобирать его. Сначала на ограниченном языке C создаётся программа. Она компилируются в объектный файл с инструкциями eBPF с помощью инструментария Clang и LLVM. При загрузке в ядро инструкции объектного файла подвергаются статическому анализу. Успешно загруженная в ядро программа исполняется в определенных пользователем местах. Исполнением занимается виртуальная машина, преобразующая инструкции eBPF в инструкции процессора, на котором запущена система.

Статический анализ программ и определенный набор инструкций обеспечивают безопасность при использовании eBPF, а исполнение на виртуальной машине — производительность, максимально приближенную к нативной. И даже при всех своих ограничениях eBPF является самым гибким и удобным способом модификации и расширения функциональности ядра Linux.

УДК 004.852

С. С. Лаппо, Н. В. Полетова (1 курс магистратуры)
И. В. Никифоров, к.т.н., доцент

АНАЛИЗ КОРРЕЛЯЦИИ МЕЖДУ ПОЛИТИЧЕСКИМИ ПРЕДПОЧТЕНИЯМИ ПОЛЬЗОВАТЕЛЯ И ЕГО НАСТРОЙКАМИ ПРИВАТНОСТИ

В данной работе исследуется возможность построения связи между политическими предпочтениями пользователей, установленных на основании их подписок на те или иные группы, а также политических лидеров, и закрытостью их страницы, а в случае если страница удалена, то удалена ли она пользователем или заблокирована администрацией.

Работа состоит из нескольких частей, которые поэтапно демонстрируют проектирование всей системы, реализацию технологии загрузки данных, повышения её производительности и, наконец, анализ данных.

В начале выполнения была спроектирована архитектура, необходимая для реализации задуманной функциональности. Для простоты использования, а также потенциального упрощения масштабируемости все компоненты находятся в отдельных Docker-контейнерах, взаимодействие между которыми осуществляется через сетевые протоколы. Для хранения собранных данных используется база данных Cassandra. Скраппер, написанный на Kotlin сохраняет данные в базу данных и, кроме того, с использованием Rabbit MQ отправляет загруженные данные в модуль, занимающийся анализом данных и реализованный на Python. Таким образом, реализованная архитектура позволяет при необходимости заменить один из компонентов, не внося изменения в другой.

В качестве набора данных для исследования были взяты пользователи 11 различных групп социальной сети VK, а также 9 различных политиков, которым была дана

предварительная классификация по типу политических предпочтений на основе общедоступной информации.

В результате анализа выбранных групп и политиков было выяснено, что общее количество пользователей превышает 7 миллионов, что, учитывая специфику доступа к данным социальной сети VK, которая не создана, для того чтобы получать из неё данные в автоматическом режиме, создаёт некоторые сложности.

Для преодоления данных трудностей был реализован собственный скраппер, который позволяет выгружать данные со скоростью до 75 запросов в секунду по одному API ключу.

Кроме выгрузки пользователей для повышения точности исследования были выгружены все подписки пользователей на различные группы. В результате было установлено что рассматриваемые пользователи суммарно подписаны более чем на 6 млн различных источников, а общее число подписок достигло более чем 550 млн.

В дальнейшем все источники классифицировались на 3 типа: провластный, оппозиционный и неизвестный.

Для этого к 22000 постов из 22 групп был применён метод машинного обучения «случайный лес», при помощи которого была достигнута значительная точность в классификации групп по типу на тестовой выборке.

После разбора групп по типам произведена финальная классификация пользователей и анализ зависимости их предпочтений и скрытости страницы, показавший, что пользователи, которых мы отнесли к оппозиционно-настроенным в 1.57 раз чаще скрывают свою страницу.

УДК 681.3.06

Д. А. Мухин (1курс магистратуры)
В. А. Семёнова-Тян-Шанская, к.т.н., доцент

РАЗРАБОТКА И ОБЕСПЕЧЕНИЕ БЕЗОПАСНОГО ФУНКЦИОНИРОВАНИЯ WEB-ПРИЛОЖЕНИЯ

Хорошо известно, что абсолютно неуязвимых ресурсов не существует, как в силу технического обеспечения, так и в силу человеческого фактора. Однако существует ряд наиболее распространённых атак, от которых всё же существует защита и которые нужно предотвращать на любом ресурсе.

В данной работе рассматриваются некоторые наиболее часто встречающиеся виды атак на Web-приложения.

Во-первых – это SQL-инъекции, которые случаются, когда текст запроса формируется склеиванием неэкранированных строк.

Во-вторых – это CSRF (межсайтинговая подмена запросов). Идея заключается в том, что многие приложения предполагают, что запросы, приходящие от браузера на сервер, отправляются самим пользователем. Иногда это может быть не так.

В-третьих – это XSS или кросс-сайтинговый скриптинг, который становится возможен, когда неэкранированный выходной HTML-код попадает в браузер. Например, если пользователь должен ввести своё имя, но вместо этого он вводит <script>alert('Hello!');</script>, тогда все страницы, которые его выводят без экранирования, будут выполнять JavaScript alert('Hello!'). В результате будет выводиться окно сообщения в браузере. В зависимости от сайта, вместо невинных скриптов с выводом всплывающего hello, злоумышленниками могут быть отправлены скрипты, похищающие личные данные пользователей сайта, либо выполняющие операции от их имени.

Для исследования способов предотвращения указанных уязвимостей в данной работе было создано Web-приложение, публикующее складскую базу данных некоторого предприятия. При разработке Web-приложения использовались следующие технологии: СУБД MYSQL, Web сервер APACHE, язык сценариев PHP. Для облегчения разработки Web-

приложения был выбран Yii2-фреймворк, который представляет собой среду с открытым исходным кодом для PHP7, позволяющий создавать оптимизированные Web- приложения. В контексте нашей задачи по обеспечению безопасности Web-приложений, фреймворк Yii предоставляет большие возможности. Так основная часть запросов в фреймворке Yii к базе данных происходит через Active Record, который правильно использует статические запросы. Также фреймворк Yii автоматически проверяет наличие допустимого маркера CSRF для всех небезопасных методов HTTP-запроса (PUT, POST, DELETE), генерирует и выводит маркер при использовании метода ActiveForm::begin () для создания тега формы открытия. Для же защиты от XSS-атак фреймворк Yii предоставляет другой вспомогательный класс, называемый HtmlPurifier.

В работе произведена оценка эффективности рассмотренных методов предотвращения атак. Для тестирования работы приложения был использован фреймворк для тестов Codeceprion. В работе представлены тесты по оценки безопасности web- приложения. Например, для проверки XSS уязвимости нужно отобрать страницы, на которых есть формы ввода. Для проверки возьмем страницу с товарами на складе. Введем в строку поиска «Вредоносный скрипт» следующего вида (рисунок 1) '><">"><script>alert("cookie: "+document.cookie)</script>

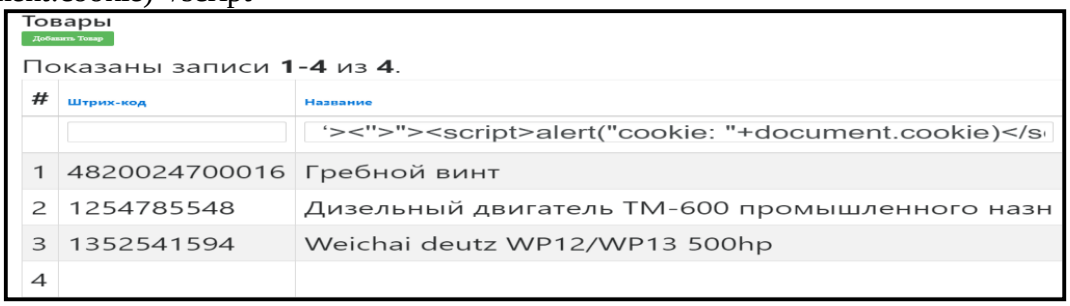


Рисунок 1 – Предварительный ввод «вредоносного скрипта».

Рисунок 2 – Результат после выполнения запроса со скриптом.

Как видно из рисунка 2 обрезались теги <script>, а, следовательно, скрипт не выполнился. В противном случае бы отобразилось окно с cookie.

Для проверки, как сделана защита от CSRF атаки понадобилось создать html страницу с запросом к нашему сайту. Код страницы приведен на рисунке 3.

```
<html>
<meta http-equiv='cache-control' content='no-cache'>
<meta http-equiv='pragma' content='no-cache'>
<body onload=document.test.submit()>
<form name=test action='http://yii2frontend.com/product/create' method='POST' >
<input name="_csrf-frontend" value="u1cbJ5e2qP6WDDU8XUbiTwD4EMHihLj9St6tIWre6tDCMzZT-
sHLma9hc1kcM4cDdqpCmIfgil4ai-BWMqiDlQ==" type=hidden >
<input name="Product[barcode]" value="pppp" type=hidden >
<input name="Product[Name]" value="pppp" type=hidden >
<input name="Product[Product_group]" value="pppp" type=hidden >
<input name="Product[Units]" value="pppp" type=hidden >
<input name="Product[Manufacturer_Name]" value="pppp" type=hidden >
<input name="Product[Country_of_Origin]" value="pppp" type=hidden >
<input type='submit' value='CSRF payload will execute now'>
</html>
```

Рисунок 3 – Код страницы для проверки CSRF атаки.

После открытия представленной страницы при нажатии кнопки submit данные попытаются добавиться на сервер. В результате появится сообщение об ошибке, так как пришел запрос с неправильным token.

УДК 004.75

П. А. Панков (2 курс магистратуры)
И. В. Никифоров, к.т.н., доцент
Д. Ф. Дробинцев, ст. преподаватель

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ ОДНОПЛАТНЫХ МИКРОКОМПЬЮТЕРОВ В КАЧЕСТВЕ ОБУЧАЮЩЕЙ ПЛАТФОРМЫ ДЛЯ РАБОТЫ С РАСПРЕДЕЛЁННОЙ ОБРАБОТКОЙ ДАННЫХ.

На сегодняшний день в сети Интернет генерируются огромные количества данных, источником которых являются организации, веб сервисы, фирмы, устройства IoT [1] и т.д. Поэтому разработка инструментов для хранения и обработки информации из больших объёмов данных является важной задачей в исследовании информационных технологий. Для решения данной задачи создаются новые центры обработки данных (далее ЦОД). Традиционные ЦОД прекрасно выполняют свои задачи в коммерческом применении, однако имеют ряд недостатков. Они состоят из мощного и дорогостоящего оборудования, требуют большого количества электроэнергии для работы, требуют мощного охлаждения аппаратного обеспечения и занимают большую площадь. Кроме того, использование мощного оборудования подразумевает постоянное его использование, поскольку эксплуатация таких систем без выполнения полезных задач может обходиться дорого.

Когда возникает необходимость распределённой обработки данных в научных, исследовательских и академических целях, мощные ЦОД могут быть избыточны, так как время выполнения той или иной задачи может быть не важно. В таких сферах применения чаще важна возможность проверки работы алгоритмов распределённой обработки данных, обучение работе с кластерной инфраструктурой, настройкой ЦОД и т.д.

Целью работы является исследование подходов к распределённой обработке данных, возможности и преимуществах использования одноплатных микрокомпьютеров в качестве аппаратного обеспечения для системы распределённой обработки данных в научных, исследовательских и академических целях. Одноплатный компьютер [2] – это полный аналог обычных компьютеров, но выполненный на одной печатной плате. Эти устройства зачастую имеют малую мощность, но они недорогие и энергоэффективные, что может позволить создать недорогой ЦОД для описанных выше целей.

При выборе одноплатного микрокомпьютера были выделены следующие критерии: процессор (К1), объём оперативной памяти (К2), интерфейсы доступа к сети Интернет (К3), поддерживаемые операционные системы (К4), цена (К5).

Таблица 1 – Сравнение одноплатных микрокомпьютеров.

	Banana Pi M1 +	**Orange Pi PC2**	**Raspberry Pi 3 Model B+**
К1	A20 ARM Cortex -A7 Dual-Core 1GHz	Allwinner Cortex-A53 64-bit Quad-Core 1.2 GHz	Broadcom Cortex-A53 64-bit Quad-Core 1.4GHz
К2	1 Гб DDR3	1 Гб DDR3	1 Гб LPDDR2
К3	Ethernet, Wi – Fi	Ethernet	Ethernet, Wi – Fi
К4	Android, Armbian, Debian, Other Linux	Android, Ubuntu, Debian	Raspbian, Ubuntu Mate, Win10 IoT, Other Linux
К5	3000-4000 р.	2900-3400 р.	3000-4000 р.

Основываясь на проведённом анализе, представленном в Таблице 1, был выбран микрокомпьютер Raspberry Pi. Кроме того, данный одноплатный микрокомпьютер является наиболее распространённым и доступным с большим сообществом программистов, что является большим преимуществом Raspberry Pi над другими микрокомпьютерами.

Кроме представленных одноплатных микрокомпьютеров, существуют также Odroid. Эти одноплатные микрокомпьютеры имеют большую производительность и лучшие характеристики, однако и более высокую стоимость, большее энергопотребление,

тепловыделение и требуют лучшего охлаждения, они менее доступны и сообщество разработчиков сильно меньше.

В качестве операционной системы была выбрана ОС Raspbian [3]. Эта операционная система является оптимизированной под низко производительные ARM процессоры версией ОС Debian Linux и считается операционной системой «по умолчанию» для Raspberry.

Для распределённой обработки данных на узлах использовался набор утилит, библиотек и фреймворков Apache Hadoop. Это проект фонда Apache Software Foundation, он используется многими компаниями для производственных и научных целей, как инструмент надёжных и масштабируемых распределённых вычислений, и распределённого хранения данных.

Для тестов созданной системы был выбран стандартный алгоритм подсчёта слов в тексте. Этот алгоритм позволит проверить эффективность работы системы с алгоритмами, построенными на базе MapReduce [4]. Тесты проводились в несколько этапов. Алгоритм запускался с учётом того, что файл с текстом не разбивался на блоки. Далее, конфигурация файловой системы была настроена так, что файл разбивался на 3 блока, работа с которыми распределялась по разным узлам. Тестировались также разные конфигурации режимов работы узлов. 3 одноплатных компьютера выполняли на каждом этапе разную роль. На каждом этапе был 1 «мастер узел» - он занимается распределением задач и мониторингом узлов. Кроме того, менялось количество «рабочих узлов», которые отвечают за обработку данных. На 1, 3 и 5 этапах «мастер узел» являлся одновременно и «рабочим узлом».

В Таблице 2 показано время обработки файла размером 38.5 мегабайт с различной конфигурацией системы.

Таблица 2 –. Собранные метрики работы тестового алгоритма подсчёта слов в тексте

	1 рабочий узел	1 мастер узел 1 рабочий узел	2 рабочих узла	1 мастер узел 2 рабочих узла	3 рабочих узла
1 блок	3м 37с	2м 38с	2м 44с	2м 46с	2м 49с
3 блока	-----	2м 13с	2м 15с	2м 14с	2м 13с

Как видно из таблицы, при работе на одном узле, время работы с одним блоком наибольшее. Лучшее время показала конфигурация из 1 «мастер узел» и 1 «рабочий узел». В других условиях, так как файл не разбивается на блоки, то мы лишь теряем время на работу «мастер узел» по распределению задач. В случае, с разбиением файла на 3 блока, время выполнения примерно одинаковое, так как в каждом случае все 3 блока сразу распределялись на все узлы, однако видно уменьшение времени работы алгоритма по сравнению с 1 блоком.

Проект находится на стадии разработки. В дальнейших исследованиях планируется выполнить следующие задачи:

– увеличить количество узлов;
– использовать внешние накопители (flash / HDD / SSD);
– увеличить количество тестовых задач для сбора метрик производительности;
– использовать средства для тестирования распределённых систем и приложений [5].

ЛИТЕРАТУРА

1. P. Pankov, I. Nikiforov, Y. Zhang, “Hardware and software system for collection, storage and visualization meteorological data from a weather stand”, Сборник статей “TELECCON 2019” (в печати).
2. W. P. Birmingham and D. P. Siewiorek, “MICON: a knowledge based single board computer designer,” in Proceedings of the 21st Conference on Design Automation, Albuquerque, NM, 1984, pp. 565571.
3. Roberto Morabito, “Virtualization on Internet of Things Edge Devices with Container Technologies: a Performance Evaluation”, IEEE Access, 2017
4. J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” in Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI), San Francisco, CA, 2004, pp. 137-150
5. К.С. Кобышев, И.В. Никифоров, О.В. Прокофьев, “Инструмент для автоматизации тестирования компонент распределённого приложения, использующего esb-шину”, Сборник статей “Современные технологии в теории и практике программирования”, 2019

УДК 004.925

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

РАЗРАБОТКА И АВТОМАТИЗАЦИЯ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДАННЫМИ ДЛЯ СКВОЗНОЙ АНАЛИТИКИ

Цель работы. Необходимо разработать систему для своевременной постоянной загрузки данных в единое хранилище для дальнейшего анализа в рамках решения задачи сквозной аналитики. Сквозная аналитка – это метод анализа эффективности маркетинговых инвестиций (ROMI) на основе данных, прослеживающий полный путь клиента, начиная от просмотра рекламного объявления, посещения сайта и заканчивая продажей, а также повторными продажами. Сюда входит предсказание чистого дохода, свзяанного со всеми будущими отношениями с клиентом, определение финансовой ценности каждого из клиентов, что в свою очередь стимулирует компании сосредоточится с квартальных отчетов на долгосрочные отношения со своими клиентами.

Для решения этой задачи требуется объединять с ежедневной периодичностью информацию из рекламных каналов (Google Реклама, Яндекс.Директ, Facebook, Google Analytics, Criteo, Admitad), каналов коммуникации (коллтрекинг, электронная почта, формы обратной связи) и CRM-систем, где находится информация о продажах.

Имеются следующие требования к системе:

1. Автоматическое управление рекламными кампаниями. Применение формулы расчета окупаемости рекламы (ROAS) с разными уровнями агрегации – для каждого канала, кампании, ключевой фразы.
2. Возможность создавать автоматизированные отчеты для расчета маркетинговых KPI вроде количества кликов, трат, дохода, конверсий, продаж, достижения целей, ROI и т.д.
3. Данные должны быть полными за весь период маркетингового анализа и актуальными за вчерашний день.
4. Факты целостности и актуальности данных должны быть проверяемы.

Анализ аналогов. В рамках работы проведен подробный анализ 30 аналогов, с ним можно подробно ознакомиться непосредственно в докладе.

Разработка. Была разработана система на основе Workflow Management Platform (WMP) Luigi, которая распределенно получает по REST API из различных источников данные и копирует их базу данных PostgreSQL. Некоторые таблицы этой БД являются частью хранилища данных (в которую входят еще другие БД – MySQL и Clickhouse). Существенная часть преобразований данных выполняется на уровне PostgreSQL в специальных таблицах, откуда данные в обработанном виде попадают в защищенные от попадания дубликатов таблицы, которые входят в хранилище.

Формирование отчетов таким образом происходит за счет создания в хранилище специального среза данных, который называется витриной данных. Технически их роль исполняют такие структуры данных PostgreSQL как обычные представления, так и материализованные представления.

Данные из этих представлений попадают в инструмент для построения дэшбордов Microsoft Power BI с помощью технологии DirectQuery, что позволяет напрямую подключаться к базе данных и не копировать эти данные в какую-либо промежуточную БД. В PowerBI формируются все необходимые для клиента меры и измерения и формируются в отчет. Также к этим данным можно получить доступ из других инструментов таких как Tableau.

В рамках доклада описывается техническая реализация алгоритма управления данными с помощью WMP Luigi, а также стратегии преобразования и проверки целостности полученных данных.

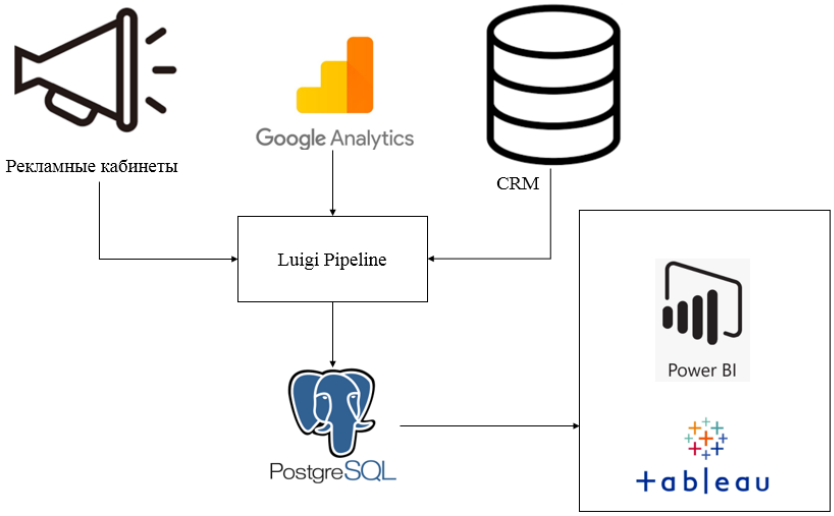


Рисунок 1 – Концептуальная схема системы.

Анализ результатов. Эффективность работы системы оценивается по общему времени преобразования сырых данных в удобный для анализа вид, общему времени перевыгрузки всех данных в случае их повреждения в хранилище и процент покрытия мониторингом целостности и актуальности данных.

ЛИТЕРАТУРА

1. Liubov Zhovtonizhko – Through data to goals: Business Intelligence Guide [Электронный ресурс] URL: https://www.owox.com/blog/use-cases/end-to-end-analytics-implementation (дата обращения: 17.03.2020)
2. NIFTIT – Connecting Power BI To PostgreSQL [Электронный ресурс] URL: http://niftit.com/connecting-power-bi-to-postgresql/ (дата обращения 17.03.2020)

УДК 004.415.04

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

РАЗРАБОТКА СИСТЕМЫ ПОСТРОЕНИЯ И ГЕНЕРАЦИИ ПАЙПЛАЙНОВ

Целью работы является исследование CI/CD [1] инструментов и разработка системы предназначенной для ускорения и упрощения создания сред непрерывной разработки и развертывания, выработки и обеспечения переиспользования корпоративных шаблонов приложения, внедрения лучших практик, визуализации настроенных рабочих процессов сборки приложения (pipeline) и их запуска, и мониторинга с помощью различных конкретных инструментов (Jenkins, TeamCity, Travis CI и др.), представления аналитики по CI/CD процессам и глубокого анализа проблем сборки.

На данный момент, использование цифровых инструментов в принятии торговых решений играет решающую роль. Наряду с усложнением высокотехнологичной платформы важным фактором стала непрерывность работы критических ИТ-систем. Эта тенденция была затронута и при разработке программного обеспечения. На этапе разработки возможность постоянно тестировать и доставлять код позволяет увеличить скорость разработки и качество готового решения. Принцип работы CI/CD аналогичен конвейеру: методика делает интеграцию, включающую различные типы автоматического тестирования, с последующей доставкой и развертыванием кода для завершенного готового продукта для конечного пользователя. CI/CD-платформа, на базе которой применяется концепция, поддерживает регулярную автоматическую сборку проекта для быстрого выявления дефектов и

интеграционных задач. [2] При стандартном, когда разработчики работают независимо над разными частями системы, этап интеграции является окончательным, и при обнаружении ошибок он может непредсказуемо задержать завершение работы. Переход к непрерывной интеграции позволяет снизить трудоемкость работы и делает ее более предсказуемой за счет раннего и непрерывного обнаружения и устранения ошибок и противоречий.

Главные плюсы CI/CD:

1. Скорость вывода новой функциональности от запроса клиента до запуска в эксплуатацию [3].
2. CI/CD позволяет запускать обновления за считанные дни или недели по сравнению с целым календарным годом при классическом waterfall-подходе. Новые сервисы – новые конкурентные бизнес-преимущества. Появляется возможность не просто воспроизводить функциональность решений конкурентов, но и значительно опережать их в разработке и внедрении новых инструментов.
3. Возможность выбора оптимального варианта за счет оперативного тестирования и большего числа итераций. Отказавшись от работы над бесперспективными решениями, вы сэкономите ресурсы компании.
4. Качество итогового результата выше: автотестирование охватывает все аспекты продукта, что труднореализуемо при стандартном релизном подходе. Все ошибки и тонкие места выявляются и удаляются ещё на ранних этапах разработки.

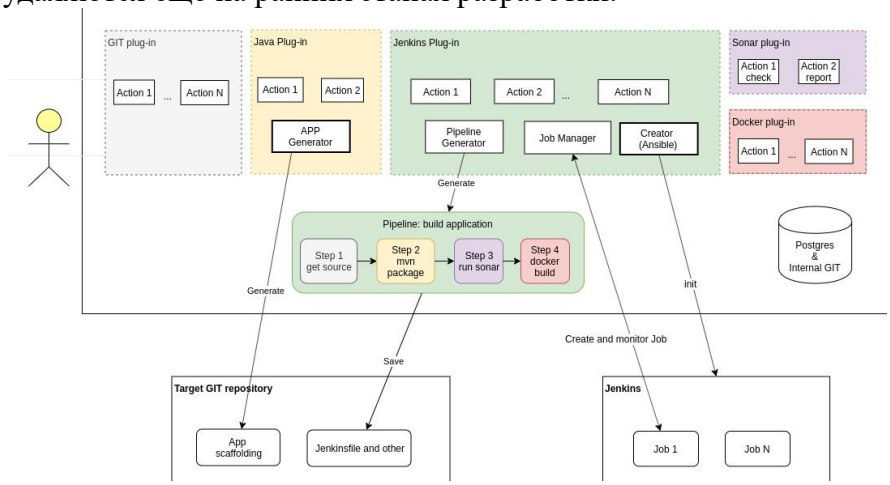


Рисунок 1 – Архитектура CI/CD системы

Разрабатываемая система обеспечивает возможность хранения, сравнения, поиска, создания и редактирования шаблонов приложения и пайплайнов с помощью графического интерфейса, запуска и мониторинга имеющихся пайплайнов, а также на основе шаблонов конфигурировать проект или пайплайн. Система генерирует в выбранном GIT репозитории исходные коды приложения и позволяет просматривать логи создания. Система предоставляет возможность перегенерации приложения и пайплайнов в указанную ветку или репозиторий.

Основные составляющие системы: job manager - коннектор к CI/CD инструменту, который позволяет получить информацию о настроенных проектах и Pipeline в конкретном инструменте, он может создавать Pipeline в CI/CD инструменте. Он также может запустить указанный Pipeline или действие, может отслеживать состояние и посылать сообщения о состоянии процессов сборки, generator - обеспечивает генерацию исходного кода. Содержит список параметров, которые влияют на кодогенерацию, action - конечное действие, которое может сконфигурировать исходный код pipeline и/или выполнить определенное действие, component — это часть приложения, которая обслуживается несколькими пайплайнами, пайплайн - описывает правила запуска действий одного за другим или параллельно, состоит из шагов. JobManager может запустить его, предварительно инстанциировав его на конкретном CI/CD инструменте, step - описывает шаг Pipeline, может содержать зависимости с другими Step, config provider - посредник, который обеспечивает чтение/сохранение в

каталог настроек данного проекта / шаблона проекта. В своем роде это просто файловая система, config provider manager - обеспечивает координацию использования локальных GIT репозиториев в рамках одной сессии редактирования, поддерживает список ConfigProvider, git storage - обеспечивает аутентификацию и GIT операции над конкретным GIT репозиторием.

В современном мире разработку крупных проектов нельзя представить без CI/CD. Несмотря на множество CI/CD-систем, которые помогают автоматизировать процесс сборки, сама практика внедрения методологии — задача нетривиальная. Наша же система уже из “коробки” предоставляет преднастройки для различных и часто использующихся наборов инструментов в различных проектах, позволяет гибко настраивать процесс сборки и следить за процессом его выполнения.

ЛИТЕРАТУРА

1. Мартынов Т.Е. Непрерывная интеграция и доставка. Сравнение инструментов CI/CD, Екатеринбург: Материалы XII международной научно-практической конференции, 2019. -8.
2. Статья с сайта Хабр «От репозитория до CI/CD-инфраструктуры в продакшене за неделю.» [Электронный ресурс] –URL: https://habr.com/ru/company/itsumma/blog/333650/ (дата обращения: 15.03.2020).
3. Статься с сайта IT Global “Методология разработки CI/CD”[Электронный ресурс] – URL: https://itglobal.com/ru-ru/company/blog/development-method-ci-cd/ (дата обращения: 14.03.2020)

УДК 004.4

Е. В. Ригин (4 курс бакалавриата)
С. А. Фёдоров, ст. преподаватель

АНАЛИЗ КАЧЕСТВА ОБРАБОТКИ НОВОСТНЫХ СТАТЕЙ НА РУССКОМ ЯЗЫКЕ NLP-СИСТЕМАМИ

Целью работы является методика качественного выделения персон, топонимов и организаций из новостных статей на русском языке с помощью NLP-систем. Эта методика должна обеспечивать уровень выделения именованных сущностей максимально близкий к тому, которого может добиться человек.

Ввиду наличия большого количества различных NLP-систем введем следующие критерии для их выбора:

1. Наличие достаточной документации и активного сообщества.
2. Поддержка русского языка.
3. Наличие встроенных в систему грамматик для распознавания персон, топонимов и организаций.

Под эти требования попадают Pullenti SDK[1], Eureka Engine[2] и Natasha[3].

Затем были отобраны пять новостных статей из открытых онлайн СМИ, критериями выбора которых были:

1. Русский язык текста.
2. Объем не менее 1000 знаков.
3. Наличие в статье упоминания в двух и более вариантах по крайней мере одной персоны, одного топонима и одной организации. Под вариантами упоминания подразумевается различные формы указания на одну и ту же сущность, например: Министерство иностранных дел, МИД.

Под эти требования попадают следующие новостные статьи:

[1] http://www.pullenti.ru/
[2] http://eurekaengine.ru
[3] https://github.com/natasha

1. Российско-турецкие переговоры продолжаются пятый час (https://tass.ru/politika/7909447)
2. Синоптики предсказали «малиновое» лето в 2020 году
(https://www.rbc.ru/rbcfreenews/5e60f0c19a7947456f58d8b7)
3. К рекламщикам-миллиардерам из Петербурга по делу о неуплаченных налогах пришел СОБР с кувалдой (https://www.fontanka.ru/2020/03/05/69017464/)
4. В России новый случай коронавируса: похоже, он в Петербурге
(https://www.fontanka.ru/2020/03/05/69016939/)
5. Командующий КСИР считает, что коронавирус может быть биологическим оружием США (https://tass.ru/mezhdunarodnaya-panorama/7906981)

В работе предложен следующий показатель качества распознавания именованных сущностей:

$$Q = \frac{\sum_{i=N} \frac{N_{NLP}}{N_H}}{N},$$

где N_{NLP} – количество распознанных именованных сущностей NLP-системой, N_H – количество выделенных именованных сущностей человеком, N – количество рассматриваемых статей.

Выделим вручную из каждой новости персоны, топонимы и организации в целях использования полученных значений как эталонных.

Таблица 1 – Результаты работы выбранных NLP-систем

	Персоны	Топонимы	Организации
Pullenti SDK	0.6	0.86	**0.87**
Eureka Engine	**1**	**0.9**	0.72
Natasha	0.9	0.1	0

По результатам таблицы видно, что платная система Eureka Engine является наиболее эффективной для выделения персон и топонимов, а бесплатная для некоммерческого использования Pullenti SDK – для выделения организаций. Поэтому для закрытых коммерческих систем предлагается следующая методика выделения именованных сущностей:

1. Выделяем персоны и топонимы с помощью Eureka Engine
2. Выделяем организации с помощью Pullenti SDK

Однако, для систем, требующих использование открытого исходного кода, наиболее предпочтительной будет следующая методика:

1. Выделяем персоны с помощью NLP-системы Natasha
2. Выделяем топонимы и организации с помощью Pullenti SDK

Вывод. В работе были сформулированы критерии выбора NLP-систем, критерии подбора новостных статей для анализа выбранных систем распознавания именованных сущностей, проведен их анализ, предложены две методики для наиболее эффективного выделения именованных сущностей из новостных статей на русском языке.

ЛИТЕРАТУРА

1. Kapetanios, Epaminondas; Tatar, Doina; Sacarea, Christian (2013-11-14). Natural Language Processing: Semantic Aspects. CRC Press. p. 298. ISBN 9781466584969.
2. Schank, Roger C.; Abelson, Robert P. (1977). Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures. Hillsdale: Erlbaum. ISBN 0-470-99033-3.

УДК 004.054

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

МУТАЦИИ КОДА КАК МЕТОД ПРОВЕРКИ ЭФФЕКТИВНОСТИ НАБОРА АВТОМАТИЧЕСКИХ ТЕСТОВ

Ни для кого не секрет, что при разработке любого программного продукта одним из важных этапов является тестирование. Юнит тесты позволяют убедиться, что код работает правильно, в соответствии с поставленными задачами. Для того чтобы понять насколько полно юнит тесты проверяют приложение используется метрика процента покрытия кода (Code Coverage).

Но точен ли этот показатель, и можно ли ему доверять? Для ответа на этот вопрос используется мутационное тестирование. Мутационное тестирование позволяет выяснить, как будут реагировать юнит тесты на всевозможные изменения кода, и сохранится ли достоверность их результатов в случае, если код изменится. Если после изменения кода тесты продолжат выполняться успешно, значит написанный набор тестов, вероятно, неэффективен.

Целью данной работы является разработка системы мутационных тестов для проверки эффективности существующего набора автоматических юнит тестов и выявление слабо покрытых тестами участков кода для их дальнейшего улучшения.

В процессе мутационного тестирования происходят незначительные изменения фрагментов кода. Каждое изменение фрагмента кода называется мутацией (или мутантом). Самым простым примером мутации является замена знаков и операций на противоположные (+ на -, > на < и т.д.). На рисунке ниже приведена общая схема использования мутаций.

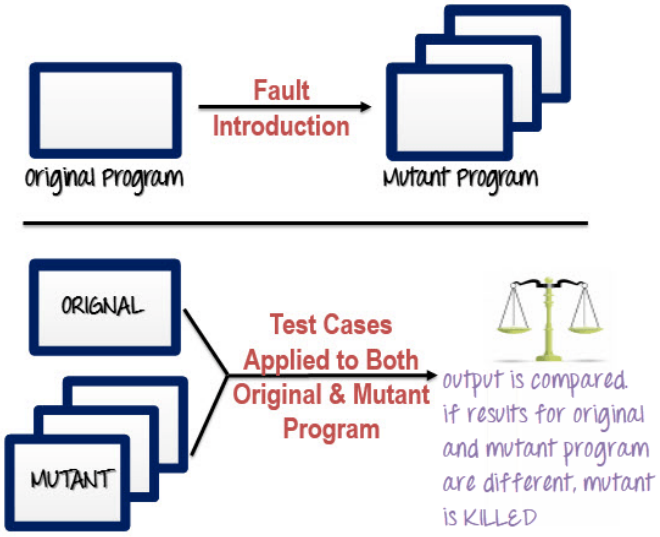


Рисунок 1 – Общая схема проведения мутационного тестирования

Если описать алгоритм мутационного тестирования более подробно, то он будет состоять из следующих шагов:

Шаг 1: ошибки вводятся в исходный код программы путем создания множества версий-мутантов. Каждый мутант должен содержать одну ошибку.

Шаг 2: тесты применяются к исходной программе, а также к программе с мутациями. Тестовый случай должен быть адекватным, настроенным на обнаружение ошибок в программе

Шаг 3: Результаты тестов оригинальной программы и программы с мутациями сравниваются. Производится оценка результатов.

Если исходная программа и версии-мутанты генерируют разные результаты тестов, то мутант считается убитым юнит тестом. Следовательно, тестовый случай достаточно хорош, чтобы обнаружить изменение между исходной программой и мутантом. В ином случае мутант

остается живым. Тогда необходимо создавать более эффективные тест-кейсы, которые убивают всех мутантов.

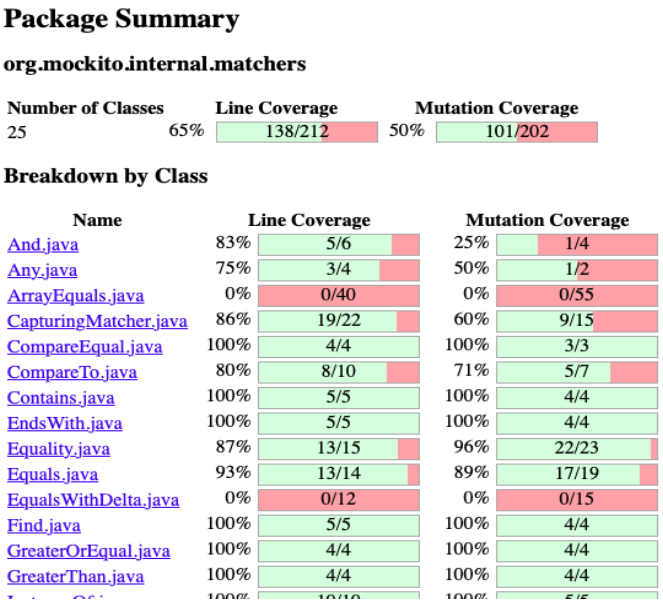
После оценки результатов происходит вычисление показателя эффективности набора автоматических тестов - Mutation Score Indicator (MSI):

$$MSI = \frac{\text{количество убитых мутантов}}{\text{общее количество мутаций}} * 100$$

Именно данный показатель позволяет сделать выводы об эффективности набора юнит тестов. Если MSI слишком низкий, вероятно, стоит поправить тесты, сделать их более универсальными.

Для достижения поставленной цели в качестве исходного приложения выбран популярный JUnit фреймворк Mockito ввиду высокого покрытия автоматическими юнит тестами и библиотека PITest для осуществления мутационного тестирования.

Для фреймворка Mockito были запущен стандартный набор юнит тестов и определен базовый процент покрытия кода. Затем, исходное приложение было подвергнуто мутациям. Часть мутаций с найденными уязвимостями приведена на следующем рисунке.



Package Summary

org.mockito.internal.matchers

Number of Classes	Line Coverage		Mutation Coverage	
25	65%	138/212	50%	101/202

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
And.java	83%	5/6	25%	1/4
Any.java	75%	3/4	50%	1/2
ArrayEquals.java	0%	0/40	0%	0/55
CapturingMatcher.java	86%	19/22	60%	9/15
CompareEqual.java	100%	4/4	100%	3/3
CompareTo.java	80%	8/10	71%	5/7
Contains.java	100%	5/5	100%	4/4
EndsWith.java	100%	5/5	100%	4/4
Equality.java	87%	13/15	96%	22/23
Equals.java	93%	13/14	89%	17/19
EqualsWithDelta.java	0%	0/12	0%	0/15
Find.java	100%	5/5	100%	4/4
GreaterOrEqual.java	100%	4/4	100%	4/4
GreaterThan.java	100%	4/4	100%	4/4

Рисунок 2 – выявленные уязвимости юнит тестов и процент выживших мутантов.

Даже из вышеописанных данных можно сделать вывод, что исходный процент покрытия кода – величина не стопроцентно объективная, поскольку не всегда при изменении кода, даже самом незначительном, юнит тесты могут качественно выявлять эти изменения и реагировать на них.

На данный момент ведется улучшение процессов определения мутантов юнит тестами (как простых, так и более сложных) с целью повышения качества исходного кода.

В данной работе был произведен обзор методологии мутационного тестирования. Конечно, этот метод имеет и ряд недостатков, таких как трудоемкость и затраты на автоматизацию процесса. Но в целом, использование мутационных тестов позволяет сделать юнит тесты более универсальными и качественными, что немаловажно для разработки качественного программного продукта.

ЛИТЕРАТУРА

1. «Mutation Testing» - Filip van Laenen
2. «A Practical System for Mutation Testing: Help for the Common Programmer» - A. Jefferson Offutt.

УДК 004.416

А. Е. Соколова, М. С. Соломонов (4 курс бакалавриата)
А. Д. Ковалев (2 курс аспирантуры)
И. В. Никифоров, к. т. н., доцент

ИСПОЛЬЗОВАНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ НА ЭТАПЕ ОБСЛУЖИВАНИЯ ПРОГРАММНОГО ПРОДУКТА

Сложность этапа обслуживания в области предоставления услуг и продуктов конечному потребителю возрастает с каждым годом. Это вызвано увеличением количества сложных современных технологий, используемых в услугах и продуктах, и, как следствие, увеличением количества ошибок в реализации и документообороте. Каждая ошибка в продукте или услуге, а также вопрос к продукту является для заказчика поводом для обращения в компанию поставщика за помощью.

Время и усилия, которые необходимо потратить компании поставщика на поиск решений по запросам заказчика и обучение новичков в отделе поддержки пониманию и работе с предоставленным продуктом или услугой огромны на этапе жизненного цикла и обслуживания. Это занимает около 40% времени разработки и требует больших человеческих ресурсов.

Основная проблема, которую решает данный проект, - это сокращение времени и усилий, затрачиваемых поставщиком на фазу поддержания продукта. Это достигается за счет разработки подхода к помощи в разрешении запросов заказчика, основанного на интеллектуальных методах и алгоритмах машинного обучения, которые реализуются в программном решении. Эти методы не только позволяют решить проблему эффективнее, но и помогают студентам и новичкам быстро освоиться с проектом благодаря семантическому поиску в системе отслеживания ошибок и проектной документации, что также снижает затраты на фазу сопровождения продукта или услуги.

Повышение качества поддержки помогает построить долгосрочные отношения между поставщиком услуг и продуктов и заказчиком, а также облегчает учебный процесс для новых инженеров поддержки. Таким образом, предлагаемое решение по автоматизации и помощи в отладке проблем и ошибок продуктов можно рассматривать как фактическую задачу на местах.

Задачи проекта можно разделить на три части в зависимости от его результатов:

- поиск похожих запросов клиентов в базе данных уже решенных вопросов. Подход классифицирует запросы клиентов с помощью алгоритма машинного обучения Doc2Vec на основе проблем с исторической базой данных и предоставляет список семантически похожих случаев. Они помогают понять, была ли проблема уже решена, дают краткий обзор принятых подходов, соответствующих инженеров для этой задачи и шагов по ее решению;
- семантический поиск по документации. Результатом поиска является таблица с отображением названия документа и номера страницы, содержащая семантически связанную информацию. Это помогает инженеру быстро предоставить нужные ссылки заказчику и подтвердить, является ли запрос ошибкой или нет;
- применение набора статических правил для предоставления формальных рекомендаций.

Основываясь на статических правилах, система предлагает инженеру перевести запрос в надлежащее состояние или запросить дополнительную информацию. Например, когда клиент жалуется на ошибку в продукте, но забывает прикрепить журнал со стековой трассировкой.

Все подходы реализованы в программном инструменте, позволяющем автоматизировать решение запросов клиентов, что, в свою очередь, позволяет сократить время, затрачиваемое инженерами службы технической поддержки и разработчиками продукта на поиск решения по запросам клиента.

На данный момент проект содержит: коннекторы для систем отслеживания ошибок (Jira и Bugzilla); процессоры для поиска семантически схожих разрешенных проблем, поиск связанных с ними файлов документации и страниц, движок правил, позволяющий применять правила "when-then" для каждой проблемы; генераторы отчетов в форматах HTML и TXT.

В качестве главного алгоритма мы используем Doc2Vec[1] для семантического поиска. Для обработки документации используется Apache POI[2], который работает с текстовыми файлами, из которых создается датасет документации.

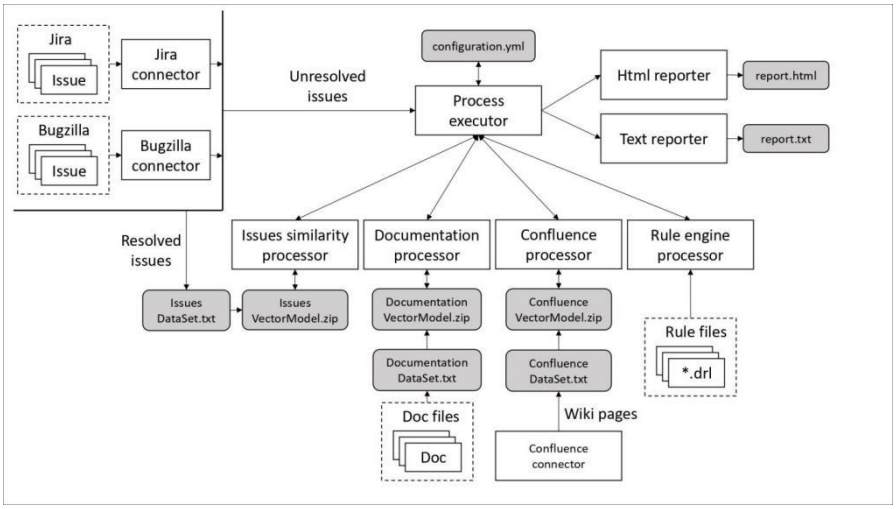


Рисунок 1 – Архитектурная схема

Рассмотрим рисунок 1, на котором изображены модули и объекты системы, показана ее структура, какие файлы необходимы для работы. Пользователь выставляет необходимые конфигурации в файле configuration.yml, решенные заявки из источников Jira[3] и Bugzilla[4] собираются в текстовый файл, проходят процесс векторизации, по итогу которого векторная модель записывается в архив. Файлы документации также собираются в текстовые файлы из указанной папки и страниц Confluence с последующей векторизацией.

Нерешенные заявки из Jira и Bugzilla собираются и передаются процессорам. Далее процессоры привлекают свои векторные модели, подсчитывают схожесть и проводят анализ рекомендаций. Наконец, обработчик процессов передает результаты в репортеры форматов html и text, которые в свою очередь выдают финальный отчет в необходимом формате.

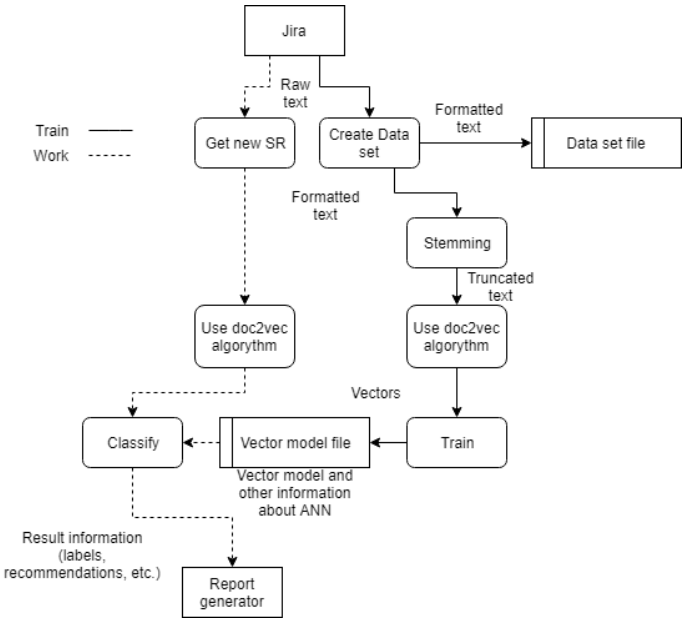


Рисунок 2 – Схема потока данных в системе

На рисунке 2 можно увидеть основной поток данных в системе, какие файлы создаются, какие процессы происходят с данными. От Jira/Bugzilla мы получаем текст, который преобразуется в файл датасета. Этот текст проходит стемминг и тренировку в doc2vec, на выходе процесса получаем файл с векторной моделью. Нерешенные заявки собираются как текст, вместе с векторными моделями передаются классификатору. Его выход затем передан в модуль, отвечающий за создание отчета, выдающий конечный результат.

Наша система имеет несколько пользовательских интерфейсов. Один из них — это конфигурационный файл. Пользователь может задать настройки в файле для регулировки производительности системы. Например, задать какую информацию выбирать и откуда, настроить алгоритм doc2vec, задать пути для входных и выходных файлов.

Другим интерфейсом является командная строка. Пользователь запускает приложение через консоль с помощью команд или скриптов, к примеру:

- createDataSet - создает датасет из решенных заявок;
- createVectorModel - создает векторную модель на основе датасета решенных заявок;
- run - собирает нерешенные заявки, векторизует их, сравнивает с решенными, а также документацией, в конце выдает финальный отчет.

Последним используемым интерфейсом является браузер с финальным отчетом, в котором указаны похожие заявки, инженеры, к которым можно обратиться, ссылки на полезную документацию и статьи confluence, а также рекомендации по дальнейшим действиям.

На данный момент в результате работы были выполнены следующие задачи:

- осуществлен поиск похожих запросов клиентов в базе данных уже решенных вопросов;
- реализован семантический поиск по документации;
- применен набор статических правил.

Дальнейшие исследования будут направлены на оптимизацию работы системы.

ЛИТЕРАТУРА

1. Doc2Vec, or Paragraph Vectors, in Deeplearning4j. [Электронный ресурс]. Режим доступа: https://deeplearning4j.org/docs/latest/deeplearning4j-nlp-doc2vec
2. Apache POI - the Java API for Microsoft Documents. [Электронный ресурс]. Режим доступа: https://poi.apache.org/
3. JIRA. [Электронный ресурс]. Режим доступа: https://www.atlassian.com/software/jira
4. Bugzilla. [Электронный ресурс]. Режим доступа: https://www.bugzilla.org/

УДК 004.045

В. К. Сычев, Ю. В. Ленькова, Е. А. Цветкова (2 курс магистратуры)
И. В. Никифоров, к.т.н., доцент

АНАЛИЗ ДАННЫХ ТРЕНДОВ YOUTUBE

В наше время, когда на рынке много конкурирующей продукции, как никогда, актуальна реклама. Наиболее предпочтительной, для рекламодателей, является реклама, размещенная на видеохостингах. Но как показывает практика, таргетированная реклама, предлагаемая самим видеохостингом, не всегда оказывает ожидаемый эффект. В некоторых случаях такая реклама может вызвать негативное воздействие на популярность рекламируемого продукта. Решением этой проблемы на данный момент выступает заказная реклама у создателей контента.

Чтобы получить максимальную прибыль, рекламодателям приходится учитывать много факторов, основными из которых являются тематика канала, средний возраст зрителей и их активность на данном канале.

На примере такого видеохостинга как YouTube был рассмотрен один из способов сбора информации о каждом видео, находящегося в специальной зоне (вкладка “тренды”), где собраны 50 самых популярных роликов. Был проведен анализ публичных данных о каждом ролике и канале, которые попали в “тренды”.

Таким образом, сформулировав цель исследования: ппроанализировать данные о популярных видео на платформе YouTube и оформить полученные результаты анализа в доступной и понятной форме, были поставлены следующие задачи:

- исследовать методы сбора данных на платформе YouTube
- сформировать базу данных из публичных данных о каждом видео
- провести обработку и анализ нсобранных данных
- отобразить данные в наглядной и удобной форме

Чтобы организовать работу по сбору и анализу данных был использован следующий стек технологий:

- Java
- JSoup - java библиотека для извлечения данных, хранящихся в документах HTML
- YouTube Data API v3 - API для доступа к публичной информации видеохостинга YouTube
- Postgresql - реляционная база данных
- Tableau - система интерактивной аналитики, позволяющая проводить глубокий и разносторонний анализ больших массивов данных.

В процессе сбора данных с YouTube было выявлено несколько проблем:

- ограничение по числу запросов в день, для бесплатного пользования API
- ограничение данных, которые можно получить веб-скраппингом

Чтобы избежать эти проблемы были совмещены использование API и скраппинг.

Из полученных данных создана БД со структурой, изображенной на рисунке 1.

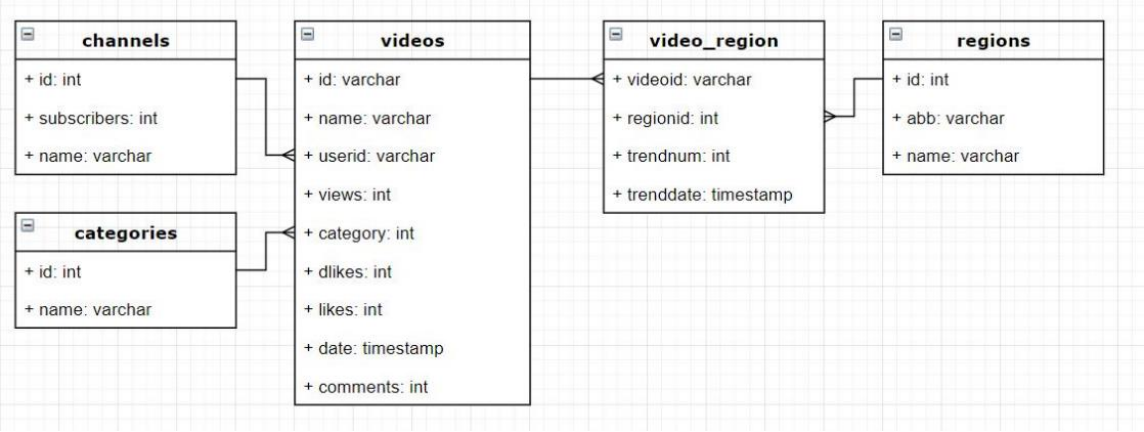


Рисунок 1 – Структура базы данных

После сбора данных в течение недели получено несколько результатов анализа:

- самые популярные категории видео по активности и просмотрам
- зависимость позиции видео в “трендах” от активности пользователей
- средний показатель активности пользователей на просмотр
- наиболее часто выходящие в “тренды” каналы в России

Выводы: В процессе работы были исследованы методы сбора данных с платформы YouTube, собраны и обработаны данные о видео в разделе “тренды” за неделю. Также было выявлено, что видео с большим количеством просмотров может не вызывать ожидаемой активности пользователей. Это обусловлено таким явлением как “накручивание просмотров”.

ЛИТЕРАТУРА

1. YouTube Data API v3 documentation [Электронный ресурс]. Режим доступа: https://developers.google.com/youtube/v3/getting-started.
2. Ганс-Юрген Шениг. PostgreSQL 11. Мастерство разработки -Москва: ДМК Пресс

УДК 004.62

Е. В. Тимощук (1 курс магистратуры)
И. В. Никифоров, к.т.н., доцент
Д. Ф. Дробинцев, ст. преподаватель

АНАЛИЗ ДАННЫХ ПРОФИЛЕЙ СОТРУДНИКОВ В СОЦИАЛЬНОЙ СЕТИ LINKEDIN

На сегодняшний день социальная сеть LinkedIn [1] представляет собой популярный в IT-сегменте портал профессионального характера, где пользователи формируют сообщество из коллег по местам работы или учёбы, общаются на профессиональные темы, следят за карьерными изменениями своих коллег, а также ищут дальнейшие возможности для трудоустройства. Поэтому открытая информация, содержащаяся на рассматриваемом сервисе, можно считать актуальной, валидной, а главное – представляющей интерес для анализа с целью выявления скрытых зависимостей и закономерностей. Несмотря на то, что самих данных в профиле не так много, на основе массива таких данных собранных воедино можно построить модели изменения различных факторов во времени и получить дополнительную информацию о компаниях и их сотрудниках, которую нельзя найти в открытом доступе. Полученная информация может быть использована при процессе поиска нового места работы, помогая соискателю в процессе принятия решения. Также эта информация может быть полезна непосредственно работодателю для более полного понимания текущей картины на рынке труда в сфере IT. Таким образом, целью данной работы является сбор актуальной информации о компаниях и сотрудниках для дальнейшего её визуального представления в удобном графическом виде.

Для анализа были выбраны 40 компаний IT-сектора, расположенных на территории Санкт-Петербурга. Веб-скрапинг [2] осуществляется при помощи инструмента PhantomBuster [3] в режиме онлайн, принимая на вход доступную для моментального изменения Google-таблицу и отдавая на выход постоянно обновляющийся и пополняющийся файл с результатами, доступный по неизменяющийся ссылке. Была построена гибкая архитектура

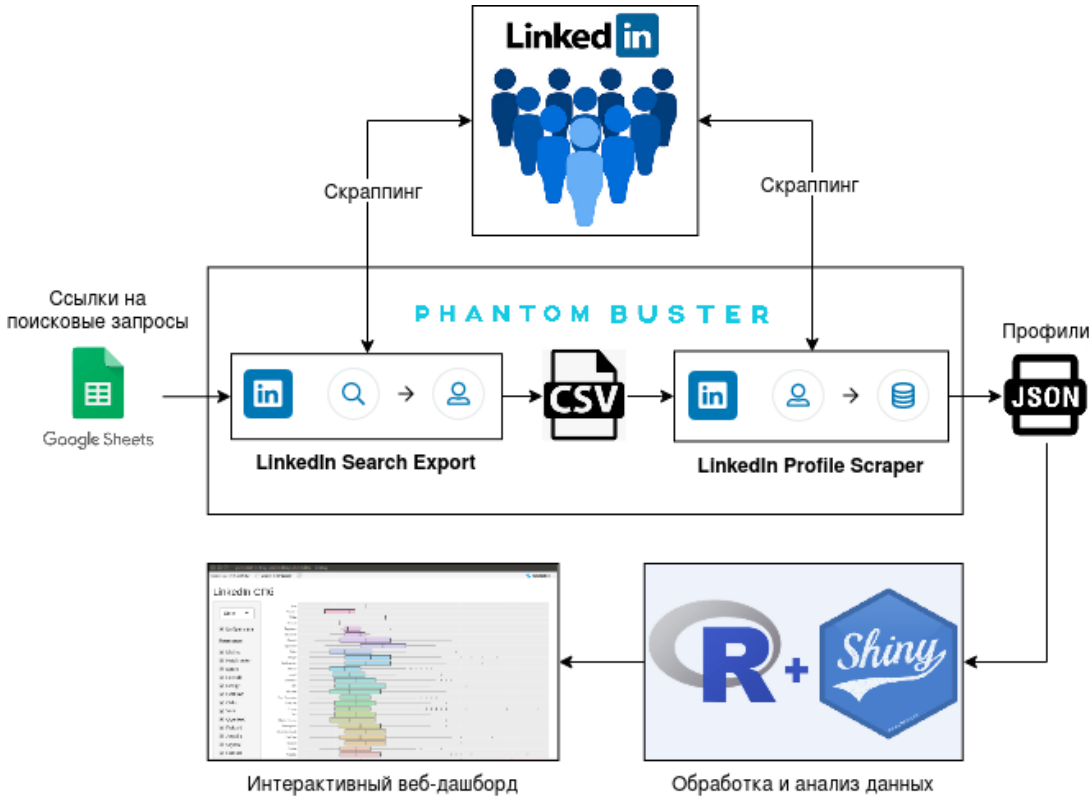


Рисунок 1 – Обобщенный алгоритм работы системы

системы, которая позволяет вносить быстрые изменения в модель данных. В процессе сбора информации профилей были проанализированы следующие данные:

- **Имена сотрудников.** Подавляющее большинство имён указано на латинице. У некоторых профилей фамилия и имя поменяны местами. Одно и то же русское имя в собранных данных было представлено в нескольких вариациях, которое нужно было привести к единому формату. Для точного определения кириллического имени был создан маппинг между русским именем и его английским форматом.
- **Пол сотрудников.** Информация о половой принадлежности не указана в профиле пользователя. Пол был определён на основании имени пользователя при помощи библиотеки “genderizeR” [4], которая содержит в себе большую статистическую базу соответствия между именем человека и его полом. Если пол не был определен с первого раза, имя и фамилия менялись местами и процесс определения пола проводился заново.
- **Опыт работы сотрудников.** Явной информации об опыте работы пользователя не содержится в LinkedIn, но, очевидно, это сумма всех лет работы в тех компаниях, которые указаны у сотрудника в профиле.
- **Возраст сотрудников.** В профилях социальной сети доступны только дата и месяц рождения, но многие пользователи не указывают даже этого. Было сделано предположение, что среднестатистический специалист начинает своё первое высшее образование в 18 лет. Таким образом, возраст сотрудника вычисляется по принципу вычитания из номера текущего года первого года начала первого высшего образования и прибавления числа 18. Если образование не указано, за дату 18-тилетия бралась дата начала первой работы.
- **Навыки сотрудников.** Каждый пользователь социальной сети может указать несколько своих основных профессиональных навыков. На основе этих навыков можно выявить основные компетенции и сферы разработки компаний в которых работают сотрудники.
- **Образование сотрудников.** У большинства пользователей указано место учебы включая ту степень образования, которую они получали. Однако степень образования может быть указана в нескольких вариациях в зависимости от желания пользователя.
- **Должность сотрудников.** В этом поле пользователь социальной сети может написать абсолютно любой текст. В процессе анализа данных была предпринята попытка извлечь из этого поля полезную информацию. Были выявлены группы сотрудников, относящиеся к классу разработчиков, менеджеров и тестировщиков на основе содержания основных ключевых слов.
- **Переходы сотрудников между компаниями.** На основании указанных в профилях местах работы пользователей можно отследить все их перемещения между компаниями. Была составлена матрица смежности, где столбцы и строки – это названия компаний, а ячейки – количество переходов из одной компании в другую. Переходы внутри компании тоже учитывались при подсчете.
- **Динамика изменения численности компаний.** На основании данных о датах смены мест работы также был проведен анализ динамики изменения численности сотрудников в каждой компании по годам.

Для графического представления данных использовались различные типы графиков, такие как boxplot, histogram, bubble plot, scatter plot, wordcloud, barplot, radar chart, donut chart, pie chart, chord diagram, time series, sankey diagram, cadlestick и другие их разновидности. Были использованы следующие графические библиотеки R [5] для построения графиков: ggimage, circlize, chorddiag, dygraphs, wordcloud2, chorddiag.

На первоначальном этапе разработки весь код по обработке, агрегации, нормализации и визуализации данных включая построение графиков был написан в отдельно стоящем R-скрипте. На последнем этапе работы части скрипта были перенесены в динамическое Shiny-приложение с интерактивным веб-интерфейсом, которое было опубликовано в сети Интернет. Некоторые из представленных графики имеют интерактивные элементы, такие как отображение дополнительной численной информации при наведении.

При помощи элементов checkbox можно выбирать те компании, данные сотрудников которых нужно учитывать при построении графиков. При изменении состояния любого элемента страницы графики динамически перестраиваются. Процесс перестроения занимает некоторое время (до 5 сек.), в это время пользователю выводится анимация ожидания загрузки.

Тип отображаемого графика пользователь может выбрать в выпадающем меню. В зависимости от выбранного типа графика изменяются доступные для изменения элементы веб-страницы. Подобный подход позволяет реализовать реактивное программирование на фреймворке R Shiny [6].

В результате проделанной работы был произведен сбор информации о профилях сотрудников 40 компаний расположенных на территории Санкт-Петербурга путем технологий веб-скрапинга. Собрано 10274 ссылок на профили, из них сохранена краткая информация по 9712 пользователям и полная информация по профилям 4861 пользователя. Основным результатом работы является веб-приложение, представляющее собой аналитическую панель для работы с визуализированными данными, которые могут быть динамически изменены и дополнены, так как приложение подключено к сервису веб-скрапинга, а весь массив постоянно пополняемых данных хранится на серверах AWS [7] онлайн.

ЛИТЕРАТУРА

1. https://ru.linkedin.com
2. Москаленко Алексей Анатольевич, Лапонина Ольга Робертовна, Сухомлин Владимир Александрович РАЗРАБОТКА ПРИЛОЖЕНИЯ ВЕБ-СКРАПИНГА С ВОЗМОЖНОСТЯМИ ОБХОДА БЛОКИРОВОК // Современные информационные технологии и ИТ-образование. 2019. №2.
3. https://phantombuster.com/phantombuster
4. https://cran.r-project.org/web/packages/genderizeR/index.html
5. https://www.r-graph-gallery.com/index.html
6. https://shiny.rstudio.com/articles
7. https://aws.amazon.com/ru/

УДК 004

Г. Д. Тузов (2 курс магистратура)
Н. В. Воинов, к.т.н., доцент

РЕАЛИЗАЦИЯ АСИНХРОННОГО ВЫПОЛНЕНИЯ СОБЫТИЙ В ВЫСОКОНАГРУЖЕННОМ РАСПРЕДЕЛЕННОМ ВЕБ-ПРИЛОЖЕНИИ

Представьте себе большое нагруженное распределенное веб-приложение приложение, такое как социальная сеть. Ее пользователи могут выполнять много тяжелый действий (например, разослать приглашения в группу всем своим друзьям). Тяжелое действие - значит, что для его выполнения требуется обработать много данных. В хорошо спроектированной социальной сети не представляется возможным выполнять такие действия по запросу (сразу), потому что необходимо вернуть контроль пользователю как можно быстрее, чтобы не оказать пагубного воздействия на UX. Таким образом, мы видим необходимость создания решения, которое могло бы выполнять пользовательские запросы позже или асинхронно. Данное решение должно быть масштабируемым, отказоустойчивым, иметь простой программный интерфейс и давать гарантии выполнения запроса. Целью данной работы является разработка такого механизма для Одноклассников на базе инфраструктуры и кодовой базы Одноклассников.

Важно отметить, что В Одноклассниках уже есть решение, которое выполняет описанную задачу - оно называется “Job executors”. Это решение использует реляционную базу MSSQL для промежуточного сохранения информации, необходимой для обработки

пользовательского события. Также, при сохранении используется “плоская“ схема представления информации, наподобие java properties. Очевидно, что это неудобный в использовании, неэффективный механизм, использующий устаревшие технологии, который был разработан за короткое время. Поэтому возникла необходимость в его замене.

Новое решение использует:

- Внутреннюю разработку Одноклассников - One Queue.
  One-queue это хранилище очередей. Умеет хранить в себе произвольные задания со временем активации задания и, соответственно, по запросу отдавать активные задания. Умеет складывать ненужное на диск, так что может хранить много информации, умеет пересортировывать события, отказоустойчивое, расширяемое и т.д.
- Внутреннюю разработку Одноклассников - cassandra-one AKA NewSQL AKA C1.
  Вкратце это система, которая использует cassandra 2.0 для хранения данных, но, дополнительно поддерживает:
  - ACID транзакции с откатами и пессимистичными блокировками
  - глобальные индексы ("глобальные" для того чтобы отличать от secondary indexes, которые в кассандре есть)
  - партиционированный sequence генератор
- Кодогенерацию для удобной сериализации/десериализации передаваемой для выполнения пользовательского действия информации

Разработанное решение позволяет распределенной системе функционировать корректно даже когда версии передаваемых объектов, для представления событий, на разных узлах разные. Также, оно является более эффективным и имеет гораздо более простой интерфейс для использования разработчиками, нежели старое решение.

УДК 004.42

А. В. Хоменко (4 курс бакалавриата)
А. В. Самочадин, к.т.н., доцент
Д. А. Тимофеев, ст. преподаватель

РЕДАКТОР ДИАЛОГА ДЛЯ КОРПОРАТИВНОГО АССИСТЕНТА

Работа представляет собой часть выполняемого в СПбПУ проекта создания платформы повышения эффективности деятельности компании на основе технологии гибридного интеллекта. Основной задачей системы является повышение эффективности выполнения корпоративных процессов управления задачами сотрудников, задачами офисной и технической поддержки. Одной из основных частей системы является интеллектуальная система управления диалогом, выполняющая функции распознавания запросов к корпоративным приложениям в форме диалога на естественном языке и формирования ответов на запросы пользователей. Для обеспечения процесса извлечения информации из текстов запросов и построения статистических моделей разработан редактор разметки текста, описание которого и является предметом настоящей работы.

Входная информация формируется диалоговым компонентом системы, который обеспечивает общение с пользователем на естественном языке. Затем диалоги обрабатываются редактором, который обеспечивает разметку текста и сохранение размеченных диалогов.

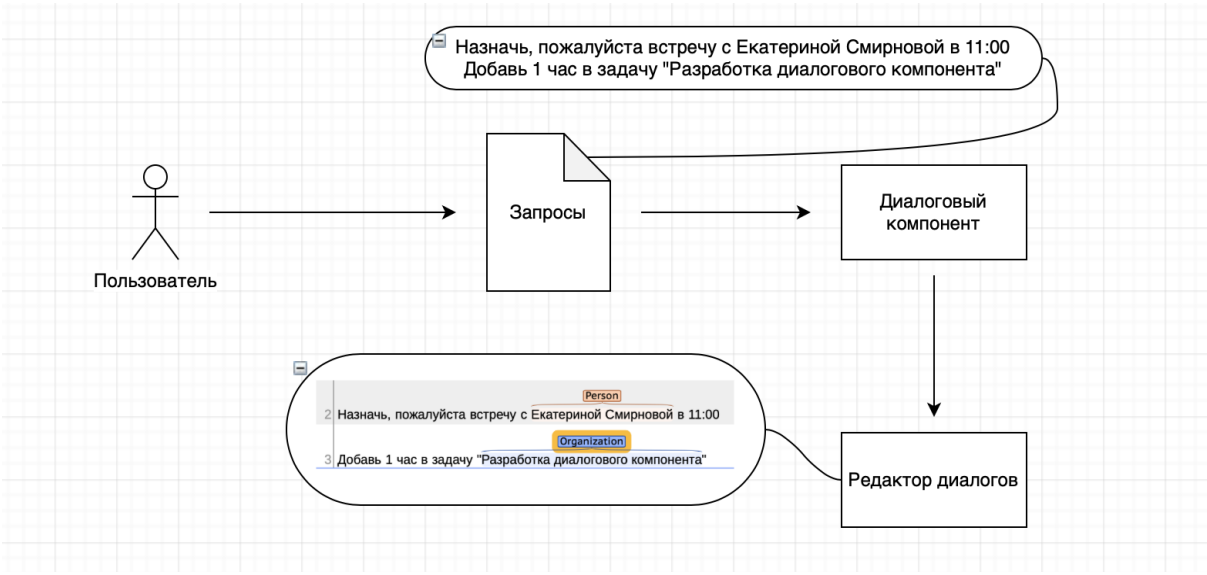


Рисунок 1 – Взаимодействие компонентов системы

Разметка – это добавление заметок к существующим текстовым документам. Разметка не является произвольным текстом: она имеет фиксированную форму, которую можно интерпретировать автоматически.

Основными функциями редактора являются:

– сохранение бесед пользователей на естественном языке из диалогового компонента проекта;
– поиск по репликам;
– разметка диалогов тегами, которые характеризуют предметную область диалога;
– выгрузка диалогов в архив;
– интеграция редактора диалогов с brat annotation rapid tool (brat) – веб-инструментом для добавления заметок к текстовым документам [1].

Технологические требования к разработке редактора включают:

– клиент-серверную архитектуру, взаимодействие между клиентской и серверной частью – протокол HTTP;
– разработку на языке Java;
– использование фреймворков Angular для клиентской части и Spring для серверной.

В редакторе есть три основные структуры данных: диалог, реплика и тег, они реализованы в виде классов в Java. Модель данных компонента представлена на изображении ниже.

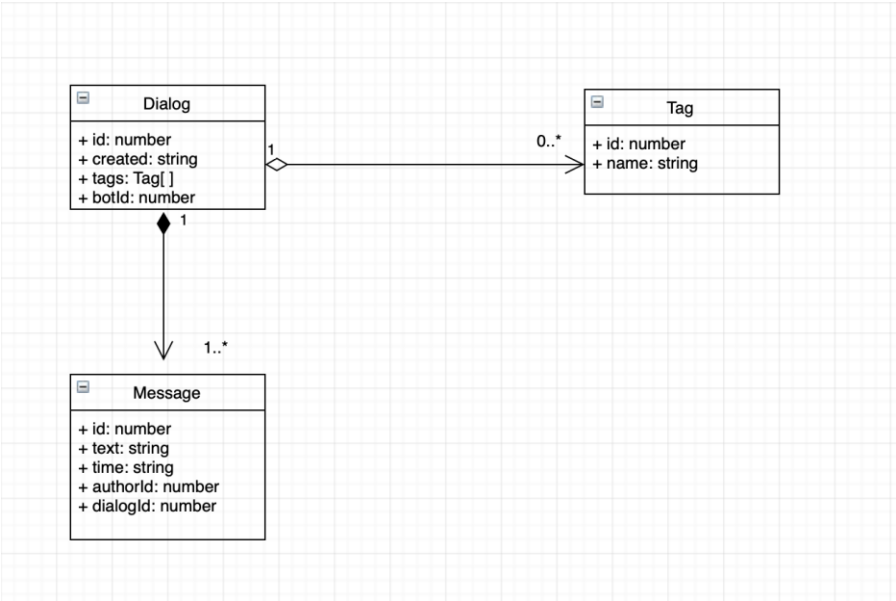


Рисунок 2 – Модель данных редактора диалогов

Редактор разметки текста позволяет работать с коллекцией диалогов, динамически подгружать их из диалогового компонента создавать и назначать теги из предметной области, использовать веб-инструмент brat для сложных текстовых аннотаций, например, аннотаций отношений.

Предполагаемое решение позволяет расширить возможности существующего решения – инструмента для текстовых аннотаций brat [1] – путем добавления возможности разметки тегами, которые описывают предметную область и относятся ко всему документу в целом, а также возможности поиска по репликам в документе и выгрузки размеченного текста в архив. Также преимуществом редактора разметки текста является то, что он является частью разрабатываемой системы, а не отдельным приложением.

Для проверки корректности работы разработанный редактор диалога был протестирован на небольшом количестве текстовых документов, которые относятся к разным предметным областям. В ходе тестирования к текстовым документам была добавлена разметка тегами в том числе с использованием инструмента brat, протестирована функция поиска по репликам, динамического сохранения текстов диалогов из диалогового компонента и выгрузки размеченных диалогов в архив. Таким образом, разработанный редактор диалогов отвечает поставленным требованиям.

ЛИТЕРАТУРА

1. Brat rapid annotation tool [Электронный ресурс]. – Режим доступа: https://brat.nlplab.org/index.html – (Дата обращения: 25.11.2019).
2. Introduction to the Angular Docs [Электронный ресурс]. – Режим доступа: https:// https://angular.io/docs– (Дата обращения: 05.11.2019).

УДК 621.319

Р. Д. Чусов (2 курс магистратуры)
Н. В. Воинов, к.т.н., доцент

РАЗРАБОТКА АЛГОРИТМА И ПРИЛОЖЕНИЯ ДЛЯ ПОСТРОЕНИЯ ФИЛОГЕНЕТИЧЕСКИХ ДЕРЕВЬЕВ ПО БОЛЬШИМ ОБЪЕМАМ ДАННЫХ

У каждого живого организма есть набор генов или геномная последовательность. Данные, содержащиеся в геномной последовательности, содержат информацию об этом организме. В процессе эволюции организмы изменялись, изменялись их геномные последовательности. Было обнаружено, что у близких организмов могут быть схожие геномные последовательности. Если научиться считать расстояния между такими последовательностями, то можно получить инструмент, позволяющий анализировать ход эволюции живых организмов по их геномам.

Целью данной работы является проектирование облачной системы, которая автоматизирует метод построения филогенетических деревьев и позволит проводить сравнение нескольких биологических объектов. На данный момент существуют инструменты, позволяющие автоматизировать этот процесс (FastTree, RaxML7, PhyML), однако они не приспособлены для запуска на кластере, это затрудняет масштабирование инструмента. Реализованная система лишена этого недостатка, так как изначально заточена под кластерные вычисления.

В данной статье рассмотрены существующие способы построения филогенетических деревьев, проведено краткое сравнение существующих алгоритмов построения филогенетических деревьев. В статье будет представлено описание реализованного и

модифицированного для кластерных вычислений алгоритма Neighbor Joining, который лежит в основе также представленного в данной статье приложения.

В задачах филогенетики как правило приходится сталкиваться с решением проблем, связанных с эволюцией символьных цепочек. Эволюция цепочек может быть вычислена на методах, основывающихся на вычислении всех дистанций между цепочками, либо на символьных методах, основывающихся на анализе цепочек посимвольно. В данной работе использован метод дистанций.

Дистанционный метод требует построения новой матрицы, представленной, в которой попарно сравниваются все рассматриваемые последовательности, также представленные на Рисунке 1. Таблица получается путем вычисления метрики расстояния попарно между каждым из геномов, представленных на входе. После этого происходит объединение узлов в дерево при помощи метода ближайших соседей, как показано на Рисунке 1.

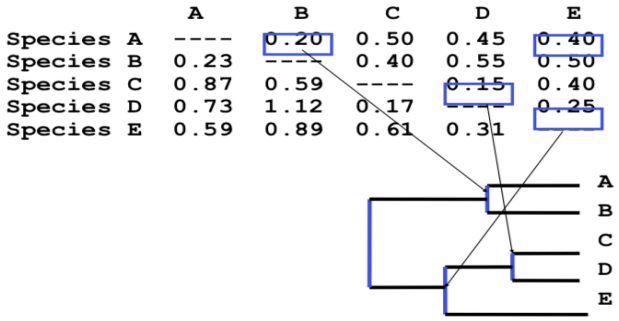


Рисунок 1 – Дистанционный метод построения дерева.

В верхней части находятся результаты сравнения последовательностей при помощи вычисления расстояния Левенштейна, в нижней части находится скорректированное по формуле Джукеса-Кантора: $K(A,B) = -\frac{3}{4}\ln\left(1-\frac{4}{3}D(A,B)\right)$ это же расстояние.

В ходе данной работы были изучены принципы работы уже существующих инструментов для задач построения филогенетических деревьев. В таблице 1 представлена сводная информация для основных существующих инструментов.

Таблица 1 – Популярные инструменты для работы с филогенетическими деревьями.

Длинна	250	1250	5000	78132
Тип	a.a.	a.a.	a.a.	n.t.
RaxML7	90.5%	88.4%	84.4%	--
FastTree	86.9%	83.7%	84.3%	92.1%
PhyML	81.7%	80.1%	--	--
FastME	79.6%	77.7%	75.3%	--
BIONJ	76.6%	73.0%	72.3%	--

Инструмент на данный момент находится в разработке в стадии “Proof of Concept”. К настоящему моменту был развернут кластер Apache Spark, реализован алгоритм построения матрицы расстояний, а также Neighbor Joining алгоритм. Также реализована простая веб-форма, позволяющая загружать данные для анализа и получать дерево в скобочном Newick-формате. Приложение запущено на облачной платформе Heroku и доступно в сети Интернет [https://blooming-wave-47898.herokuapp.com].

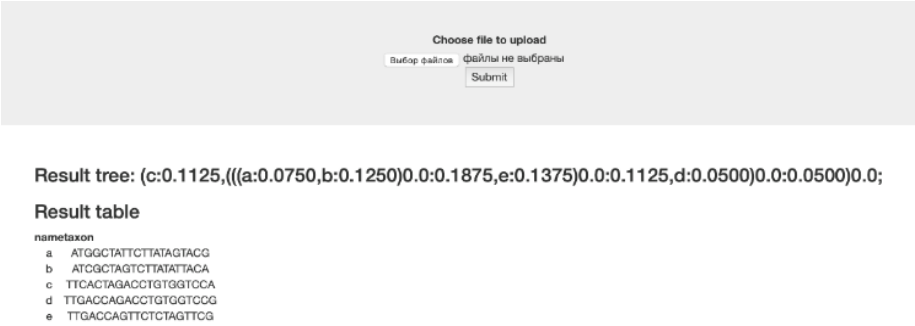


Рисунок 2 – Веб-интерфейс приложения.

В ближайшем будущем планируется:

- Добавить модуль отрисовки филогенетического дерева по матрице расстояний.
- Увеличить количество кластеров, тем самым увеличить производительность.
- Провести анализ открытых данных и сравнить с существующими инструментами.
- Проработать графический интерфейс.

УДК 004.934

Г. С. Шандакова (4 курс бакалавриата)
Т. В. Леонтьева, к. т. н., доцент

РАЗРАБОТКА ГРАФИЧЕСКОГО ВИЗУАЛИЗАТОРА ПО ТЕКСТОВОМУ ФРАГМЕНТУ

Известно, что лучше всего человек воспринимает информацию, представленную графически. Именно поэтому в большинстве обучающих пособий текст иллюстрируется. Эта задача требует дорогостоящей работы художников-иллюстраторов, поэтому актуальным является создание системы, позволяющей автоматически визуализировать небольшие текстовые фрагменты. В существующей литературе системы визуализации текста получили название TTP-систем (англ. Text-To-Picture). Одними из самых популярных TTP систем для английского языка являются WordsEye[1], генерирующая 3D изображения, и CWI[2], генерирующая коллаж. Для русского языка в данный момент существует только Utkus[3], использующая онтологии для синтеза предложения. Эта система генерирует изображения на основе ограниченной базы примитивов, используя наклон объектов для передачи действий и отношений, из-за чего результат работы не является наглядным. Поэтому было решено разработать TTP-систему, направленную на работу с русским языком и при этом обладающую более репрезентативным результатом. В первую очередь такая система может быть применена в области изучения русского языка, позволяя передавать основной смысл простых предложений для людей, которые только начинают своё обучение.

Необходимо заметить, что работа с русским языком имеет свои особенности. В построении предложений большую роль играют окончания, однако после синтаксического анализа текста, для получения корректных изображений, мы вынуждены приводить слова в начальную форму. Учесть все конструкции русского языка не представляется возможным, поэтому в первой версии системы предлагается ограничиться сравнительно небольшим объёмом словаря, состоящим из наиболее употребляемых слов.

Общий процесс построения изображения в разрабатываемой системе выглядит следующим образом:

1. Подготовка текста
2. Синтаксический разбор текста
3. Нахождение семантических шаблонов
4. Поиск подходящих изображений
5. Склеивание коллажа

Первые шаги всех ТТР-систем довольно схожи. Подготовка текста включает в себя удаление вводных конструкций и замену всех заглавных букв на строчные. Далее производится синтаксический разбор текста: определяют главные и второстепенные члены предложения, которые помогут для нахождения семантических шаблонов.

Для следующего шага необходимо составить базу шаблонов, которые представляют собой некоторую словесную конструкцию и макет для её визуального отображения. Причём шаблоны располагаются в зависимости от возможного количества слов, которые могут находиться внутри макета. Таким образом, в начале базы будут находиться шаблоны, соответствующие подчинительным союзам (потому, что они связывают простые предложения в составе сложноподчинённого предложения). Далее располагаются шаблоны, соответствующие сочинительным союзам (они связывают меньшие синтаксические единицы), и затем все остальные.

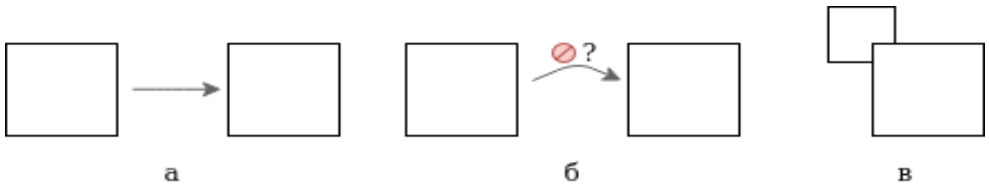


Рисунок 1 — примеры шаблонов для: а) подчинительного союза «потому что», б) сочинительного союза «если не _ , то» , в) квазиагентивного синтаксического отношения.

Перед следующим этапом должны быть удалены: союзы, не имеющие шаблонов в базе и синтаксические отношения, обработка которых не предусмотрена, а все слова приведены к начальной форме.

Далее начинается поиск подходящих изображений для оставшихся слов, который состоит из двух частей. Сначала происходит поиск в собранной вручную базе картинок, так как многие слова абстрактны или не дают репрезентативного представления при автоматическом поиске.

Рисунок 2 — Примеры слов в собранной вручную базе.

Если подходящего изображения не нашлось, то происходит автоматический поиск в интернете. При условии, что оба поиска ничего не дали, происходит поиск картинки для синонима слова. Если же и дальнейшие поиски не дали результата, то вместо изображения слова появится его текстовое отображение. Изображения некоторых слов проходят доработку: например, к географическим названиям добавляется значок «расположения», а к именам собственным добавляется значок пола человека. При этом цифры и символы не проходят через поиск и визуализируются, как и в обычном тексте.

В дальнейшей реализации системы было решено использовать следующие средства: Google Custom Search API [4] для поиска изображений, ISPRAS Texterra [5] для выполнения синтаксического разбора и программу MyStem [6] для определения начальной формы слова. В качестве критериев выбора было обращено внимание на поддержку русского языка, удобство использования, наличие бесплатных версий.

ЛИТЕРАТУРА

1. Coyne B., Sproat R. WordsEye: an automatic text-to-scene conversion system // In proc. of The 28th ACM Annual Conf. on Computer graphicsand interactive techniques 2001 — P. 487–496.

2. Jiang Y, Liu J, Lu H (2014) Chat with illustration. Multimedia Systems 22: P. 5–16
3. Ustalov D. A Text-to-Picture System for Russian Language // Proceedings of the Sixth Russian Young Scientists Conference in Information Retrieval (RuSSIR 2012). – 2012. – P. 35–44.
4. Custom Search | Google Developers URL: https://developers.google.com/custom-search (дата обращения: 16.03.2020)
5. ISPRAS API URL: https://api.ispras.ru/products (дата обращения: 16.03.2020).
6. MyStem - Технологии Яндекса URL: https://yandex.ru/dev/mystem/ (дата обращения: 16.03.2020).

Секция «Подходы к разработке программного обеспечения на основе технологий Dell Technologies»

УДК 004.414

Д. Ю. Агеев (4 курс бакалавриата)
Т. А. Вишневская, ст. преподаватель

СИСТЕМА МОНИТОРИНГА СОСТОЯНИЯ КОМПОНЕНТОВ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

Введение. Целью проекта является создание системы по сбору данных(метрик) с персональных компьютеров, на которых предустановлено специальное ПО, для наблюдения за основными атрибутами компонентов в течение длительного промежутка времени. Такими атрибутами могут быть температура/нагрузка/частота процессора, видеокарты, ОЗУ, загруженность этих компонентов. А также хотелось бы отображать и оценивать общее состояние отслеживаемого компьютера основываясь на анализе изменения этих метрик в течение какого-то временного периода.

Обзор подобных систем. Существуют похожие аналоги по мониторингу комплектующих, такие как AIDA64[1], Speccy, прикладные решения для собственных комплектующих от AMD(AMD System Monitor)[2] и NVIDIA(NVIDIA System Monitoring)[3], и native решения от Microsoft. Особенность всех перечисленных выше приложений в том, что они показывают информацию в реальном времени и не сохраняют исторические данные.

Структура приложения. Для реализации приложения предлагается использовать микросервисную архитектуру, а также MVC схему для разделения данных приложения. В нашем случае получается следующая архитектура для backend (Рис. 1):

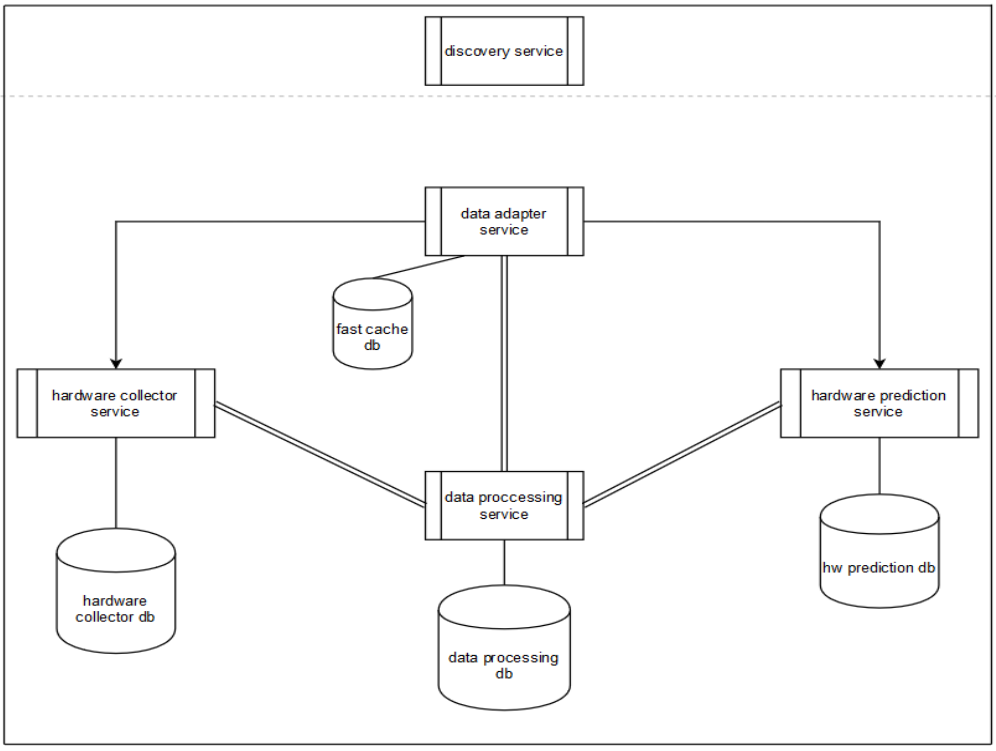


Рисунок 1 – Архитектура Backend

Также для отображения данных конечному пользователю мы будем использовать веб интерфейс. Помимо этого, мы должны собирать статистику с компьютеров с помощью клиента. Тогда, общая схема нашего сервиса выглядит следующим образом (Рис. 2):

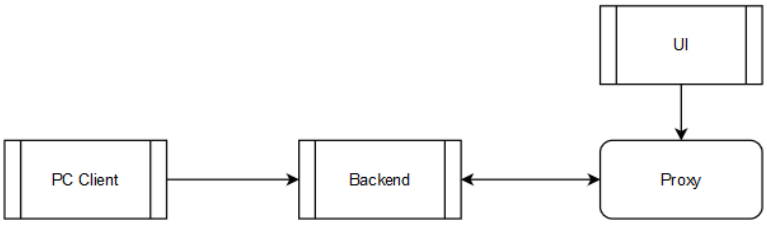


Рисунок 2 – Схема Сервиса

Используемые инструменты. Для разработки микросервисов выбрана версия Java 11 с использованием библиотек Spring Boot 2.2.5, позволяющей легко описывать REST API архитектуру, и Hibernate, с помощью которой удобно работать с объектами базы данных. Помимо этого, платформа Angular 8 отлично подходит для разработки веб интерфейса одностраничных сайтов. Предполагается использовать СУБД PostgreSQL[4] для хранения данных, так как это одна из популярных и удобных в использовании реляционных БД с оглядкой на объектно-ориентированный подход к реализации.

ЛИТЕРАТУРА

1. AIDA64 URL: https://www.aida64.com/products
2. NVIDIA URL: https://www.nvidia.com/en-us/drivers/system-monitor-ru/
3. AMD URL: https://www.amd.com/ru/technologies/ryzen-master
4. PostgreSQL URL: https://www.postgresql.org/docs/

УДК 004.08

В. Е. Артемьева (4 курс бакалавриата)
С. А. Молодяков, д.т.н., профессор

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СРЕДСТВ ХРАНЕНИЯ ДАННЫХ НА ОСНОВЕ ВИРТУАЛЬНЫХ ЛЕНТОЧНЫХ УСТРОЙСТВ

Целью работы является изучение средств хранения данных и реализация программного обеспечения для их настройки и установки. Современные системы хранения данных (СХД) – это комплексное программно-аппаратное решение по организации надёжного хранения информационных ресурсов и предоставления гарантированного доступа к ним.

Необходимость в СХД возникла, когда массивы хранимой и передаваемой информации превысили все мыслимые на тот момент пределы. Согласно данным TAdviser, с 2010 г. объем хранимой информации каждый год возрастает примерно на 50% от ее первоначального объема. Растет и стоимость информации, поскольку от нее напрямую зависят все бизнес-процессы. Системы хранения данных по типу накопителей информации делятся на три больших группы: дисковые, флэш, ленточные (кассетные).

Широкое распространение ленточных накопителей было связано с большими ЭВМ и, в частности, мейнфреймами IBM. Мейнфрейм — это большая универсальная высокопроизводительная практически безотказная электронная вычислительная машина

(часто это сервер) с большими ресурсами ввода-вывода, значительным объёмом оперативной и внешней памяти.

Использование современных ленточных накопителей имеет следующие отличительные черты:

– большая ёмкость;

– низкая стоимость и широкие условия хранения информационного носителя;

– стабильность работы;

– надёжность;

– низкое энергопотребление у ленточной библиотеки большого объёма.

Недостатками выступают низкая скорость произвольного доступа к данным из-за последовательного доступа (лента должна прокрутиться к нужному месту) и сравнительно высокая стоимость устройства записи (стримера).

Накопитель на магнитной ленте, поддерживающий работу одновременно с несколькими лентами, называется ленточной библиотекой. Роботизированные ленточные библиотеки могут содержать хранилища с тысячами магнитных лент, из которых робот автоматически достаёт требуемые ленты и устанавливает в одно или несколько устройств чтения-записи. С программной точки зрения такая библиотека выглядит, как один накопитель с огромной ёмкостью и значительным временем произвольного доступа.

Виртуальная ленточная библиотека (ВЛБ, VTL, Virtual tape library) — это технология виртуализации хранения данных. При этом различные устройства хранения данных (обычно жесткие диски) представлены как ленточные приводы либо ленточные библиотеки с целью сохранения совместимости с существующими системами резервного копирования.

Преимущества ВЛБ перед ленточной библиотекой:

– большая скорость сохранения и восстановления данных;

– масштабируемость;

– невысокая стоимость при небольшом объёме данных;

– простота произвольного доступа, в отличие от ленты, где поиск нужного фрагмента требует гораздо больше времени.

Преимуществом дискового накопителя является также то обстоятельство, что запись на дисковый накопитель может вестись с переменной скоростью. Благодаря этому, процесс восстановления данных происходит быстрее, чем резервирование вне зависимости от реализации.

Семейство продуктов Dell EMC Disk Library for mainframe (DLm) предоставляет заказчикам мэйнфреймов IBM System z возможность заменить свои физические ленточные библиотеки, в том числе традиционные виртуальные ленточные библиотеки на динамические виртуальные ленточные решения без магнитных лент, устраняющие проблемы, связанные с традиционной ленточной обработкой.

Основными компонентами системы DLm8500 являются контроллер виртуальной ленточной эмуляции (VTEC) и внутренняя система хранения. VTEC — это подсистема, которая подключается к IBM или IBM-совместимому мэйнфрейму и обеспечивает эмуляцию ленточных накопителей.

Все модели DLm построены с использованием программного пакета на магнитной ленте (tape-on-disk software), известного как Virtuent. Программное обеспечение Virtuent работает на базовом аппаратном контроллере, который обеспечивает четыре соединения FICON (Fibre Connection — последовательный канал передачи данных) с мэйнфреймом.

DLm бесперебойно работает со средой мэйнфреймов, включая основные системы управления лентами и приложения резервного копирования, без необходимости изменять какие-либо операторы JCL (Job Control Language — язык программирования, применяющийся в операционных системах мейнфреймов фирмы IBM) клиента.

Нет необходимости запускать задачу или определять конкретную подсистему для работы с DLm, поскольку хост мэйнфрейма видит DLm как ленточные устройства.

DLm предлагает много преимуществ по сравнению с традиционной лентой, включая:

– более быстрая обработка запросов на монтирование ленты;

– нет требований, как к физическим лентам (снижение стоимости, хранения и потенциальной возможности потери лент и данных);

– поддержка обмена данными между несколькими VTE (virtual tape emulators) (создание уровня доступности данных, которого не было в предыдущих виртуальных ленточных системах мэйнфреймов);

– поддержка низкого уровня доступа к внешним физическим лентам, что позволяет мэйнфрейму записывать и читать физические ленты;

– целостность данных поддерживается за счет хранения данных на лентах во внутренних массивах хранения и использования технологии RAID 6 для защиты данных от сбоев физических дисков;

– встроенные технологии мониторинга и отчетности, такие как простой протокол управления сетью и ConnectEMC, которые генерируют предупреждения, когда требуется внимание в среде DLm;

– поддержка репликации данных ленты между системами DLm вплоть до двух локальных или удаленных систем DLm;

– нет единой точки отказа данных ленты мэйнфреймов, если в системе DLm более одного VTE.

В результате исследования виртуальных ленточных библиотек будут разработаны инструменты, отвечающие за настройку и установку программного обеспечения DLm, высокую доступность системы и обеспечение качества и исправности релизов Dlm.

ЛИТЕРАТУРА

1. Disk Library for mainframe (DLm) product documentation. [Электронный ресурс]. – Режим доступа: https://community.emc.com/docs/DOC-63995

УДК 621.319

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

МЕТОД АВТОМАТИЧЕСКОЙ ПРОВЕРКИ ЛИМИТОВ РЕСУРСОВ ОБЛАЧНОГО ПРОВАЙДЕРА AWS

В настоящее время многими компаниями при разработке и обслуживании ПО используются различные облачные провайдеры такие, как Google Cloud Provider (GCP), Amazon Web Services (AWS), Microsoft Azure. Самым популярным из них является Amazon Web Services (AWS). Все облачные компании предоставляет огромное количество сервисов, которые позволяют полностью или частично автоматизировать многие процессы, при для пользователя, например, крупной компании в данном случае пропадает необходимость держать собственные физические сервера, то есть плюсом идёт сокращение расходов.

При этом у каждого ресурса в сервисе есть лимит по использованию. Запрос на создание ресурса, лимит которого превышен, вызовет ошибку. В AWS есть возможность создания запроса на увеличение лимита, но при этом данная операция занимает минимум один день. В таком случае работы команд, использующих ресурс с превышенным лимитом, встаёт на долгое время.

Целью работы является разработка средства для предотвращения ситуаций неожиданного исчерпания лимитов, то есть средство должно в определенный период времени, который мог бы назначаться пользователем, опрашивать все или выбранные ресурсы AWS, проверяя, не приближается ли значение количества ресурсов к какому-либо из двух критических показателей, значение которых тоже можно задавать. Одно значение отвечает за

предупреждение, а второе за опасность. При этом должна быть настройка оповещений при превышении показателей. Также необходимо иметь возможность создания запроса на увеличение лимита, если пользователь хочет этого. По факту должен производиться сбор метрик, их анализ и оповещение пользователя.

Особенности инструмента:

Использование Golang - Golang является распространенным языком, по этой людям, пишущим на нём, было бы удобнее использовать его для работы с библиотекой.

Максимально возможное использование AWS Service Quotas API - получение достоверных метрик из-за того, что данный сервис является наиболее новым в AWS по сравнению с другими сервисами сбора данных о лимитах.

Возможность автоматически создавать запрос на увеличение лимита с ограничениями от пользователя на верхнее значение - с помощью API AWS Serivce Quotas есть возможность создавать запрос для увеличения лимита, можно дать пользователю выбрать данную опцию для автоматизации при переходе границы использования ресурса

Добавить возможность посылать результаты в другие приложения/сервисы: первая цель - Slack, так как Slack является наиболее распространенным корпоративным мессенджером, для работы с ним можно использовать API или библиотеки для работы с ним.

На рисунке 1 представлены различные способы разворачивания инструмента для работы в системе. Есть возможность использовать сервис AWS Lambda, где есть возможность запускать код и задать таймер для запуска с помощью менеджера времени cron. Также предусматривается возможность работы docker-контейнера с приложением-библиотекой, который можно развернуть на машине из сервиса AWS EC2 или прямо в кластере AWS ECS. Все решения имеют связь со Slack.

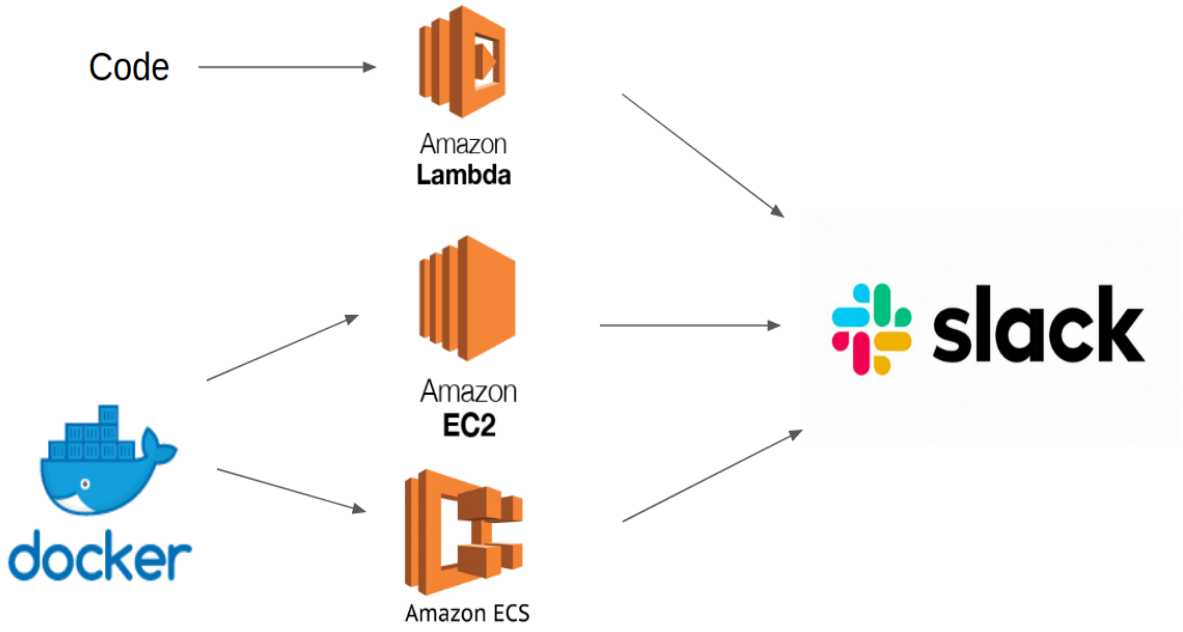


Рисунок 1 – Возможности запуска инструмента

Для создания инструмента используется Golang вместе с AWS Golang SDK. Данный SDK предоставляется Amazon и реализует функции для работы с API сервисов AWS. AWS Golang SDK используется для работы с API Quota Service, где можно получить все возможные лимиты, значения использования сервисов, лимитов по умолчанию и модифицированных лимитов, а также создать запрос на увеличение лимита. Для работы со Slack используется API Slack. Также используется сервис AWS IAM для работы с разрешениями в облачных решениях.

УДК 004.738.5: 004.451.55

Д. А. Васина (2 курс магистратуры)
Т. С. Горавнева, к.т.н., доцент

РАЗРАБОТКА ПРИЛОЖЕНИЙ ВАЛИДАЦИИ ФАЙЛОВ ХРАНЕНИЯ ИЛИ ОБМЕНА ИНФОРМАЦИИ ОБ ИЗДЕЛИЯХ САПР СУДОСТРОИТЕЛЬНОЙ ТЕМАТИКИ

Тема данной научно-исследовательской работы дополняет разработку, выполненную в процессе подготовки и защиты бакалаврской выпускной работы (ВКР), целью которой являлось создание Интернет-проекта с базой данных судостроительного проката (полособульба, швеллера, двутавра, тавра, уголка, полосы), фирм, поставляющих и изготавливающих судостроительный прокат и предприятий, использующие данный прокат для постройки судов.

Разработка данного валидатора является актуальной, так как, проанализировав другие валидаторы, в каждом из них есть недостатки. В одних приложениях нет загрузки из файла, в других ограничен объем файла, нет проверки синтаксиса у схем, нет в принципе проверки соответствия XML-документа XML-схемам или возможность проверить только по одной схеме.

В ходе данной работы решены следующие задачи:

1. Выполнен поиск и обзор существующих на современном этапе инструментов валидации XML-файлов обмена или хранения информации.
2. Сделан сравнительных анализ инструментов валидации, выявление их достоинств и недостатков.
3. Выполнено выявление характерных ошибок в структуре и содержании XML-файлов обмена или хранения информации.
4. Получены методики и рекомендации по их обнаружению на основе информации об изделиях САПР судостроительной тематики.

XML широко используется для отделения данных от HTML, для хранения и передачи данных. Его широкое применение можно объяснить 3-мя важными плюсами:

– XML расширяем;
– есть возможность хранить данные независимо от того, как они будут представлены;
– XML является общедоступным стандартом.

Для практической реализации XML-валидатора в качестве предметной области выбраны изделия судостроительной тематики (судостроительный прокат и его характеристики, рассмотренные в докладе сборника [1]).

Условно работа валидатора разделена на 3 части:

– проверка синтаксиса XML-документа (правила вложенности, наличие корневого элемента, соответствие регистру, наличие всех скобок и так далее);
– проверка синтаксиса XML-схемы. В данном приложении выбрано две наиболее используемые схемы: DTD и XSD (проверка синтаксиса сложных и простых элементов, наличие всех скобок и так далее);
– проверка соответствия XML-документа XML-схеме.

Ввод данных как самого XML-документа, так и схемы возможен в самом приложении, так и из файла. После ввода документа, запускается проверка синтаксиса и при обнаружении ошибки, выводится название ошибки (например, «Открывающий и закрывающий тег имеют одинаковое имя в разном регистре») и ее позиция, также проверка синтаксиса происходит и у XML-схем.

Приложение написано на языке программирования C# Microsoft Visual Studio 2019.

ЛИТЕРАТУРА

1. Васина Д.А., Горавнева Т.С. Разработка клиентской части web-сайта судостроительного проката. «Современные технологии в теории и практике программирования», Сборник материалов конференции 19 апреля 2019 года, ПОЛИТЕХ-ПРЕСС, 2019 – 248с., с. 174-175, тираж 200, ISBN 978-5-7422-6508-5

УДК 004.658.6

А. К. Голев (4 курс бакалавриата)
Т. В. Коликова, ст. преподаватель

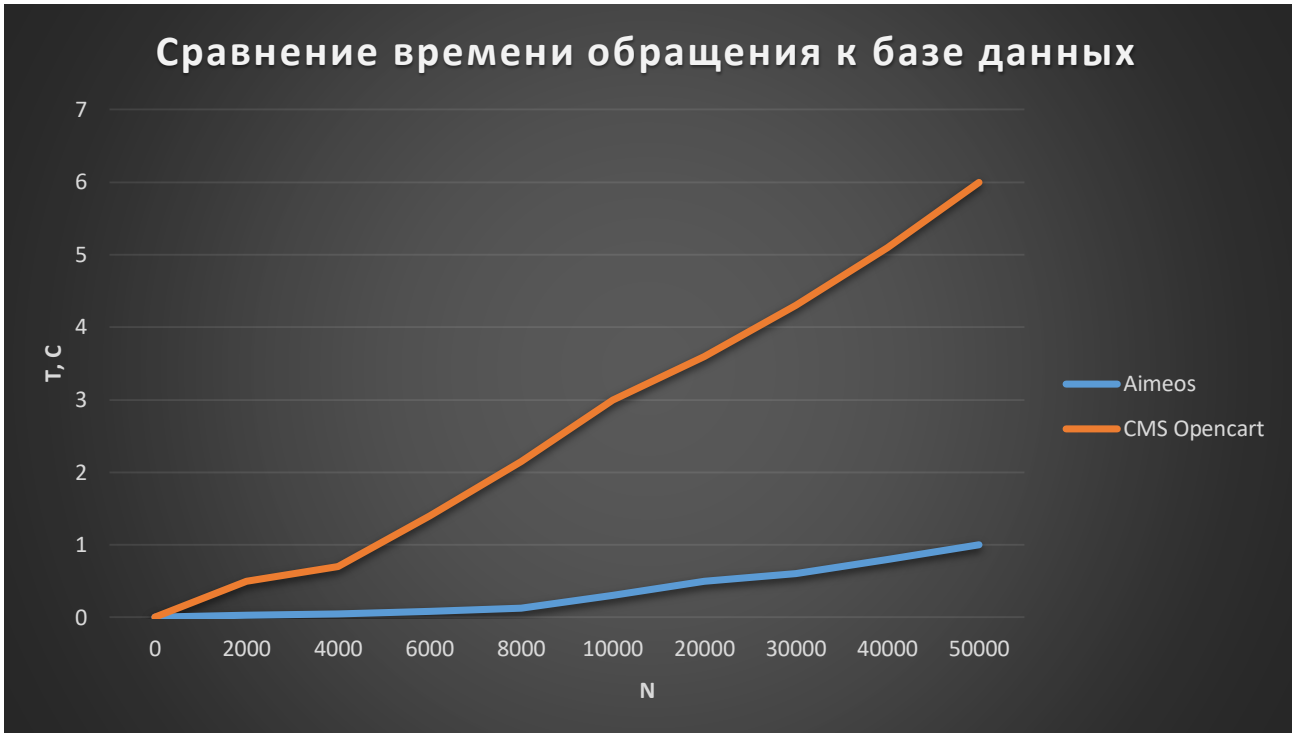
СРАВНЕНИЕ РАЗРАБОТКИ RESTFUL API ДЛЯ E-COMMERCE ПЛАТФОРМ

Целью работы является сравнение двух популярных CMS платформ для интернет-магазина на языке программирования PHP с открытым исходным кодом и особенности написания RESTful API для них. С целью выяснить целесообразность перехода с первой платформы на вторую.

В настоящий момент потребность в создании интернет-магазина очень высока, как и создание своего мобильного приложения, для нескольких платформ. Сервер должен уметь общаться со всеми платформами и быть независимым от них. Для этого применяется архитектурный стиль REST (Representational State Transfer). А общение с каждой платформой осуществляется через конкретный API (Application programming interface). Также для интернет-магазина очень важна скорость взаимодействия с сервером так как она напрямую влияет конверсию.

В первой части работы используется CMS Opencart, очень популярная платформа для создания интернет-магазинов и часто используемая малым бизнесом. У платформы имеются преимущества в скорости разработки, но трудности в виде расширения функционала и производительности. В ходе разработки API для мобильного приложения возникли проблемы с аутентификацией и было принято решение создать собственный апи, а не использовать готовые решения платформы. Также было замечено, что встроенные методы обращения и работы с базой данных в CMS Opencart настолько не оптимизированные, что работа с 10000 товарами приводит к тому что обращение к базе данных занимает около 10 секунд что неприемлемо для e-commerce платформы. Вследствие чего была предпринята попытка переписать встроенные методы обращения и работы, приведшая к уменьшению времени обращения до примерно 3 секунд, что все еще не является минимально удовлетворительным временем обращения к базе данных. Дальнейшая оптимизация времени нецелесообразна так как основную задержку создает структура базы данных CMS Opencart. В итоге данная задержка привела к выбору, или произвести значительные изменения в платформе CMS Opencart, или изменению платформы.

Во второй части используется специализированный инструментарий для создания интернет-магазина Aimeos для Фреймворка Laravel PHP. Являющимся одним из самых популярных для данного Фреймворка. Преимущества в виде высокой производительности, масштабируемости и кастомизации. Из минусов можно выделить ограниченную документацию и перегруженность, что негативно сказывается на времени разработки. В Aimeos присутствовали готовые методы для API, но их оказалось недостаточно, что привело необходимости их расширения. Также необходимо отметить, что время освоения инструментария была в несколько раз выше чем у конкурента, но плюсом была высокая, устраивающая нас производительность. Обращение к базе данных с 10000 товаров у Aimeos происходит за 0,3 секунды, что являет собой вполне достойный результат.



ЛИТЕРАТУРА

1. Официальный сайт Opencart // [официальная документация]/— URL: https://www.opencart.com/ (дата обращения:16.03.2020).
2. Официальный сайт Aimeos // [официальная документация]/— URL: https://aimeos.org/ (дата обращения:16.03.2020).
3. Официальный сайт Laravel // [официальная документация]/— URL: https://laravel.com/ (дата обращения:16.03.2020).
4. Официальный сайт PHP // [официальная документация]/— URL: https://www.php.net/ (дата обращения:16.03.2020).
5. 30+ кейсов: как скорость сайта влияет на конверсию// [статья на habr]/— URL: https://habr.com/ru/post/300210/ (дата обращения:16.03.2020).

УДК 004.457

Р. А. Долматов (4 курс бакалавриата)
С. А. Молодяков, д.т.н., профессор

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ ПРИМЕНИЯ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ TERRAFORM И ANSIBLE ПРИ РАЗВЕРТЫВАНИИ ПРИЛОЖЕНИЯ В ПУБЛИЧНОЕ ОБЛАКО

Развертывание - это все действия, которые делают программную систему готовой к использованию. Данный процесс является частью жизненного цикла. В целом, процесс развертывания состоит из нескольких взаимосвязанных действий с возможными переходами между ними. Раньше, чтобы развернуть какое-либо приложение, приходилось покупать и настраивать собственные физические серверы. Такой подход обладал большим количеством недостатков, например, если для нормальной работы приложения ему достаточно одного сервера, платить все равно приходилось за два - расходы на содержание и обслуживание инфраструктуры оказывались неоправданно высокими. Публичное облако - инфраструктура, предназначенная для свободного использования широкой публикой. Оно может находиться в собственности или управлении различными организациями: коммерческими, научными,

правительственными и так далее. Однако, публичное облако не означает, что данные пользователей доступны всем, так как здесь реализованы механизмы безопасности для контроля доступа. Достоинством использования данного типа облака является простота настройки и низкая стоимость. Изучение возможностей использования облачных сервисов и программ представляется очень актуальным.

Схема модели обслуживания публичного облака представлена на рис.1. На нижнем уровне находится инфраструктура услуг IaaS (Infrastructure-as-a-Service). Она позволяет создавать виртуальные машины с любой операционной системой по своему выбору. Выше располагается PaaS (Platform as a Service). Она включает в себя все, за исключением самого кода приложения, пользователей и данных. На верхнем уровне находится платформа SaaS (software as a service), которая представляется моделью обслуживания, при ограниченном доступе к программному обеспечению. Сервисы четырех известных гигантов - Amazon Web Services(AWS), Google, Azure и Heroku поддерживают облачные платформы в соответствии с описанной моделью (рис.1).

Целью работы является изучение возможностей использования утилит Ansible и Terraform [1, 2], а также их сравнительный анализ при развертывании приложения. Обе утилиты работают в рамках модели.

Рисунок 1 – Схема модели обслуживания.

Ansible — это система оркестрации. Она написана на Python, присутствует в популярных дистрибутивах и не требует клиента на целевых машинах. Для работы с Ansible достаточно передать скрипт (сценарий, файл конфигурации (config)) с перечислением действий, который он должен выполнить на целевой машине. Файл конфигурации Ansible называется playbook. Он описывается на языке YAML. Ключевой объект файла - задача. Это массив (список) вложенных блоков-задач, каждая из которых описывает одно действие. Задачи могут, в свою очередь, содержать блок подзадач. Задач может быть сколько угодно, и Ansible выполнит каждую по очереди. Для работы с Ansible необходимо: Linux, Python и в некоторых случаях SSH - сетевой протокол прикладного уровня, который дает возможность шифрования передаваемых данных и паролей, а также машина (master), **с которой** мы осуществляем управление, другими словами — на которой мы запускаем Ansible со скриптом. Она выполняет команды на удаленной (целевой) машине и target ,**над которой** мы выполняем задачи, другими словами — целевая машина, на которой по скрипту нужно развернуть рабочее окружение.

Terraform - инструмент, позволяющий разрабатывать, изменять инфраструктуру сервиса в формате кода. Инфраструктура описывается в конфигурационных файлах и может применяться, как к работающей системе, так и к совершенно новой.

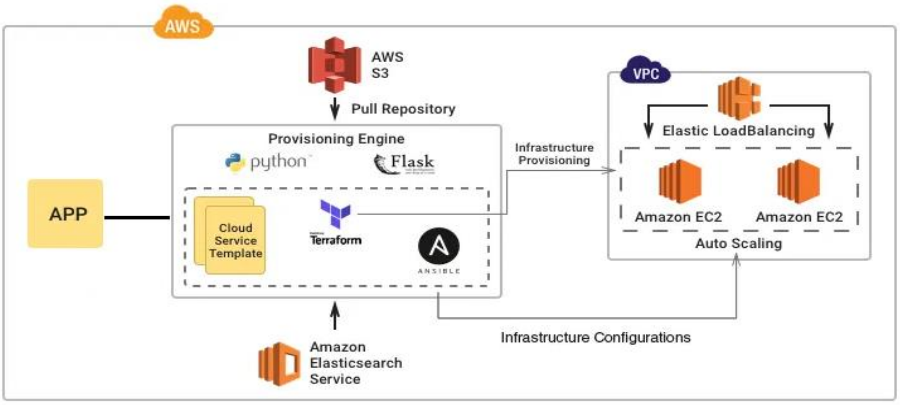


Рисунок 2 – Архитектура развертывания.

В ходе работы разрабатывается приложение для публичного облака Amazon с применением утилит Ansible и Terraform. Для развертывания приложения используется специально разработанный язык программирования HashiCorp Configuration Language (HCL) (рис. 2). Конфигурация состоит из 4 файлов: mail.tf – основной файл, какие инстансы нужны, variables.tf – описание переменных и значений по-умолчанию, outputs.tf – описание выходных переменных, выбор параметр из созданного инстанса , например IP , слзданного в облаке инстанса. После того, как Terraform создает ресурсы виртуальной машины, он часто вызывает своего связанного "провайдера" для настройки ресурсов виртуальной машины, например, изменение конфигурации системы, установка переменных среды или тоже развертывание приложения, но в "механизм подготовки" (provisioning engine) Ansible вызывает «провайдеров» на разных облачных платформах (общедоступных или частных) для создания и управления ресурсами инфраструктуры, а также вызывает «провайдеров» для настройки вновь создаваемых ресурсов. Не все ресурсы поддерживают механизм подготовки. По сути, ресурсы виртуальных машин, такие как ECS, EC2 поддерживают предоставление этого механизма. Используется существующий "провайдер" Terraform для создания динамических файлов Ansible для вновь создаваемых виртуальных машин и добавления их в текущую папку ресурсов Ansible (Ansible объединит все файлы ресурсов, добавленные текущим файлом ресурсов во время выполнения), и, наконец, на основе динамических ресурсов выполняется Ansible - скрипт для настройки вновь созданной виртуальной машины. Итогом данной работы считается развертывание Splunk, которое разворачивается на виртуальной машине через утилиты в публичное облако. Данный процесс нужен для кастомизации, усиления контроля версий.

ЛИТЕРАТУРА

1. Официальный сайт Ansible https://www.ansible.com
2. Официальный сайт Terraform https://www.terraform.io

УДК 004.771

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

СИСТЕМА КРОССПЛАТФОРМЕННОГО МОНТИРОВАНИЯ УДАЛЁННЫХ ФАЙЛОВЫХ СИСТЕМ

Целью работы является проектирование системы кроссплатформенного монтирования удалённых файловых систем и разработка её подсистемы на платформе ОС Android.

Создание системы стало следствием анализа рынка решений по работе с файлами на уровне широкого пользователя. Большинство программных и аппаратных продуктов в данной предметной области можно разделить на несколько категорий – это облачные хранилища,

сетевые хранилища и синхронизаторы файлов. Все они недостаточно удобно решают простую задачу: иметь удобный, быстрый доступ ко всем файлам на всех устройствах пользователя.

Облачные хранилища, такие как Google Drive, Microsoft One Drive вынуждают пользователя загружать свои файлы на удалённые сервера. Очевидно, что данное решение подойдёт не всем, существуют группы пользователей, для которых оно приводит к следующим проблемам. Если документов много по количеству, и они часто меняются, добавляются или устаревают, то постоянно загружать их в хранилище и чистить его — это чрезмерная затрата времени. Если файлы большого размера, они могут не уместиться в лимит предоставленного объема. Если файлы особо ценные, хранение их вне личного, физически защищённого хранилища – это уязвимость информационной безопасности.

Синхронизаторы файлов, такие как Resilio Sync и rsync, решают многие проблемы облачных хранилищ. Работают они по принципу синхронизации папок и файлов между устройствами. Если пользователю необходимо открыть удалённый файл, приходится сначала скачать его на устройство. Однако, и в этом случае возникают нерешённые трудности. Если места на используемом устройстве недостаточно для скачивания файла – пользователь не сможет его открыть. Если файл слишком большой, ожидание скачки покажется слишком долгим, особенно, если нужно срочно просмотреть маленькую часть большого файла.

Сетевые хранилища, или Network Attached Storage (NAS) представляют собой сервер, предназначенный для хранения данных, который работает под управлением специализированных операционных систем или подключается к компьютеру по кабелю. Для того чтобы решить поставленную задачу с помощью NAS, пользователь должен купить такое хранилище и отдельно организовать сервис доступа к этим файлам. Очевидно, что это решение - не для широкого круга пользователей.

В данной работе проектируется система, предназначенная для решения поставленной задачи и перечисленных проблем. Для создания максимально удобного и безопасного доступа к файлам на удалённых устройствах система должна выполнять следующие требования:

– Система должна организовать хранение файлов на устройствах пользователя, а не на выделенных удалённых серверах

– Система должна монтировать удаленные файловые системы целиком, чтобы не вынуждать пользователя выделять специальные папки, переносить туда файлы и синхронизировать их между устройствами.

Система должна иметь клиентские приложения на разных платформах со следующей функциональностью:

– управление учётной записью

– просмотр удалённых файловых систем

– просмотр удалённых и локальных файлов во встроенных средствах просмотра

– управление удалёнными и локальными файлами

– работа в режиме файлового сервера, обеспечение удалённым клиентам доступа к файловой системе

Дополнительные требования к системе:

– Удалённая файловая система должна монтироваться как локальная, для просмотра содержимого не только в специализированном клиенте, но и в системных и других диспетчерах файлов

– Удалённые файлы должны предоставлять случайный доступ к содержимому для мгновенного начала просмотра больших файлов без ожидания их скачки на используемое устройство

На основе приведённой функциональной спецификации была спроектирована система со следующей архитектурой:

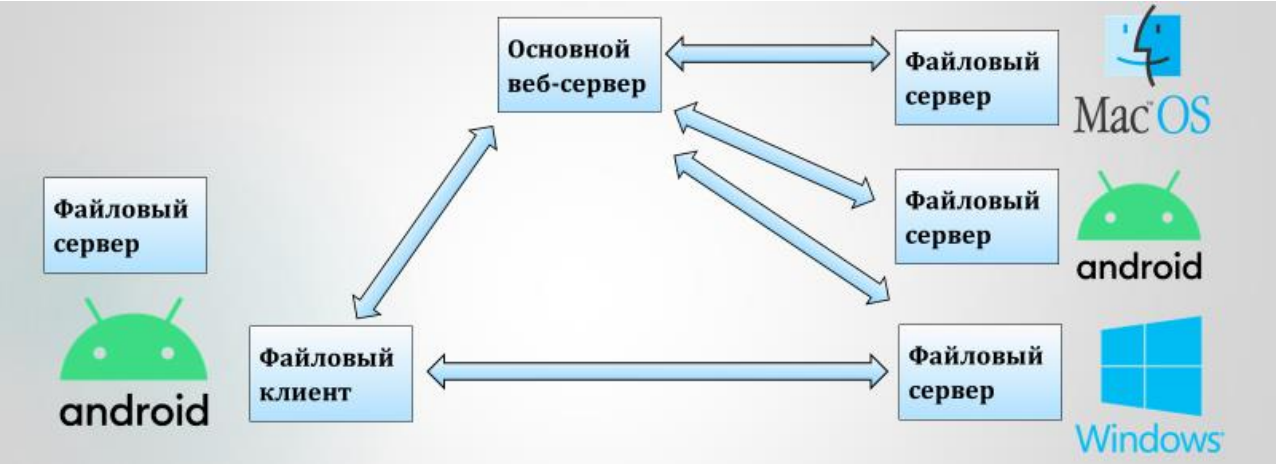


Рисунок 1 – Система кроссплатформенного монтирования удалённых файловых систем.

Приложения на разных платформах выступают в качестве файловых серверов, которые обеспечивают удалённым клиентам доступ к своей файловой системе. И наоборот, файловые клиенты позволяют обозревать удалённые файловые серверы, как указано в спецификации. Основной веб-сервер используется для авторизации файловых клиентов и регистрации файловых серверов в системе. После получения данных о файловых серверах от основного веб-сервера, файловый клиент может установить P2P соединение по протоколу HTTPS с любым из них. После установки соединения, пользователь может напрямую управлять удалёнными файлами, качество доступа к которым будет зависеть только от скорости Интернет-соединения у файлового сервера и файлового клиента. Более того, если и клиент, и сервер зарегистрированы в одной локальной сети, то обмен данными будет происходить на скорости этой локальной сети.

Реализована подсистема монтирования удалённых файловых систем на платформе ОС Android. В том числе, создано клиентское приложение, построенное по принципам современной архитектуры приложений на Android. Взаимодействие с удалёнными серверами реализовано через использование динамической библиотеки C++ с интерфейсом JNI. Решены проблемы монтирования удалённых файловых систем на разных версиях ОС Android с помощью технологий Content Provider, Storage Manager и запуска отдельного медиа сервера на Android устройстве. Реализованы встроенные средства просмотра файлов для следующих MIME-типов: text/*, image/*, application/pdf, audio/*, video/*. Просмотр файлов осуществляется с ограничениями, которые накладывают старшие версии ОС Android и сторонние библиотеки. Файловый сервер реализует интерфейс монтируемой файловой системы посредством вызовов к файловой системе Android и запускается через Foreground Service.

УДК: 004.454

К. В. Забелин (4 курс бакалавриата)
Т. В. Коликова, ст. преподаватель

РЕАЛИЗАЦИЯ IPC ДЛЯ РАБОТЫ В ВИРТУАЛЬНОМ ОКРУЖЕНИИ

На сегодняшний день существует огромное количество технологий виртуализации разного рода. Начиная от виртуальных машин и заканчивая легковесными контейнерами. Технологии виртуализации дают множество различных преимуществ: безопасность, повышение изоляции, оптимальное управление ресурсами, повышение качества администрирования, возможность легко создавать специальное окружение. Существуют системы, состоящие из целых кластеров виртуальных машин или контейнеров. При

разработке ПО для таких систем часто поднимается вопрос взаимодействия программ, находящихся на разных виртуальных машинах. Цель данной работы изучить возможные способы такого взаимодействия и то каким образом его можно улучшить.

Перед тем как начать стоит разобраться в вопросах терминологии. ПО которое позволяет создать виртуальное окружение называется – гипервизор. Это программа, которая управляет ресурсами и распределяет их между всеми виртуальными машинами. Машина на которой запускается гипервизор, и которая предоставляет ресурсы, называется – хост. ОС, которая запущена внутри виртуальной машины называется гостевой ОС. Сейчас так же широко развивается направление контейнеризации. Основное отличие контейнера от виртуальной машины, то что контейнер эмулирует лишь пространство пользователя, в то время как на виртуальной машине так же запускается пространство ядра.

Когда появляется задача настроить процесс взаимодействия между двумя программами, находящимися на разных операционных системах первым что, приходит в голову это взаимодействие по сети. Современные технологии виртуализации позволяют эмулировать различные виртуальные устройства для гостевых операционных систем. Таким образом возможна эмуляция сетевых устройств и настройка сети между гостевыми системами и хостом. Тот факт, что сеть работает в рамках одной физической машины позволяет оптимизировать обработку сетевых пакетов. Однако, сетевой трафик все равно проходит значительную часть сетевого стека, что делает использование сети не самым быстрым способом обмена данными в виртуальном окружении. Но при этом обладает достаточной гибкостью и позволяет использовать все преимущества обычной физической сети.

Работу гипервизора можно условно разделить на две части: управление ресурсами, такими как RAM, CPU, Storage и эмуляция различных устройств. Для примера рассмотрим стандартный гипервизор в Linux – KVM. KVM это гипервизор реализованный в пространстве ядра. Однако, KVM не позволяет эмулировать устройства, эту работу на себя берет QEMU. QEMU – это ПО для эмуляции виртуальных устройств, которое запускается в пространстве пользователя. Таким образом каждая гостевая машина в Linux при использовании QEMU/KVM это отдельный процесс.

В Linux при использовании QEMU/KVM разновидность видов взаимодействия процессов на хосте и гостевой системе может быть шире. Концепция “everything is file” позволяет добиться большей гибкости. QEMU помимо PCI устройств может предоставлять гостевой системе так называемые character devices (символьные устройства), которые дают возможность посимвольного обмена информацией. QEMU позволяет эмулировать подобного рода устройства на гостевой системе и использовать в качестве backend символьные устройства с хоста. Что позволяет использовать одни и те же устройства как программам на хосте, так и программам в гостевой системе.

Однако, подобного рода устройства приспособлены лишь для взаимодействия двух процессов. При необходимости связать между собой большее число процессов, необходимо создавать новое устройство для каждой пары процессов. В больших системах, которые состоят из множества виртуальных машин и процессов как внутри них, так и на хосте это может стать проблемой. Большое количество устройств увеличивает нагрузку на QEMU и на процессы, которые их используют, а также увеличивает расход памяти.

Для того чтобы избежать данных недостатков, в данной работе рассматривается следующий вид IPC – Shared Queue Messages. В основе данного вида IPC лежит классическая очередь сообщений. Но её преимущество в том, что достаточно лишь одного такого символьного устройства для корректной связи неограниченного количества процессов. Ниже приведена схема как подобное взаимодействие может выглядеть в рамках виртуального окружения.

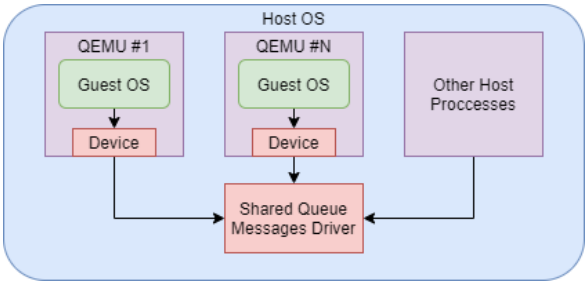


Рисунок 1

Необходимо реализовать новый драйвер символьного устройства. Данный драйвер должен работать на хосте. Для того чтобы процессы в гостевых системах могли так же использовать данное устройство необходимо реализовать эмуляцию нового типа символьного устройства в QEMU.

Внутреннее устройство подобного драйвера может выглядеть следующим образом:

– Реализуется глобальный кольцевой буфер для хранения сообщений. Каждое сообщение состоит из четырех полей: от какого процесса отправлено данное сообщение, кому оно отправлено, указатель на данные и их размер
– Для каждого нового процесса создается очередь в виде связанного списка, где каждый элемент очереди указывает на сообщение в кольцевом буфере, которое должен получить данный процесс. Это необходимо для быстрого получения новых сообщений.
– Глобальная хеш-таблица где ключом является имя процесса, а значением указатель на очередь данного процесса. Это поможет снизить время отправки сообщений.

Подобная реализация увеличит размер передаваемых данных, за счет того, что необходимо вместе с сообщением передавать имя отправителя и получателя. Однако, это позволит множествам процессов использовать одно устройство для взаимодействия, снизив нагрузку на память и CPU.

ЛИТЕРАТУРА

1. Uchit Gandhi Mr., Mitul Modi Mr., Mitali Raval Ms., Paavan Maniar Mr., Narendra Patel Dr., Kirti Sharma Prof, Distributed Virtualization Manager for KVM Based Cluster, Procedia Computer Science, 2018. P. 182-189

УДК 004.051

Д. А. Коптев (2 курс магистратуры)
Т. В. Коликова, ст. преподаватель

ПОВЫШЕНИЕ МАСШТАБИРУЕМОСТИ СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ ПУТЕМ РЕАЛИЗАЦИИ СТЕКА ПРОТОКОЛОВ SCSI В ПРОСТРАНСТВЕ ПОЛЬЗОВАТЕЛЯ

Введение. Системы хранения данных (СХД) обеспечивают эффективное и надежное хранение больших объемов информации, что в настоящее время необходимо практически любой корпорации. Взаимодействие серверов СХД с пользователями осуществляется с помощью протоколов передачи данных. Большое распространение получил стандарт iSCSI [1], обеспечивающий управление хранилищами и передачу данных на большие расстояния с помощью стека TCP/IP. iSCSI инкапсулирует команды и наследует архитектуру стандарта SCSI. Программное обеспечение СХД, как правило, базируется на операционных системах с ядром Linux, для которых существует несколько некоммерческих реализаций iSCSI и SCSI. В рассматриваемом проекте был выбран драйвер SCST [3], так как данный продукт наиболее полно реализует функциональность стандарта.

С учетом накладных расходов, зачастую присутствующих в программном обеспечении, компоненты которого работают в пространстве ядра [4], был выдвинут тезис о том, что полноценное использование функционала SCST может оказывать пагубное влияние на масштабируемость СХД. Таким образом, *цель работы* – повышение масштабируемости СХД путем частичной реализации стека протоколов SCSI в пространстве пользователя. Для достижения поставленной цели были выполнены следующие *задачи*:

1. Анализ предметной области и существующих решений;
2. Формирование концепции переноса логики SCSI в пространство пользователя;
3. Проектирование архитектуры модифицируемой части системы;
4. Программная реализация проекта;
5. Снятие метрик и анализ результатов.

Основная часть. Протокол SCSI описывает клиент-серверную модель взаимодействия. Клиент именуется инициатором, а сервер – целевым устройством. Для унификации доступа инициаторы могут объединяться в группы. Устройства хранения абстрагируются логическими единицами. Доступ группы инициаторов к устройству обеспечивается путем создания отображения логической единицы, характеризуемого локальным для группы номером [2].

Реализация SCSI, драйвер SCST, включает в себя модули, отвечающие за разные типы фабрик и устройств. Модули фабрик характеризуют особенности передачи данных по сети (iSCSI, Fiber Channel [5]). Модули типов устройств, например, файловые или блочные, определяют специфику работы с ними. Обобщенная архитектура SCST представлена на рис.1.

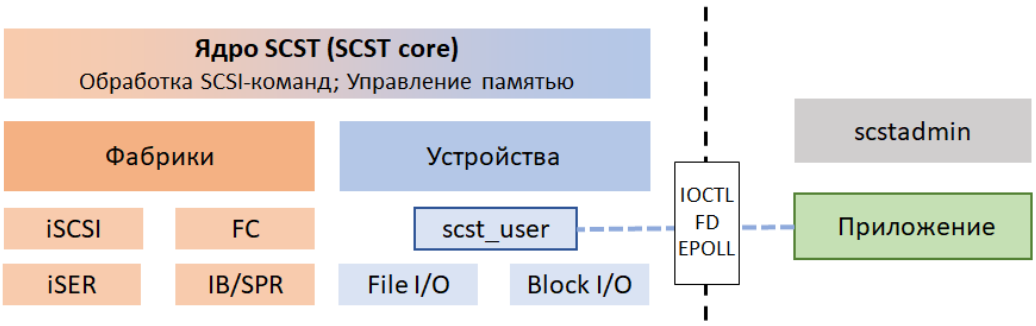


Рисунок 1 – Обобщенная архитектура SCST

Важной особенностью SCST является поддержка устройств в пространстве пользователя – модуль scst_user. Драйвер делегирует приложению ответственность за обработку чтения, записи и других специфичных служебных команд путем их перенаправления в пространство пользователя с использованием IOCTL и файловых дескрипторов Linux [3]. Описанный принцип наиболее важен в контексте рассматриваемого проекта. ПО СХД содержит собственный модуль работы с операциями ввода-вывода, так как продукт должен поддерживать репликацию и дедупликацию. Кроме того, СХД должна обеспечивать устойчивость к ошибкам и высокую доступность. Сохранение конфигурации в постоянную память делает возможным ее полное восстановление после сбоев.

В процессе разработки проекта были полностью изучены исходный код и архитектура SCST. Исследование показало, что некоторые инженерные решения, применяемые в реализации драйвера, действительно оказывают пагубное влияние на производительность и масштабируемость системы. В частности, многие конфигурационные операции в драйвере влекут за собой глобальную блокировку функциональности, в том числе операций ввода-вывода. Кроме того, для доступа к сущностям SCST часто использует линейные алгоритмы.

С целью оценки влияния размера конфигурации на скорость ее создания был проведен простейший тест, заключающийся в цикличном добавлении логических томов в систему. Далее была выполнена оценка времени, затрачиваемого ядром для создания каждой сущности, которая показала наличие экспоненциальной зависимости от количества томов в системе. Так, добавление тома из первой тысячи занимает примерно 100 мс, но уже для 30 тысяч время возрастает в 10 раз. Данная ситуация говорит о потенциально плохой масштабируемости системы.

Подход к решению описанной проблемы довольно прост и основывается на том факте, что большая часть конфигурации дублируется в ПО СХД. Идея заключается в организации логики, отвечающей за создание большинства конфигурационных сущностей, и реализации обработки SCSI-команд, относящихся к ним, в пространстве пользователя. В этом случае SCST будет использоваться в основном в качестве туннеля для передачи команд от хостов. Подход подразумевает отключение конфигурирования в ядре тех сущностей, которые представлены в системе в большом количестве. Таковыми являются инициаторы, их группы и логические тома с отображениями. Реализация обработки SCSI-команд в пространстве пользователя для данных сущностей тривиальна, так как вся необходимая информация предоставляется стандартом [2].

Механизм туннелирования состоит в создании «служебных» отображений томов – туннелей – в ядре по количеству физических ядер процессора. Алгоритм в пространстве пользователя заключается в поиске «реальной» логической единицы, которой предназначается переданная по «служебному» туннелю команда. Модуль использует эффективные хэш-таблицы с 64-битным ключом, не применяя алгоритмы линейного поиска.

Описанный подход обладает существенными преимуществами. Во-первых, за счет значительного сокращения объема логики, выполняемой драйвером в пространстве ядра, происходит уменьшение временных затрат, сокращение использования памяти в ядре и снижение вероятности возникновения критических ошибок. Во-вторых, исчезает необходимость в расширении и модификации кода драйвера. Данный фактор имеет большое значение, так как SCST распространяется по лицензии GPL [3], и все его модификации должны быть опубликованы. Кроме того, появляется возможность устранить глобальные блокировки во время конфигурирования системы. Например, были использованы отдельные механизмы блокировки RW-lock для каждой группы инициаторов, обеспечивающие поддержку параллельного конфигурирования. В случае с SCST, помимо необходимости публиковать модификации, данная задача являлась бы существенно более трудоемкой. Что более важно, становится возможным использование более эффективного метода работы с памятью, заключающегося в ее предварительном распределении на старте системы. Кроме того, производится упаковка структур с выравниванием и оптимальное размещение полей.

На момент написания статьи велась активная работа над реализацией описанного подхода. Промежуточные результаты подтверждают высокий потенциал разработанной концепции. На рис.2 показано сокращение времени создания томов прототипом по сравнению с исходным решением. Время создания последнего тома из 30 тысяч сократилось с 1 с. до 320 мс. Дальнейшие планы направлены на сведение к нулю времени, затрачиваемого ядром на создание томов.

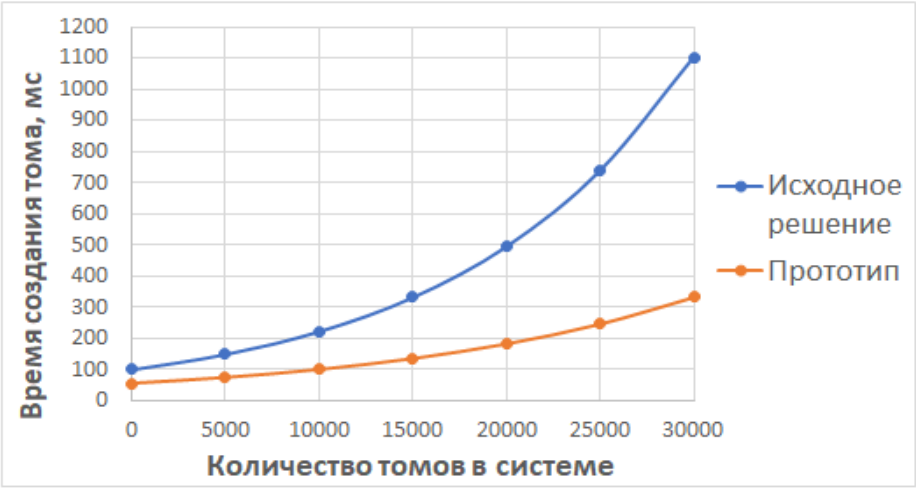


Рисунок 2 – График времени создания томов для исходного решения и прототипа

ЛИТЕРАТУРА

1. Internet Small Computer Systems Interface (iSCSI), RFC 3270, 2004.

2. SCSI Architecture Model - 4 (SAM-4), ANSI INCITS 447, 2008.
3. Сайт продукта SCST. URL: http://scst.sourceforge.net (дата обращения: 05.10.2019).
4. Передача и обработка SCSI-пакетов в области приложения систем хранения данных с использованием драйвера Linux-IO, Д.А. Коптев, 2018, с.36-37.
5. Fibre Channel Framing and Signaling (FC-FS), ANSI INCITS 373, 2003.

УДК 004.45

М. Б. Литвинов (4 курс СПбПУ)
Н. В. Воинов, к.т.н., доцент

СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ЗАЯВОК ПОЛЬЗОВАТЕЛЕЙ ТЕЛЕКОММУНИКАЦИОННЫХ УСЛУГ

Целью работы является решение следующей проблемы: компания “Deutsch Telekom” предоставляет своим клиентам телекоммуникационные сервисы. В случае возникновения проблемы, клиент обращается в техподдержку, оставляя заявку, которую затем рассматривает оператор службы техподдержки. В действительности, некоторая часть таких заявок остаётся необработанной, что приводит к недовольству клиентов.
Есть предположение, что заявки пропадают из-за некоторых комбинаций значений, которые обрабатываются некорректно. В таком случае, мы можем попробовать вычислить зависимость, которая приводит к данной ошибке.

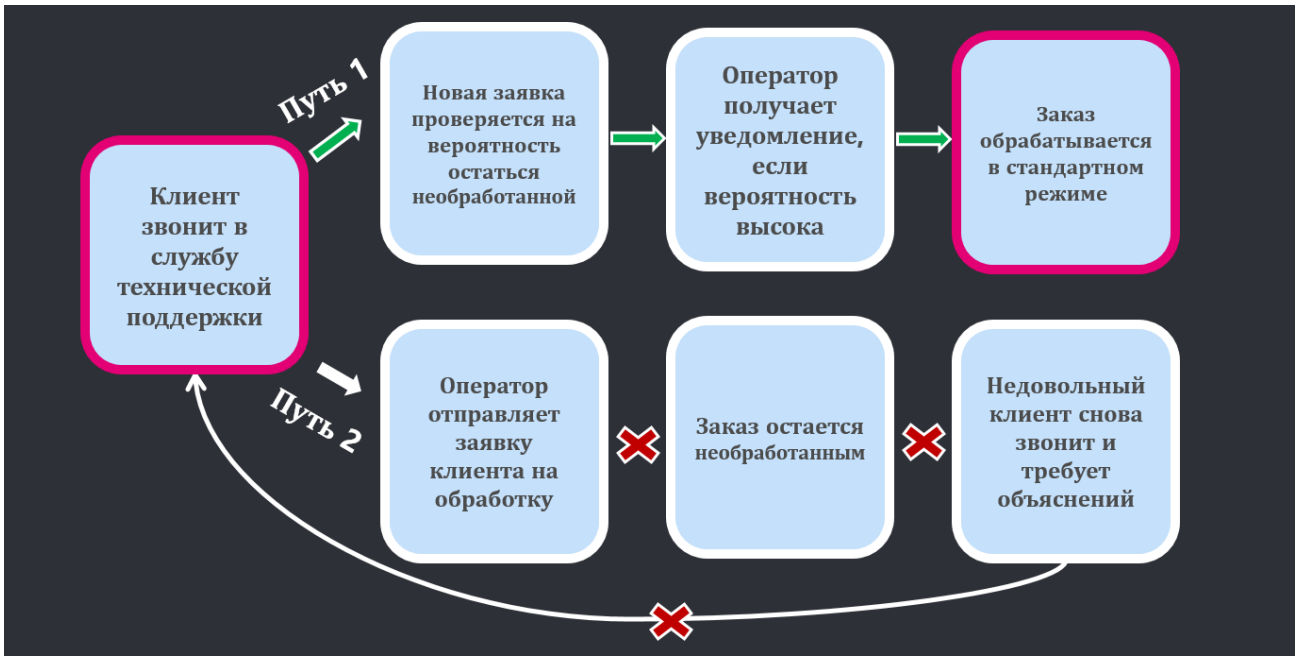


Рисунок 1 – Система предотвращения пропажи заявок клиентов телекоммуникационных услуг.

Особенности реализации системы управления:

Предлагается создать алгоритм, предотвращающий ситуацию, в которой заявка остаётся необработанной. Для этого необходимо, чтобы все новые заявки проверялись регулярно. В случае выявления заявок, имеющих вероятность остаться необработанными, оператор техподдержки получает оповещение.

Для предсказания вероятности пропажи заявки, будут использованы алгоритмы классификации и/или нейронные сети.

УДК 004.052.3

В. А. Максимчук (4 курс бакалавриата)
И. В. Никифоров, к.т.н., доцент

РАЗРАБОТКА СИСТЕМЫ МАСШТАБИРОВАНИЯ РЕПЛИК ПРИЛОЖЕНИЙ С ЦЕЛЬЮ УВЕЛИЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ УЗЛОВ КЛАСТЕРА KUBERNETES

Введение. В наши дни ежесекундно обрабатывается огромное количество информации. В связи с этим появляется задача реализации систем, способных выдержать подобные нагрузки. Благодаря контейнерной виртуализации – создания изолированных окружений на базе одного физического устройства на уровне процессов операционной системы, появляется возможность распределять нагрузки, увеличивая количество обрабатываемой информации без сбоев. Управление группами контейнеров осуществляется оркестраторами, которые так же учувствуют в распределении нагрузки. В современной индустрии одним из самых популярных оркестраторов контейнеров является Kuberenetes, выпущенный Google в 2014 году, как Open-source проект, на основе предшественников Borg и Omega. Данный инструмент позволяет производить развертывание кластера, а также включает в себя средства для управления нагрузкой на узлы [1]. Одним из таких является Horizontal Pod Autoscaler (HPA), который на основе метрик и заданных параметров осуществляет масштабирование реплик приложения, увеличивая их количество или уменьшая. Метрики, используемые данным инструментом, собираются и хранятся в Metric Server, самого Kubernetes, однако представляют собой лишь базовый набор показаний CPU. Таким образом для получения дополнительных метрик необходимо подключить сторонний инструмент мониторинга, при помощи Custom Metric API [2]. Одним из таких является Prometheus, который представляет собой базу данных временных рядов – данные хранятся по временной шкале [3]. Главным его отличием является то, что он сам собирает данные с целевых объектов с заданной частотой.

Цели и задачи. Целью данной работы является реализация подхода повышения параметра отказоустойчивости кластера при различных нагрузках. Данная цель достигается за счет решения следующих задач:

–Исследование HPA Kubernetes
–Подключение Prometheus к кластеру Kubernetes
–Разработка алгоритма масштабирования узлов кластера
–Реализация данной системы автомасштабирования

Реализация. Исследование и анализ HPA Kubernetes показал, что у него присутствуют некоторые проблемы, возникающие из-за архитектурных особенностей системы (рис. 1). Главная из них заключается в скорости реагирования на событие. После наступления события формируется метрика, передается в Metric server, далее в HPA, который после ее анализа начинает репликацию подов, однако развертывание одного пода занимает до 90 секунд, не учитывая запуск приложения, а, если необходимо дополнительно развернуть instance, это

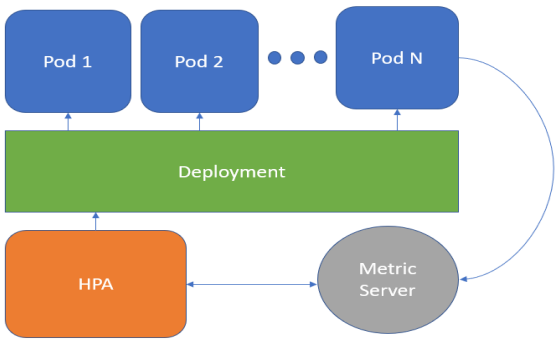


Рисунок 1 – Архитектура системы масштабирования Kubernetes

может занимать до 12 минут [4]. Следовательно, появляется задержка масштабирования

системы, которая приводит к некорректной работе и может вызывать сбои. Однако, нагрузка не может постоянно расти, а имеет скачковый вид и при данной конфигурации работы системы это приводит к двум периодам: задержки развертывания, вызывающей сбои, и снижения количества подов, в связи с их нерационально большим количеством. Данная проблема решается в Kubernetes путем наличия двух констант роста, то есть кластер не масштабируется в необходимое количество подов за один момент времени, а делает это пошагово, и наличием регулируемого окна стабилизации, которое не дает HPA уменьшать количество реплик до следующего скачка. Однако данные параметры реализованы для кластера целиком, что не позволяет выполнять на кластере разные задачи с оптимальной производительностью [5]. Данное решение не решает проблему задержки развертывания при первом скачке нагрузке, а также порождает ситуацию, когда ресурсы системы, тратятся на ожидание следующего скачка.

Разрабатываемый алгоритм направлен на решение вышеупомянутых проблем. При подключении Prometheus, согласно архитектуре системы (Рис. 2), появляется возможность отслеживать дополнительные метрики необходимые для определенной системы, при помощи которых, поды будут начинать масштабирование до получения максимальной допустимой нагрузки на них, а также регулярные нагрузки будут отслеживаться и запоминаться системой для дальнейшего применения [6]. Далее необходимо разделить алгоритм для различных задач, так как при определенных условиях снижение количества реплик необходимо синхронно со снижением нагрузки на кластер, например, при обработке запросов на сервис, развернутый на кластере, а при других оптимальным решением будет регулируемое окно стабилизации для каждого набора реплик.

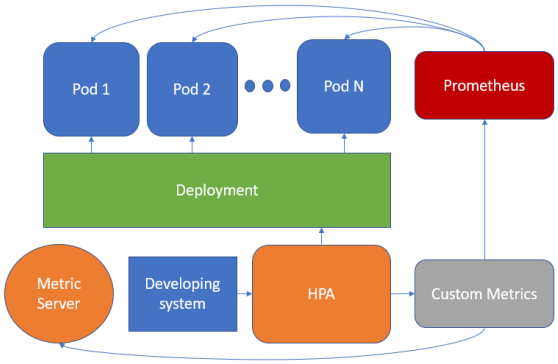


Рисунок 2 – Архитектура разрабатываемой системы масштабирования

Будущие планы. На данный момент система реализована в виде прототипа на платформе OpenStack, на развернутом кластере Kubernetes. В дальнейшем планируется подбор параметров и метрик для оптимальной работы системы, а также ее тестирование и сравнение с существующими решениями и HPA.

ЛИТЕРАТУРА

1. Burns B., Grant B., Oppenheimer D. Borg, omega, and Kubernetes // RightLink. New York: Association for Computing Machinery. Vol. 14. 2016. 24 p.
2. Kubernetes docs [Электронный ресурс]. URL: https://kubernetes.io/docs/home/ (дата обращения: 17.03.2020)
3. Naqvi S., Yfantidou S. Time Series Databases and InfluxDB. Belgium: Universite libre de Bruxelles, 2017. 45 p.
4. Mohamed A. Kubernetes Autoscaling 101: Cluster Autoscaler, Horizontal Autoscaler, and Vertical Pod Autoscaler [Электронный ресурс]. URL: https://www.magalix.com/blog/kubernetes-autoscaling-101 (дата обращения: 15.03.2020)
5. Configurable Horizontal Pod Autoscaler [Электронный ресурс]. URL: https://blog.postmates.com/configurable-horizontal-pod-autoscaler-81f48779abfc (дата обращения: 14.03.2020)
6. Casalicchio E., Perciballi V. Auto-scaling of containers: the impact of relative and absolute metrics. New Jersey: Institute of Electrical and Electronics Engineers Inc, 2017. P. 207-214

УДК 004.772

Я. В. Опарко, С. А. Вязов (4 курс бакалавриата, СПбПУ)
Б. М. Медведев, к.т.н., доцент

ПЕРЕДАЧА ФАЙЛОВ ПО ДИНАМИЧЕСКИ ОРГАНИЗУЕМОЙ БЕСПРОВОДНОЙ ЛОКАЛЬНОЙ СЕТИ

Часто возникает ситуация, когда необходимо организовать обмен файлами между ноутбуками студентов в условиях отсутствия доступной беспроводной сети WiFi. Использование сотовых сетей существенно увеличивает время передачи и имеет высокую стоимость при больших размерах файлов. В качестве решения можно использовать технологию Virtual WiFi, которая позволяет развернуть виртуальную точку доступа на одном из компьютеров. Для безопасности сети используется протокол защиты WPA2.

Самыми известными приложениями для передачи файлов между устройствами являются SHAREit и ShareMe. Приложение ShareMe не поддерживает передачу файлов с компьютера на компьютер (поддерживается только режим смартфон-смартфон и смартфон-компьютер). Приложение SHAREit имеет версию для ПК под управлением Windows и OS X. Однако, передать файл между двумя компьютерами у авторов статьи не получилось. Процесс передачи останавливался на стадии подключения к WiFi сети. Таким образом, для передачи файлов между двумя компьютерами без использования интернета и USB-накопителя необходимо разработать приложение с использованием Virtual WiFi.

Целью работы является создание клиент-серверного приложения, обеспечивающее передачу файлов через беспроводную локальную сеть WiFi, образованную двумя взаимодействующими компьютерами.

Решаемые задачи:

1. Разработка архитектуры приложения.
2. Разработка программного обеспечения, работающего под управлением ОС Windows 7,8,10 и Ubuntu версии от 12.04.
3. Оценка скорости передачи файлов разрабатываемым приложением.

Разработанное приложение состоит из 5 модулей: модуль отправки файла, модуль приема файла, модуль создания точки доступа, модуль подключения и модуль взаимодействия с пользователем (см. рисунок.1). При запуске программы первым активируется модуль взаимодействия с пользователем. Задачей этого модуля является обработка действий пользователя и запуск других модулей (модуля создания точки доступа или модуля подключения). Данный модуль активен на протяжении работы всей программы и завершается только вместе с завершением программы. Модуль создания точки доступа отвечает за развертывание виртуальной Wi-Fi сети. Запускается из модуля взаимодействия с пользователем. В случае успешного завершения работы запускает модуль получения файла.

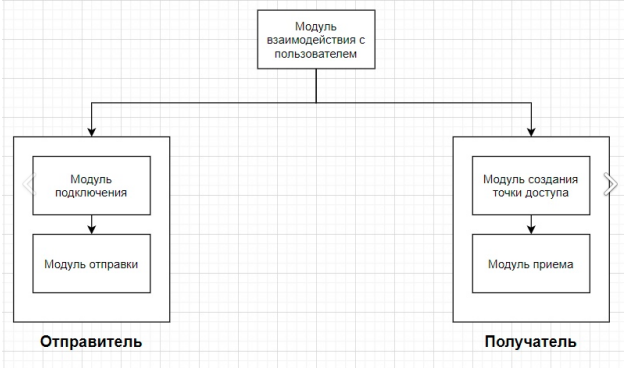


Рисунок 1– Архитектура ПО

Модель OSI

Уровень	Использование
Прикладной	Custom
Транспортный	TCP
Сетевой	IPv4
Канальный	WLAN
Физический	Радиоканал

Рисунок 2 – Стек протоколов

В модуле получения файла реализованы все функции для приема пакетов файла. Модуль подключения к сети запускается из модуля взаимодействия с пользователем, как только пользователь выбирает файл для отправки. В случае успешного завершения запускается модуль отправки. Модуль отправки – отвечает за передачу файла, запускается из модуля подключения.

Рассмотрев доступные протоколы прикладного уровня, было принято решение разработать собственный протокол, который был бы прост в реализации и обеспечивал более высокую скорость передачи данных по сравнению, например, с протоколом FTP. Разработанный протокол предусматривает, что получатель, который работает в режиме сервера, ожидает подключения клиента. Отправитель, который работает в режиме клиента, подключается к получателю и отправляет пакет, содержащий имя файла и его размер, затем отправитель ожидает согласие получателя. Получатель отправляет отправителю решение пользователя и переходит в режим приема данных в случае положительного ответа пользователя. Когда отправитель получает подтверждение от получателя, то немедленно начинает отправку файла. Файл отправляется пакетами по 1450 байт каждый (размер пакета выбран в соответствии с MTU). Для контроля доставки пакетов через сеть WiFi используется протокол TCP.

На рисунке 3 представлена диаграмма последовательности для сценария передачи файла. Две копии программы работают в режимах отправителя и получателя на двух разных устройствах. Копия программы в режиме получателя разворачивает точку доступа Wi-Fi, выводит пароль сети на экран. Другая копия программы в режиме отправителя подключается к сети Wi-Fi. Затем начинается обмен файлами в соответствии с вышеописанным протоколом.

По результатам проделанной работы можно сделать следующие выводы:

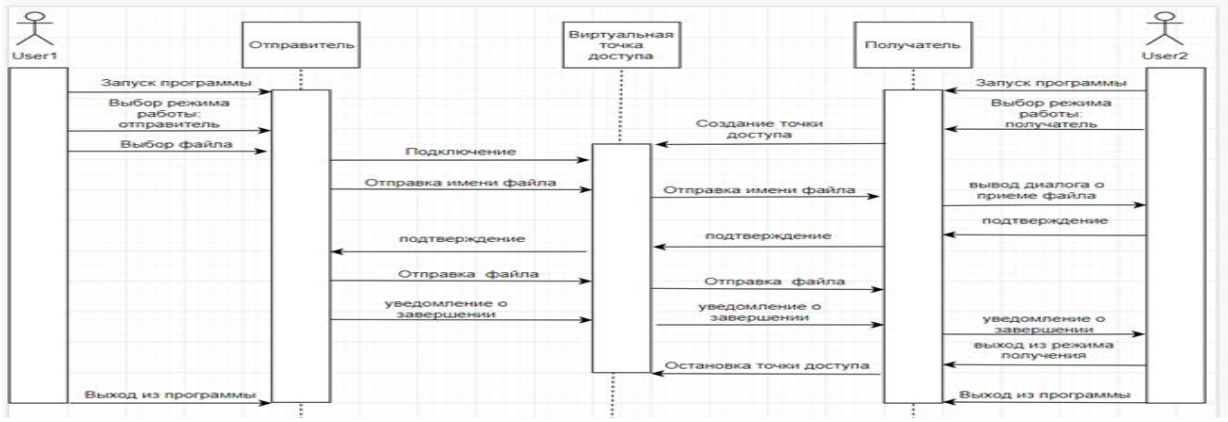


Рисунок 3 – Диаграмма последовательности

1. Разработано приложение, обеспечивающее передачу данных между компьютерами на высокой скорости по технологии virtual WiFi для ОС windows 7,8,10 и Ubuntu версии от 12.04.

2. Во время тестирования программы была выявлена сильная зависимость скорости передачи данных от аппаратных и программных средств устройств. Так, если оба устройства соответствуют стандарту WiFi 5 и адаптеры имеют поддержку установленной операционной системы, то скорость передачи превышает 100 Мб/с. Для режима WiFi 4 средняя скорость передачи файла уменьшается до 40 Мб/с. В ряде случаев было обнаружено существенное уменьшение средней скорости передачи до 2 Мб/с в тех же условиях тестирования. Такое уменьшение скорости зарегистрировано для Wi-Fi адаптера Realtek rtl8723de, который изначально не поддерживается операционной системой Ubuntu (установлен драйвер производителя).

ЛИТЕРАТУРА

1. Кручинин, С.В. Семиуровневая модель OSI/ISO и стек протоколов TCP/IP: исследование взаимоотношения и интерпретации [Текст] / С.В. Кручинин // Journal of Scientific Research Publications. – 2015. - № 5 (25). – С. 115-120.

УДК 004.42

А. А. Петров (4 курс бакалавриата)
И. А. Селин, ст. преподаватель

РАЗРАБОТКА ПРОГРАММЫ ПО ОБЪЕДИНЕНИЮ ОБЛАЧНЫХ ХРАНИЛИЩ ДАННЫХ

В настоящее время повсеместно используются системы хранения данных (СХД) [1], представленные как облачными системами хранения, например, Яндекс.Диск [2], Microsoft OneDrive или Google Drive, так и промышленными системами хранения данных. При этом облачные СХД предоставляют API для взаимодействия с ними, что позволяет реализовать приложения-клиенты по взаимодействию с ними [3, 4].

При этом стоить отметить, что по ряду причин пользователям может не хватать места на имеющемся облачном хранилище или же у него есть ещё и другое хранилище, с которым он хотел бы объединить свое хранилище. Эти обстоятельства привели к необходимости создания приложения, которое предоставило бы возможность программного объединения имеющихся облачных хранилищ и монтирования их в файловую систему пользователя в виде каталога (папки). При этом этот каталог использует свободное пространство на всех имеющихся облачных хранилищах.

Такое приложение должно обеспечивать надежное и эффективное взаимодействие с разными СХД, такими как: Яндекс.Диск [2], Google Drive, Облако Mail.Ru а также не облачными (аппаратными) СХД, например, Dell EMC Unity, Xtremio [5], IBM DS8900F и другими. Для эффективного использования имеющегося объема памяти программа должна разделять получаемый файл на части (возможно, несколько раз) и сохранять их в соответствии с алгоритмом на разных хранилищах, соответственно, при запросе пользователем файла программа должна получить необходимую информацию из всех систем хранения, построить исходный файл и предоставить его пользователю в исходном состоянии. Так как пользователь имеет также и прямой (в обход нашего приложения) доступ к своим СХД, риск потери одной или нескольких частей исходного файла возрастает. Это является значимой проблемой для разрабатываемого приложения, т.к. отсутствие даже одной части исходного файла может не позволить приложению корректно восстановить исходный файл. В свою очередь это приводит к необходимости организовать хранение нескольких копий одной и той же части в нескольких хранилищах

Реализованный на данный момент прототип такого приложения обеспечивает подключение к облачному хранилищу посредством использования REST API и объединение нескольких таких хранилищ (разные аккаунты имеют разные хранилища), а также объединение с любыми каталогами уже находящимися (смонтированными) в файловой системе. Это позволяет объединять вместе практически любые существующие системы хранения данных, как аппаратные, так и облачные за счет того, что часть из них может быть смонтирована в файловую систему при помощи сторонних утилит, например, davfs2 [6] или ExpandDrive [7]. При этом разработанное приложение само предоставляет REST API для реализации возможности взаимодействия с ним посредством HTTP запросов. Это позволит в будущем реализовать для него графический интерфейс, что увеличит удобство использования. Так как мы хотели бы, чтобы пользователь имел возможность использовать наше хранилище так же, как и любой другой каталог в файловой системе, было решено предоставить протокол WebDAV, который предоставляет пользователю сетевую файловую систему, которую он может смонтировать на свой компьютер, воспользовавшись сторонними приложениями, например, уже указанным выше davfs2. [6]

Одной из проблем в ходе реализации был выбор алгоритма резервного копирования блоков. В итоге был выбран вариант создания нескольких копий блока на других доступных хранилищах, при этом на одном хранилище должно храниться не более 1 копии блока, что

означает, что если пользователь подключил только 1 хранилище, то резервное копирование осуществляться не будет.

Полученное приложение позволило объединить несколько хранилищ, что дало возможность записывать на такое объединенное хранилище файлы, которые по объему больше, чем любое из имеющихся хранилищ. Ещё одним преимуществом является возможность объединять хранилища, предоставляемые разными провайдерами (масштабирование производится реализацией всего 1 интерфейса). Также его можно использовать для временного расширения доступного объема памяти для хранения, в случаях, когда у пользователя уже имеется СХД.

Недостатком такого приложения является необходимость дублирования файлов, что ограничивает возможный объем использования по памяти до 50% от имеющегося объема. Но в случае, когда несанкционированный доступ к данным будет ограничен, а сами хранилища гарантируют отсутствии потери данных, репликацию (копирование блоков) можно отключить, что позволит ещё больше увеличить эффективность приложения.

ЛИТЕРАТУРА

1. Techcrunch // Google updates Drive with a focus on its business users, 2017. https://techcrunch.com/2017/03/09/google-drive-now-has-800m-users-and-gets-a-big-update-for-the-enterprise/
2. Яндекс Помощь // Что такое Яндекс.Диск, 2020. https://yandex.ru/support/disk/
3. Google Drive API // Главная, 2020. https://developers.google.com/drive
4. Яндекс // API Яндекс.Диска, 2020. https://yandex.ru/dev/disk/api/concepts/about-docpage/
5. Dell Technologies // Системы хранения данных, ускоряющие переход от аналитики к инновациям, 2020. https://www.dellemc.com/ru-ru/storage/data-storage.htm
6. ArchLinux // Davfs2, 2020. https://wiki.archlinux.org/index.php/Davfs2
7. ExpanDrive // Главная страница // Connect to any Cloud, 17.03.2020. https://www.expandrive.com/

УДК 004.042

Д. Сафронов, К. М. Стоноженко (4 курс бакалавриата)
И. В. Никифоров, к.т.н., доцент

АВТОМАТИЧЕСКАЯ БАЛАНСИРОВКА НАГРУЗКИ МЕЖДУ ПОТОКОВОЙ ОБРАБОТКОЙ ДАННЫХ И ВНУТРЕННИМИ ЗАДАЧАМИ КЛАСТЕРА С ИСПОЛЬЗОВАНИЕМ KUBERNETES

Введение. В последнее время доля систем, построенных на базе машинного обучения, значительно увеличивается. Это связано как с ростом мощности вычислительных ускорителей, так и с большим объемом исследований, проводимых в данной области [1].

Тем не менее включение нейросетевых модулей в отлаженные коммерческие сервисы все еще представляет собой крайне сложную задачу. Это связано с тем, что для успешной работы такого модуля необходим процесс обучения, на который по крайней мере должны быть выделены вычислительные ресурсы. А зачастую даже построена отдельная архитектура и применён дополнительный технологический стек.

Одним из примеров является попытка улучшения модели, с помощью изменения гиперпараметров. С одной стороны, в разрезе коммерческих рекомендательных систем даже 1-2% точности важны, так как они выливаются в привлекаемую аудиторию, а значит в доход (или убытки). С другой небольшие современные нейронные сети имеют уже порядка 200000 внутренних параметров [2]. Ввиду этой сложности математические алгоритмы, по которым можно аппроксимировать точность модели исходя из задаваемых параметров, только развиваются. Один из самых используемых способов улучшения точности – это простая проверка на различных входных параметрах, либо с учетом новых поступивших данных.

Второй пример – построение предсказывающих моделей для каждого отдельного пользователя. Например, широко обсуждаемой, но до сих пор нерешенной проблемой

является идентификация человека по логам, которые он оставляет на сайте. Это позволило бы гораздо быстрее определять взломы аккаунтов, а также другие неправомерные действия.

Таким образом *целью работы* является реализация системы аналитики данных, которая была бы способна обеспечить должную эффективность использования кластера в вышеописанных задачах.

Система должна поддерживать обработку потока поступающих заявок в реальном времени. Планируется, что обработка будет основана на применении алгоритмов глубокого машинного обучения. Также на том же кластере должны исполняться задачи по улучшению обученных моделей. При этом при отсутствии нагрузки на обработчики может быть реализована возможность уменьшения их количества на некоторых узлах кластера и передача высвобождаемых мощностей на другие внутренние процессы.

Для подобной системы лучше всего подходит архитектура на основе контейнеров. Их главное преимущество в том, что они могут обеспечить скрытие окружения, что соответственно позволяет запускать и удалять их быстрее, а также, что самое важное, автоматизировать этот процесс.

В современной индустрии одним из самых популярных инструментов для оркестрации большого количества контейнеров считается Kubernetes [3]. Он содержит средства для создания наборов контейнеров, способных контролировать количество реплик.

Это дает возможность, с одной стороны, эффективнее управлять потоком данных. С другой, если организовать разбиение исторической информации на батчи, можно также ускорить их обработку перед задачами обучения.

Для организации взаимодействия коннектор-обработчик традиционно используют брокеры сообщений, такие как Kafka, Rabbit MQ. В нашем случае, так как еще понадобится дополнительно хранить историческую информацию, можно воспользоваться технологией Pravega [4], которая предлагает автоматическую автоматическое разбиение данных на уровни хранения. Таким образом можно организовать взаимодействие всех компонентов (рис. 2).

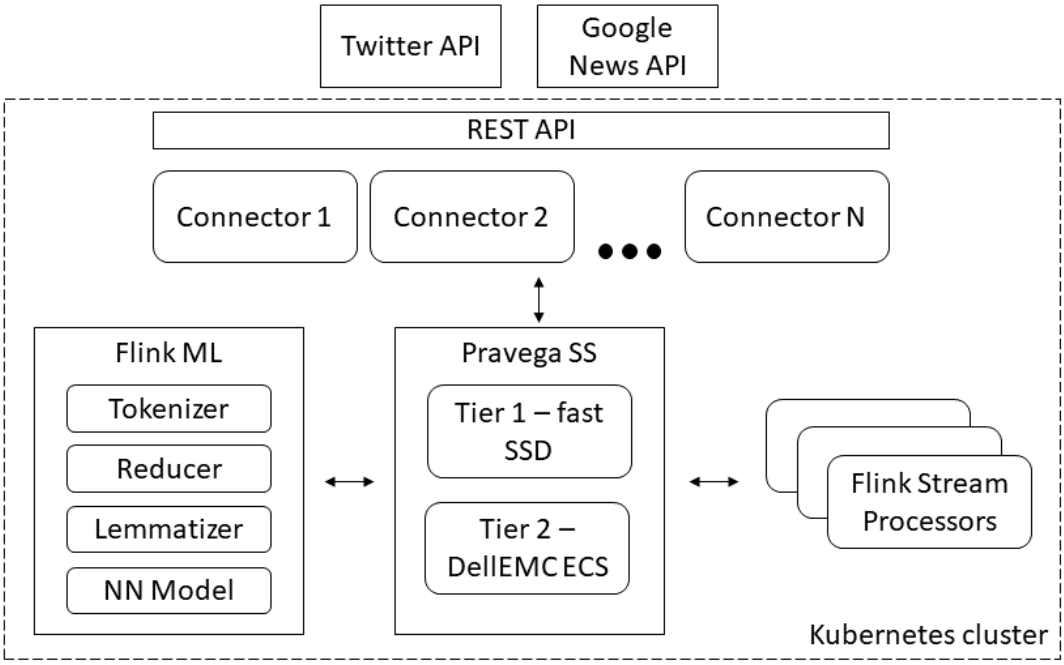


Рисунок 1 – Архитектура системы.

На сегодняшний день имеется достаточно много фреймворков для работы с большими данными, например, Hadoop, Spark, Storm. Однако в рамках данного проекта требуются инструменты работы с потоками данных в реальном времени. С этой задачей хорошо справляется Apache Flink. Кроме того, он имеет поддержку со стороны Pravega, что является большим плюсом.

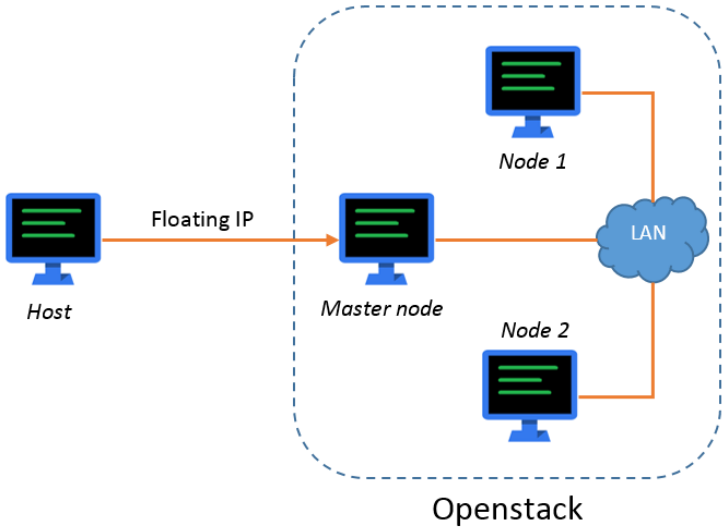


Рисунок 2 – Структура кластера.

На данный момент система реализована на уровне работающего прототипа, запущенного на платформе Openstack, согласно архитектуре, представленной на рис. 2. Средства, которые были использованы для развертывания, – Kubespray и Ansible.

Задачи по дальнейшему развитию системы сводятся к двум основным группам: тестирование и автоматизация. Планируется две основные группы тестов: модульные/интеграционные и нагрузочные.

ЛИТЕРАТУРА

1. D. Shewan, "10 Companies Using Machine Learning in Cool Ways," 12 August 2019. [Online]. Available: https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications.
2. D. Rao and B. McMahan, Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning, O'Reilly Media, 2019.
3. "Kubernetes docs," [Online]. Available: https://kubernetes.io/. [Accessed 2020].
4. "Pravega," [Online]. Available: http://pravega.io/. [Accessed 2020].

УДК 004.457

А. С. Шемякинская (4 курс бакалавриата)
И. В. Никифоров к.т.н., доцент

РЕАЛИЗАЦИЯ ПАТТЕРНА «ОПЕРАТОР» ДЛЯ ОТСЛЕЖИВАНИЯ СОСТОЯНИЯ ДИСКОВ В СИСТЕМЕ ОРКЕСТРАЦИИ КОНТЕЙНЕРОВ KUBERNETES

В настоящее время все большую популярность набирают системы оркестрации контейнеров для автоматического развертывания приложений в кластерной инфраструктуре. К таким системам относят: Kubernetes[1], Docker Swarm[2], Mesos[3].

Приложениям всегда требуется хранить данные на дисковых носителях. Например, базу данных, лог-файлы и т.д, а системам оркестрации контейнеров необходимо знать состояние дисков на голом железе для создания томов. Из этого следует, что системным администраторам и администраторам систем оркестрации контейнеров необходимо следить за состоянием дисковых носителей в системе.

Поэтому задача мониторинга дисковых носителей в системе является актуальной.

Целью работы является сокращения времени и трудоемкости анализа информации о дисковых носителях за счет внедрения «оператора» в систему оркестровки контейнеров. Эта цель достигается за счет решения следующих задач:

1) Осуществить обзор существующих программных средств мониторинга дисковых носителей.

2) Предложить подход для мониторинга в системе оркестрации контейнеров.

3) Реализовать программное средство для мониторинга дисковых носителей в системе оркестрации контейнеров.

4) Провести тестирование полученнего продукта на реальных системах.

5) Сравненить предложенный подход с существующими решениями.

В таблице 1 представлен результат обзора существующих подходов к мониторингу дисков.

Таблица 1 – Обзор существующих решений мониторинга дисковых носителей
(«+» - программное средство удовлетворяет критерию, «-» - программное средство не удовлетворяет).

Программное средство	Поддержка кроссплат-форменности	Наличие зависимости от оборудования	Поддержка графического интерфейса	Наличие бесплат-ной лицензии	Интеграция с системой оркестровки контейнеров Kubernetes
smartctl	-	-	-	+	-
ipmi	+	-	+	-	-
lsblk	-	+	-	+	-
hdparm	-	+	-	+	-
lshw	-	+	-	+	-

Из таблицы 1 можно сделать вывод, что существующие решения обладают рядом недостатков: отсутствие кроссплатформенности, интеграции с Kubernetes и графического интерфейса, поэтому в работе предлагается подход автоматизации анализа состояния дисковых носителей за счет реализации Kubernetes «оператора» инструмента мониторинга дисков.

Данный подход имеет ряд отличительных особенностей:

1) Автоматический и эффективный способ мониторинга дисков без создания дополнительной нагрузки на кластер Kubernetes;

2) Кроссплатформенность и независимость от используемого оборудования.

«Оператор» — это программное расширение Kubernetes, которое объединяет пользовательские ресурсы и пользовательские контроллеры [4]. Пользовательский ресурс (англ., Custom Resource) - это расширение API Kubernetes. В свою очередь ресурс в Kubernetes хранит коллекцию объектов API определенного вида. Например, встроенный ресурс pods содержит коллекцию объектов pod. Контроллеры (англ., Custom Controller) следят за заданными ресурсами и поддерживают их состоянием в соответствии с тем, что задал пользователь. Когда происходит какое-либо событие с ресурсом (CREATE, GET, DELETE и т.д.), «оператор» реагирует и выполняет определенное действие.

На рисунке 1 представлена архитектура «оператора» мониторинга дисков. Были реализованы два контроллера: контроллер инструмента мониторинга дисков, который собирает информацию о дисковых носителях, и контроллер ресурсов дисков, который создает пользовательские ресурсы в Kubernetes с информацией, полученной от первого контроллера. Реализация выполнена на языке Golang.

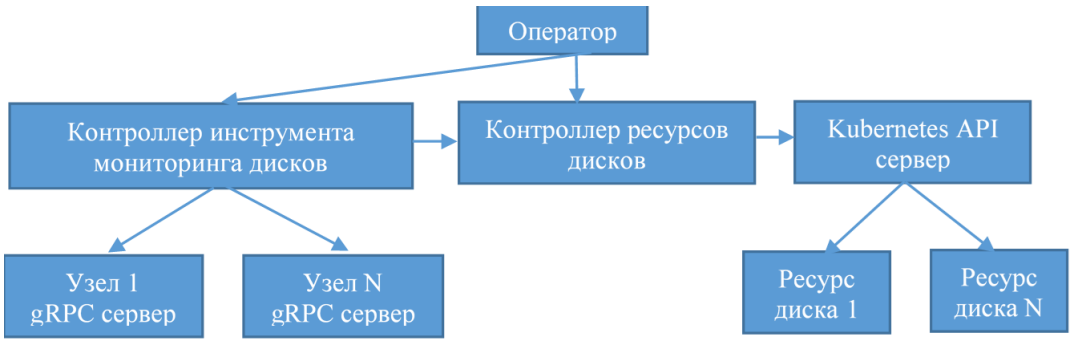


Рисунок 1 – Архитектура «оператора»

В таблице 2 представлен сравнительный анализ существующего программного обеспечение с подходом внедрения «оператора» в Kubernetes.
T_a – время ручного анализа информации, $T_в$ – время ввода команды,
$T_о$ – общее время анализа в тестируемой системе

Таблица 2 – Сравнительный анализ

Время\Решение	Утилита smartctl	ipmi	Подход использования «оператора»
$T_в$ (секунд)	5	7	5
$T_а$ (секунд на один диск)	10	2	2
$T_о$ (секунд) в тестируемой системе	125	31	29

Таким образом был реализован «оператор» для Kubernetes, который предоставляет информацию о дисковых носителях в кластерной инфраструктуре. Данный подход требует в 4 раза меньше времени, чем использование утилиты smartctl и на 2 секунды меньше, чем ipmi. Однако полученный подход имеет ряд недостатков по сравнению с существующими. Например, утилита smartctl и реализация ipmi предоставляют больше информации о дисковых носителях, в то время как «оператор» только состояние. В дальнейшем планируется для уменьшения количества времени анализа информации о дисках создать графический интерфейс.

ЛИТЕРАТУРА

1. Документация Kuberntes [Электронный ресурс]. Режим доступа: https://kubernetes.io/
2. Документация Docker Swarm [Электронный ресурс]. Режим доступа: https://docs.docker.com/engine/swarm/
3. Документация Mesos [Электронный ресурс]. Режим доступа: http://mesos.apache.org/
4. Operator Pattern [Электронный ресурс]. Режим доступа: https://kubernetes.io/docs/concepts/extend-kubernetes/operator/

СОДЕРЖАНИЕ

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

Сборник материалов конференции
23 апреля 2020 года

Налоговая льгота – Общероссийский классификатор продукции
ОК 005-93, т. 2; 95 3004 – научная и производственная литература

Подписано в печать 22.04.2020. Формат 60×84/16. Печать цифровая.
Усл. печ. л. 11,0. Тираж 150. Заказ 0746.

Отпечатано с готового оригинал-макета, предоставленного редколлегией,
в Издательско-полиграфическом центре Политехнического университета.
195251, Санкт-Петербург, Политехническая ул., 29.
Тел.: (812) 552-77-17; 550-40-14.