Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

|  |
| --- |
| *К защите допустить*: |
| Заведующая кафедрой ПОИТ |
| \_\_\_\_\_\_\_\_\_\_\_\_Н. В. Лапицкая |

Пояснительная записка

к дипломному проекту

на тему

Программное средство Организации работы студентов над групповыми проектами под ОС Android

БГУИР ДП 1-40 01 01 01 084 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | Д.О. Пендо |
| Руководитель |  | К.Е. Колотыгин |
| Консультанты: |  |  |
| *от кафедры ПОИТ* |  | К.Е. Колотыгин |
| *по экономической части* |  | В.А. Палицын |
|  |  |  |
| Нормоконтролер |  | С.А. Медведев |
|  |  |  |
| Рецензент |  |  |

Минск 2017

**Р Е Ф Е Р А Т**

Пояснительная записка 122 с., 24 рис., 14 табл., 24 источника

ПРОГРАММНОЕ СРЕДСТВО ОРГАНИЗАЦИИ РАБОТЫ СТУДЕНТОВ НАД ГРУППОВЫМИ ПРОЕКТАМИ ПОД ОС ANDROID

Объектом исследования является программное средство организации работы студентов над групповыми проектами под ОС Android.

Ключевые слова:

Java, Groovy, JavaFX, WebSocket, RESTful, SSL, автоматизированное рабочее место, реализуемый проект, электронная сделка, сервис реализации проектов.

Цель работы – решение проблемы автоматизации и организации работы администратора по учету данных системы и управлению реализуемыми клиентами проектами, что позволит снизить временные затраты и в целом упростить организацию процесса управления контентом системы.

Предлагаемое программное средство позволяет исключить человеческий фактор из процесса учета сделок между клиентами, упрощает работу по информированию клиентов об изменениях в статусах проектов, а также значительно упрощает модерирование наполнения сервиса.

Проведен анализ достоинств и недостатков существующих программных продуктов. С их помощью разработаны и спроектированы функциональные требования к приложению.

На основе функциональных требований разработана архитектура программного средства и модель базы данных.

Разработаны тесты для проверки соответствия функциональным требованиям и корректности работы приложения.

Приведено технико-экономическое обоснование эффективности разработки и использования программного средства.

Разрабатываемое программное средство должно позволить упростить работу с модерированием проектов, которые реализуют пользователи сервиса, и автоматизировать процесс учета электронных сделок клиентов.

Министерство образования Республики Беларусь

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| Факультет | КС и С | | | Кафедра | | | ПОИТ | | | | | | | | | | |
| Специальность | 1-40 01 01 | | | Специализация | | | | | | 01 | | | | | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | | |
|  | | | | | | | | |  | | | | | Н.В.Лапицкая | | | |
| « | | | | | | | | |  | | » | |  | | | 20 | г. |
| ЗАДАНИЕ | | | | | | | | | | | | | | | | | |
| по дипломному проекту студента | | | | | | | | | | | | | | | | | |
| **Пендо Дмитрия Олеговича** | | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | | | | | | | |
| 1. Тема проекта: | | **Программное средство организации работы студентов над** | | | | | | | | | | | | | | | |
| **групповыми проектами под ОС Android** | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| утверждена приказом по университету от | | | | « | 21 | » | | апреля | | | | 2017 г. | | | № | 651-c | |
| 2 . Срок сдачи студентом законченной работы | | | | | 01 июня 2017 года | | | | | | | | | | | | |
| 3. Исходные данные к проекту | | | Тип операционной системы – Android OS; | | | | | | | | | | | | | | |
| Язык программирования – Javascript, JSX; Перечень выполняемых функций: | | | | | | | | | | | | | | | | | |
| а) функция расчетного центра; б) модерация контента сервиса; в) оповещение клиентов об | | | | | | | | | | | | | | | | | |
| изменении статусов их проектов; г) аккумулирование данных от внешних систем. | | | | | | | | | | | | | | | | | |
| Назначение разработки: автоматизация учета данных сервиса совместной реализации | | | | | | | | | | | | | | | | | |
| кредитуемых бизнес-проектов | | | | | | | | | | | | | | | | | |
| 4. Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | | |
| 1 Анализ прототипов, литературных источников. Формирование требований к ПС | | | | | | | | | | | | | | | | | |
| 2 Анализ требований к ПС и разработка функциональных требований | | | | | | | | | | | | | | | | | |
| 3 Проектирование программного средства | | | | | | | | | | | | | | | | | |
| 4 Создание программного средства | | | | | | | | | | | | | | | | | |
| 5 Тестирование программного средства | | | | | | | | | | | | | | | | | |
| 6 Руководство по установке и использованию | | | | | | | | | | | | | | | | | |
| 7 Технико-экономическое обоснование | | | | | | | | | | | | | | | | | |
| Заключение | | | | | | | | | | | | | | | | | |
| Список использованных источников | | | | | | | | | | | | | | | | | |
| Приложение А Текст программы | | | | | | | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. Перечень графического материала (с точным указанием наименования) и обозначения | | | |
| вида и типа материала) | |  | |
| Диаграмма вариантов использования. Плакат - формат А1, лист 1. | | | |
| Схема взаимодействия АРМ с внешними система при открытии лота. | | | |
| Плакат - формат А1, лист 1. | | | |
| Информационная модель предметной области. Плакат - формат А1, лист 1. | | | |
| Программное средство автоматизации учета данных платформа по совместной  реализации кредитуемых бизнес-проектов. Схема программы - формат А1, лист 1. | | | |
| Схема входа с использованием сервера LDAP - формат А1, лист 1. | | | |
| Схема обработки проектов клиентов - формат А1, лист 1. | | | |
|  | | | |
|  | | | |
| 6. Содержание задания по технико–экономическому обоснованию | | | |
| Расчет экономической эффективности от внедрения программного средства | | | |
| Задание выдал |  | | / В.А. Палицын / |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
| Задание выдал: |  | |  |

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа в % | Срок выполнения этапа | Примечание |
| Анализ предметной |  |  |  |
| области, разработка технического задания | 15-20 | 01.02–14.02 |  |
| Разработка функциональных требований, |  |  |  |
| проектирование архитектуры программы | 20-15 | 15.02–06.03 |  |
| Разработка схемы программы, алгоритмов, |  |  |  |
| схемы данных | 20-15 | 07.03–27.03 |  |
| Разработка программного средства | 15-20 | 28.03–24.04 |  |
| Тестирование и отладка | 10 | 25.04–08.05 |  |
| Оформление пояснительной записки |  |  |  |
| и графического материала | 20 | 09.05–31.05 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 1 февраля 2017 г. | | Руководитель | |  | / К.Е. Колотыгин / | |
| Задание принял к исполнению | |  | | / Д.О. Пендо / | | |

СОДЕРЖАНИЕ

[Введение 8](#_Toc451797466)

[1 Анализ прототипов, литературных источников. Формирование требований к проектируемому ПС 9](#_Toc451797467)

[1.1 Анализ существующих аналогов 9](#_Toc451797468)

[1.2 Аналитический обзор литературы **Ошибка! Закладка не определена.**](#_Toc451797469)

[1.3 Формирование требований к проектируемому ПС 19](#_Toc451797470)

[2 Анализ требований к ПС и разработка функциональных требований 20](#_Toc451797471)

[2.1 Описание функциональности ПС 20](#_Toc451797473)

[2.2 Спецификация функциональных требований 21](#_Toc451797474)

[3 Проектирование программного средства 21](#_Toc451797475)

[3.1 Разработка программной архитектуры 25](#_Toc451797476)

[3.2 Проектирование системы **Ошибка! Закладка не определена.**](#_Toc451797477)

[3.3 Разработка модели базы данных 28](#_Toc451797478)

[3.4 Разработка схемы алгоритма работы с программой **Ошибка! Закладка не определена.**](#_Toc451797479)

[3.5 Разработка алгоритма входа пользователя в систему 27](#_Toc451797480)

[3.6 Разработка алгоритма обработки проектов **Ошибка! Закладка не определена.**](#_Toc451797481)

[3.7 Жизненный цикл перепродаваемых вкладов **Ошибка! Закладка не определена.**](#_Toc451797482)

[4 Создание программного средства 33](#_Toc451797483)

[4.1 Технология RESTful Web Service 33](#_Toc451797484)

[4.2 Протокол шифрования SSL 33](#_Toc451797485)

[4.3 Java Persistence Query Language 34](#_Toc451797486)

[4.4 Описание классов и методов 35](#_Toc451797487)

[4.5 Используемые модули и фреймворки 38](#_Toc451797488)

[4.3 Описание взаимодействия с использованием WebSocket 39](#_Toc451797489)

[5 Тестирование программного средства 41](#_Toc451797490)

[6 Руководство по установке и использованию 47](#_Toc451797491)

[6.1 Серверная часть 47](#_Toc451797492)

[6.2 Клиентская часть 47](#_Toc451797493)

[7 Технико-экономическое обоснование 59](#_Toc451797494)

[7.1 Общая характеристика программного средства 59](#_Toc451797495)

[7.2 Расчет сметы затрат и цены программного продукта 59](#_Toc451797496)

[7.3 Расчет экономического эффекта от применения ПС у заказчика 70](#_Toc451797497)

[7.4 Вывод по технико-экономическому обоснованию 75](#_Toc451797498)

[Заключение 76](#_Toc451797499)

[Список использованных источников 78](#_Toc451797500)

[Приложение А. Текст программного модуля клиента 79](#_Toc451797501)

Определения и сокращения

В настоящей пояснительной записке применяются следующие определения и сокращения.

*Аутентификация* – проверка подлинности предъявленного пользователем идентификатора.

*Бонус* – вознаграждение, предоставляемое владельцем инвестпроекта физическим лицам, разместившим денежные средства во вклад в рамках финансирования инвестпроекта.

*Вклад* – срочный банковский вклад, открытие которого производится с использованием возможности дистанционного банковского обслуживания сервиса совместной реализации кредитуемых бизнес-проектов.

*Владелец инвестпроекта* – юридическое лицо, индивидуальный предприниматель, получающий финансирование в Банке.

*Инвестпроект* – инвестиционный проект, зарегистрированный на портале.

*Клиент* – физическое лицо, заключившее с банком договор дистанционного банковского обслуживания.

*Контент* – информационное содержание сайта.

*Лот* – вклад, выставленный владельцем на продажу.

*Модерация* – это контроль над информацией, размещаемой пользователями на интернет-сайтах. Модерация включает контроль цензуры, а также соответствие информации тематике ресурса и установленным правилам.

*Покупатель* – клиент, желающий осуществить покупку лота.

*Программа* – данные, предназначенные для управления конкретными компонентами системы обработки ин­формации в целях реализации определенного ал­горитма.

*Программирование* – научная и практическая деятельность по созданию программ.

*Продавец* – клиент, ранее оформивший вклад и выставивший данный вклад на продажу на портале.

*Проект* – мероприятия, работы, услуги, блага и др., которые будут реализовываться автором за счёт средств, привлечённых на портале.

*Сайт ДБО* – банковский ресурс, используемый для ведения учет клиентов банка, их авторизации, проверки статуса подписания новой формы универсального договора ДБО.

*Сервис, портал, платформа* – сайт, расположенный в доменной зоне Банка и предназначенный для заведения и демонстрации проектов, вкладов и их условий, выставления лотов на продажу.

*Счет обслуживания* – текущий счет, который используется для осуществления операций по счетам, вкладам Клиента с использованием систем ДБО.

*Фреймворк* – программное обеспечение, облегчающее разработку и объединение различных компонентов большого программного проекта.

*Forpost* – система аналитического учета Банка, используемая для учета клиентов, счетов и операций физических лиц.

*Gate* – шлюз, используемый для установки взаимодействия между АРМ и системой Forpost. Содержит сервисы по заведению карточки клиента и открытию депозитных контрактов в системе Forpost.

MPI – процессинговый центр Банка, используется для совершения финансовых операций по выкупу лота.

БПК – банковская пластиковая карта.

ДБО – дистанционное банковское обслуживание.

СОУ – сервер оплаты услуг.

СУБД – система управления базой данных.

NoSQL – not only SQL (не SQL).

CRUD – create, read, update, delete (создание, чтение, обновление, удаление).

Введение

В связи с усложнением технологий разработки программ и повышением конкурентноспособности приложений на рынке для создания современных технологических проектов требуется больше специалистов, зачастую концентрирующих свою деятельность в более узконаправленных областях, как, например, дизайн, разработка мобильных приложений для Android OS и т.д. Для обеспечения более высокого качества программных продуктов и оптимизации процесса их разработки представляется целесообразным выполнять задания дипломного проектирования в группах, состоящих из определенного набора специалистов, необходимых для написания полноценного программного продукта. Однако несмотря на это, в большинстве случаев многим студентам приходится самостоятельно прорабатывать все аспекты их курсовых и дипломных работ, от дизайна до конечной реализации. В таком случае на это уходит больше времени по сравнению с выполнением того же объёма работы в команде. В связи с этим многие студенты не успевают полностью изучить предметную область, множество вопросов остаются нерешёнными, из-за чего страдает качество конечного продукта. Другой проблемой, также представляющейся обоснованием для темы данного дипломного проекта, является недостаток практики командной разработки во время обучения в университете, что влечет за собой необходимость дополнительно обучаться этому на курсах вне университета.

Целью данной работы является разработка программного средства, которое помогало бы студентам в процессе выполнения курсовых, дипломных и других видов групповых работ. Среди возможностей данного приложения следующие: поиск проектов, вакансий на студенческих проектах, пользователей, создание проектов, их изменение, удаление, создание вакансий, управление существующими проектами и другое. Также характерной чертой данного приложения является функция отслеживания руководителем группы степени прогресса в работе над поставленными задачами.

Проанализировав достижения в области данного дипломного проекта, можно утверждать, что данная сфера является относительно новой и на данный момент пока ещё не существует большого количества подобных приложений. Среди существующих приложений смежной тематики можно выделить Overv.io, Trello, однако данные приложения выполняют более общие задачи по управлению проектами. В то время как данный дипломный проект нацелен на использование в сфере IT-образования.

Данный проект даст возможность студентам создавать свои собственные проекты в команде, улучшить взаимодействие друг с другом в режиме реального времени, структурировать и задокументировать процесс участия студентов в проекте. Создание приложения такого рода позволит подготовить выпускников к старту карьеры в технологических компаниях, упростить подготовку к защите диплома, получить достойный проект в портфолио, а со стороны руководителей проекта – получить более полную информацию о прогрессе подопечных и принимать участие и направлять студентов в планировании задач.

1 Анализ прототипов, литературных источников. Формирование требований к проектируемому ПС

## Анализ существующих аналогов

Разрабатываемое приложение не имеет известных полных аналогов, однако возможно выделить в нем две основные части: поиск и подбор команды для проекта и управление разработкой проекта. Рассмотрим ряд программ, соответствующих по функционалу этим частям приложения. Таким образом, аналогом для части, связанной с подбором команды является сеть профессиональных контактов LinkedIn, а среди аналогов части, ответственной за управление командной разработкой можно выделить такие приложения, как Jira, Trello, Overvio.

LinkedIn — социальная сеть для поиска и установления деловых контактов. В LinkedIn зарегистрировано более 400 млн пользователей (по состоянию на конец 2015 года), представляющих 150 отраслей бизнеса из 200 стран[1].

LinkedIn предоставляет возможность создавать и поддерживать список деловых контактов. Пользователь может добавлять новые контакты как из существующих на сайте с помощью встроенного поиска или выбрав подходящие из предложенных, так и извне, например, путем рассылки приглашений знакомым на email адреса. Вне зависимости от способа приглашения, правила сайта требуют предварительное знакомство с приглашаемыми. Если пользователь имеет возможности указать связь с контактом, общий контакт может представить одного другому.

Пользователи LinkedIn могут использовать список контактов для следующих целей:

расширять круг профессиональных связей;

осуществлять поиск людей, компаний, тематических групп;

публиковать резюме и осуществлять поиск работы;

оставлять рекомендации о коллегах и быть рекомендованными;

делиться идеями, знаниями и мыслями о событиях в мире и профессиональной сфере деятельности;

опубликовывать вакансии.

Так как LinkedIn преследует несколько иные цели, в частности подбор кандидатов для закрытия вакансий и расширение круга профессиональных контактов, нельзя сказать, что это является аналогичным приложением разрабатываемому. Однако, можно выделить некоторые схожести, к примеру, разделы поиска сотрудников и отображения навыков конкретного пользователя (Рисунок 1.1).

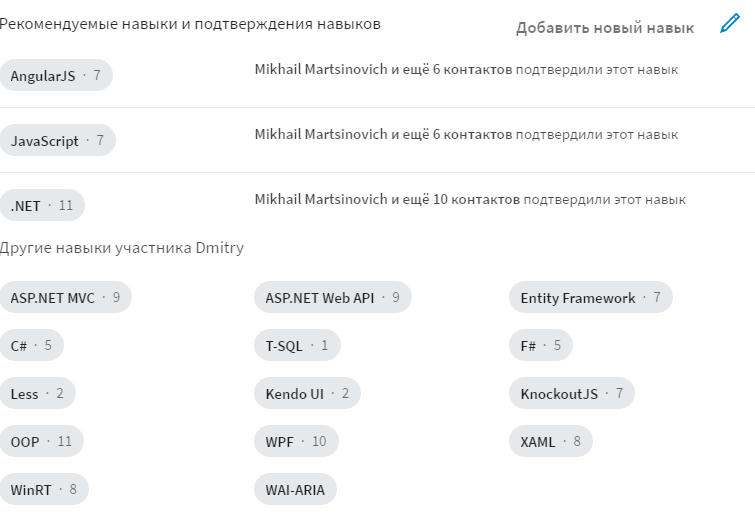


Рисунок 1.1 – Иллюстрация раздела навыков

В разрабатываемом приложении аналогичный функционал, связанный с поиском кандидатов будет осуществляться при поиске кандидатов для заполнения вакансий на проектах. Схожая секция навыков будет представлена на странице пользователя и будет доступна для просмотра пользователями. Также будет реализован поиск кандидатов для проекта исходя из заявленных навыков в профиле пользователя.

Далее рассмотрим приложения для управления командной разработкой.

Jira — коммерческая система отслеживания ошибок, предназначена для организации взаимодействия с пользователями, хотя в некоторых случаях используется и для управления проектами. Разработана компанией Atlassian, является одним из двух её основных продуктов (наряду с вики-системой Confluence). Имеет веб-интерфейс (рисунок 1.2) и мобильное приложение для тех пользователей, которые используют версию продукта, размещенную в облаке.[2]

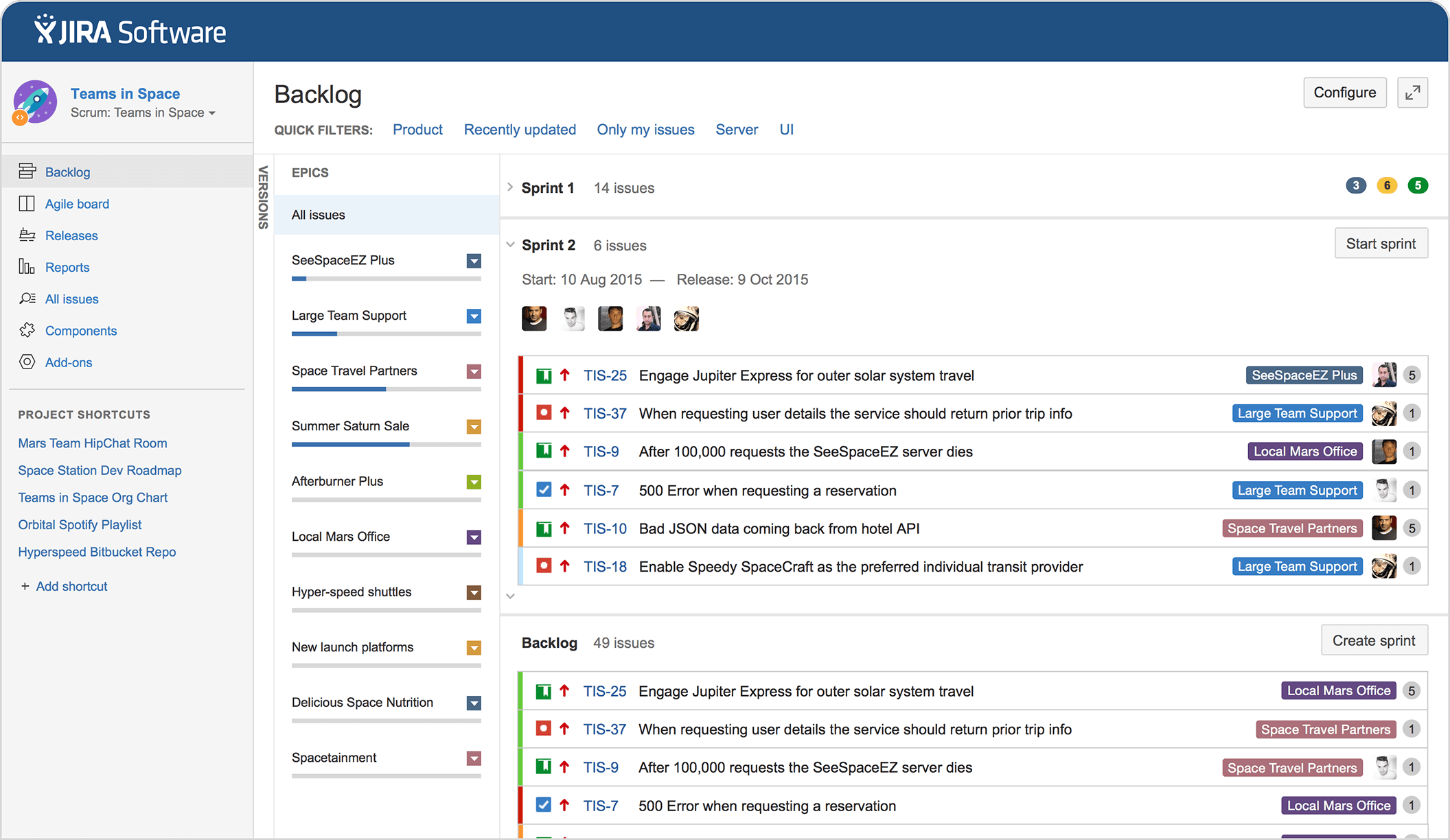


Рисунок 1.2 – Пример интерфейса пользователя JIRA

Задача (англ. ticket или issue) является основным элементов в проекте в JIRA. Задача содержит поля, достаточные для задания полной характеристики цели для большинства случаев, например, название проекта, тему, тип, приоритет, компоненты и содержание. Задача может быть расширена дополнительными или определенными пользователями полями, приложениями (фотографиями, снимками экрана, документами) или комментариями. Задача может быть отредактирована в любой момент или перемещена в иной статус, например, из «открыта» в «закрыта». Множество возможных переходов определяется через настраиваемый рабочий процесс. Все изменения в задаче вносятся в журнал.

Jira имеет большое количество возможностей конфигурации: для каждого приложения может быть определен отдельный тип задачи с собственным рабочим процессом, набором статусов, несколькими видами представления (англ. screens). Кроме того, с помощью так называемых «схем» можно определить для каждого индивидуального Jira-проекта поведение и видимость полей, собственные права доступа и другое. Также для Jira существует большое количество плагинов, которые позволяют существенно расширить и без того немалый функционал системы.

Благодаря высокой гибкости Jira может использоваться для многих непрофильных задач, от реализации небольшой системы бронирования до автоматизации процесса рекрутинга.

Достоинства:

* Гибкость. Jira имеет множество вариантов настройки рабочих процессов и содержания задач, что позволяет использовать ее для любых целей.

Поддержка основных методологий разработки.

Огромное количество плагинов (расширений), которые расширяют функциональные возможности Jira.

Дружественный интерфейс и функциональная система поиска и фильтрации цадач в проекте.

Прозрачное слежение за исполнением задач и производительностью.

Интеграция со сторонними приложениями для разработки, такими как Stash и Bitbucket.

Недостатки:

Дороговизна.

Высокий порог входа.

Мобильное приложение существует только для версии JIRA, располагающейся в облаке.

Необходимость тщательной настройки рабочего процесса под собственные нужды пользователя.

Tréllo — это условно-бесплатное веб-приложение для управления проектами небольших групп, разработанное Fog Creek Software (рисунок 1.3).

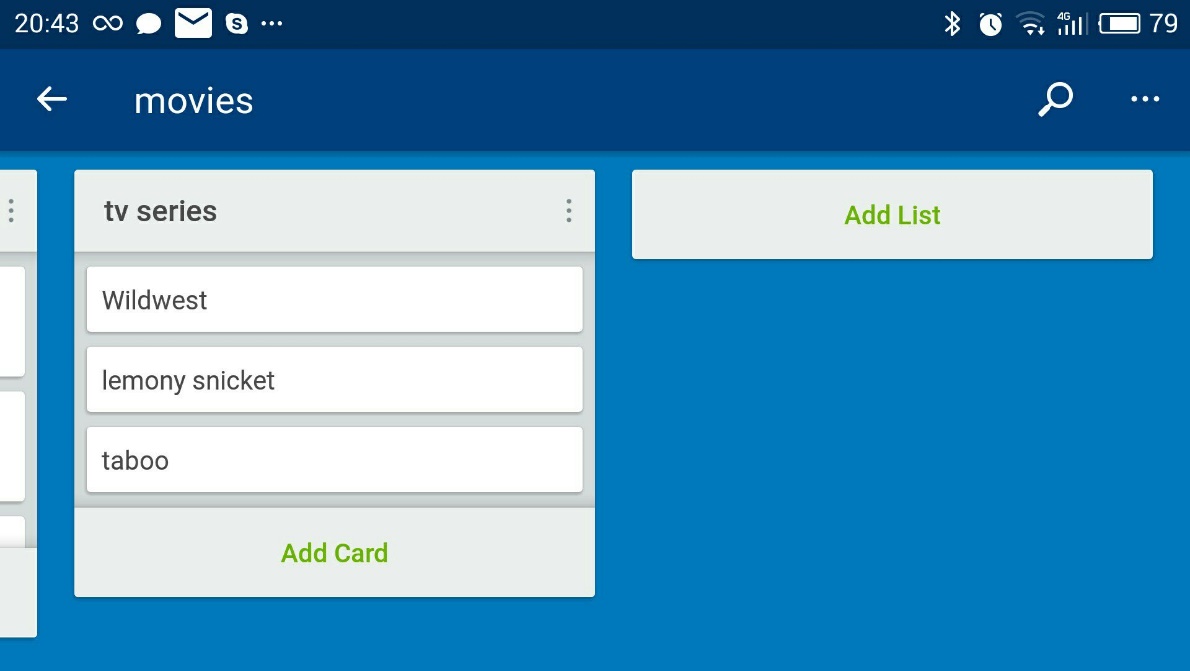


Рисунок 1.3 – Пример интерфейса пользователя мобильного приложения Trello

Trello использует парадигму управления проектами, известную как канбан. Этот метод первоначально был популяризирован Toyota в 1980-х для управления цепочками поставок. Trello использует freemium-бизнес-модель, платные услуги были запущены в 2013 году. [3]

Проекты Trello — это наборы карточек. Каждый набор отображает состояние любого проекта. В каждом наборе карточки сгруппированы по какому-либо признаку в колонки. Например, если вы отбираете кандидатов на работу, то в первой колонке будут карточки кандидатов, во второй — кандидаты, которых вы отобрали для собеседования, в третьей — с кем назначили встречи, в четвертой — с кем встретились, а в пятой — небольшой набор тех, о ком вы всерьез задумываетесь, как о своем будущем сотруднике.

За простотой карточек скрываются множество возможностей. Вы в них можете проводить обсуждения, голосования, загружать файлы данных, задавать дедлайны, назначать текстовые и цветовые метки. Чтобы к любой задаче назначить исполнителя, нужно в карточку выбрать его из списка вашей рабочей команды или просто справа перетащить на задачу аватар коллеги.

Важно заметить, что все члены рабочей группы видят в реальном времени изменения, вносимые в проект, и могут наблюдать состояния друг друга так же в реальном времени.

Достоинства:

Гибкость. Карточки Trello не ограничивают обязательными полями или определенным рабочим процессом.

Простота и низний порог входа.

Есть как бесплатная, так и платная версия приложения.

Удобство в планировании задач

Мобильный клиент

Недостатки:

Плохо подходит для больших команд и управления большими проектами на протяжении долгого времени.

Плохой поиск по сообщениям в списках.

Не принуждает работников четко следовать рабочему процессу, из-за чего исчезает мотивация заполнять карточки должным образом.

Overvio – бесплатное веб-приложение для управления проектами, основанных на гибкой методологии разработки с тесной интеграцией с Github (рисунок 1.4).

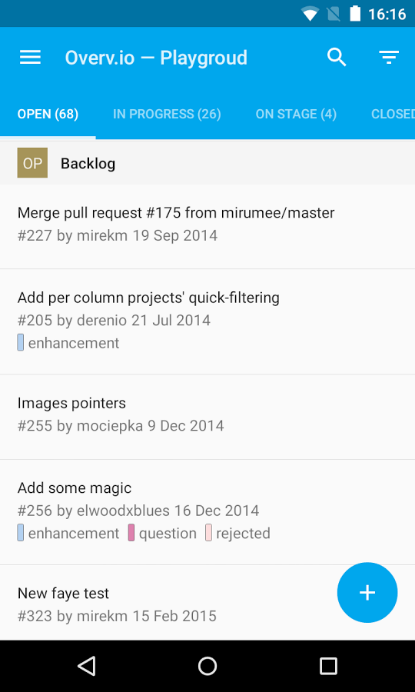


Рисунок 1.4 – Пример интерфейса пользователя мобильного приложения Overvio

Достоинства:

Высокий уровень настраиваемости рабочего процесса в рамках гибкой методологии

Интеграция с Github

Бесплатное распространение

Низкий порог входа

Мобильный клиент

Недостатки:

Сильная зависимость от Github

Отсутствие возможности использовать

Низкая производительность мобильного приложения

Анализируя функционал данных программ можно прийти к выводу, что для эффективной организации командной разработки студенческих проектов необходимо что-то среднее между обширным функционалом и большими возможностями и, как следствие, сложностью JIRA, простотой и излишней гибкостью Trello и удобством Overvio.

С учетом того, что на данный момент не существует программного средства, которое бы включало в себя в полной мере вышеперечисленные возможности и было направлено на упрощение координации студенческих команд в сфере совместной разработки программного обеспечения, разработка на основе рассмотренных примеров функционального, но в то же время простого в использовании программного продукта полезна в прикладном плане. Целью дипломного проекта будет создание мобильного клиента для приложения, которое позволяет подбирать команды для совместного написания студенческих проектов и развивать сотрудничество студентов в написании этих проектов и организовывать последующую разработку программного средства.

## 1.2 Аналитический обзор литературы

Большое внимание при разработке нового программного продукта необходимо уделить пользовательскому интерфейсу. Программы и продукты, которые легко понятны конечным пользователям, повышают удовлетворённость и производительность труда при одновременном снижении расходов на обучение и техническую поддержку, фактически устраняя необходимость в ней, так как у пользователей не будет возникать никаких затруднений при работе с продуктом.

Для снижения расходов на разработку приложения представляется целесообразным выбрать подход, в котором исполнитель сможет частично переиспользовать уже написанную логику для веб-приложения, поэтому основным языков разработки был выбран JavaScript.

JavaScript - [прототипно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [сценарный](https://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [язык программи- рования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Является реализацией языка [ECMAScript](https://ru.wikipedia.org/wiki/ECMAScript) (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам [приложений](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Наиболее широкое применение JavaScript находит в [браузерах](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) как язык сценариев для придания [интерактивности](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [веб-страницам](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0).

Основные архитектурные черты: [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [слабая типизация](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B0%D0%B1%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [прототипное программирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), функции как [объекты первого класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0).

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

объекты, с возможностью интроспекции;

функции как объекты первого класса;

автоматическое приведение типов;

автоматическая сборка мусора;

анонимные функции [4].

За время своего существования JavaScript претерпел множество изменений. В основе данного стандарта Ecma лежит несколько технологий, послуживших для него источниками, наиболее известными из которых являются JavaScript и JScript. Изобрел язык Бренден Айк (Brendan Eich), который на тот момент работал в компании Netscape. Впервые этот язык появился именно в браузере этой компании – Navigator 2.0. В дальнейшем он использовался во всех последующих версиях браузера от Netscape и во всех браузерах от Microsoft, начиная с версии Internet Explorer 3.0.

Разработка [ECMAScript](https://ru.wikipedia.org/wiki/ECMAScript) началась в ноябре 1996г. В июне 1997 года на общем собрании Ecma была принята первая редакция стандарта.

Эта версия стандарта Ecma была представлена на рассмотрение ISO/IEC JTC 1 для ее принятия в соответствии с ускоренной процедурой, и в апреле 1998г. она была утверждена в качестве международного стандарта ISO/IEC 16262. В целях полного соответствия с ISO/IEC 16262, в июне 1998г. на общем собрании Ecma была утверждена вторая редакция ECMA-262. Изменения, внесенные во вторую редакцию стандарта по сравнению с первой, имеют характер редакторской правки.

В третьей редакции стандарта появились мощные регулярные выражения, улучшенная обработка строк, новые операторы управления, обработка исключений try...catch, более строгое определение ошибок, форматирование для цифровых результатов вычислений. Кроме того, были внесены незначительные изменения, предвосхищающие возможности языка в будущем, связанные с его развитием и предстоящей интернационализацией. На общем собрании Ecma в декабре 1999 года была принята третья редакция стандарта ECMAScript. Она была опубликована в июне 2002г. как ISO/IEC 16262:2002.

С момента публикации третьей редакции стандарта язык ECMAScript получил широкое применение в связи с использованием всемирной сети Интернет, где он стал языком программирования, поддерживаемым практически всеми браузерами. Большая работа была произведена по разработке четвертой редакции ECMAScript. Несмотря на то, что эта работа не была завершена и не была опубликована как четвертая редакция ECMAScript, она свидетельствует о непрерывном развитии языка. Пятая редакция ECMAScript (опубликованная как 5-я редакция ECMA-262 5th) систематизирует и оформляет интерпретации языка, которые уже используются к этому времени, и которые стали общепринятыми в реализациях языка. Кроме того, эта редакция предлагает дополнительную поддержку новых возможностей, возникших после публикации третьей редакции. К таким возможностям, в частности, относятся: свойства-аксессоры, рефлексивное создание и инспекция объектов, программное управление атрибутами свойства, дополнительные функции для работы с массивами, поддержка формата кодирования JSON, а также строгий режим языка, обеспечивающий улучшенный контроль ошибок и безопасность программы.

JavaScript – живой язык, который продолжает развиваться. По-прежнему будут появляться существенные технические усовершенствования, которые будут отражены в последующих редакциях этой спецификации. [5]

Следующая редакция языка, ECMAScript 6, принесла в него классы, символы, итераторы, стрелочные функции и многое другое. Однако, полностью стандарт поддерживается только в самых свежих версиях браузеров, поэтому для обратной совместимости необходимо использовать транскомпиляторы, такие, как Babel.

Для написания проекта были рассмотрены основные представители средств разработки кроссплатформенных приложений для мобильных устроиств с использованием JavaScript как основного языка разработки: PhoneGap, NativeScript и React Native.

PhoneGap (называемый также Apache Callback, основанный на Apache Cordova) — бесплатный open-source фреймворк для создания мобильных приложений, созданный Nitobi Software. Позволяет создать приложения для мобильных устройств используя JavaScript, HTML5 и CSS3, без необходимости знания «родных» языков программирования (например, Objective-C), под все мобильные операционные системы (iOS, Android, Bada и т. д.). Готовое приложение компилируется в виде установочных пакетов для каждой мобильной операционной системы. [6]

Плюсы PhoneGap:

Кроссплатформенность.

Быстрая разработка.

Минусы:

Низкая производительность в сравнении с нативными приложениями.

Различия в браузерах различных платформ.

Низкокачественный пользовательский опыт.

Проблемы с утверждением приложения при выпуске в магазины приложений.

Слабая интеграция с внутренними функциями и программными интрефейсами операционной системы.

NativeScript это фреймворк с открытым исходным кодом, разрабатываемый компанией Telerik, для разработки приложений на платформах Android и iOS. Приложения NativeScript разрабатываются на платформонезависимых языках, таких как Javascript или TypeScript. В NativeScript реализована полная поддержка фреймворка AngularJS. Мобильные приложения, построенные с NativeScript, имеют полный доступ к API платформы так, будто они были разработаны в XCode или в Android Studio. Также разработчики могут включать в свои приложения сторонние библиотеки с таких ресурсов, как Cocoapods, Android Arsenal, Maven и npm.js, без создания дополнительных прослоек. [7]

Стоит подчеркнуть, что приложения можно писать для Android 4.2 и выше, и для iOS 7.1 и выше.

Плюсы NativeScript:

Кроссплатформенность.

Быстрая разработка.

Использование Angular2.

Прямой вызов нативного кода из кода Nativescript.

Минусы:

Низкая производительность в сравнении с нативными приложениями.

Довольно малое сообщество и молодость технологии.

React Native – это JavaScript фреймворк от компании Facebook, основанный на подходе к построению приложений как в библиотеке React. React использует раздельные компоненты для построения приложения и одностороннее связывание данных, что делает её отличным вариантом для создания пользовательских интерфейсов. На данный момент React является одним из самых мощных и эффективных способов разработки интерфейсов пользователя, а React Native – достойный его преемник в мире мобильной кроссплатформенной разработки.

Плюсы React Native:

Кроссплатформенность.

Быстрая разработка.

Схожие принципы и совместимость с кодом React приложений.

Использование нативных компонентов.

Высокая производительность приложения

Поддержка сформировавшимся сообществом и технологическим гигантом Facebook.

Минусы:

Ограниченные возможности в стилизации компонентов.

Довольно малое сообщество и молодость технологии.

Исходя из вышеперечисленных характеристик рассматриваемых проектов и того, что веб-приложение будет написано с помощью React, что позволит переиспользовать часть кода вместе с веб-клиентом, что сократит время разработки и вероятность возникновения ошибки при написании, качестве фреймворка для написания мобильного приложения был выбрал React Native. Расмотрим основные концепции React:

Элементы — это объекты JavaScript, которые представляют HTML-элементы. Их не существуют в браузере. они описывают DOM-элементы, такие как h1, div, или section.

Компоненты — это элементы React, написаные разработчиком. Обычно это части пользовательского интерфейса, которые содержат свою структуру и функциональность. Например, такие как Header, Grid, или SubmitButton.

JSX — это надстройка над синтаксисом JavaScript, позволяющая облегчить написание элементов и компонентов React. JSX – XMl-подобный язык, и JSX код внешне выглядит схожим с HTML, но, при этом им не является и, в итоге, транслируется в цепочку вызовов функции React.createElement(), которая и создает элементы интерфейса пользователя. С JSX требуется гораздо меньше усилий на обучение фронтенд-разработчика работе с использованием библиотеки React.

Virtual DOM — это дерево React элементов на JavaScript. React отрисовывает Virtual DOM в браузере, чтоб сделать интерфейс видимым. React следит за изменениями в Virtual DOM и автоматически изменяет DOM в браузере так, чтоб он соответствовал виртуальному. Изменения в React изолируются в тех частях DOM которые и были фактически изменены, что в свою очередь существенно увеличивает производительность системы.

После успеха с React для веб-приложений сообщество взглянуло на него с совершенно другой стороны. По своей сути React – это слой отображения. Весь код, отвечающий за отображение преобразуется из промежуточного языка JSX в вызовы функций создания графических элементов интерфейса пользователя. Таким образом, если заменить при преобразовании JSX создание html-тегов на использование нативных компонентов для определенной платформы, появляется возможность создавать пользовательские интерфейсы с помощью React-кода на любой платформе, для которой можно перегрузить вызов React.createElement. Так, кроме React для веба и React Native для мобильных устройств на базе ОС Android и iOS.

Redux является предсказуемым контейнером состояния для JavaScript приложений. Это позволяет создавать приложения, которые ведут себя одинаково в различных окружениях (клиент, сервер и нативные приложения), а также просто тестируются. Кроме того, это обеспечивает большой опыт отладки, например, редактирование кода в реальном времени в сочетании с перемещенем про истории состояний приложения. Redux можно использовать вместе с React, React-Native или с любой другой view-библиотекой. [8]

Babel - это JavaScript транскомпилятор. Это означает, что он переводит код из новой версии в более старый - стабильный. Babel состоит из 2 частей: транскомпилятор и полифилл. Транскомпилятор преобразует код указанной версии (в данном проекте используется стандарт EcmaScript 6) в код, соответствующий стандарту EcmaScript 5, который на данный момент поддерживается большинством браузеров. Полифилл в свою очередь добавляет отсутствующие методы в стандарте ES5. Таким образом, мы имеем возможность использовать уже сейчас функциональные возможности нового стандарта ES6 для написания JavaScript-кода и JSX для написания компонентов React Native. [9]

## 1.3 Формирование требований к проектируемому программному средству

Разрабатываемое программное средство включает выступает как инструмент хранения данных о всех пользователях и их типах, о истории создания и реализации совместных проектов, о самих проектах, их структуре и заданиях, выполненных в ходе реализации проекта. ПС включает в себя как функции подбора команд для разработки и учета их для различных целей университета, так и функции управления проектами.

Входными данными системы являются данные, вводимые пользователями, т.к. по большому счету приложение представляет собой интерфейс, с помощью котого пользователь манипулирует данными о проектах и задачах. Данные, вводимые пользователем, должны проверяться на корректность, как в процессе аутентификации, так и перед осуществлением каких-либо действий в системе.

Основными выходными данными системы являются экранные формы представления существующих проектов, данных о задачах, решаемых в ходе выполнения проекта и пользователей и всех данных об их взаимодействии.

Разрабатываемая система должна соответствовать следующим требованиям, сформированным на основании изучения существующих аналогов:

1. удобный и интуитивно понятный интерфейс;
2. высокое быстродействие;
3. осуществление функций создания и управления проектами;
4. осуществление поиска исполнителей для проекта;
5. осуществление поиска;
6. осуществление управления проектом путем управления задачами;
7. осуществление организации коллективной работы путем распределения задач между участниками;
8. возможность осуществлять слежение за прогрессом проекта;
9. защищенность данных пользователей;
10. разграничение прав доступа к различным данным и функциям приложения посредством ролей пользователя;
11. дизайн не должен противоречить стилевым рекомендациям мобильной операционной системы;
12. наличие руководства пользователя.

2 Анализ требований к ПС и разработка

функцинальных требований

## 2.1 Описание функциональности ПС

Для представления функциональной модели была выбрана диаграмма вариантов использования UML [14], которая отражает отношения между актерами и прецедентами и позволяет описать систему на концептуальном уровне. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. UML предназначен для определения, визуализации, проектирования и документирования программных систем.

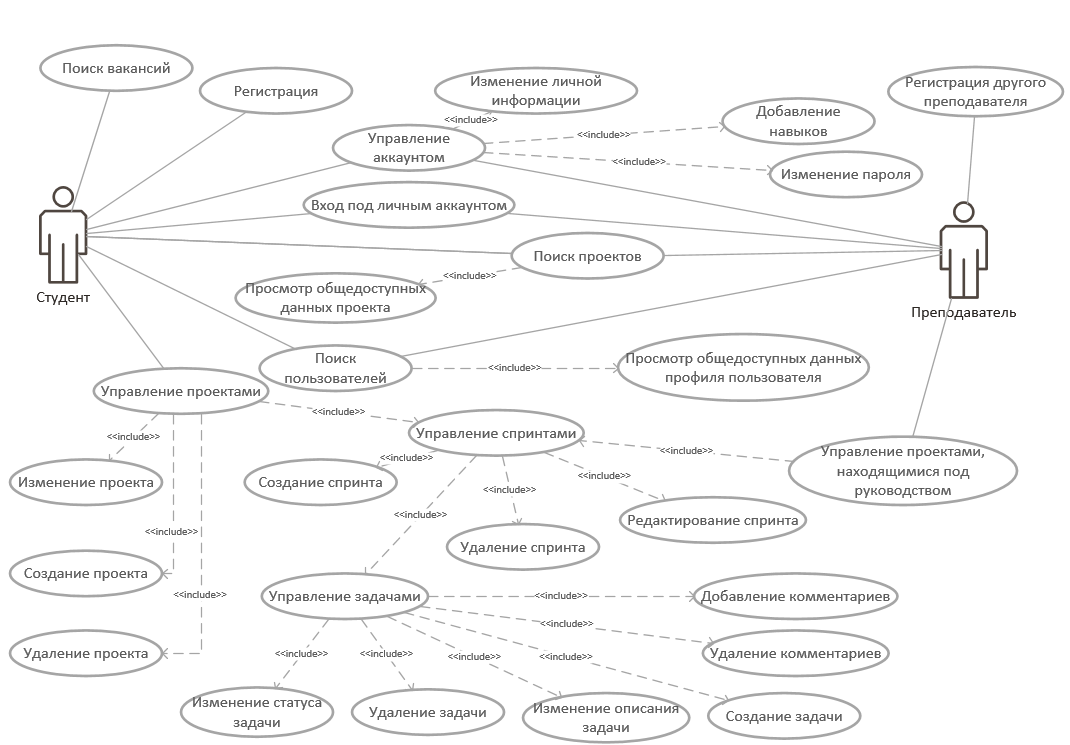


Рисунок 2.1 – Диаграмма вариантов использования

Диаграмма вариантов использования разрабатываемого программного средства представлена на рисунке 2.1.На диаграмме можно выделить два основных составляющих элемента – актор и прецедент. Актор – стилизованный человек, обозначающий набор ролей пользователя, взаимодействующего с некоторой сущностью. Прецедент – эллипс с надписью, обозначающий выполняемые системой действия, приводящие к наблюдаемым актером результатам.

На основании представленной диаграммы вариантов использования можно сделать вывод, что в системе будет существовать два основных актера:

1. Пользователь с уровнем прав «Студент».

2. Пользователь с уровнем прав «Преподаватель».

Рассмотрим каждый из прецедентов более подробно для каждого актера.

Пользователю с уровнем прав «Студент» предоставляются следующие возможности:

регистрация;

вход в систему под личным аккаунтом;

CRUD проектов, спринтов, задач;

добавление и удаление комментариев к задаче;

изменение статуса задачи;

редактирование личной информации;

изменение пароля;

добавление навыков;

просмотр общедоступной информации профилей других пользователей системы;

просмотр общедоступной информации о проектах, в которых пользователь не принимает участие;

поиск свободных вакансий на других проектах;

поиск по пользователям с заданными параметрами;

поиск по проектам с заданными параметрами.

Администратор получает следующие возможности:

вход в систему под личным аккаунтом;

регистрация другого преподавателя;

CRUD спринтов, задач;

добавление и удаление комментариев к задаче;

изменение статуса задачи;

редактирование личной информации;

изменение пароля;

добавление навыков;

просмотр общедоступной информации профилей других пользователей системы;

просмотр общедоступной информации о проектах, в которых пользователь не принимает участие;

поиск по пользователям с заданными параметрами;

поиск по проектам с заданными параметрами.

## 2.2 Спецификация функциональных требований

На основании анализа исходных данных для проектируемого ПС можем выделить, что основной целью является создание качественного программного продукта, позволяющего решить следующие задачи:

В ходе разработки требуется реализовать следующие возможности:

* автоматизировать процесс учета и контроля данных платформы по реализации совместных бизнес-проектов;
* автоматизировать процесс учета движения денежных средств;
* предоставить возможность оповещения клиентов о статусах их проектов и вложений.

В ходе разработки необходимо реализовать следующие возможности:

* функциональность учета данных пользователей площадки;
* функциональность учета проектов;
* функциональность расчетного центра;
* функции экспорта документов в популярные форматы работы с данными (Word, Excel, PDF);
* функциональность по настройке и администрированию системы и пользователей;
* функциональность агрегирования данных с внутренних систем Банка;
* возможность идентификации и аутентификации субъектов доступа.

Разрабатываемый проект представляет собой автоматизированное рабочее место, предназначенное для учета данных платформы по реализации совместных бизнес-проектов.

Основные функции данного программного средства:

АРМ должен поддерживать учет данных пользователей площадки.

АРМ должен предоставлять функционал расчетной системы.

АРМ должен поддерживать учёт проектов.

АРМ должен поддерживать информационную интеграцию с внутренними системами Банка.

Функция идентификации и аутентификации.

Возможность протоколирования событий системы.

Функция отправки уведомлений клиентам об измении статусов их вкладов и проектов.

Дополнительные функции разрабатываемого программного обеспечения:

Предоставление интерфейса доступа к данным расчетного центра для площадки и внешних систем.

Обеспечение безопасности за счет логирования состояний системы и действий пользователей.

1. Осуществление создания проекта:
   1. Пользователь должен иметь возможность создать новый проект.
   2. Создание проекта должно осуществляться на отдельном экране.
   3. Форма с созданием проекта должно обязательно содержать следующие поля:

* название проекта;
* описание проекта;
* дата начала проекта;
* дата окончания проекта;
* вакансии, доступные на проекте.
  1. Форма с созданием проекта может опционально содержать ссылку на репозиторий системы контроля версии.
  2. Проект должен создаваться как отдельная сущность.
  3. Пользователь должен иметь возможность заполнять поля с параметрами для проекта на любом языке.
  4. При отсутствии заполнении одного из обязательных полей проекта пользователь должен увидеть следующее сообщение об ошибке гласящее о том, что поле обязательно для заполнения.
  5. Сообщение об ошибке должно располагаться под полем, в котором допущена ошибка.
  6. При правильном заполнении всех обязательных полей проекта должен произойти запрос на сервер на сохранение проекта в базу данных и последующее сохранение этого проекта в базе данных.
  7. При успешном создании проекта пользователь должен быть перенаправлен на страницу Dashboard, в которой будет отображаться список всех текущий проектов, созданных пользователем.
  8. При нажатии на поля «Дата старта» и «Дата окончания» пользователю должен показываться элемент, осуществляющий выбор даты.

1. Осуществление поиска исполнителей для проекта:
   1. Пользователь должен иметь возможность поиска исполнителей для проекта.
   2. На странице «Поиск исполнителей» должно присутствовать поле для ввода пользовательских данных и поле для отображения результата поиска.
   3. Пользователь должен иметь возможность поиска с использованием следующих фильтров:

* По имени или фамилии пользователя;
* По электронной почте;
* По телефону;
* По университету/факультету/специальности/группе;
* По навыкам.
  1. При пустом результате поиска пользователю должно отобразиться сообщение «Пользователь не найден».

1. Осуществление поиска проекта пользователем:
   1. Пользователь должен иметь возможность поиска проекта.
   2. На странице, где может производиться поиск проекта должно присутствовать поле для ввода данных о проекте и поле для отображения результата поиска.
   3. Пользователь должен иметь возможность поиска с использованием следующих фильтров:

* По названию проекта;
* По дате старта проекта;
* По дате окончания проекта;
* По навыкам, используемым на проекте;
* По вакансиям, доступным на проекте.
  1. При пустом результате поиска пользователю должно отобразиться сообщение «Проект не найден».

1. Осуществление управления проектом путем управления задачами:
   1. Пользователь должен иметь возможность создания задачи.
   2. Пользователь должен иметь возможность переносить выбранные задачи из секции backlog в выбранный спринт на странице «Project Board».
   3. Пользователь должен иметь возможность перетаскивать задачи между секциями на странице «Project Board».
   4. Пользователь должен иметь возможность ассайнить другого пользователя на выполнение задачи.
2. Осуществление создания задачи:
   1. Пользователь должен иметь возможность создать новую задачу.
   2. Создание проекта должно осуществляться в отдельном окне.
   3. Окно с созданием задачи должно обязательно содержать следующие поля:

* Название задачи;
* Тип задачи;
* Подзадача в иерархии;
* Родительская задача;
* Исполнитель задачи;
* Приоритет задачи;
* Спринт.
  1. Окно с созданием проекта может опционально содержать следующие поля:
* Описание задачи;
* Список тегов, принадлежащих задаче;
* Оценка времени на выполнение задачи;
* Конечные сроки выполнения задачи;
* Комментарии.
  1. Задача должна создаваться как отдельная сущность
  2. Пользователь должен иметь возможность заполнять поля с параметрами для проекта на любом языке.
  3. При неправильном заполнении одного из обязательных полей задачи пользователь должен увидеть следующее сообщение об ошибке – «This field is required».
  4. Сообщение об ошибке должно быть располагаться под полем, в котором допущена ошибка.
  5. При правильном заполнении всех обязательных полей задачи должен произойти запрос на сервер на сохранение задачи в базе данных и последующее сохранение этого задачи в базе данных.
  6. При успешном создании задачи, она перемещается в backlog проекта, пользователь должен быть перенаправлен на страницу «Board» проекта, в которой будет отображаться список всех текущих задач.

**3 Проектирование программного средства**

## 3.1 Разработка программной архитектуры

Прежде чем приступать к непосредственной реализации программного средства, необходимо определиться с архитектурой системы в целом, а также компонентов, на основе которых будет построена серверная часть приложения.

В первую очередь, необходимо провести анализ необходимой аппаратной конфигурации, на которой будут работать части конечного программного средства, и описать их взаимодействие между собой. Для описания узлов и их связей будем использовать диаграмму развертывания (Рисунок 3.1):

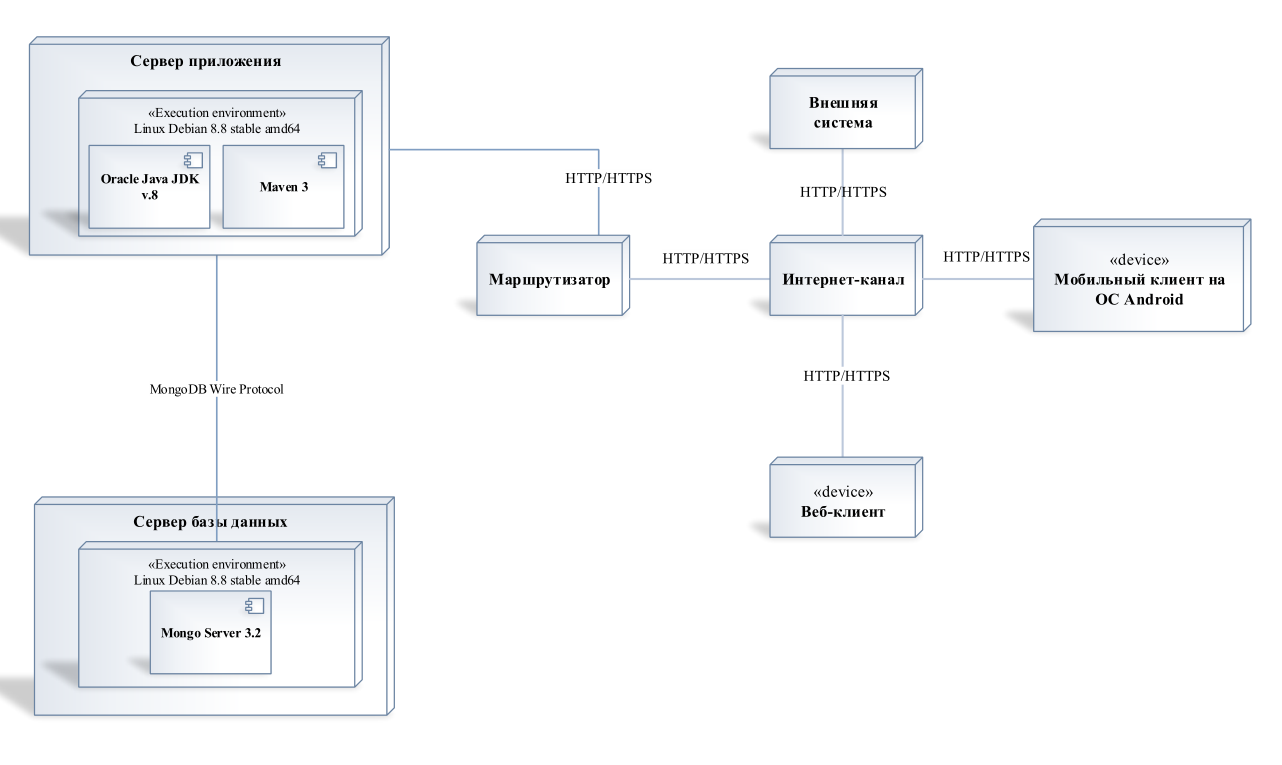


Рисунок 3.1 – Диаграмма развертывания

Данное ПС разрабатывается в виде клиент-серверного приложения. Для запуска веб-сервиса требуется рабочая станция с утановленной JVM и контейнером приложений Tomcat. Приложение запускается с сгенерированного при сборке jar-файла. Для работы приложения требуется наличие на рабочей станции всех библиотек зависимостей данного прилоежения, а также наличия файла со свойствами приложения.

Данные приложения хранятся на сервере базы данных MongoDB. Веб-сервис имеет доступ к серверу базы данных с помощью соединения по протоколу SSL. Подключение к базе данных осуществляется с помощью ввода требуемых имени пользователя и пароля. Изменить имеющиеся имя пользователя, пароль и адрес соединения с базой данных можно в файле свойств приложения.

Клиентские приложения передают и получают данные от веб-сервиса с помощью протокола HTTP. Данные возвращаются на клиентские приложения в формате JSON. В данном случае в роли клиентских приложений выступают веб-приложение и мобильное приложение.

Конфигурация веб-сервиса доступна для двух профилей запуска. Первый профиль, dev, сконфигурирован для локального запуска, является более удобным для процесса разработки и отладки приложения. Профиль prod содержит конфигурацию для работы веб-сервиса на удаленном хостинге (heroku) и является финальной версией приложения, доступной для использования различными клиентскими приложениями. На удаленный сервер загружаются проверенные и реализованные до конца части приложения. Наличие двух профилей разработки позволяет содержать различные окружения для разработки и работы, а также позволяет избежать рисков повреждения данных или предоставления пользователям версии приложения с частичным или неверно работающим функционалом.

В первую очередь, необходимо провести анализ необходимой аппаратной конфигурации, на которой будут работать части конечного программного средства, и описать их взаимодействие между собой.

Были сформулированы следующие требования:

1. Программное обеспечение «Сервер Back-end части приложения» и «Сервер Front-end части приложения» могут быть установлены, как на один физический (либо виртуальный) сервер, так и на различные выделенные сервера.
2. Программное обеспечение «MongoDb Database» устанавливается на отдельный выделенный сервер, либо на кластер серверов.
3. Узлы могут располагаться на отдельных серверах, однако целесообразно всю инфраструктурную часть приложения поместить в единую локальную сеть.
4. Для снижения сетевых нагрузок сервер базы данных вынесен в отдельный узел.
5. Клиент, осуществляет работу с системой по протоколам HTTP, HTTPS.
6. Узлы системы взаимодействуют посредством протокола HTTPS с целью осуществления должного уровня безопасности.

Минимальные рекомендуемые требования к аппаратной платформе:

Для работы сервера приложений:

* процессор уровня Intel Xeon E3 (рекомендуется Intel Xeon E5) с поддержкой 64-битной архитектуры и с тактовой частотой – 2,2ГГц (4 ядра) либо более производительный;
* объем ОЗУ – не менее 4 Гбайт;
* сетевой адаптер (допускается встроенный) с производительностью не менее 100 Мбит/сек (рекомендуется 2 “Gigabit Ethernet”-адаптера);
* объем свободного пространства на жестком диске – не менее 5 Гбайт.

Для работы сервера базы данных:

* один, либо более процессора уровня Intel Xeon E5-2620 с поддержкой 64-битной архитектуры, и частотой не менее 2 GHz (либо более производительных);
* объем ОЗУ – не менее 6Гбайт;
* 2 сетевых адаптера (допускаются встроенные), допускающих работу со скоростью 1Гбит/сек;
* свободное дисковое пространство объёмом не менее 1 Гбайт для установки ПО “MongoDb”.
* свободное дисковое пространство объёмом не менее 100 Гбайт для размещения файлов данных.

Для работы клиентской части:

* компьютер класса Pentium IV тактовая частота – не менее 1000МГц;
* объем ОЗУ – не менее 1 ГБ (для “Windows Vista” и “Windows 7” – не менее 2 Гбайт);
* объем свободного места на жестком диске - 2Гбайт;
* видеоадаптер (допускается встроенный), позволяющий разрешающую способность не менее 1024х824;
* монитор размером не менее 17'', обеспечивающий указанное выше разрешение (рекомендуется 19'' и более);
* сетевой адаптер (допускается встроенный);
* клавиатура, манипулятор типа «мышь».

Для работы мобильного приложения:

* устройство на базе Android OS версии не менее 5.0;
* объем ОЗУ – не менее 1 ГБ;
* объем свободного места на карте памяти – 50 Мбайт;
* экран размером не менее 4'';
* подключение к сети Интернет;

**3.2 Разработка алгоритма входа пользователя в систему**

Вход пользователя в систему состоит из аутентификации и авторизации. Аутентификация представляет собой соответствия лица названному им идентификатору. Авторизация – предоставление этому лицу возможностей в соответствии с положенными ему правами.

Данный алгоритм реализует функцию аутентификации в системе с использованием JWT токена.

JSON Web Token (JWT) — маркер, который содержит в зашифрованном виде всю минимально необходимую информацию для аутентификации и авторизации. При этом не требуется хранить в сессии данных о пользователе, так как маркер самодостаточный (self-contained).

На сервер возлагаются определенные задачи по управлению доступом:

Предоставление возможности установления уровня сложности пароля (невозможность использования только цифровых символов и др.).

Сервер осуществляет контроль и подсчет попыток входа в систему (успешный или неуспешный – несанкционированный).

При достижении заданного числа неуспешных попыток доступа сервер блокирует вход пользователя на определенное (настраиваемый параметр) время.

При авторизации клиентское приложение отправляет запрос на сервер, передавая в сообщении логин и пароль пользователя с использованием шифрования SSL. Сервер формирует запрос на проверку существования учетной записи с данными логином и паролем. При наличии соответствующей записи сервер возвращает данные о клиенте и роль пользователя, зашифрованные в токене. Таким образом, для дальнейших обращений к защищенному от неавторизованного доступа API необходимо передавать полученный токен в секции Header каждого запроса. На сервере токен будет обработан, пройдет проверку и будет вынесено решение о том, имеет ли пользователь, приславший запрос права на получение информации. В случае, если токен не действителен, ответ сервера будет имень код 401 Unauthorised.

## 3.3 Разработка модели данных, получаемых с API

Также, неотъемлемой частью конечного программного средства являются данные, в этой случае, получаемы с API путем запросов на сервер. Диаграмма классов данных, получаемых в результате запросов на сервер представлена на рисунке 3.2

Рисунок 3.2 - Модель базы данных

3.3.1 Сущность «Пользователь»:

* идентификатор − идентификатор клиента (первичный ключ);
* имя;
* фамилия;
* отчество;
* состояние (подтвержден, не подтвержден);
* роль («Студент», «Преподаватель»)
* email - электронный ящик пользователя;
* пароль доступа;
* номер телефона;
* список связанных аккаунтов в соцсетях;
* ссылка на аккаунт Github;
* ссылка на фото профиля;
* информация о университете (сущность «Университет»)
* список навыков (список сущностей «Навык»)
* список проектов (присок сущностей «Проект»)

Пользователь может иметь следующие состояния:

* + подтвержден
  + не подтвержден

Пользователь через свой личный кабинет на портале может заполнить все оставшиеся незаполненными необязательные реквизиты или изменить их.

3.3.2 Сущность «Проект»:

* + идентификатор проекта – первичный ключ сущности;
  + название проекта;
  + описание проекта;
  + дата старта проекта;
  + дата закрытия проекта;
  + список вакансий(позиций) проекта - список сущностей «Вакансия»;
  + ссылка на Github репозиторий.

3.3.3 Сущность «Задача»:

* + идентификатор задачи – первичный ключ сущности;
  + название задачи;
  + приоритет задачи;
  + тип задачи;
  + список идентификаторов подзадач;
  + идентификатор родительской задачи;
  + список идентификаторов смежных задачи;
  + описание задачи;
  + идентификатор автора задачи;
  + идентификатор ответственного за задачу;
  + планируемое время на решение задачи;
  + уровень задачи;
  + список тегов задачи;
  + статус задачи;
  + идентификатор спринта задачи;
  + идентификатор проекта;
  + дата, к которой задача должна быть решена.

3.3.4 Сущность «Университет»:

* + название университета – первичный ключ сущности;
  + название факультета;
  + название спеуиальности;
  + номер группы;
  + год начала обучения;
  + год окончания обучения.

3.3.5 Сущность «Проект-уменьшеннная» (уменьшенная версия для превью):

* + идентификатор проекта – первичный ключ сущности;
  + название проекта;
  + описание проекта;
  + дата старта проекта;
  + дата закрытия проекта;
  + ссылка на Github репозиторий.

3.3.6 Сущность «Пользователь-Проект» - сущность, объединяющая пользователей и проекты:

* + идентификатор проекта – первичный ключ сущности;
  + информация о позиции; - сущность «Вакансия-Информация»
  + текущий ли проект;
  + дата старта проекта;
  + дата закрытия проекта.

3.3.7 Сущность «Пользователь-уменьшеннная»:

* + идентификатор пользователя – первичный ключ сущности;
  + адрес электронной почты;
  + пароль;
  + номер группы;
  + имя пользователя;
  + фамилия пользователя.

3.3.8 Сущность «Вакансия»:

* + идентификатор пользователя – первичный ключ сущности;
  + название вакансии;
  + описание вакансии;
  + тип вакансии;
  + идентификатор назначенного пользователя.

Тип вакансии может иметь следующие значения

* + программист
  + тестировщик
  + дизайнер
  + бизнес-аналитик
  + инженер
  + лидер проекта

3.3.9 Сущность «Вакансия-Информация»:

* + название вакансии;
  + описание вакансии;
  + тип вакансии;

3.3.10 Сущность «Навык»:

* + идентификатор навыка – первичный ключ сущности;
  + название навыка;
  + уровень владения навыком;
  + список идентификаторов пользователей, подтвердивших владение навыком;

3.3.11 Сущность «Аккаунт социальной сети»:

* + идентификатор записи – первичный ключ сущности;
  + ссылка;
  + тип социальной сети;

## 3.4 Описание алгоритма работы компонента Project list

При переходе на экран Project list запускается очередность действий, характерная для любого React-компонента. Всего можно выделить 4 состояния в компоненте:

* Инициализация
* Обновление
* Простой
* Деинициализация

В процессе инициализации последовательно вызываются следующие методы:

* constructor()
* componentWillMount()
* render()
* componentDidMount()

В методе componentWillMount() проиходит вызов API c целью получить список проектов. Статус компонента устанавливается в loading. Далее компомент наступает фаза «обновление». Эта фаза наступает в ответ на изменение параметров компонента или его состояния. В фазе обновления просходят следующие вызовы:

* componentWillReceiveProps()
* shouldComponentUpdate()
* componentWillUpdate()
* render()
* componentDidUpdate()

К этому моменту данные еще не получены, так что в методе render() происходит проверка на то, находится ли компонент в статусе loading. В этом случае отрисовывается уведомление, что данные еще не загружены и отображается индикатор прогресса.

Когда сервер возвращает данные, ообновляется основное хранилище данных. Все компоненты, подписанные на изменения в соответвствующих частях хранилища, получают обновленные данные. В компоненте запускается фаза обновления и заново вызывается метод render(). Если этом случае данные есть и статус – fulfilled – список проекта отображается на экране устройства.

При закрытии приложения или переходе на другой экран запускается процесс разрушения компонента и он входит в фазу деинициализации. В ней вызывается метод componentWillUnmount(), в котором можно освободить ресурсы, если требуется.

4 Создание программного средства

Спецификация функциональных требований и спроектированная архитектура программного средства служат фундаментом, на котором основывается выбор наиболее подходящих технологий для разработки программного средства. Успешное и обоснованное завершение данных этапов позволит создать расширяемое, надежное и функциональное приложение, призванное решать поставленные задачи.

4.1 Технология RESTful Web Service

Веб-службы представляют собой компоненты веб-сервера, доступные извне клиентским приложениям. Их можно использовать для интеграции компьютерных приложений, иссполняемых на различных платформах, написанных на различных языках с использованием различных технологий. Веб-службы не зависят от языка и платформы, так как между поставщиками существует договоренность об общих стандартах веб-служб.

REST [!!!https://spring.io/understanding/REST] (сокр. англ. Representational State Transfer, «передача состояния представления» или «передача репрезентативного состояния») – стиль построения архитектуры распределенных систем.

Веб-службы на основе REST («RESTful») представляют собой коллекцию веб-ресурсов, идентифицируемых по своим URI. Каждая единица информации однозначно определяется уникальным идентификатором URI. Этими веб-ресурсами можно управлять с помощью действий, указанных в заголовке HTTP.

REST не накладывает явных ограничений на формат данных, используемый для передачи данных между клиентом и сервером. Единственными требованиями, накладываемыми на формат данных, являются возможность обработки данных как клиентом, так и сервером и возможность полного описания данных в этом формате, вне зависимости от сложности данных.

Используемый сетевой протокол должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии клиента между парами «запрос-ответ». Клиент отправляет запросы на сервер, когда есть необходимость в переходе в новое состояние. Запросы клиента должны быть составлены так, чтобы сервер получил всю необходимую ему информацию для формирования ответа. Ответы сервера должны быть помечены как кешируемые или нет для избежания устаревания данных. Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями. Наиболее часто используемым протоколом является HTTP. [!!!https://en.wikipedia.org/wiki/Representational\_state\_transfer]

Создание веб-служб с применением подхода RESTful является популярной альтернативой использованию технологий развертывания служб в сети Интернет на основе SOAP, поскольку этот подход отличается простотой и удобством, а также предоставляет возможность передачи данных непосредственно по HTTP.

4.2 Redux

Как уже было сказано выше, приложение написано с использованием библиотеки React Native. React Native – это библиотека для построения мобильных приложений. Но, как и React, она не берет на себя контроль за состоянием различных частей приложения в тот или иной момент времени. Поэтому для написания приложения был выбран redux.

Redux – предсказуемый контейнер состояния для приложений, написанных на языке JavaScript. Вес данной библиотеки составляетвсего 2Кб вместе со всеми зависимостями, что сравнительно мало. Это является большим плюсом для браузерных клиентских приложений. Рассмотрим основные понятия redux: действия, редьюсеры, хранилище.

Действие – это простой обьект, который описывает факт свершения какого-либо события, которое влечет за собой определеннные мутации хранилища. В действии обязательно должно быть поле, указывающее на тип действия. С помощью вызова функции dispatch действие можно передать в редьюсеры.

Редьюсер – чистая функция, принимающая на вход состояние хранилища, произошедшее действие и возращающая новое состояние. Новое состояние должно быть новым объектом, оно должно быть софрмировано без внесения каких-либо изменений в объект прошлого состояния. Каждый редьюсер получает все действия, переданные с помощью dispatch. Решение о применении каких-либо изменений на состояние должно быть принято внутри самого редьюсера.

Хранилище – объект, содержащий состояние приложения. Хранилище позволяет доступ к текущему состоянию, регистрировать слушателей изменений и обновлять состояние с помощью вызовов dispatch. В redux хранилище существует только в одном экземпляре, все взаимодейтствия происходят с этим объектом.

Таким образом поток данных в redux является однонаправленных. При возникновении события, требующего изменения состояния создается действие. Действие передается во все существующие редьюсеры с помощью функции dispatch. Редьюсеры применяют необходимые мутации на объекте хранилища, который, в свою очередь, оповещает всех подписчиков о изменении данных. Также рассмотрим принципы redux.

Первый принцип называется «единственний источник правды». Это принцип гласит о том, что состояние приложение должно быть сохранено в дереве объектов внутри одного хранилища. Это является большим плюсом при написании больших приложений, в разных частях которого происходит работа с одним и тем же набором данных – единственное хранилище позволяет всем частям приложения оперировать одной версией данных. К тому же, для ускорения разработки состояние приложения можно сохранять.

Второй принцип называется «состояние только для чтения». Этот принцип гласит о том, что единственным способом изменить состояние хранилища должно быть применение объекта, описывающего изменения. Это гарантирует, что функция, реагирующие на события сети никогда не изменят состояние хранилища напрямую. Т.к. все изменения централизованы и применяются в определенном порядке, то подследовательность действий можно сохранить для последующего воспроизведения при отладке.

Третий принцип имеет название «мутации как чистые функции». Этот принцип налагает ограничения на функции, изменяющие дерево состояния в хранилище. Такие фунции называются редьюсеры. Редьюсер – это чистая функция, которая принимает прошлое состояние и действие и возвращает новое состояние. Чистая функция предполагает то, что она является детерминированной и не обладает побочными эффектами.

Исходя из вышесказанного, можно отметить, что redux позволяет писать крупные и масштабируемые приложения, предоставляющие полный контроль над состоянием и его обновлениями в любой момент времени.

4.3 Используемые модули и библиотеки

Как было сказано выше, одним из основных языков разработки будет язык программирование JavaScript. Для установки зависимостей используется пакетный менеджер npm. Npm имеет крупнейший реестр модулей программного обеспечения во всем мире [!!!https://www.npmjs.com/]. Большое количество библиотек и модулей, находящихся в каталоге npm значительно упрощает разработку системы.

Таким образом, в программном продукте используются следующие библиотеки и модули:

* axios – популярный http-клиент, основанный на системе обещаний (англ. promise);
* react-native – бибилотека для построения клиентских приложений для OC Android и iOS;
* redux – предсказуемый контейнер состояний;
* redux-thunk – библиотека, позволяющая писать функции, создающие действия, возвращающие функции вместо объектов;
* react-native-button – React Native компонент, представляющий собой кнопку, предоставляющий большие возможности в настройке внешнего вида;
* react-native-image-crop-picker – React Native компонент, предоставляющий возможность использовать стандартный для ОС Android диалог для выбора изображений;
* react-native-side-menu – React Native компонент, представляющий собой боковое меню с широкими возможностями настройки стилей и содержания меню;
* react-navigation – JavaScript-библиотека для навигации между состояниями в React Native приложениях. Каждое состояние представляется в виде компонента; при переходе между состояниями создаются новые компоненты, в которые есть возможность передать параметры навигации.

4.4 Описание компонентов

Данное приложение имеет компонентную структуру, присущую React приложениям. Благодаря этому структура кода становится более понятной и логичной. В свою очередь, компоненты условно делятся на «умные» и «глупые» компоненты. «Умные» компоненты – это компоненты, которые тем или иным способом взаимодействуют с redux и хранилищем данных приложения. В свою очередь, «глупые» компоненты принимают все данные только через параметры и взаимодействуют с хранилищем только через «умные» компоненты. Рассмотрим некоторые из «умных» компонентов приложения, так как они представляют больший интерес для исследования.

4.4.1 Компонент LoginComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillReceiveProps – если произошел вход в приложение, производит переход к главному экрану приложения;
* componentWillMount – добавить подписку к событиям клавиатуры;
* componentWillUnmount – удалить подписку к событиям клавиатуры;
* keyboardDidShow – обработчик события открытия клавиатуры;
* keyboardDidHide – обработчик события закрытия клавиатуры;
* loginPressed – обработчик входа в приложения;
* setModalVisibility – показать или скрыть уведомление об ошибке.

Свойства компонента:

* auth – авторизационная информация из хранилища данных приложения;
* onSubmit – отправка запроса на вход в приложение.

4.4.2 Компонент RegisterComponent

Компонент для регистрации пользователя в приложении. Содержит следующие методы:

* componentWillReceiveProps – если произошел вход в приложение, производит переход к экрану входа;
* registerPressed – обработчик входа в приложения;
* setModalVisibility – показать или скрыть уведомление об ошибке.

Свойства компонента:

* auth – авторизационная информация из хранилища данных приложения;

onSubmit – отправка запроса на вход в приложение.

4.4.3 Компонент BoardComponent

Компонент для просмотра доски заданий для выбранного проекта. Содержит следующие методы:

navigationOptions – настройка отображения заголовка экрана;

componentWillMount – отправка запроса на получение задач при создании компонента;

viewTask – переход к экрану просмотра задачи;

createTask – переход к экрану создания задачи.

Свойства компонента:

* tasks – информация о списке задач из хранилища данных приложения;
* updateTask – отправка запроса на обновление задачи;
* getTasks – отправка запроса на получение задач для проекта.

4.4.4 Компонент MenuComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillMount – отправка запроса на получение информации о текущем пользователе при создании компонента;
* logOut – удалить подписку к событиям клавиатуры;
* search – переход к экрану поиска;
* registerTutor – переход к экрану регистрации пользователя.

Свойства компонента:

* profile – информация из хранилища данных приложения;
* fetchProfile – отправка запроса на получение информации о текущем пользователе;
* dispatchLogOut – отправка события выхода из аккаунта пользователя.

4.4.5 Компонент EditProfileComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillReceiveProps – если изменение данных пользователя прошло успешно, производит переход к экрану просмотра профиля;
* navigationOptions – настройка отображения заголовка экрана;
* changeImage – открытие диалога выбора изображения;
* submit – отправка запроса на изменение профиля.

Свойства компонента:

* profile – информация о текущем пользователе из хранилища данных приложения;
* newProfile – информация о статусе запроса на обновление пользователя;
* onSubmit – отправка запроса на изменение информации о пользователе.

4.4.6 Компонент ViewProfileComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillMount – отправить запрос на получение профиля пользователя при создании компонента;
* navigationOptions – настройка отображения заголовка экрана;
* openUrl – открыть ссылку в стандартном Android браузере.

Свойства компонента:

* profile – информация из хранилища данных приложения;
* getProfile – отправка запроса на получение информации о пользователе.

4.4.7 Компонент EditProjectComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillReceiveProps – если создание или изменение проекта прошло успешно, производит переход к экрану просмотра списка проектов и заполнение списка доступных пользователей для вакансий;
* navigationOptions – настройка отображения заголовка экрана;
* componentWillMount – отправить запрос на получение проекта при создании компонента;
* submit – отправка запроса на изменение проекта.

Свойства компонента:

* projects – информация о проектах из хранилища данных приложения;
* users – информация о проектах из хранилища данных приложения;
* getUsers – отправка запроса на получение списка пользователей;
* onSubmit – отправка запроса на создание или изменение информации о профиле.

4.4.8 Компонент ViewProjectComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillMount – отправить запрос на получение проекта при создании компонента;
* navigationOptions – настройка отображения заголовка экрана;
* openBoard – переход к экрану доски задач проекта;
* getFormattedDate – вывести дату в формате YYYY-MM-DD.

Свойства компонента:

* project – информация о выбранном проекте из хранилища данных приложения;
* getProject – отправка запроса на получение информации о выбранном проекте.

4.4.9 Компонент ProjectListComponent

Компонент для входа в приложение. Содержит следующие методы:

* componentWillMount – если изменение данных пользователя прошло успешно, производит переход к экрану просмотра профиля;
* createProject – переход к экрану создания проекта;
* openBoard – переход к экрану доски задач проекта;
* openProject – переход к экрану просмотра проекта.

Свойства компонента:

* projects– информация о текущем пользователе из хранилища данных приложения;
* fetchProjectsByUser – отправка запроса на получение всех проектов текущего пользователя.

5 Тестирование программного средства

Проведено тестирование программного средства. Целью данного испытания было ознакомление с программным средством и проверка его работоспособности.

Установка и тестирование программного средства производилась на мобильном устройстве с установленной операционной системой Android 5.1.

Таблица 5.1 – Набор тест-кейсов функциональности «Регистрация»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый  результат | Получен-  ный результат |
| 1 | Регистрация. Валидация ошибок | 1. Запустить приложение.  2. Нажать на кнопку «Зарегистриро-ваться».  3. Нажать на кнопку «Зарегистриро-ваться». | Сообщение под полем ввода «Электронная почта»: «Пожалуйста, введите значение поля «Электронная почта»». | Тест успешно пройден |
| 2 | Регистрация. Валидация ошибок | 1. Запустить приложение.  2. Нажать на кнопку «Зарегистриро-ваться».  3. Ввести разные значения в поля «Пароль и «Повторите пароль»  4. Нажать на кнопку «Зарегистриро-ваться». | Сообщение в всплывающем окне «Пароль и подтверждение не совпадают». | Тест успешно пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый  результат | Получен-  ный результат |
| 3 | Регистрация. Успешная регистрация. | 1. Запустить приложение.  2. Ввести верную комбинацию адреса электронной почты, пароля и подтверждения пароля  3. Нажать на кнопку «Войти» | Загружен экран входа в приложение. | Тест успешно пройден |

Таблица 5.2 – Набор тест-кейсов функциональности «Авторизация»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый  результат | Получен-  ный результат |
| 4 | Авторизация. Валидация ошибок | 1. Запустить приложение.  2. Нажать на кнопку «Войти». | Сообщение под полем ввода логина «Поле Email обязательно для заполнения». | Тест успешно пройден |
| 5 | Авторизация. Валидация ошибок | 1. Запустить приложение.  2. Ввести ложную комбинацию логина и пароля.  3. Нажать на кнопку «Войти». | Сообщение в всплывающем окне «Неправильный Email  или пароль». | Тест успешно пройден |

Продолжение таблицы 5.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 6 | Авторизация. Успешная авторизация. | 1. Запустить приложение.  2. Ввести верную комбинацию логина и пароля  3. Нажать на кнопку «Войти» | Загружен главный экран приложения. | Тест успешно пройден |

Следующие наборы тест-кейсов рассматриваются с выполненным шагом ввода корректного пароля и логина.

Таблица 5.3 – Набор тест-кейсов функциональности «Проект»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 7 | Проект. Просмотр информации о проекте. | 1.Нажать на карточку проекта на странице «Ваши проекты» | 1. Отображение экрана информации о проекте.  2. Заголовок страницы отображает название проекта | Тест успешно пройден |
| 8 | Проект. Создание проекта. Выбор навыка и работника при создании позиции на проекте. | 1. Нажать на иконку «+» в заголовке экрана «Ваши проекты»  2. В выпадающих списках выбрать любые значения | 1. Отображение формы создания проекта.  2. Подгруженные списки работников и типов навыков.  3. Корректное отображение списка навыков и работников.  4. После выбора навыка или работника фиксируется нужные значения. | Тест успешно пройден |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 9 | Проект. Создание проекта. Валидация корректных данных. | 1. Выполнить действия из пункта 8.  2. Заполнить корректно остальные поля.  3. Нажать на кнопку «Создать». | 1. Отсутствие сообщений об ошибках создания проекта.  2. Проект добавлен в список проектов. | Тест успешно пройден |
| 10 | Проект. Редактирова-  ние проекта. Загрузка формы с изменяемыми данными. | 1. Выполнить действия из пункта 7.  2. Выбор иконки редактирования проекта в правом углу заголовка экрана. | 1. Отображение формы редактирования проекта.  2. Подгружены списки пользователей, типов навыков.  3. Корректно отображены остальные поля, заполнены старыми значениями. | Тест успешно пройден |
| 11 | Проект. Редактирова-  ние проекта. Сохранение измененных данных. | 1. Повторение действий из пункта 10.  2. Внести изменения в каждое поле и выпадающий список.  3. Нажать кнопку «Сохранить». | 1. Отображение формы редактирования проекта.  2. Подгружены списки пользователей, типов навыков.  3. Корректно отображаются внесенные изменения на форме.  4. Проект успешно сохранен и отображен в списке проектов с измененными данными. | Тест успешно пройден |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 12 | Проект. Удаление проекта. | 1. Выполнить действия из пункта 7.  2. Ввести в поле текстового ввода название проекта.  3. Нажать «Удалить проект». | 1. Переход кнопки «Удалить проект» в активное состояние, если имя проекта совпадает с введенным.  2. Отображение сообщения с текстом «Проект успешно удален!»  3. Переход на экран списка проектов после нажатия кнопки «Ок» | Тест успешно пройден |

Таблица 5.4 – Набор тест-кейсов функциональности «Пользователь»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 13 | Пользователь. Просмотр данных пользователя | 1. Открыть левое боковое меню.  2. Нажать «Профиль» | 1. Отображение экрана просмотра профиля. | Тест успешно пройден |
| 14 | Пользователь. Редактирование данных пользователя | 1. Повторение действий из пункта 10.  2. Выбор иконки редактирования проекта в правом углу заголовка экрана. | 1. Отображение формы редактирования проекта.  2. Подгружены списки университетов, факультетов, типов аккаунтов.  3.Корректно отображены остальные поля, заполнены старыми значениями. | Тест успешно пройден |

Продолжение таблицы 5.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 15 | Пользователь. Редактирование данных пользователя.Валидация | 1. Повторение действий из пункта 14.  2. Удалить содержимое поля «Электронная почта».  3. Нажать кнопку «Сохранить». | 1. Сообщение под полем «Электронная почта»: «Пожалуйста, введите значение поля «Электронная почта»». | Тест успешно пройден |
| 16 | Пользователь. Редактирова-  ние данных пользователя. Добавление навыка. | 1. Повторение действий из пункта 14.  2. Внести название навыка в соответствующее поле.  3. Нажать кнопку с знаком «+». | 1. Навык добавится в список навыков на экране изменения профиля пользователя | Тест успешно пройден |
| 17 | Пользователь. Редактирова-  ние данных пользователя. Добавление аккаунта социальных сетей. | 1. Повторение действий из пункта 14.  2. Внести ссылку на аккаунт в соответствую-  щее поле.  2. Выбрать тип аккаунта.  3. Нажать кнопку «Добавить». | 1. Аккаунт добавится в список аккаунтов на экране изменения профиля пользователя | Тест успешно пройден |

Продолжение таблицы 5.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 18 | Пользователь. Редактирова-  ние данных пользователя. Сохранение измененных данных. | 1. Повторение действий из пункта 14.  2. Внести изменения в каждое поле и список.  3. Нажать кнопку «Сохранить». | 1. Отображение формы редактирования профиля.  2. Подгружены списки университетов, факультетов, типов аккаунтов.  3. Корректно отображаются внесенные изменения на форме.  4. Профиль успешно сохранен и отображен на экране просмотра профиля пользователя с измененными данными. | Тест успешно пройден |

Таблица 5.5 – Набор тест-кейсов функциональности «Задачи»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 19 | Задачи. Просмотр списка задач. | 1.Нажать на кнопку «Открыть доску заданий» на активном проекте на экране «Ваши проекты» | 1. Отображение доски заданий с списком задач. | Тест успешно пройден |
| 20 | Задачи. Просмотр информации о задаче. | 1. Повторение действий из пункта 19.  2. Нажать на карточку задачи | 1. Отображение экрана информации о задаче.  2. Заголовок страницы отображает название задачи | Тест успешно пройден |

Продолжение таблицы 5.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 21 | Задачи. Создание задачи. Выбор типа задачи, приоритет и исполнителя при создании задачи. | 1. Повторение действий из пункта 19.  2.Нажать кнопку «Добавить задание»  3. В выпадающих списках выбрать любые значения | 1. Отображение формы создания задачи.  2. Подгруженные списки типов задач, приоритетов и участников проекта, и корректное их отображение.  3. После выбора элемента списка фиксируется нужные значения. | Тест успешно пройден |
| 22 | Задачи. Создание задачи. Валидация корректных данных. | 1. Выполнить действия из пункта 21.  2. Заполнить корректно остальные поля.  3. Нажать на кнопку «Создать». | 1. Отсутствие сообщений об ошибках создания задачи.  2. Задача добавлена на доску задач. | Тест успешно пройден |
| 23 | Задачи. Редактирова-  ние задачи. Загрузка формы с изменяемыми данными. | 1. Выполнить действия из пункта 20.  2. Нажать на кнопку редактирования задачи в правом углу заголовка экрана. | 1. Отображение формы редактирования проекта.  2. Подгруженные списки типов задач, приоритетов и участников проекта, и корректное их отображение.  3. Корректно отображены остальные поля, заполнены старыми значениями. | Тест успешно пройден |

Продолжение таблицы 5.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональ-  ность | Последователь-  ность действий | Ожидаемый результат | Получен-  ный результат |
| 24 | Задачи. Редактирова-  ние проекта. Сохранение измененных данных. | 1. Повторение действий из пункта 23.  2. Внести изменения в каждое поле и выпадающий список.  3. Нажать кнопку «Сохранить». | 1. Отображение формы редактирования задачи.  2. Корректно отображаются внесенные изменения на форме.  3. Задача успешно сохранена и отображена в списке задач с измененными данными. | Тест успешно пройден |
| 25 | Задачи. Удаление задачи. | 1. Выполнить действия из пункта 20.  2. Нажать «Удалить задачу». | 1. Отображение сообщения с текстом «задача успешно удалена»  2. Переход на экран списка задач после нажатия кнопки «Ок» | Тест успешно пройден |
| 26 | Задачи. Смена статуса задачи. | 1. Выполнить действия из пункта 19.  2. Нажать кнопку «⋮..».  3. Выбрать статус из предложенных | 1. Перемещение задачи в соответствующую новому статусу колонку | Тест успешно пройден |

6 Руководство по установке и использованию

6.1 Руководство по установке

Данное приложение было разработано для ОС Android. Таким образом, можно выделить 2 способа его установки: с помощью установочного apk файла и с помощью Google Play Market.

Для установки с помощью установочного файла необходимо скачать установочный файл в память телефона. Для этого можно воспользоваться USB-кабелем и компьютером. Необходимо подключить утройство к компьютеру в режиме подключения «USB-носитель» или аналогичном ему и скопировать установочный файл в память устройства. Далее необходимо разрешить установку из сторонних источников, если эта опция отключена. Данная опция находится в настройках безопасности устройства либов в настройках приложенией. В зависимости от производителя устройства путь к данной опции может различаться. Следующим шагом будет установка приложение из установочного файла. С помощью файлового менеджера необходимо найти установочный файл и запустить.

Google Play – официальный и крупнейший магазин приложений для Android, по умолчанию установлен на большистве устройств на базе данный ОС. Для установки через магазин приложений необходимо иметь аккаунт Google. Устройство предложит его ввести при первоначальной активации устройства, иначе его можно задать в настройках. Для установки приложения необходимо открыть приложение Google Play Market, с помощью поиска найти приложение «Coworkio» и нажать кнопку «Установить».

6.2 Руководство по использованию

6.2.1 Экран регистрации пользователя

При нажатии кнопки «Зарегистрироваться» открывается экран регистрации пользователя. Для успешной регистрации необходимо ввести имя, фамилию, адрес электронной почты, пароль и подтверждение пароля. Все поля обязательны к заполнению. Значение полей пароль и подтверждение пароля должны совпадать. После успешной регистрации пользователь перенаправляется на экран входа в систему с предзаполненными полями логина и пароля.

6.2.2 Экран входа в систему

Работа с сайтом начинается со экрана, где можно авторизоваться для дальнейшей работы (смотреть рисунок 6.1).

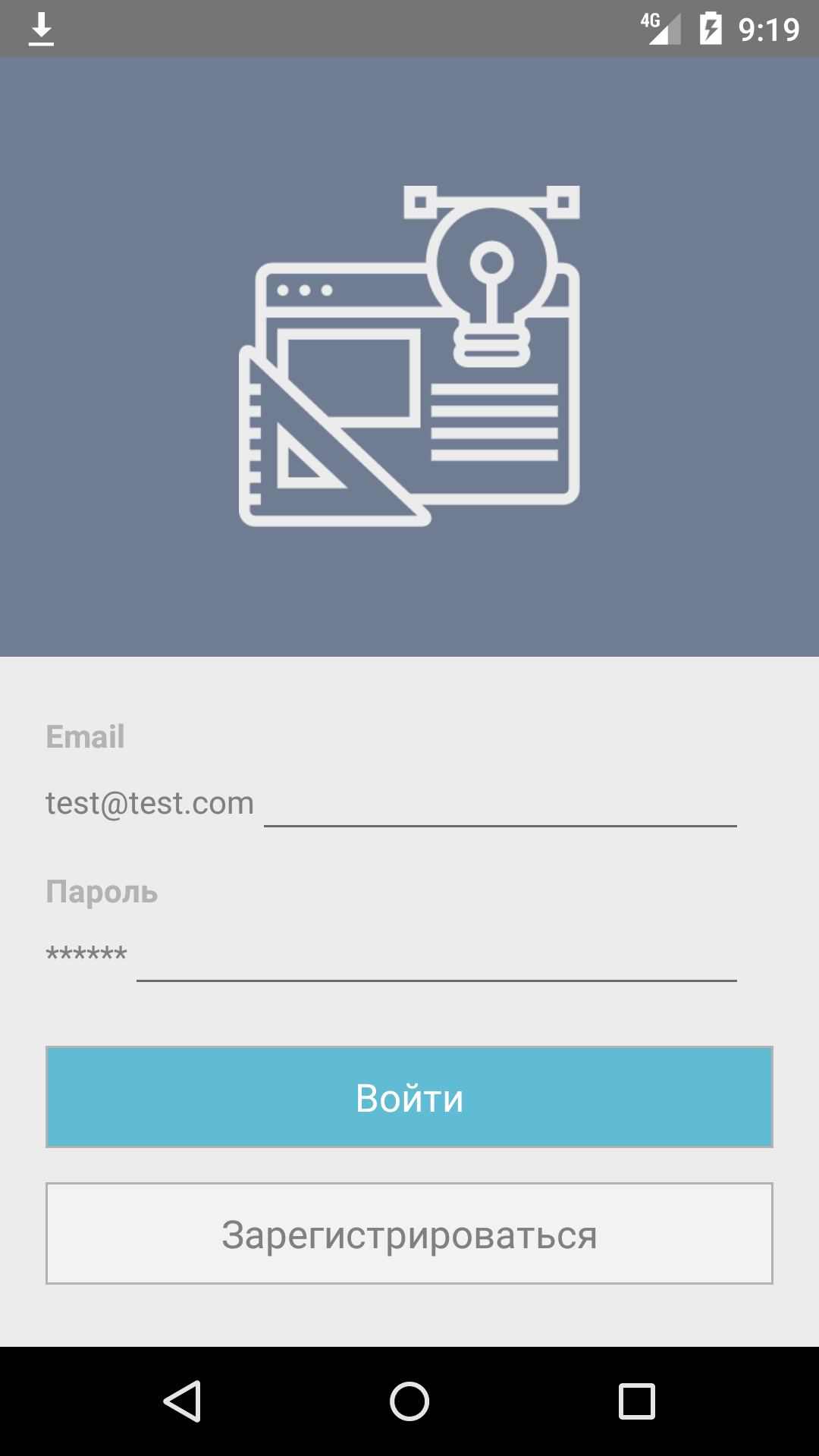


Рисунок 6.1 – Экран входа в систему

После авторизации осуществляется переход на страницу с списком проектов пользователя.

6.2.3 Экран списка проектов пользователя

На данном экране пользователь видит полный список активных и завершенных проектов (смотреть рисунок 6.2).

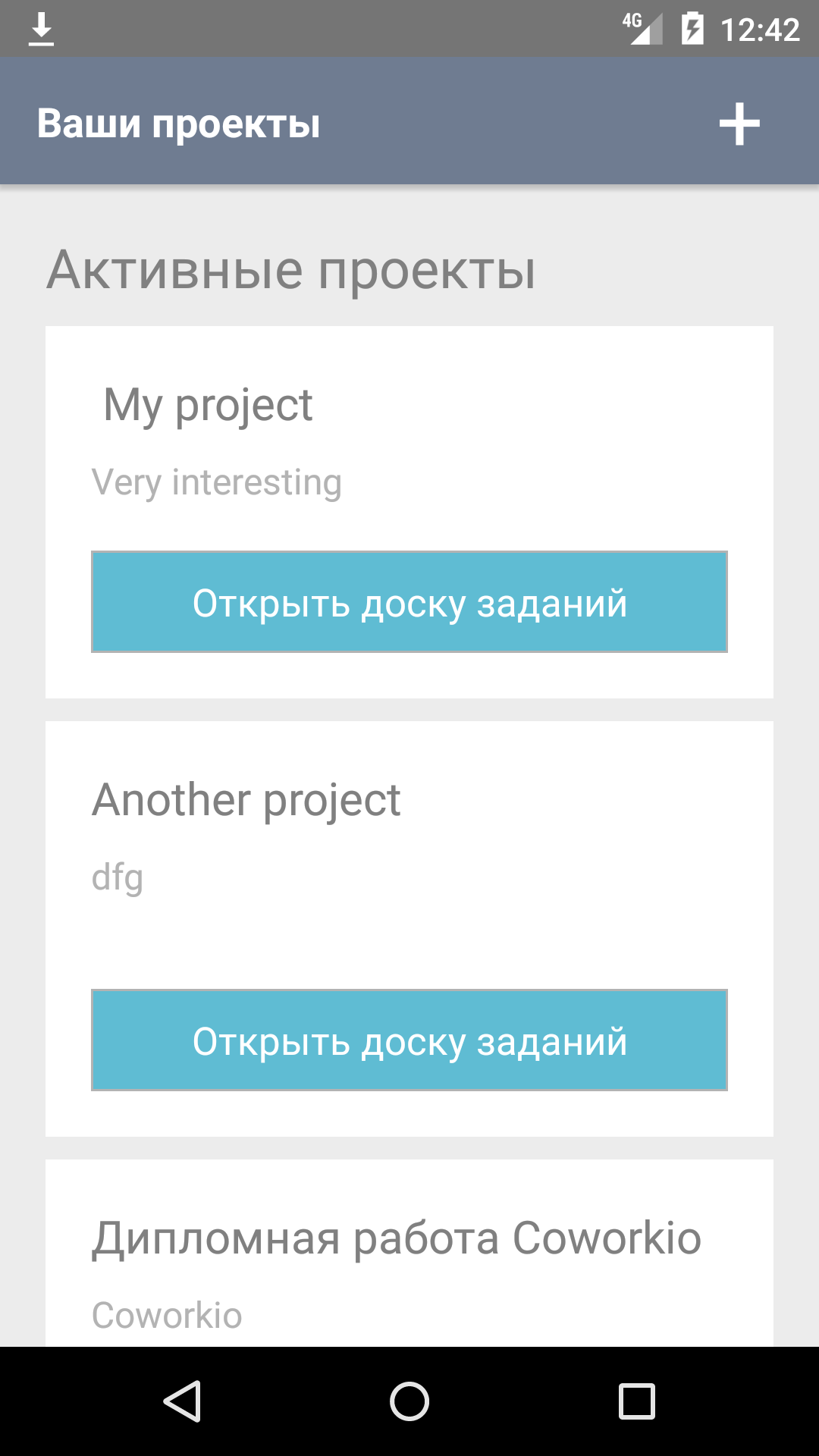


Рисунок 6.2 – Экран списка проектов пользователя

Проект считается активным, если сроки проекта не истекли на настоящий момент времени. При нажатии на карточку проекта происходит переход к экрану детального описания проекта. При нажатии на кнопку «Открыть доску задания» происходит переход к экрану просмотра доски заданий. При нажатии на символ «+» в заголовке экрана происходит переход к экранусоздания нового проекта. При протягивании слева направо от левого края экрана открывается боковое меню (рисунок 6.3).

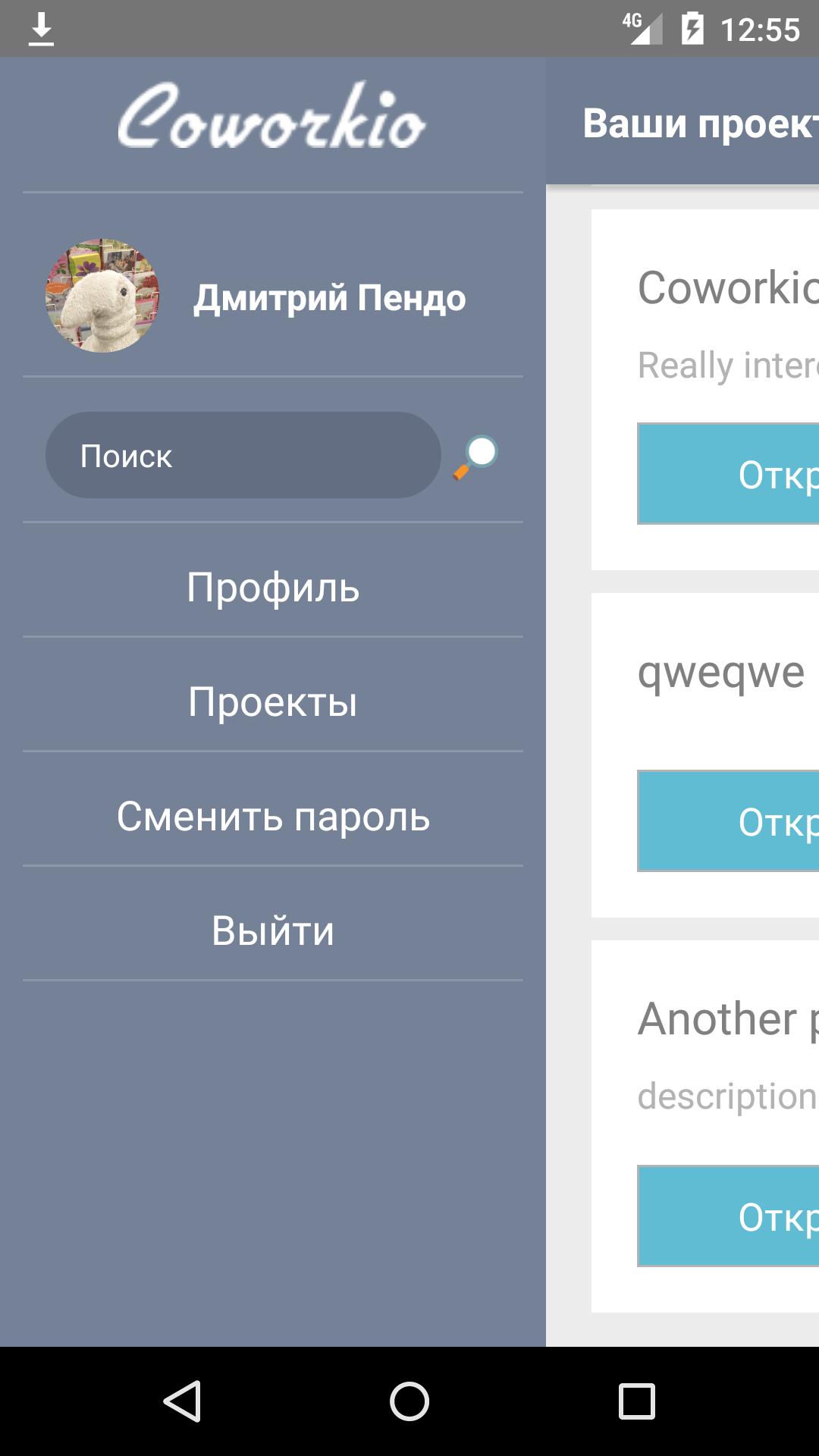


Рисунок 6.3 – Внешний вид открытого бокового меню

6.2.4 Боковое меню

Боковое меню предоставляет удобный способ перемещаться на экраны поиска, информации о профиле пользователя, списка проектов пользователя, мены пароля и функцию выхода из системы. В первом блоке после логотипа приложения размещается картинка профиля пользователя и имя и фамилия пользователя. Если пользователь имеет роль «преподаватель», то в список ссылок в меню добавляется переход к экрану регистрации для регистрации другого пользователя с ролью «преподаватель».

6.2.5 Экран создания и редактирования проекта

Экран создания и редактирования проекта (рисунок 6.4) предназначен для создания и внесения модификаций в описание уже существующих проектов. На этом экране можно создавать вакансии, назначать людей на уже существующие вакансии, изменять и удалять их. После нажатия кнопки «Сохранить», данные отсылаются на сервер и в случае успешного завершения операции происходит переход на экран списка проектов.

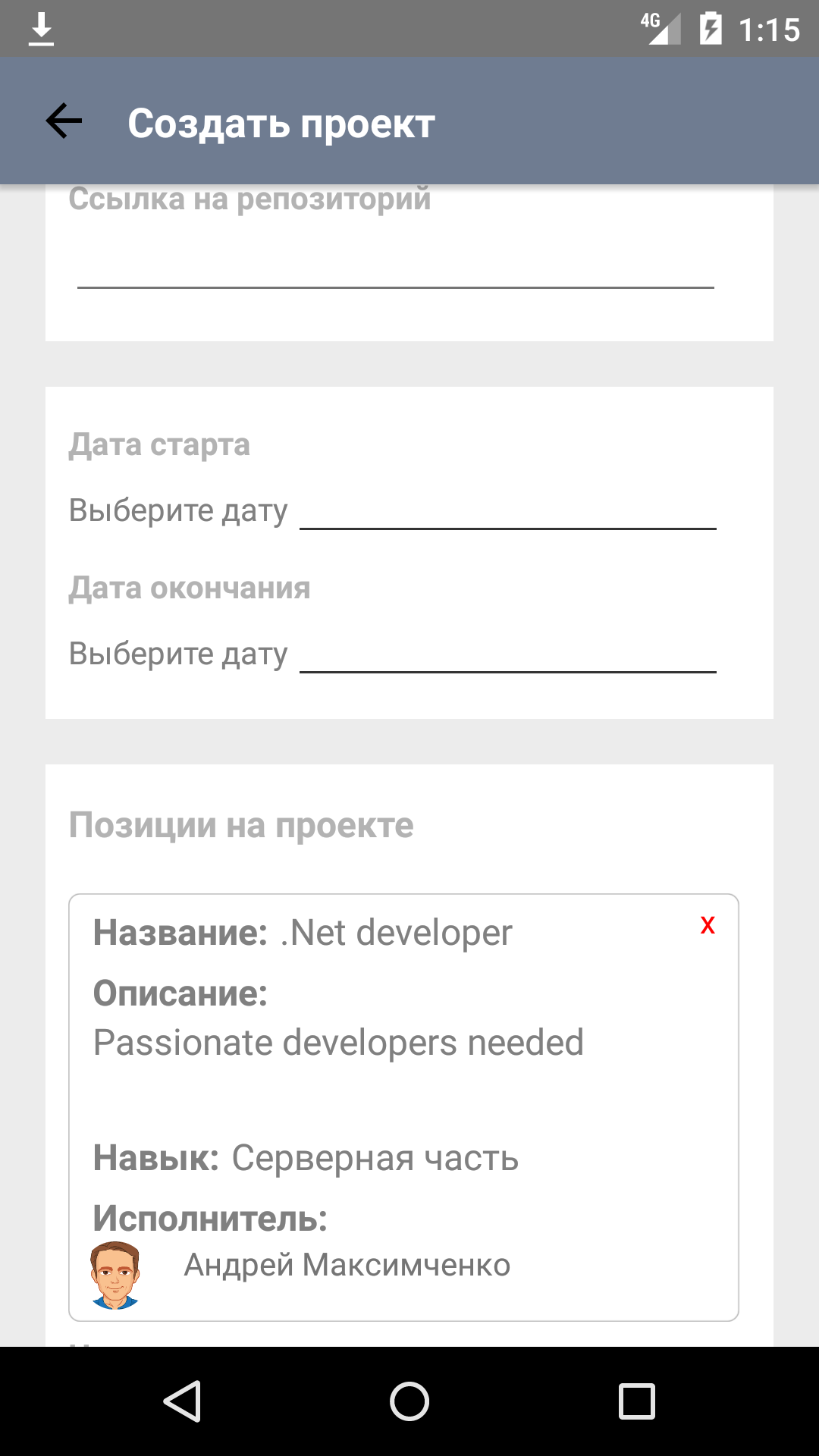


Рисунок 6.4 – Экран создания и редактирования проекта

6.2.6 Экран просмотра информации о проекте

Экран просмотра информации о проекте представлен на рисунке 6.5. На данном экране отображена основная информация о проекте, включая название, описание, дату начала и окончания, ссылку на репозиторий и количество спринтов. При наличии позиций на проекте, кроме создателя проекта, отображается секция «Позиции». В ней можно просмотреть детальную информацию о каждой позиции и перейти к профилю человека, занимающего позицию. При нажатии на кнопку «Открыть доску заданий» происходит переход на экран просмотра доски задач.

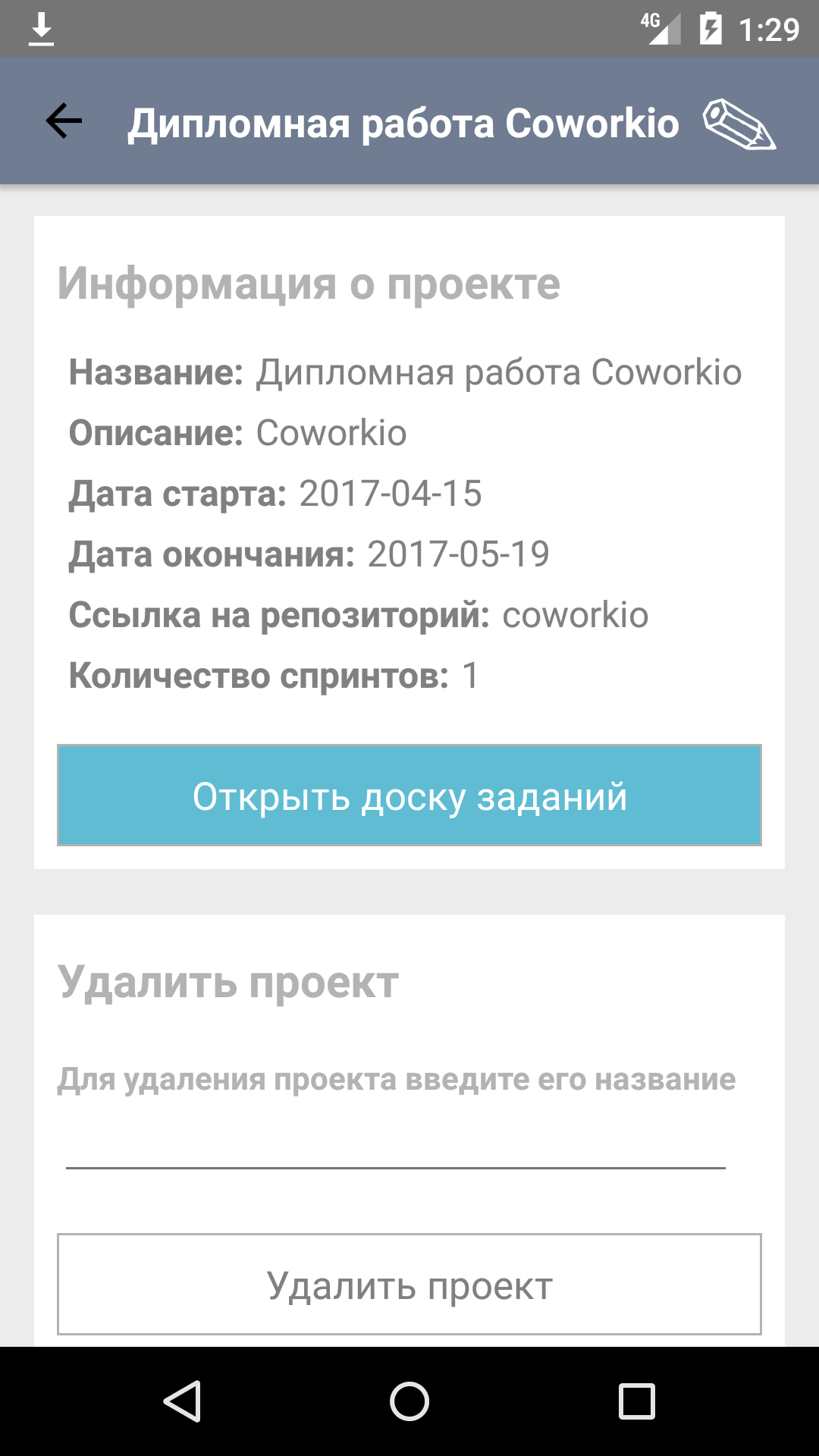


Рисунок 6.5 – Экран просмотра информации о проекте

Также на этом экране есть возможность удалить проект. Для этого в соответствующее поле необходимо ввести название проекта и нажать кнопку «Удалить проект». Кнопка будет оставаться заблокированной до тех пор, пока введенный текст не будет совпадать с названием проекта. При нажатии на иконку редактирования в заголовке экрана происходит переход к экрану редактирования проекта.

6.2.7 Экран просмотра доски заданий

Экран просмотра доски заданий (смотреть рисунок 6.6) является основным в процессе управления задачами проекта. На данном экране отображаются все задачи, сгруппированные по спринтам и колонкам статусов. При нажатии на карточку задания происходит переход на экран просмотра деталей задания. При нажатии на кнопку в левой части карточки задания отображается выпадающий список с названия ми колонок, в которые можно перенести задачу. При нажатии на кнопку «Добавить задание» происходит переход на экран создания задачи. Задача будет создана в той колонке, в который находится выбранная кнопка «Добавить задание». В кадом заголовке столбца отображается текущее количество задач в нем. Каждая задача имеет цветовой индикатор в левой чати карточки, означающий приоритет задачи. К примеру задача с низким приоритетом имеет зеленый индикатор.

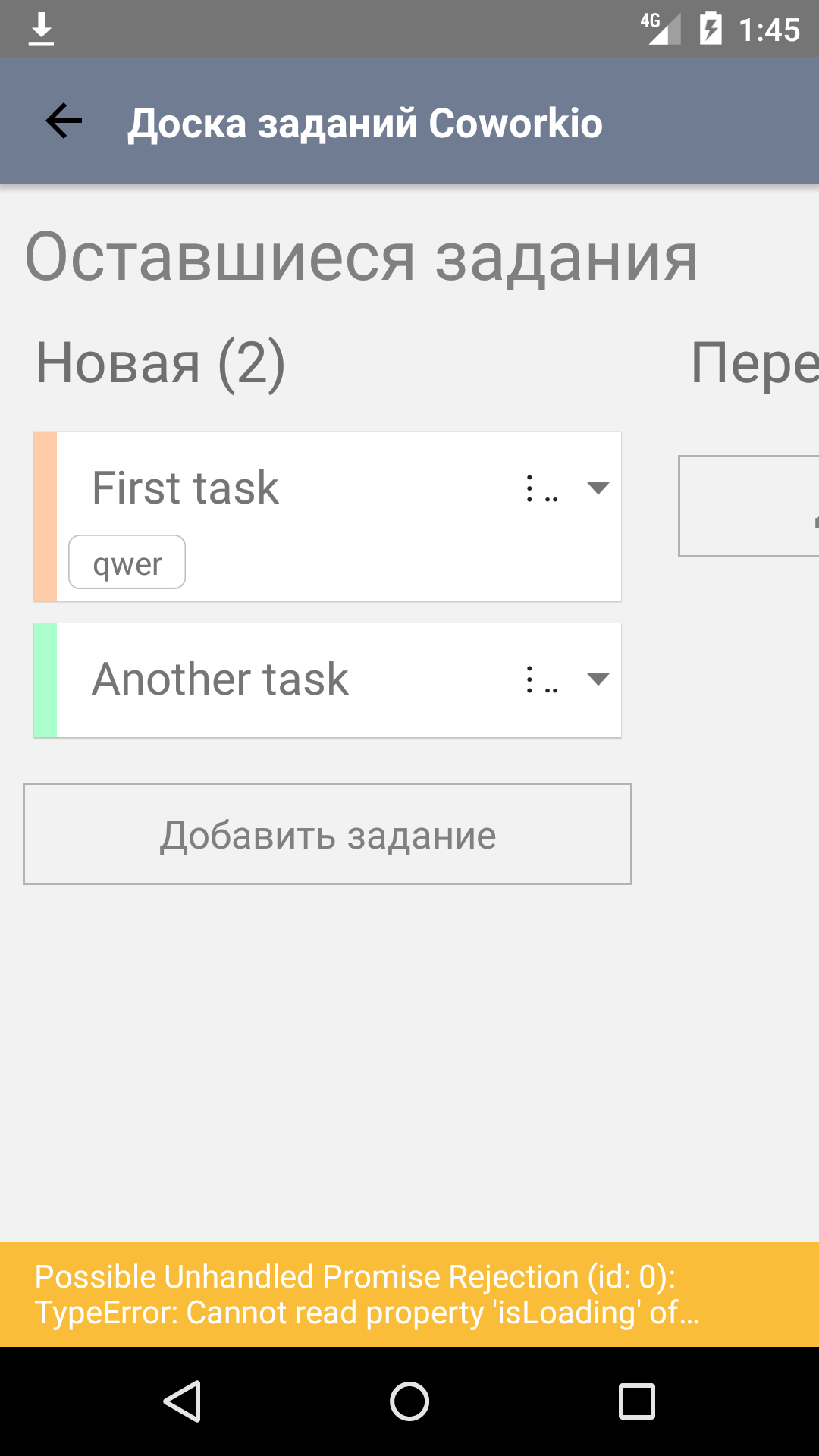


Рисунок 6.6 – Экран просмотра доски заданий

6.2.8 Экран просмотра информации о задаче

Экран просмотра информации о задаче (смотреть рисунок 6.7) отображает детальную информацию о выбранной задаче.

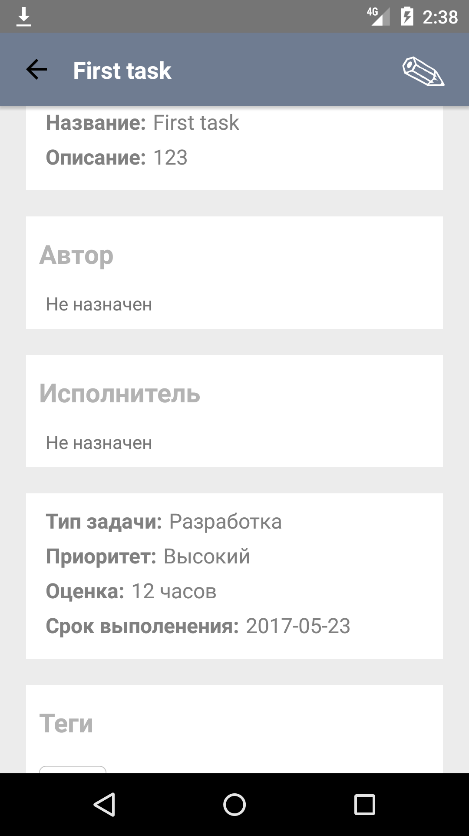


Рисунок 6.7 – Экран просмотра информации о задаче

На данном экране отображаются все поля, характеризующие задачу, сгруппированные по логическим блокам. Так в блоке общая информация указывается название о писание задачи. Далее, при наличии информации а авторе и исполнителе задачи отображаяется краткая информация о них: имя, фамилия, изображение профиля пользователя. Далее располагается секция о типе задачи, ее приоритете и оценке. При наличии добавленных тегов к задаче, они отображаются а отдельной карточке с помощью перечисления. В конце страницы располагается секция комментариев. Пользователь может добавить комментарий ввея его в соответвующее текстовое поле и нажав кнопку «Добавить». Пользователь, который создал комментарий может его удалить. В заголовке стрницы отображается имя задачи, просматриваемой в текущий момент. При нажатии на кнопку с значком редактирования, расположенную в заголовке страницы происходит переход на страницу изменения данных задачи.

6.2.9 Экран создания и редактирования задачи

Экран создания и редактирования задачи (смотреть рисунок 6.6) предназначен для создания задач и внесения изменений в существующие.

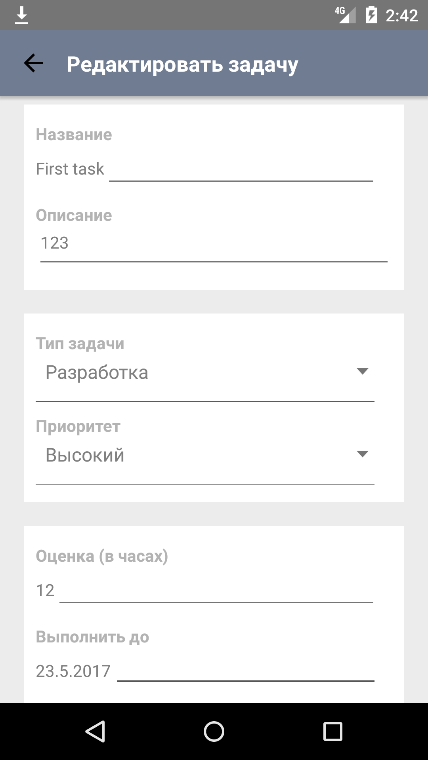


Рисунок 6.8 – Экран просмотра доски заданий

При открытии экрана в режиме редактирования, все исеющиеся данные будут предзагружены заранее и отобразятся в полях формы. При нажатии на кнопку «Сохранить» будет отослан запрос на сохранение данных и, в случае успешного завершения, пользователь будет перенаправлен на доску задач.

6.2.10 Экран просмотра профиля пользователя

Экран просмотра профиля пользоватлея (смотреть рисунок 6.9) является основным в процессе управления личными данными. На данном экране отображаются вся личная информация, которая была введена в приложение, сгруппированная в логические блоки в виде карточек. При нажатии на кнопку с значком редактирования, расположенную в заголовке страницы происходит переход на страницу изменения данных профиля.

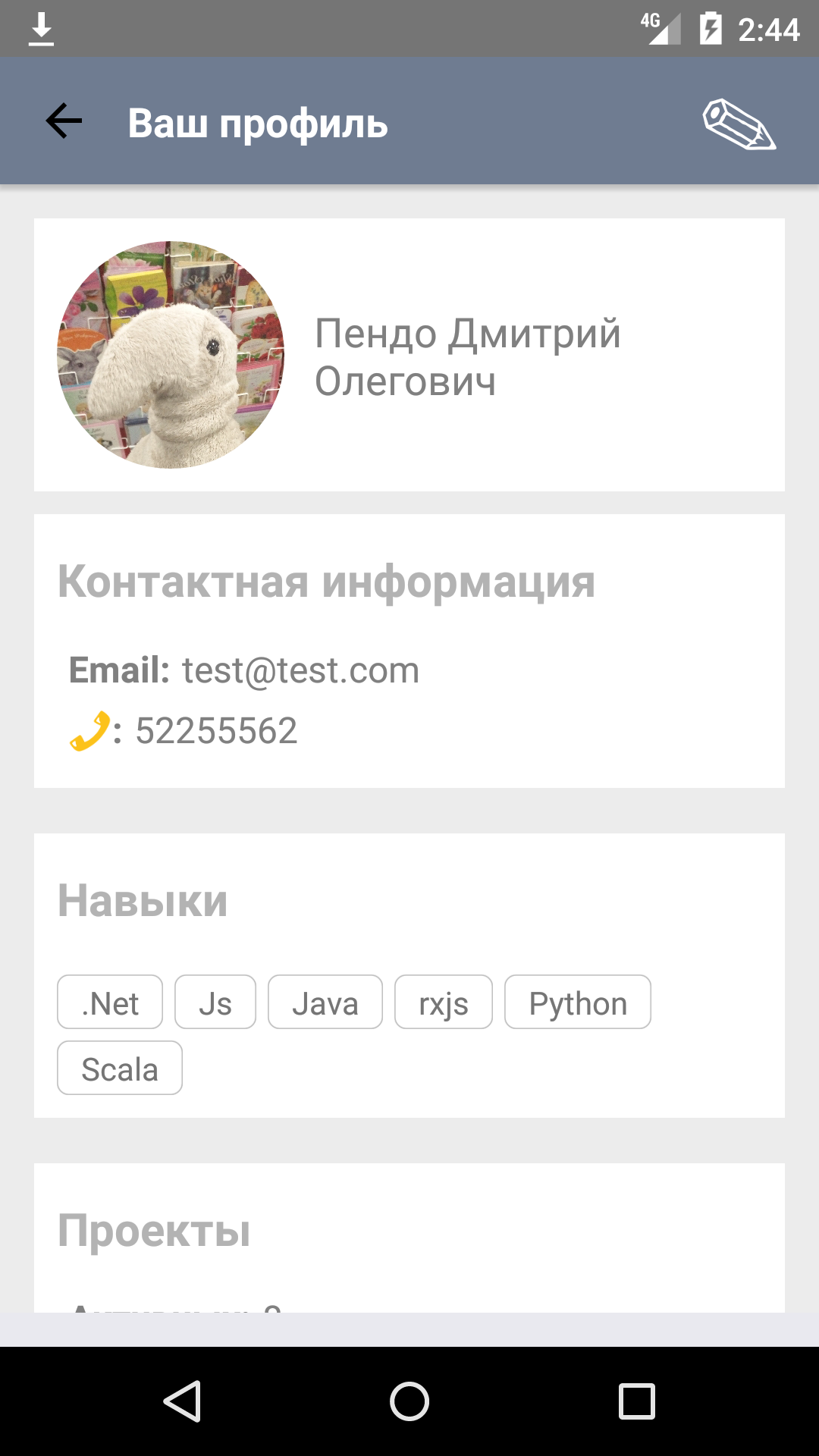


Рисунок 6.9 – Экран просмотра профиля пользователя

6.2.11 Экран редактирования данных профиля

Экран редактирования данных профиля (смотреть рисунок 6.10) позволяет вносить изменения в личные данные аккаунта пользователя. Так, в профиле пользователя можно установить картинку профиля, добавить отчесто к имени и фамилии, введенным при регистрации, номер телефона для связи, ссылку на аккаунт GitHub и аккаунты социальных сетей. Кроме того, для облегчения поиска проекта можно добавить основные навыки, которыми владеет пользователь. Последним логическим блоком данных пользователя являются данные об университете. Пользователь может выбрать университет, факультет, ввести название специальности, номер группы и срок обучения.

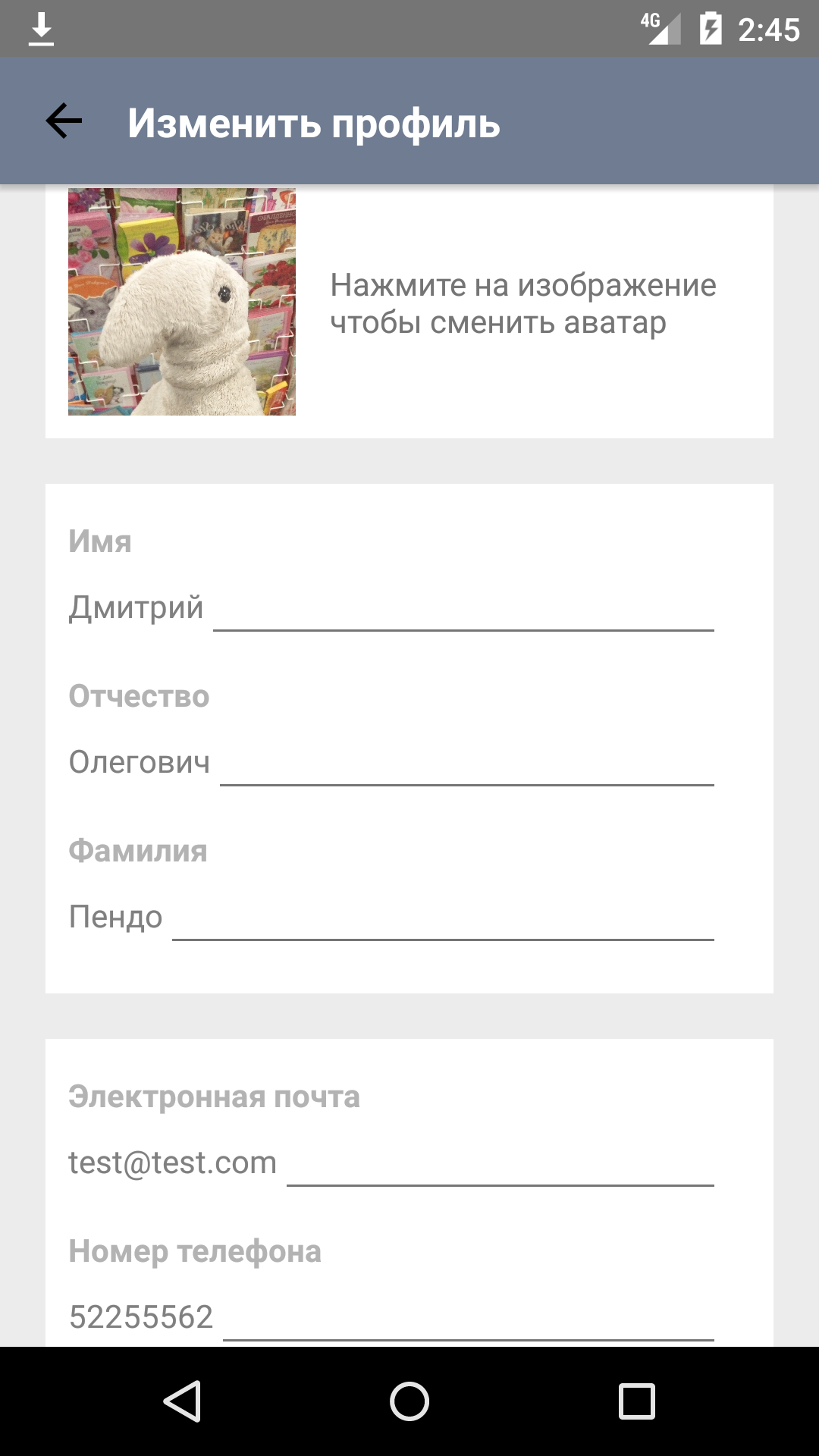
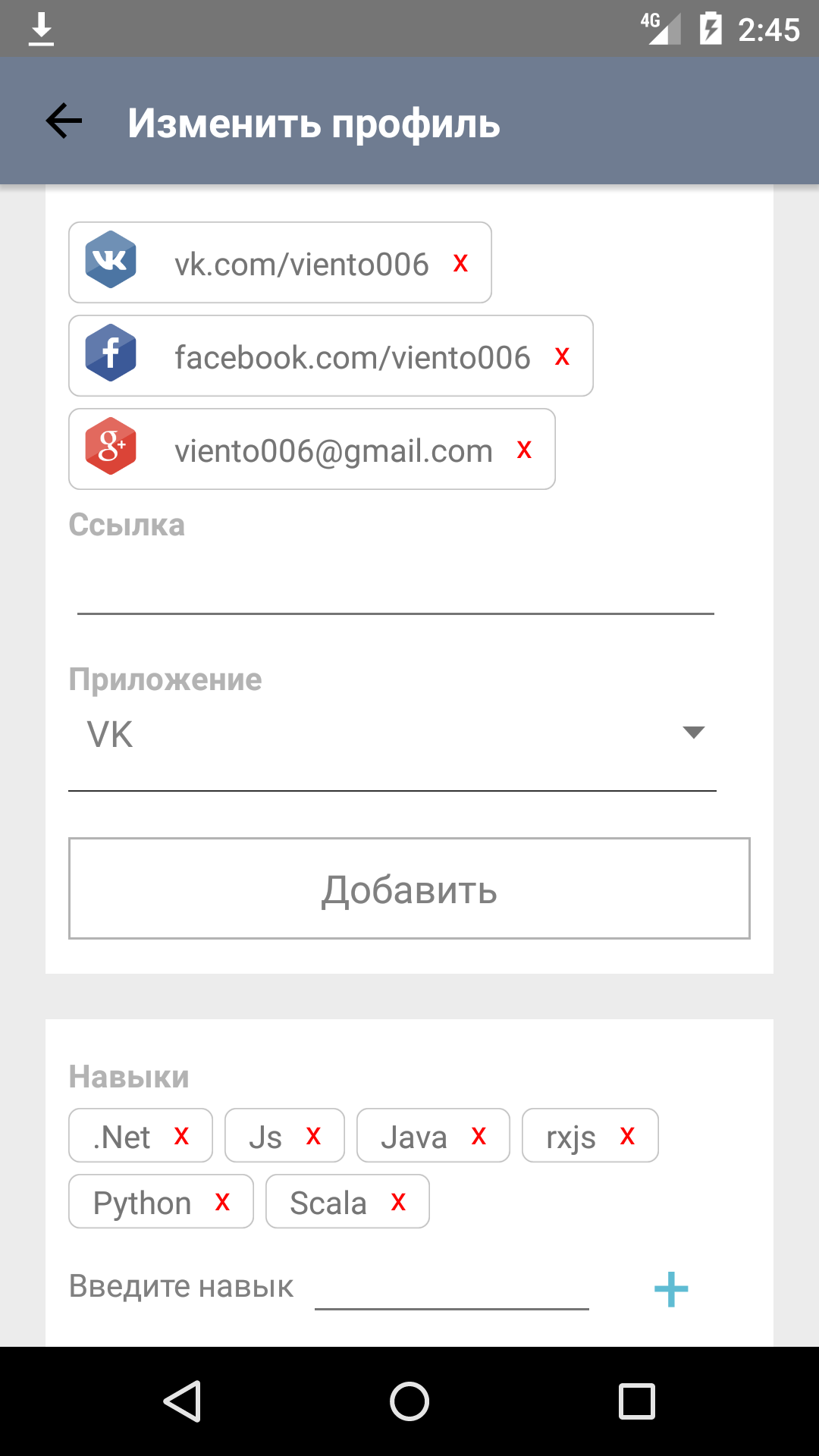


Рисунок 6.10 – Экран редактирования данных профиля

Для смены изображение профиля пользователя нажмите на изображение. Далее откроется диалог выбора изображения (смотреть рисунок 6.11). После выбора изображения и обрезки его под подходящий формат, изображение отправляется на сервер и, через некоторое время, если отправка прошла успешно, картинка предпросмотра изображения обновится на выбранную.



7 Технико-экономическое обоснование

эффективности разработки и внедрения ПС

7.1. Характеристика программного продукта

Программное средство, разрабатываемое в рамках рассматриваемого дипломного проекта, предназначено для управления процессом разработки дипломных и курсовых проектов. Основная цель заключается в объединении функционала приложения, позволяющего организовывать команды студентов для совместной работы над курсовыми либо дипломными работами, управлять процессом разработки и контролировать его ход руководителю проекта, являющемуся сотрудником университета.

Особенностью проекта является то, что студенты-пользователи данного программного средства, ознакамливаются с процессами управления проектами на практике в рамках курсовых и дипломных работ и получают ценный опыт командной разработки. Это позволит при дальнейшем трудоустройстве на работу максимально подготовить студентов к командной работе над проектами.

Воспользоваться программным средством смогут как и одиночные разработчики, проекты которых не предусматривают командную работу по тем либо иным причинам, так и группы учащихся, которые могут объединяться в команды как по договоренности, так и путем подбора кандидатов в команды с помощью встроенного поиска кандидатов на позиции проекта. Также данный проект несет пользу и для руководителя проекта от университета, позволяя ему прозрачно наблюдать за выполнением заданий в установленные сроки и помогая при затруднениях.

7.2. Расчет затрат и отпускной цены программного средства

7.2.1. Объем ПО (строки исходного кода, LOC)

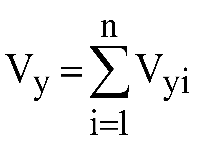


Таблица 7.1 - Перечень и объем функций программного модуля

|  |  |  |  |
| --- | --- | --- | --- |
| № функции | Наименование (содержание) функции | Объем функции (LОС) | |
| По каталогу Vi | УточненныйVyi |
| 1 | 2 | 3 | 4 |
| 101 | Организация ввода информации | 150 | 110 |
| 104 | Преобразование операторов входного языка и команды другого языка | 980 | 960 |
| 109 | Организация ввода/вывода информации в интерактивном режиме | 320 | 200 |
| 207 | Манипулирование данными | 9550 | 5750 |
| 506 | Обработка ошибочных и сбойных ситуаций | 430 | 610 |
| 707 | Графический вывод результатов | 300 | 1460 |
|  | Итого | 11730 | 9090 |

Среда разработки ПО (Visual Studio Сode), ПО функционального назначения. Vi = 11730 LOC.

1.2.2. Трудоемкость разработки ПО

Категории сложности ПО – 3-я категория, Vi = 11730 LOC;  
Нормативная трудоемкость ПО (Tн) – 260 чел./дн.;

Коэффициент сложности (Kc) – 0,12;

Коэффициент, учитывающий степень использования при разработке

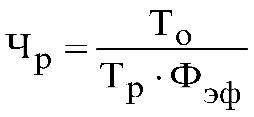
ПО стандартных модулей (Kт) – 0,6;

Коэффициент новизны разрабатываемого ПО (Kн) – 0,7;

Общая трудоемкость:

Tо = Tн⋅Kс⋅Kт⋅Kн = 260⋅1,12⋅0,6⋅0,7 = 122 чел./дн.;

1.2.3. Численность исполнителей проекта (Чp)

,

где Tp− срок разработки проекта (лет);

Эффективный фонд времени работы одного работника (Фэф):

Фэф = Дг-Дп-Дв-До = 236 дн.;

Срок разработки установлен 3 мес. (Tp = 0,25 г.):

https://lh3.googleusercontent.com/byTXgftj56AruBuBExgluMwl92FNVNVvw8zBIMtpzoWDvSFM9LLpdZ9MlevzcEjPRFsYhXBDKjBatVMFeMbFn6P_Ge_12Wwy4HkOuqRM-AMQfZAWtb6MogV3b7kvUeYEnfM8wfCH.

**1.3.** Расчет сметы затрат и цены заказного ПО

1. Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

                           Зоi  =   ∑ni=1 \* Зci  \* Фрi \* К,

где n− количество исполнителей, занятых разработкой конкретного ПО;

Зci −среднедневная заработная плата i-го исполнителя (*д.е.*);

Фрi − плановый фонд рабочего времени i-го исполнителя (дн.);

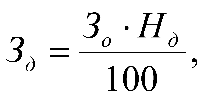
К − коэффициент премирования (1,2).

Расчет основной заработной платы представлен в табл. 1.3.1.

Таблица 1.3.1**.** Расчет основной заработной платы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Средне-  дневная заработная плата, руб. | Плановый фонд рабочего времени, дн. | Коэф-  фициент преми-  рования | Основная заработ-  ная плата, руб. |
| Ведущий JavaScript разработчик | 40 | 61 | 1,2 | 2928 |
| Разработчик программного средства на языке программирования JavaScript | 35 | 61 | 1,2 | 2562 |
| Итого с премией (20%), Зо | - | - | - | 5490 |

2. Дополнительная заработная плата исполнителей проекта определяется по формуле:

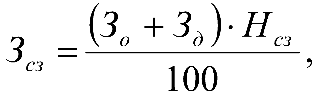


где НД – норматив дополнительной заработной платы (20%).

Дополнительная заработная плата составит:

Зд = 5490·20/100= 1098 руб.

3. Отчисления в фонд социальной защиты населения и на обязательное страхование (ЗС) определяются в соответствии с действующими законодательными актами по формуле:



где НСЗ – норматив отчислений в фонд социальной защиты населения и на обязательное страхование (34 +0,6%).

Зсз = (5490 +1098) ·34,6 / 100=2279.45 руб.

4. Расходы по статье «Машинное время» (РМ) включают оплату машинного времени, необходимого для разработки и отладки ПС, и определяются по формуле:

Рм = Цм \* Тч \* Ср,

где Цм – цена одного машино-часа;

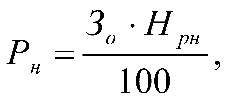
Тч – количество часов работы в день;

Ср – длительность проекта.

Стоимость машино-часа на предприятии составляет 1.4 руб.. Разработка проекта займет 61 день. Определим затраты по статье “Машинное время”:

Рм=1.4·8·61=683,2 руб.

5. Затраты по статье «Накладные расходы» (РН), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (РН), определяются по формуле



где НРН – норматив накладных расходов (50%).

Накладные расходы составят:

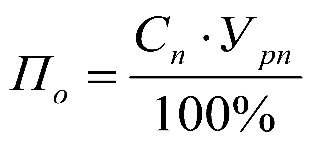
Рн= 5490· 0.5=2745 руб.

Общая сумма расходов по всем статьям сметы (Сп) на ПО рассчитывается по формуле:

https://lh6.googleusercontent.com/fdpGt_QBU8kuGcnyVr_gs6jvMRmCae7Yti6ncV8FnKMkW5N6bNTXCm6uuZasqaQDE50CtGvEdqsZty4zVQiwXPhMnw_5pWjNxcvoNpytpWY_dziZixwG_p1LLr3QQC2auU5hCtk5

Сn =5490 + 1098 + 2279.45 + 683,2 + 2745 = 12295,65 руб.

Прибыль ПС рассчитывается по формуле:

,

где Ппс – прибыль от реализации ПС заказчику (руб.);

УР – уровень рентабельности ПС (25%);

СП – себестоимость ПС (руб.).

Ппс =12295,65 · 25/100= 3073,91 руб.

Прогнозируемая отпускная цена ПС вычисляется по формуле:

https://lh5.googleusercontent.com/EjHlKebv_eURZM_yXVYopMekWg6OWEz56KCmhZzLmxbgSIWsmwPbTq1cKzqu450_vOHbmVnZlE8dGQVlutocvf9L_gQE7KgbSuMo4a2qKm6avJ4zyQ80RH_ib8c2PWkNX_GjGAqz.

Цп =12295,65 + 3073,91 = 15369,56 руб.

Налог на добавленную стоимость (НДСi):

НДС= ЦП \* НДС : 100,

где Hдc− норматив НДС (%).

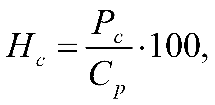
НДС= 15369,56 \* 20% : 100% = 3073,91 руб.

Прогнозируемая отпускная цена (Цo):

Цo = Цп+ НДС

Цo = 15369,56 + 3073,91= 18443,47 руб.

Кроме того, организация-разработчик осуществляет затраты на сопровождение ПС (РС), которые определяются по нормативу (НС), где Нс – норматив расходов на сопровождение и адаптацию (20%).



где Рс – расходы на сопровождение и адаптацию ПС в целом по организации (руб.);

СР – смета расходов в целом по организации без расходов на сопровождение и адаптацию (руб.).

Рс = 12295,65 · 20 / 100= 2459,13 руб.

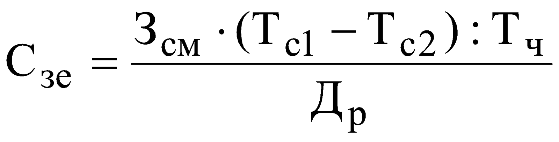
**2.** Оценка экономической эффективности применения программного средства у пользователя

**2.1.** Расчет экономии основных видов ресурсов в связи   
с использованием нового программного средства

Исходные данные для расчета экономии ресурсов в связи с применением нового программного средства

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование показателей | Обозна-  чения | Единицы измерения | Значение показателя | | Наименование источника информации |
| в базовом варианте | в новом варианте |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1. Капитальные вло-  жения, включая зат-  раты пользователя на приобретение | Kпp | руб. |  | 18443,47 | Договор заказчика с разработ-  чиком |
| 2. Затраты на сопро-  вождение ПО | Кс | руб. |  | 2459,13 | Договор заказчика с разработ-  чиком |
| 3. Время простоя сервиса, обуслов-  ленное ПО,  в день | П1,П2 | мин | 50 | 10 | Расчетные данные поль-  зователя и паспорт ПО |
| 4. Стоимость одного часа простоя | Сп | руб. | 30,1 | 30,1 | Расчетные данные поль-  зователя и паспорт ПО |
| 5. Среднемесячная ЗП одного програм-  миста | Зсм | руб. | 853,0 | 853,0 | Расчетные  данные поль-  зователя |
| 6. Коэффициент на-  числений на зарплату | Kнз |  | 1,5 | 1.5 | Рассчитывает-  ся по данным пользователя |
| 7. Среднемесячное количество рабочих дней | Д р | день |  | 21.5 | Принято для расчета |
| 8. Количество типо-  вых задач, решаемых за год | Зт1 ,Зт2 | задача | 6000 | 6000 | План пользо-  вателя |
| 9. Объем выполня-  емых работ | А1, А2 | задача | 6000 | 6000 | План пользо-  вателя |
| 10. Средняя трудоем-  кость работ на задачу | Tc1,  Тс2 | Человеко-часов | 0,5 | 0,2 | Рассчитывает-  ся по данным пользователя |
| 11. Количество часов работы ты в день | Tч | ч | 8 | 8 | Принято для расчета |
| 12. Ставка налога на прибыль | Hп | % |  | 18 |  |

Экономия затрат на заработную платув расчете на 1 задачу (Cзe):

,

где Зcм− среднемесячная заработная плата одного программиста (руб.);

Tc1, Tc2− снижение трудоемкости работ в расчете на 1 задачу (человеко-часов);

Tч− количество часов работы в день (ч);

Дp− среднемесячное количество рабочих дней.

Сзе = 853\*(0.5-0.2)/(8\*21.5) =  1,49 руб

Экономия заработной платы при использовании нового ПО (руб.):

Сз=СзеА2,

где Сз− экономия заработной платы;

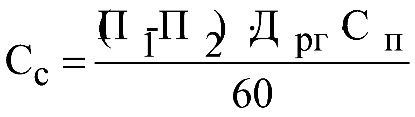
А2− количество типовых задач, решаемых за год (задач).

Сз=1,49\*6000=8940 руб.

Экономия с учетом начисления на зарплату (Сн):

*Cн= Cз****ּ*** 1,5 *=* 8940 \*1,5 *=* 13410 руб.

Экономия за счет сокращения простоев сервиса (Сс) рассчитывается по  формуле:

,

где Дрг – плановый фонд работы сервиса (дней).

Сс = (180-120)\* 21.5\*30,1 / 60 = 647,15 руб.

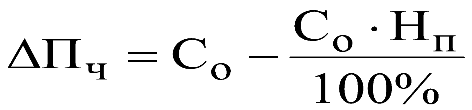
Общая готовая экономия текущих затрат, связанных с использованием  
нового ПО (Со), рассчитывается по формуле:

Co= Cн + Cc

Co= 13410 + 647,15 = 14057,15 тыс руб.

Расчет экономического эффекта.

Внедрение нового ПО позволит пользователю сэкономить на текущих затратах, т.е. практически получить на эту сумму дополнительную прибыль. Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль − дополнительная прибыль, остающаяся в его распоряжении (ΔПЧ), которая определяется по формуле:

,

где Hп− ставка налога на прибыль (%).

ΔПЧ = 14057,15 – (14057,15 \* 0.18) = 11562,86 руб.

В процессе использования нового ПО чистая прибыль в конечном итоге возмещает капитальные затраты. Однако полученные при этом суммы результатов (прибыли) и затрат (капитальных вложений) по годам приводят к единому времени − расчетному году (за расчетный год принят 2017-й год) путем умножения результатов и затрат за каждый год на коэффициент дисконтирования α. В данном примере используются коэффициенты: 2017 г. – 1, 2018-й – 0,8696, 2019-й – 0,7561, 2020 г. – 0,6575. Все рассчитанные данные экономического эффекта сводятся в таблицу 2.1.

Таблица 2.1. Расчет экономического эффекта от использования нового программного средства

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатели | Единицы  измерения | Годы | | | |
| 2017 | 2018 | 2019 | 2020 |
| *Результаты* |  |  |  |  |  |
| Прирост прибыли за счет экономии затрат (Пч) | руб. |  | 11562,86 | 11562,86 | 11562,86 |
| То же с учетом фактора времени | руб. |  | 10023.76 | 8715.46 | 7578.91 |
| *Затраты* |  |  |  |  |  |
| Приобретение ПО (Кпр) | руб. | 18443,47 |  |  |  |
| Сопровождение (Кс) | руб. | 2459,13 |  |  |  |
| Всего затрат | руб. | 20921,04 |  |  |  |
| То же с учетом фактора времени | руб. | 20921,04 |  |  |  |
| *Экономический эффект* |  |  |  |  |  |
| Превышение результата над затратами | руб. | -20921,04 | 10023.76 | 8715.46 | 7578.91 |
| То же с нарастающим итогом | руб. | -20921,04 | -10897,28 | -2181,82 | 5397,09 |
| Коэффициент приведения | единицы | 1 | 0,8696 | 0,7561 | 0,6575 |

Реализация проекта ПО позволит снизить затраты на процесс отслеживания прогресса выполнения курсовых и дипломных работ, процесс распределения заданий проектов за счет наличия единой системы коммуникации и отслеживания статусов и задач курсовых и дипломных проектов. Расчет производился для среднего количества курсовых и дипломных проектов (с учетом плановых консультаций и промежуточных оценок) для кафедры ПОИТ. Увеличение числа кафедр, использующих данное ПС для организации работы с курсовыми и дипломными проектами приведет к увеличению числа плановых задач за год, что, соответственно, приведет к ускорению срока окупаемости данного ПС.

При учете использования ПС одной кафедрой университета, все затраты окупятся на 3 год эксплуатации.

Проект представляется эффективным и полезным для использования.

Заключение

В ходе работы над дипломным проектом проанализированы системы автоматизации рабочего места администратора, исследованы различные направления и подходы к решению задач, связанных с разработкой программного средства для данных систем.

В процессе разработки проведен анализ предметной области, исследованы существующие аналоги. Результатом этого анализа явилось обобщение достоинств и недостатков существующих решений, которые учтены при разработке функциональных требований к разработанному программному средству. Наиболее часто встречающимся недостатком у имеющихся решений оказалось отсутствие комплексного решения всех поставленных требований. Из-за узкой специфики направления использования данного программного средства на отечественном рынке на сегодняшний день нет подобных средств, позволяющих коллективно обрабатывать и работать с данными сервиса совместной реализации бизнес-проектов.

На основе функциональных требований было произведено проектирование программного средства. В нем представлены разработка архитектуры ПС, разработка модели базы данных, разработка алгоритма программного средства и алгоритмов отдельных модулей. В разделе разработки архитектуры ПС детально рассмотрены схема алгоритма входа в систему, схема алгоритма работы с программным средством, схема работы с проектами. В разделе представлена также диаграмма развертывания приложения.

Согласно требованиям были сформированы тестовые наборы, которые успешно пройдены в ходе тестовых испытаний программного средства. Успешность прохождения тестов показывает корректность работы программы с реальными данными, соответствие функциональным требованиям.

На завершающем этапе подробно описана методика использования программного средства, которая позволяет за достаточно короткие сроки освоить работу с программой.

Также в ходе работы над дипломным проектом рассмотрена экономическая сторона проектирования и разработки программного средства, рассчитан экономический эффект от внедрения программного средства. В результате расчётов подтвердилась целесообразность разработки. Рентабельность разработки составила почти 50%, а инвестиции, вложенные в разработку, окупаются за три года.

Главной целью при разработке программного средства было поставлено устранение основных недостатков существующих аналогов, а также разработка и реализация всего основного функционала. В ходе работы над дипломным проектом эта цель была успешно достигнута: разработан минималистический, удобный, интуитивно понятный пользовательский интерфейс, предоставлена возможность коллективной обработки данных платформы, реализованы учет пользователей и их электронных сделок и систему уведомлений об изменениях в проектах.

Благодаря тому, что программа разработана на Java, обеспечена переносимость данного средства на различные операционные системы.

Список использованных источников

[1] LinkedIn – Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/ LinkedIn.

[2] JIRA – Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/ JIRA.

[3] Trello – Википедия [Электронный ресурс]. – Электронные дан-ные. – Режим доступа: https://ru.wikipedia.org/wiki/ Trello .

https://ru.wikipedia.org/wiki/PhoneGap

[4] React – Википедия [Электронный ресурс]. – Электронные дан-ные. – Режим доступа: https://ru.wikipedia.org/wiki/ React.

[5] Webpack– Википедия [Электронный ресурс]. – Электронные дан-ные. – Режим доступа: https://ru.wikipedia.org/wiki/Webpack.

[6] Babel – Википедия [Электронный ресурс]. – Электронные дан-ные. – Режим доступа: https://ru.wikipedia.org/wiki/ Babel.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

Текст программного модуля клиента

package com.lwo.groovy.wikipc.project

import com.lwo.groovyx.javafx.beans.FXBindable

import java.time.LocalDate

import java.time.LocalDateTime

import javax.xml.bind.annotation.\*

import javax.xml.bind.annotation.adapters.\*

import com.lwo.groovy.wikipc.dimension.\*

import com.lwo.utils.\*

@XmlRootElement(name = "Project")

@XmlAccessorType(XmlAccessType.NONE)

class Project {

@XmlElement(name = "code") String code

@XmlElement(name = "message") String message

@XmlElement(name = "ProjectID") String projectId

@XmlElement @FXBindable Long id

@XmlElement @FXBindable String name

@XmlElement @FXBindable Dimension category

@XmlElement @FXBindable Dimension status

@XmlJavaTypeAdapter(LocalDateAdapter.class)

@XmlElement @FXBindable LocalDate dtActivation

@XmlElement @FXBindable String videoLink

@XmlElement @FXBindable String previewLink

@XmlElement @FXBindable String descriptionShort

@XmlElement @FXBindable String descriptionFull

@XmlElement @FXBindable Dimension location

@XmlElement @FXBindable BigDecimal neededSum

@XmlElement @FXBindable BigDecimal minSum

@XmlElement @FXBindable Integer daysCount

@XmlJavaTypeAdapter(LocalDateAdapter.class)

@XmlElement @FXBindable LocalDate dtStartFee

@XmlJavaTypeAdapter(LocalDateAdapter.class)

@XmlElement @FXBindable LocalDate dtEndFee

@XmlJavaTypeAdapter(LocalDateAdapter.class)

@XmlElement @FXBindable LocalDate dtEnd

@XmlElement @FXBindable Integer idForpost

@XmlElement @FXBindable String latitude

@XmlElement @FXBindable String longitude

@XmlElement @FXBindable Integer videoImageId

@XmlElement @FXBindable Long imageId

@XmlElement @FXBindable Boolean favouriteFlag

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.BaseDialogController

import com.lwo.groovy.wikipc.bonus.\*

import com.lwo.groovy.wikipc.dimension.\*

import java.time.\*

import javafx.beans.value.ChangeListener

import javafx.beans.value.ObservableValue

import javafx.fxml.FXML

import javafx.scene.control.\*

import javafx.scene.control.DatePicker

import com.lwo.client.proxy.ServerScript

import javafx.collections.\*

import javafx.event.ActionEvent

import javafx.util.converter.\*

/\*\*

\*

\* @author sadovskaya\_a

\*/

class AddBonusController extends BaseDialogController{

@FXML private TextField sum

@FXML private TextArea descriptionShort

@FXML private DatePicker termOfUse

@FXML private TextField quantity

@FXML private ComboBox dtIssueType

@FXML private DatePicker dtIssue

@FXML private ComboBox inheritance

@FXML private ComboBox salePeriod

@FXML private DatePicker saleFrom

@FXML private DatePicker saleTo

@FXML private TextField depositDays

@FXML private TabPane tabPane

List dtIssueTypeList

List inheritanceList

List salePeriodList

Bonus bonus

public void initialize(){

sum.lengthProperty().addListener(new ChangeListener<Number>(){

@Override

public void changed(ObservableValue<? extends Number> observable, Number oldValue, Number newValue) {

if (sum.getText().equals('0')){

quantity.setText('1')

quantity.setDisable(true)

}else{

quantity.setDisable(false)

}

if(newValue.intValue() > oldValue.intValue()){

char ch = sum.getText().charAt(oldValue.intValue());

println("Length:"+ oldValue+" "+ newValue +" "+ch);

//Check if the new character is the number or other's

if(!(ch >= '0' && ch <= '9' )){

//if it's not number then just setText to previous one

sum.setText(sum.getText().substring(0,sum.getText().length()-1));

}

}

}

});

depositDays.lengthProperty().addListener(new ChangeListener<Number>(){

@Override

public void changed(ObservableValue<? extends Number> observable, Number oldValue, Number newValue) {

if(newValue.intValue() > oldValue.intValue()){

char ch = depositDays.getText().charAt(oldValue.intValue());

println("Length:"+ oldValue+" "+ newValue +" "+ch);

//Check if the new character is the number or other's

if(!(ch >= '0' && ch <= '9' )){

//if it's not number then just setText to previous one

depositDays.setText(depositDays.getText().substring(0,depositDays.getText().length()-1));

}

}

}

});

quantity.lengthProperty().addListener(new ChangeListener<Number>(){

@Override

public void changed(ObservableValue<? extends Number> observable, Number oldValue, Number newValue) {

if(newValue.intValue() > oldValue.intValue()){

char ch = quantity.getText().charAt(oldValue.intValue());

println("Length:"+ oldValue+" "+ newValue +" "+ch);

//Check if the new character is the number or other's

if(!(ch >= '0' && ch <= '9' )){

//if it's not number then just setText to previous one

quantity.setText(quantity.getText().substring(0,quantity.getText().length()-1));

}

}

}

});

}

public void bindObject(def object){

sum.textProperty().bindBidirectional(object.sumProperty(), new BigDecimalStringConverter())

descriptionShort.textProperty().bindBidirectional(object.descriptionShortProperty())

termOfUse.valueProperty().bindBidirectional(object.termOfUseProperty())

quantity.textProperty().bindBidirectional(object.quantityProperty(), new IntegerStringConverter())

dtIssue.valueProperty().bindBidirectional(object.dtIssueProperty())

saleFrom.valueProperty().bindBidirectional(object.dtSaleFromProperty())

saleTo.valueProperty().bindBidirectional(object.dtSaleToProperty())

depositDays.textProperty().bindBidirectional(object.depositDaysProperty(), new IntegerStringConverter())

dtIssueType.valueProperty().bindBidirectional(object.issueTypeProperty())

inheritance.valueProperty().bindBidirectional(object.inheritanceProperty())

salePeriod.valueProperty().bindBidirectional(object.salePeriodProperty())

if (object.issueType != null){

if (object.issueType.name.equals('Дата')){

dtIssue.setDisable(false)

}

}

if (object.salePeriod != null){

if (object.salePeriod.name.equals('Период')){

saleFrom.setDisable(false)

saleTo.setDisable(false)

}

}

this.bonus = object

}

public void unbindObject(def object){

sum.textProperty().unbindBidirectional(object.sumProperty())

descriptionShort.textProperty().unbindBidirectional(object.descriptionShortProperty())

termOfUse.valueProperty().unbindBidirectional(object.termOfUseProperty())

quantity.textProperty().unbindBidirectional(object.quantityProperty())

dtIssue.valueProperty().unbindBidirectional(object.dtIssueProperty())

saleFrom.valueProperty().unbindBidirectional(object.dtSaleFromProperty())

saleTo.valueProperty().unbindBidirectional(object.dtSaleToProperty())

depositDays.textProperty().unbindBidirectional(object.depositDaysProperty())

dtIssueType.valueProperty().unbindBidirectional(object.issueTypeProperty())

inheritance.valueProperty().unbindBidirectional(object.inheritanceProperty())

salePeriod.valueProperty().unbindBidirectional(object.salePeriodProperty())

this.bonus = object

}

public void loadComboboxLists() {

def doc = ServerScript.run("com.lwo.groovy.wd.gui.getDimEntries", ['DT\_ISSUE\_TYPE'], null)

//printXml(doc)

def wrapper = xmlToObject(doc, DimensionListWrapper.class)

dtIssueTypeList = wrapper.dims

dtIssueType.setItems(FXCollections.observableArrayList(dtIssueTypeList))

doc = ServerScript.run("com.lwo.groovy.wd.gui.getDimEntries", ['INHERITANCE'], null)

//printXml(doc)

wrapper = xmlToObject(doc, DimensionListWrapper.class)

inheritanceList = wrapper.dims

inheritance.setItems(FXCollections.observableArrayList(inheritanceList))

doc = ServerScript.run("com.lwo.groovy.wd.gui.getDimEntries", ['SALE\_PERIOD'], null)

//printXml(doc)

wrapper = xmlToObject(doc, DimensionListWrapper.class)

salePeriodList = wrapper.dims

salePeriod.setItems(FXCollections.observableArrayList(salePeriodList))

}

void initGUI() {

tabPane.getSelectionModel().select(0)

}

@FXML

public void onDtIssueTypeAction(ActionEvent action){

def type = dtIssueType.getSelectionModel().getSelectedItem()

if (type.name.equals('Дата')){

dtIssue.setDisable(false)

}else{

dtIssue.setValue(null)

dtIssue.setDisable(true)

}

}

@FXML

public void onSalePeriodAction(ActionEvent action){

def type = salePeriod.getSelectionModel().getSelectedItem()

if (type.name.equals('Период')){

saleFrom.setDisable(false)

saleTo.setDisable(false)

}else{

saleFrom.setValue(null)

saleTo.setValue(null)

saleFrom.setDisable(true)

saleTo.setDisable(true)

}

}

public void getObject(def object){

object.setId(0)

object.setDescriptionShort(descriptionShort.getText() == null ? null : descriptionShort.getText())

object.setTermOfUse(termOfUse.getValue() == null ? null : termOfUse.getValue())

object.setQuantity(quantity.getText().isInteger() ? quantity.getText().toInteger() : null)

object.setIssueType(dtIssueType.getValue() == null ? new Dimension() : dtIssueType.getValue())

object.setDtIssue(dtIssue.getValue() == null ? null : dtIssue.getValue())

object.setInheritance(inheritance.getValue() == null ? new Dimension() : inheritance.getValue())

object.setSalePeriod(salePeriod.getValue() == null ? new Dimension() : salePeriod.getValue())

object.setDtSaleFrom(saleFrom.getValue() == null ? null : saleFrom.getValue())

object.setDtSaleTo(saleTo.getValue() == null ? null : saleTo.getValue())

object.setDepositDays(depositDays.getText().isInteger() ? depositDays.getText().toInteger() : null)

object.setSum(sum.getText().isBigDecimal() ? sum.getText().toBigDecimal() : null)

}

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.BaseDialogController

import com.lwo.groovy.wikipc.model.News

import java.util.function.UnaryOperator

import javafx.fxml.FXML

import javafx.scene.control.\*

import javafx.util.converter.LongStringConverter

/\*\*

\*

\* @author sadovskaya\_a

\*/

class AddNewsController extends BaseDialogController{

@FXML private TextField header;

@FXML private TextArea descriptionShort;

@FXML private javafx.scene.web.HTMLEditor descriptionFull;

@FXML private TabPane tabPane

@FXML private Tab ttt

News news

com.lwo.client.gui.PageHtmlEditor htmlEditor

public void initialize(){

int limit = 200

UnaryOperator<TextFormatter.Change> filter = new UnaryOperator<TextFormatter.Change>() {

@Override

public TextFormatter.Change apply(TextFormatter.Change c) {

if ((c.isAdded() || c.isReplaced()) && c.getControlNewText().length()>limit) {

String ct = c.getText();

ct = ct.substring(0,ct.length()-(c.getControlNewText().length()-limit));

c.setText(ct);

}

return c;

}

};

descriptionShort.setTextFormatter(new TextFormatter(filter));

}

void initGUI() {

tabPane.getSelectionModel().select(0)

htmlEditor = new com.lwo.client.gui.PageHtmlEditor( ttt );

htmlEditor.setContext("")

}

public void getObject(News object){

object.setHeader(header.getText())

object.setDescriptionShort(descriptionShort.getText())

def text = htmlEditor.getText()

def encoded = text?.bytes?.encodeBase64()?.toString()

object.setDescriptionFull(encoded)

/\*

String html = descriptionFull.getHtmlText()

int p1 = html.indexOf('<body')

int p2 = html.indexOf('>', p1)

String body = html.substring(p2+1, html.indexOf("</body"));

def encoded = body.bytes.encodeBase64().toString()

object.setDescriptionFull(encoded)

\*/

}

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.BaseDialogController

import com.lwo.groovy.wikipc.model.\*

import javafx.fxml.FXML

import javafx.scene.control.\*

import javafx.util.converter.LongStringConverter

import com.lwo.client.proxy.ServerScript

import javafx.collections.\*;

/\*\*

\*

\* @author sadovskaya\_a

\*/

class AddPageController extends BaseDialogController{

@FXML private TextField id;

@FXML private TextField title;

@FXML private javafx.scene.web.HTMLEditor text;

@FXML private ComboBox parentPage

@FXML private TextField order

@FXML private TextField alias

@FXML private TabPane tabPane

@FXML private Tab ttt

TextPage page

List pageList

com.lwo.client.gui.PageHtmlEditor htmlEditor

public void initialize(){

}

public void loadComboboxLists() {

def doc = ServerScript.run("com.lwo.groovy.wd.gui.page.getPageDimList")

//printXml(doc)

def wrapper = xmlToObject(doc, DimensionListWrapper.class)

pageList = wrapper.dims

if (pageList != null) {

parentPage.setItems(FXCollections.observableArrayList(pageList))

}

}

void initGUI() {

tabPane.getSelectionModel().select(0)

htmlEditor = new com.lwo.client.gui.PageHtmlEditor( ttt );

htmlEditor.setContext("")

}

public void getObject(def object){

object.setTitle(title.getText())

object.setAlias(alias.getText())

object.setOrderPage(order.getText())

object.setParentId(parentPage.getValue())

def text = htmlEditor.getText()

def encoded = text?.bytes?.encodeBase64()?.toString()

object.setText(encoded)

/\*

String html = text.getHtmlText()

int p1 = html.indexOf('<body')

int p2 = html.indexOf('>', p1)

String body = html.substring(p2+1, html.indexOf("</body"));

def encoded = body.bytes.encodeBase64().toString()

object.setText(encoded)

\*/

}

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.BaseDialogController

import com.lwo.javafx.SelectKeyComboBoxListener

import java.net.URL;

import java.util.ResourceBundle;

import java.util.function.UnaryOperator

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.\*;

import javafx.scene.input.KeyEvent

import javafx.util.converter.\*

import javafx.collections.\*

import javafx.event.EventHandler

import com.lwo.groovy.wikipc.project.\*

import javafx.scene.control.cell.PropertyValueFactory

import javafx.scene.image.Image

import javafx.scene.image.ImageView

import javafx.beans.value.ObservableValue

import javafx.beans.property.ReadOnlyObjectWrapper

import javafx.beans.value.ChangeListener

import javafx.scene.text.Text

import javafx.stage.FileChooser

import javafx.util.Callback

public class AddProjectController extends BaseDialogController {

@FXML private TextField name

@FXML private ComboBox category

@FXML private ComboBox status

@FXML private ComboBox projectLocation

@FXML private TextField image

@FXML private TextField videoLink

@FXML private TextArea descriptionShort

@FXML private javafx.scene.web.HTMLEditor descriptionFull

@FXML private TextField neededSum

@FXML private TextField latitude

@FXML private TextField longitude

@FXML private CheckBox favouriteFlag

@FXML private TabPane tabPane

@FXML private ImageView imageView

private String fname

private String extension

Project project

public void initialize() {

neededSum.lengthProperty().addListener(new ChangeListener<Number>(){

@Override

public void changed(ObservableValue<? extends Number> observable, Number oldValue, Number newValue) {

if(newValue.intValue() > oldValue.intValue()){

char ch = neededSum.getText().charAt(oldValue.intValue())

if(!(ch >= '0' && ch <= '9' )){

neededSum.setText(neededSum.getText().substring(0,neededSum.getText().length()-1))

}

}

}

})

int limit = 200

UnaryOperator<TextFormatter.Change> filter = new UnaryOperator<TextFormatter.Change>() {

@Override

public TextFormatter.Change apply(TextFormatter.Change c) {

if ((c.isAdded() || c.isReplaced()) && c.getControlNewText().length()>limit) {

String ct = c.getText();

ct = ct.substring(0,ct.length()-(c.getControlNewText().length()-limit));

c.setText(ct);

}

return c;

}

};

descriptionShort.setTextFormatter(new TextFormatter(filter));

}

void initGUI() {

tabPane.getSelectionModel().select(0)

}

public String getFname(){

return fname

}

public String getExtension(){

return extension

}

void setCategoryItems(def list) {

if (list!=null) {

Iterator<List> itr = list.iterator()

while(itr.hasNext()){

def parent = itr.next()

if (parent.name.equals('Все')){

itr.remove()

}

}

category.setItems(FXCollections.observableArrayList(list))

}

}

void setStatusItems(def list) {

if (list!=null) {

Iterator<List> itr = list.iterator()

while(itr.hasNext()){

def parent = itr.next()

if (parent.name.equals('Все')){

itr.remove()

}

}

status.setItems(FXCollections.observableArrayList(list))

}

}

void setLocationItems(def list) {

if (list!=null) {

projectLocation.setItems(FXCollections.observableArrayList(list))

}

}

public String getImage(){

return image.getText()

}

public void getObject(def object) {

object.setId(0)

object.setName(name.getText() == null? null : name.getText())

object.setCategory(category.getValue() == null ? null : category.getValue())

object.setStatus(status.getValue() == null ? null : status.getValue())

object.setLocation(projectLocation.getValue() == null ? null : projectLocation.getValue())

object.setVideoLink(videoLink.getText() == null ? null : videoLink.getText())

object.setDescriptionShort(descriptionShort.getText() == null ? null : descriptionShort.getText())

object.setNeededSum(neededSum.getText().isBigDecimal()? neededSum.getText().toBigDecimal() : null)

object.setLatitude(latitude.getText() == null ? null : latitude.getText())

object.setLongitude(longitude.getText() == null ? null : longitude.getText())

object.setFavouriteFlag(favouriteFlag.isSelected())

String html = descriptionFull.getHtmlText()

int p1 = html.indexOf('<body')

int p2 = html.indexOf('>', p1)

String body = html.substring(p2+1, html.indexOf("</body"));

def encoded = body.bytes.encodeBase64().toString()

object.setDescriptionFull(encoded)

}

@FXML

private void onLoadImageAction(){

// try{

fname = null

extension = null

image.setText(null)

FileChooser fileChooser = new FileChooser()

fileChooser.setTitle("Выбрать картинку")

//fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("png", "\*.png"))

File file = fileChooser.showOpenDialog(stage)

String filename = file.toString()

if (file != null){

def path = file.getAbsolutePath()

def i = path.lastIndexOf('.')

def j = file.name.lastIndexOf('.')

if (i > 0) {

extension = path.substring(i+1)

}

if (j>0){

fname = file.name.substring(0, j)

}

FileInputStream fis = new FileInputStream(file)

def imageData = new byte[(int)file.length()]

fis.read(imageData)

String imageDataString = imageData.encodeBase64().toString()

image.setText(imageDataString)

Image image = new Image(file.toURI().toString())

imageView.setImage(image)

// Alert alert = new Alert(AlertType.INFORMATION);

// alert.setTitle("Сообщение");

// alert.setHeaderText("Экспорт прошел успешно");

// alert.initOwner(stage);

// alert.initModality(Modality.WINDOW\_MODAL);

// alert.showAndWait();

}

// }catch(Exception ex){

// Alert alert = new Alert(AlertType.INFORMATION);

// alert.setTitle("Сообщение");

// alert.setHeaderText("Экпорт был прерван");

// alert.initOwner(stage);

// alert.initModality(Modality.WINDOW\_MODAL);

// alert.showAndWait();

// }

}

}

import com.lwo.client.gui.BaseDialogController

import com.lwo.groovy.wikipc.model.\*

import com.lwo.groovy.wikipc.dimension.\*

import com.lwo.groovy.wikipc.author.\*

import java.awt.image.BufferedImage

import javafx.fxml.FXML

import javafx.scene.control.\*

import javafx.scene.image.Image

import javafx.scene.image.ImageView

import javafx.scene.layout.VBox

import javafx.stage.FileChooser

import javafx.util.converter.LongStringConverter

import java.io.ByteArrayInputStream

import javax.imageio.ImageIO

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

import javafx.embed.swing.SwingFXUtils

import javafx.event.ActionEvent

import com.lwo.client.proxy.ServerScript

import javafx.collections.\*

/\*\*

\*

\* @author sadovskaya\_a

\*/

class AuthorEditorController extends BaseDialogController{

@FXML private TextField surname

@FXML private TextField name

@FXML private TextField lastname

@FXML private TextField email

@FXML private TextField phone

@FXML private TextField address

@FXML private TextField unp

@FXML private TextField account

@FXML private TextField website

@FXML private ComboBox project

@FXML private ImageView imageView

@FXML private VBox imgLoader

Author author

def imgName, imgType, imgBase64

public void initialize(){

imgName = null

imgType = null

imgBase64 = null

}

void initGUI() {

imageView.setFitHeight( 200.0 )

imageView.setFitWidth( 300.0 )

imageView.setImage( null )

imageView.setVisible( false )

imgLoader.setVisible( true )

Thread.start{

try {

def image = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.image.getImage", createFilter([ imageId : author.imageid ]) ), com.lwo.groovy.wikipc.model.Image.class )

if ( null != image?.base ){

byte[] decoded = image.base.decodeBase64()

imageView.setImage( SwingFXUtils.toFXImage( ImageIO.read( new ByteArrayInputStream( decoded ) ) , null) )

} else {

imageView.setImage( new Image(getClass().getResourceAsStream("/db/icons/core/no\_image.png")) )

}

} catch ( Exception e ){

imageView.setImage( new Image(getClass().getResourceAsStream("/db/icons/core/Administrator.jpg")) )

} finally {

imgLoader.setVisible( false )

imageView.setVisible( true )

}

}

}

public void loadComboboxLists() {

def wrapper = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.project.getDimsProjects"), DimensionListWrapper.class)

if ( wrapper?.dims != null ) project.setItems(FXCollections.observableArrayList( wrapper.dims ) )

}

public void bindObject(def object){

surname.textProperty().bindBidirectional(object.surnameProperty())

name.textProperty().bindBidirectional(object.nameProperty())

lastname.textProperty().bindBidirectional(object.lastnameProperty())

email.textProperty().bindBidirectional(object.emailProperty())

phone.textProperty().bindBidirectional(object.phoneProperty())

address.textProperty().bindBidirectional(object.addressProperty())

account.textProperty().bindBidirectional(object.accountProperty())

unp.textProperty().bindBidirectional(object.unpProperty())

website.textProperty().bindBidirectional(object.websiteProperty())

project.valueProperty().bindBidirectional(object.projectProperty())

this.author = object

}

public void unbindObject(def object){

surname.textProperty().unbindBidirectional(object.surnameProperty())

name.textProperty().unbindBidirectional(object.nameProperty())

lastname.textProperty().unbindBidirectional(object.lastnameProperty())

email.textProperty().unbindBidirectional(object.emailProperty())

phone.textProperty().unbindBidirectional(object.phoneProperty())

address.textProperty().unbindBidirectional(object.addressProperty())

account.textProperty().unbindBidirectional(object.accountProperty())

unp.textProperty().unbindBidirectional(object.unpProperty())

website.textProperty().unbindBidirectional(object.websiteProperty())

project.valueProperty().unbindBidirectional(object.projectProperty())

this.author = object

}

public void getObject(Author object){

object.setSurname(surname.getText())

object.setName(name.getText())

object.setLastname(lastname.getText())

object.setEmail(email.getText())

object.setPhone(phone.getText())

object.setProject(project.getValue())

object.setAddress(address.getText())

object.setUnp(unp.getText())

object.setAccount(account.getText())

object.setWebsite(website.getText())

}

def getImageData(){

return [

name : imgName,

type : imgType,

base : imgBase64

]

}

@FXML

private void onLoadImageAction(){

try {

FileChooser fileChooser = new FileChooser()

fileChooser.getExtensionFilters().addAll(

new FileChooser.ExtensionFilter("Все форматы", "\*.jpg", "\*.gif","\*.bmp","\*.png")

)

File selectedFile = fileChooser.showOpenDialog(null);

if ( null != selectedFile ){

imgName = selectedFile.getName();

imgType = selectedFile.getName().substring(selectedFile.getName().lastIndexOf(".") + 1);

imgBase64 = toBase( selectedFile.getAbsolutePath() );

imageView.setImage( new Image( selectedFile.toURI().toString() ))

imageView.setFitHeight( 200 )

imageView.setFitWidth( 300 )

}

} catch ( Exception e ) {

showAlertWithException( e, "Ошибка при добавлении картинки" )

}

}

def toBase( String path ){

try{

File file = new File( path );

FileInputStream fileInputStreamReader = new FileInputStream(file);

byte[] b = new byte[(int)file.length()];

fileInputStreamReader.read(b);

return b.encodeBase64().toString();

} catch ( Exception e ){

e.printStackTrace()

}

}

def fillFilter ( def filter ){

println filter

}

}

package com.lwo.groovy.wikipc

import com.lwo.groovyx.javafx.beans.FXBindable

import com.lwo.client.gui.\*

import com.lwo.groovy.wikipc.author.\*

import com.lwo.groovy.wikipc.model.\*

import com.lwo.groovy.wikipc.dimension.\*

import groovy.xml.\*

import javafx.fxml.FXML

import javafx.scene.Parent

import javafx.scene.control.\*

import javafx.scene.control.cell.PropertyValueFactory

import javafx.scene.layout.AnchorPane

import javafx.scene.layout.Pane

import javafx.scene.input.MouseEvent;

import javafx.event.EventHandler;

import com.lwo.client.proxy.ServerScript

import javafx.beans.value.\*

import javafx.collections.\*

import javafx.event.ActionEvent

import javafx.scene.control.cell.CheckBoxTableCell

import javafx.stage.Modality;

import java.util.Optional;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ButtonType;

import javax.xml.bind.annotation.\*

import javax.xml.bind.annotation.adapters.\*

import java.time.\*

import java.time.format.\*

import javax.xml.parsers.DocumentBuilder

import org.xml.sax.InputSource

import org.w3c.dom.\*

import javax.xml.parsers.\*

import javafx.application.Platform

import javafx.scene.Cursor

/\*\*

\*

\* @author sadovskaya\_a

\*/

class AuthorViewController extends BaseController{

private BaseController mainController = null

@FXML private TableView authorsTable

@FXML private TableColumn idCol

@FXML private TableColumn surnameCol

@FXML private TableColumn nameCol

@FXML private TableColumn lastnameCol

@FXML private TableColumn phoneCol

@FXML private TableColumn emailCol

@FXML private TableColumn projectCol

@FXML private TableColumn addressCol

@FXML private TableColumn unpCol

@FXML private TableColumn accountCol

@FXML private TableColumn websiteCol

@FXML private TextField filterEmail

@FXML private TextField filterSurname

@FXML private ComboBox filterProject

@FXML private Label totalLabel

List projectList

def dlg, addDlg, isInitialize

public void initialize(){

idCol.setCellValueFactory(new PropertyValueFactory('id'))

surnameCol.setCellValueFactory(new PropertyValueFactory('surname'))

nameCol.setCellValueFactory(new PropertyValueFactory('name'))

lastnameCol.setCellValueFactory(new PropertyValueFactory('lastname'))

phoneCol.setCellValueFactory(new PropertyValueFactory('phone'))

emailCol.setCellValueFactory(new PropertyValueFactory('email'))

projectCol.setCellValueFactory(new PropertyValueFactory('project'))

addressCol.setCellValueFactory(new PropertyValueFactory('address'))

unpCol.setCellValueFactory(new PropertyValueFactory('unp'))

accountCol.setCellValueFactory(new PropertyValueFactory('account'))

websiteCol.setCellValueFactory(new PropertyValueFactory('website'))

//loadComboboxLists()

authorsTable.setOnMousePressed(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.isPrimaryButtonDown() && event.getClickCount() == 2) {

def author = authorsTable.getSelectionModel().getSelectedItem()

if ( null != author ) editAuthor( createFilter([id: author.id]) )

}

}

});

}

void afterRender(){

if ( isInitialize != true ) {

loadComboboxLists()

isInitialize = true

}

}

public void loadComboboxLists() {

Thread.start(){

def wrapper = xmlToObject(ServerScript.run("com.lwo.groovy.wd.gui.project.getDimsProjects"), DimensionListWrapper.class)

wrapper.dims?.add(0, new Dimension(0, 'ALL', 'Все', 'PROJECT'))

projectList = wrapper.dims

if ( projectList != null ) filterProject.setItems(FXCollections.observableArrayList(projectList))

}

}

void setMainController(BaseController controller) {

this.mainController = controller

}

@FXML

public onFilterAction(ActionEvent event) {

def filter = [

projectid : filterProject.getValue()?.id == 0 ? null : filterProject.getValue()?.id,

surname : filterSurname.getText(),

email : filterEmail.getText()

]

showAuthors(filter)

}

public void fillFilter( def filter ) {

println "Author fill filter " + filter

onClearFilterAction()

if (filter['projectId']) {

Long id = filter['projectId']

for (def item:filterProject.getItems()) {

if (item.id == id) {

filterProject.setValue(item)

break

}

}

}

}

@FXML

private void onClearFilterAction() {

filterSurname.setText(null)

filterProject.setValue(null)

filterEmail.setText(null)

}

public void showAuthors( def filter) {

println "showAuthors filter " + filter

Platform.runLater{ run ->

authorsTable.getParent().setDisable( true )

authorsTable.getScene().setCursor(Cursor.WAIT)

Thread.start{

try {

def wrapper = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.author.getAuthorList", createFilter( filter ) ), AuthorListWrapper.class)

authorsTable.getItems().clear()

if ( wrapper.code == "000" ) authorsTable.setItems(FXCollections.observableArrayList(wrapper.list))

if ( wrapper.code == "999" ) showAlert( "Ошибка" , wrapper.message , AlertType.ERROR )

} catch ( Exception e ){

Platform.runLater{ runs ->

showAlertWithException( e, null )

}

} finally {

Platform.runLater{ runs ->

authorsTable.getParent().setDisable( false )

authorsTable.getScene().setCursor(Cursor.DEFAULT)

totalLabel.setText( 'Количество: ' + authorsTable.getItems().size())

}

}

}

}

}

@FXML

private editAuthorAction(ActionEvent event) {

def filter = createFilter([

id : authorsTable.getSelectionModel().getSelectedItems()[0]?.id

])

editAuthor( filter )

}

def editAuthor( def filter) {

def author

if ( filter == null ) {

author = new Author()

} else {

def authors = xmlToObject(ServerScript.run("com.lwo.groovy.wd.gui.author.getAuthorList", filter ), AuthorListWrapper.class)

if ( authors.code != "000" ){

showAlert( "Ошибка" , "Ошибка при получении записи автора\n" + authors.message , AlertType.ERROR )

return

}

author = authors.list[0]

}

if ( dlg == null ) dlg = newDialog('Редактирование автора', 'wd.WDAuthorEditor')

dlg.controller.loadComboboxLists()

dlg.controller.bindObject(author)

dlg.controller.initGUI()

dlg.showAndWait()

dlg.controller.unbindObject( author )

if (dlg.getAction() == DialogAction.OK) {

def savedAuthor = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.author.saveAuthor", objectToXml(author) ) , Author.class )

if ( savedAuthor.code == "000" ){

def img\_data = dlg.controller.getImageData()

if ( img\_data.name != null ){

def writer = new StringWriter()

def xml = new MarkupBuilder(writer)

xml.image(){

id author.id

name img\_data.name

type img\_data.type

base img\_data.base

}

def savedImage = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.image.saveAuthorImage", stringToXml(writer.toString())), Image.class)

if ( savedImage.code != "000" ){

showAlert( "Ошибка" , "Ошибка при сохранении картинки автора\n" + savedImage.message , AlertType.ERROR )

}

}

showAuthors([id:null])

}

dlg = null

}

}

@FXML

private addAuthorAction(ActionEvent event){

editAuthor( null )

}

@FXML

private deleteAuthorAction(ActionEvent event){

Alert confirm = new Alert(AlertType.CONFIRMATION);

confirm.setTitle("Подтверждение");

confirm.setHeaderText("Данный автор будет удален!");

confirm.setContentText("Продолжить?");

confirm.initOwner(stage);

confirm.initModality(Modality.WINDOW\_MODAL);

def answer = confirm.showAndWait();

if (answer.isPresent() && answer.get() == ButtonType.OK) {

try {

def filter = createFilter([

id : authorsTable.getSelectionModel().getSelectedItems()[0]?.id

])

def result = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.author.deleteAuthor", filter ), Author.class)

if ( result.code != "000"){

showAlert( "Ошибка" , "Ошибка при удалении автора\n" + result.message , AlertType.ERROR )

} else {

showAlert( "Уведомление" , "Автор удален" , AlertType.INFORMATION )

showAuthors([id:null])

}

} catch (Exception ex) {

showAlertWithException( e, "Возникла ошибка при удалении автора" )

}

}

}

@FXML

public void goToProjectAction(ActionEvent event) {

mainController.goToProjects([

projectId : authorsTable.getSelectionModel().getSelectedItems()[0]?.project?.id

])

}

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.BaseDialogController

import com.lwo.groovy.wikipc.depositBonus.\*

import com.lwo.groovy.wikipc.model.\*

import java.time.\*

import javafx.beans.value.ChangeListener

import javafx.beans.value.ObservableValue

import javafx.fxml.FXML

import javafx.scene.control.\*

import javafx.scene.control.DatePicker

import javafx.scene.control.cell.PropertyValueFactory

import com.lwo.client.proxy.ServerScript

import javafx.collections.\*

import javafx.event.ActionEvent

import javafx.util.converter.\*

import com.lwo.groovy.core.utils.CellFactoryUtils

/\*\*

\*

\* @author sadovskaya\_a

\*/

class BonusForDepositController extends BaseDialogController{

@FXML private TableView bonusTable

@FXML private TableColumn idCol

@FXML private TableColumn clientCol

@FXML private TableColumn numDaysInDepositCol

@FXML private TableColumn sumCol

@FXML private TableColumn descriptionShortCol

@FXML private TableColumn termOfUseCol

@FXML private TableColumn quantityCol

@FXML private TabPane tabPane

public void initialize(){

idCol.setCellValueFactory(new PropertyValueFactory('id'))

clientCol.setCellValueFactory(new PropertyValueFactory('clientFullName'))

numDaysInDepositCol.setCellValueFactory(new PropertyValueFactory('numDaysInDeposit'))

sumCol.setCellValueFactory(new PropertyValueFactory('sum'))

descriptionShortCol.setCellValueFactory(new PropertyValueFactory('desc'))

termOfUseCol.setCellValueFactory(new PropertyValueFactory('termOfUse'))

quantityCol.setCellValueFactory(new PropertyValueFactory('count'))

}

void initGUI() {

tabPane.getSelectionModel().select(0)

}

void setBonusList(def wrapper){

if (wrapper.bonusList != null){

bonusTable.setItems(FXCollections.observableArrayList(wrapper.bonusList))

}else{

bonusTable.getItems().clear()

}

}

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.BaseDialogController

import java.net.URL;

import java.time.LocalDate

import java.time.format.DateTimeFormatter

import java.util.ResourceBundle;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.ComboBox

import javafx.scene.control.DatePicker

import javafx.scene.control.TextField;

import javafx.scene.input.KeyEvent

import javafx.util.StringConverter

import javafx.util.converter.NumberStringConverter

import javafx.collections.\*

import javafx.event.EventHandler

/\*\*

\* FXML Controller class

\*

\* @author lena

\*/

public class ClientEditorController extends BaseDialogController {

@FXML private TextField nick;

@FXML private TextField email;

@FXML private ComboBox clientStatus;

@FXML private ComboBox state;

@FXML private TextField lastName;

@FXML private TextField firstName;

@FXML private TextField patronymic;

@FXML private ComboBox sex;

@FXML private DatePicker dtBirth;

@FXML private ComboBox typeDoc;

@FXML private TextField personalNum;

@FXML private TextField docSeries;

@FXML private TextField docNum;

@FXML private TextField docWhoIssued;

@FXML private DatePicker docDt;

@FXML private TextField registration;

@FXML private TextField residency;

@FXML private TextField information;

@FXML private TextField vkClient;

@FXML private TextField okClient;

@FXML private TextField fbClient;

@FXML private TextField ggClient;

@FXML private TextField i24Client;

@FXML private TextField confirmationFlag;

@FXML private TextField phone;

// private StringConverter<LocalDate> converter = null

public void initialize() {

println 'editClient'

//

// dtBirth.setConverter(converter)

// docDt.setConverter(converter)

}

void setClientStatusItems(List list){

if (list!=null) {

clientStatus.setItems(FXCollections.observableArrayList(list))

}

}

void setClientStateItems(List list){

if (list!=null) {

state.setItems(FXCollections.observableArrayList(list))

}

}

void setSexItems(List list){

if (list!=null) {

sex.setItems(FXCollections.observableArrayList(list))

}

}

void setDocTypeItems(List list){

if (list!=null) {

typeDoc.setItems(FXCollections.observableArrayList(list))

}

}

public void bindObject(def object) {

nick.textProperty().bindBidirectional(object.nickProperty())

email.textProperty().bindBidirectional(object.clientEmailProperty())

clientStatus.valueProperty().bindBidirectional(object.clientStatusProperty())

state.valueProperty().bindBidirectional(object.stateProperty())

lastName.textProperty().bindBidirectional(object.lastNameProperty())

firstName.textProperty().bindBidirectional(object.firstNameProperty())

patronymic.textProperty().bindBidirectional(object.patronymicProperty())

sex.valueProperty().bindBidirectional(object.sexProperty())

dtBirth.valueProperty().bindBidirectional(object.dtBirthProperty())

typeDoc.valueProperty().bindBidirectional(object.typeDocProperty())

personalNum.textProperty().bindBidirectional(object.personalNumProperty())

docSeries.textProperty().bindBidirectional(object.docSeriesProperty())

docNum.textProperty().bindBidirectional(object.docNumberProperty())

docWhoIssued.textProperty().bindBidirectional(object.docWhoIssuedProperty())

docDt.valueProperty().bindBidirectional(object.docDtProperty())

registration.textProperty().bindBidirectional(object.registrationProperty())

residency.textProperty().bindBidirectional(object.residencyProperty())

information.textProperty().bindBidirectional(object.informationProperty())

vkClient.textProperty().bindBidirectional(object.vkClientProperty())

okClient.textProperty().bindBidirectional(object.odClientProperty())

fbClient.textProperty().bindBidirectional(object.fbClientProperty());

ggClient.textProperty().bindBidirectional(object.ggClientProperty())

i24Client.textProperty().bindBidirectional(object.i24ClientProperty())

confirmationFlag.textProperty().bindBidirectional(object.confirmationFlagProperty())

phone.textProperty().bindBidirectional(object.phoneProperty())

}

public void unbindObject(def object) {

nick.textProperty().unbindBidirectional(object.nickProperty())

email.textProperty().unbindBidirectional(object.clientEmailProperty())

clientStatus.valueProperty().unbindBidirectional(object.clientStatusProperty())

state.valueProperty().unbindBidirectional(object.stateProperty())

lastName.textProperty().unbindBidirectional(object.lastNameProperty())

firstName.textProperty().unbindBidirectional(object.firstNameProperty())

patronymic.textProperty().unbindBidirectional(object.patronymicProperty())

sex.valueProperty().unbindBidirectional(object.sexProperty())

dtBirth.valueProperty().unbindBidirectional(object.dtBirthProperty())

typeDoc.valueProperty().unbindBidirectional(object.typeDocProperty())

personalNum.textProperty().unbindBidirectional(object.personalNumProperty())

docSeries.textProperty().unbindBidirectional(object.docSeriesProperty())

docNum.textProperty().unbindBidirectional(object.docNumberProperty())

docWhoIssued.textProperty().unbindBidirectional(object.docWhoIssuedProperty())

docDt.valueProperty().unbindBidirectional(object.docDtProperty())

registration.textProperty().unbindBidirectional(object.registrationProperty())

residency.textProperty().unbindBidirectional(object.residencyProperty())

information.textProperty().unbindBidirectional(object.informationProperty())

vkClient.textProperty().unbindBidirectional(object.vkClientProperty())

okClient.textProperty().unbindBidirectional(object.odClientProperty())

fbClient.textProperty().unbindBidirectional(object.fbClientProperty());

ggClient.textProperty().unbindBidirectional(object.ggClientProperty())

i24Client.textProperty().unbindBidirectional(object.i24ClientProperty())

confirmationFlag.textProperty().unbindBidirectional(object.confirmationFlagProperty())

phone.textProperty().unbindBidirectional(object.phoneProperty())

}

}

package com.lwo.groovy.wikipc

import javafx.event.ActionEvent

import javafx.fxml.FXML

import javafx.scene.control.ComboBox

import javafx.scene.control.ContextMenu

import javafx.scene.control.MenuItem

import javafx.scene.control.TableColumn

import javafx.scene.control.TableView

import javafx.scene.control.TextField

import javafx.scene.control.cell.PropertyValueFactory

import javafx.scene.layout.AnchorPane

import com.lwo.client.proxy.ServerScript

import com.sun.javafx.scene.control.skin.TextInputControlSkin.ContextMenuItem

import javafx.collections.\*

import com.lwo.client.gui.\*

import java.util.List

import javafx.scene.control.cell.\*

import javafx.scene.control.\*

import javafx.event.\*

import javafx.scene.control.TableColumn.CellEditEvent

import com.lwo.groovy.core.utils.CellFactoryUtils

import com.lwo.groovy.wikipc.client.Client

import com.lwo.groovy.wikipc.client.ClientListWrapper

import com.lwo.groovy.wikipc.dimension.Dimension

import com.lwo.groovy.wikipc.dimension.DimensionListWrapper

import javafx.scene.control.Alert

import javafx.scene.control.Alert.AlertType

import javafx.stage.Modality

import javafx.stage.FileChooser

import java.time.LocalDate

import java.time.format.DateTimeFormatter

import javafx.stage.FileChooser

import javafx.application.Platform

import javafx.scene.Cursor

/\*\*

\*

\* @author lena

\*/

class ClientsViewController extends BaseController {

private BaseController mainController = null

@FXML private TableView clientsTable

@FXML private TableColumn idColumn

@FXML private TableColumn nickColumn

@FXML private TableColumn secretWordColumn

@FXML private TableColumn signingColumn

@FXML private TableColumn statusColumn

@FXML private TableColumn custCodeColumn

@FXML private TableColumn fullnameColumn

@FXML private TableColumn cntrColumn

@FXML private TableColumn indexColumn

@FXML private TableColumn regionColumn

@FXML private TableColumn subregionColumn

@FXML private TableColumn townTypeColumn

@FXML private TableColumn townColumn

@FXML private TableColumn streetTypeColumn

@FXML private TableColumn streetColumn

@FXML private TableColumn houseColumn

@FXML private TableColumn buildingColumn

@FXML private TableColumn roomColumn

@FXML private TableColumn birthColumn

@FXML private TableColumn emailColumn

@FXML private TableColumn phoneColumn

@FXML private TableColumn sexColumn

@FXML private TableColumn docCntrColumn

@FXML private TableColumn docTypeColumn

@FXML private TableColumn docNrColumn

@FXML private TableColumn docPlaceColumn

@FXML private TableColumn docDtColumn

@FXML private TableColumn docExpiredDtColumn

@FXML private Label totalLabel

//поля фильтра

@FXML private TextField filterId

@FXML private TextField filterFio

@FXML private TextField filterEmail

FxmlDialog dlg

public void initialize() {

idColumn.setCellValueFactory(new PropertyValueFactory('id'))

nickColumn.setCellValueFactory(new PropertyValueFactory('nickname'))

secretWordColumn.setCellValueFactory(new PropertyValueFactory('secretWord'))

signingColumn.setCellValueFactory(new PropertyValueFactory('signingFlag'))

statusColumn.setCellValueFactory(new PropertyValueFactory('status'))

custCodeColumn.setCellValueFactory(new PropertyValueFactory('bankId'))

fullnameColumn.setCellValueFactory(new PropertyValueFactory('fullname'))

cntrColumn.setCellValueFactory(new PropertyValueFactory('cntr'))

indexColumn.setCellValueFactory(new PropertyValueFactory('index'))

regionColumn.setCellValueFactory(new PropertyValueFactory('region'))

subregionColumn.setCellValueFactory(new PropertyValueFactory('subregion'))

townTypeColumn.setCellValueFactory(new PropertyValueFactory('townType'))

townColumn.setCellValueFactory(new PropertyValueFactory('town'))

streetTypeColumn.setCellValueFactory(new PropertyValueFactory('streetType'))

streetColumn.setCellValueFactory(new PropertyValueFactory('street'))

houseColumn.setCellValueFactory(new PropertyValueFactory('house'))

buildingColumn.setCellValueFactory(new PropertyValueFactory('building'))

roomColumn.setCellValueFactory(new PropertyValueFactory('room'))

birthColumn.setCellValueFactory(new PropertyValueFactory('birth'))

emailColumn.setCellValueFactory(new PropertyValueFactory('clientEmail'))

phoneColumn.setCellValueFactory(new PropertyValueFactory('phone'))

sexColumn.setCellValueFactory(new PropertyValueFactory('sex'))

docCntrColumn.setCellValueFactory(new PropertyValueFactory('docCntr'))

docTypeColumn.setCellValueFactory(new PropertyValueFactory('docType'))

docNrColumn.setCellValueFactory(new PropertyValueFactory('docNr'))

docPlaceColumn.setCellValueFactory(new PropertyValueFactory('docPlace'))

docDtColumn.setCellValueFactory(new PropertyValueFactory('docDt'))

docExpiredDtColumn.setCellValueFactory(new PropertyValueFactory('docExpiredDt'))

birthColumn.setCellFactory(CellFactoryUtils.getDateCellFactory())

docDtColumn.setCellFactory(CellFactoryUtils.getDateCellFactory())

// emailColumn.setCellFactory(TextFieldTableCell.forTableColumn())

}

public setMainController(BaseController controller){

mainController = controller;

}

public void fillFilter( def filter) {

onClearFilterAction()

if (filter['clientId']) filterId.setText(filter['clientId']==null? '': filter['clientId'].toString())

}

@FXML

private onFilterAction(ActionEvent event) {

showClients([

id : filterId.getText(),

fio : filterFio.getText(),

email : filterEmail.getText()

])

}

@FXML

private void onClearFilterAction() {

filterId.setText(null)

filterFio.setText(null)

filterEmail.setText(null)

}

@FXML

private void onExportAction(){

try{

FileChooser fileChooser = new FileChooser()

fileChooser.setTitle("Сохранить документ")

fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("CSV", "\*.csv"))

File file = fileChooser.showSaveDialog(stage)

String filename = file.toString();

if (!file.toString().toLowerCase().endsWith(".csv")) filename = file.toString() + ".csv"

if (file != null){

FileWriter writer = new FileWriter(filename);

writer.append("Ид.;Ид. в FORPOST;ФИО;Эл. почта;Телефон;Никнейм;Секретное слово;Признак подписания договора ДБО;Статус;Страна резидентства;Индекс;Область;Район;Тип населенного пункта;Населенный пункт;Тип улицы;Название улицы;Номер дома;Номер корпуса;Номер квартиры;Дата рождения;Пол;Страна документа;Тип документа;Номер документа;Кем выдан;Дата окончания документа")

writer.append("\r\n")

def separator = ";"

clientsTable.getItems().each{

def result = new StringBuilder()

result << "${it.id}" << separator

result << "${it.bankId}" << separator

result << "${it.fullname}" << separator

result << "${it.clientEmail}" << separator

result << "${it.phone}" << separator

result << "${it.nickname}" << separator

result << "${it.secretWord}" << separator

result << "${it.signingFlag}" << separator

result << "${it.status}" << separator

result << "${it.cntr}" << separator

result << "${it.index}" << separator

result << "${it.region}" << separator

result << "${it.subregion}" << separator

result << "${it.townType}" << separator

result << "${it.town}" << separator

result << "${it.streetType}" << separator

result << "${it.street}" << separator

result << "${it.building}" << separator

result << "${it.room}" << separator

result << "${it.birth}" << separator

result << "${it.sex}" << separator

result << "${it.docCntr}" << separator

result << "${it.docType}" << separator

result << "${it.docNr}" << separator

result << "${it.docPlace}" << separator

result << "${it.docDt}" << separator

result << "${it.docExpiredDt}" << separator

result << "\r\n"

writer.append( result )

}

writer.close()

showAlert( "Сообщение" , "Экспорт прошел успешно", AlertType.INFORMATION )

}

}catch(Exception ex){

showAlertWithException( ex, "Экпорт был прерван " )

}

}

public void showClients( def filter) {

Platform.runLater{ run ->

clientsTable.getParent().setDisable( true )

clientsTable.getScene().setCursor(Cursor.WAIT)

Thread.start{

def wrapper = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.client.getClientData", createFilter(filter)), ClientListWrapper.class)

clientsTable.getItems().clear()

if ( wrapper.clients != null ) clientsTable.setItems(FXCollections.observableArrayList(wrapper.clients))

Platform.runLater{ runs ->

clientsTable.getParent().setDisable( false )

clientsTable.getScene().setCursor(Cursor.DEFAULT)

totalLabel.setText( 'Количество: ' + clientsTable.getItems().size())

}

}

}

}

@FXML

public void goToPaymentsAction(ActionEvent event) {

mainController.goToPayments([

clientId : clientsTable.getSelectionModel().getSelectedItems()[0]?.id

])

}

}

package com.lwo.groovy.wikipc

import com.lwo.client.gui.\*

import com.lwo.client.proxy.ServerScript

import com.lwo.groovy.core.dim.Dimension

import com.lwo.groovy.core.dim.DimensionListWrapper

import com.lwo.groovy.core.utils.CellFactoryUtils

import com.lwo.groovy.wikipc.deposit.\*

import com.lwo.groovy.wikipc.depositBonus.\*

import java.util.concurrent.Executors

import java.util.concurrent.ExecutorService

import javafx.event.ActionEvent

import javafx.fxml.FXML

import javafx.scene.control.ComboBox

import javafx.scene.control.ContextMenu

import javafx.scene.control.MenuItem

import javafx.scene.control.TableColumn

import javafx.scene.control.TableView

import javafx.scene.control.TextField

import javafx.scene.control.cell.PropertyValueFactory

import javafx.scene.layout.AnchorPane

import javafx.scene.control.Alert

import javafx.scene.control.Alert.AlertType

import javafx.stage.Modality

import javafx.stage.FileChooser

import javafx.collections.\*

import javafx.scene.control.cell.\*

import javafx.scene.control.\*

import javafx.event.\*

import javafx.scene.control.TableColumn.CellEditEvent

import javafx.util.Callback;

import com.sun.javafx.scene.control.skin.TextInputControlSkin.ContextMenuItem

import java.util.List

import java.time.LocalDate

import java.time.format.DateTimeFormatter

import javafx.application.Platform

import javafx.scene.Cursor

/\*\*

\*

\* @author lena

\*/

class DepositViewController extends BaseController {

private BaseController mainController = null

@FXML private TableView depositTable

@FXML private TableColumn idCol

@FXML private TableColumn dtCreatedCol

@FXML private TableColumn paymentAmountCol

@FXML private TableColumn dtPaymentCol

@FXML private TableColumn clientCol

@FXML private TableColumn projectCol

@FXML private TableColumn contractNumCol

@FXML private TableColumn idForpostVariantCol

@FXML private TableColumn statusCol

@FXML private TableColumn currencyCol

@FXML private TableColumn plannedDtOpenCol

@FXML private TableColumn plannedDtCloseCol

@FXML private TableColumn intSchemeCol

@FXML private TableColumn intRateTypeCol

@FXML private TableColumn dtSaleStartCol

@FXML private TableColumn dtSaleEndCol

@FXML private TableColumn parentIdCol

@FXML private TableColumn paymentIdCol

@FXML private TableColumn minSumCol

@FXML private TableColumn repaymentDtCol

@FXML private TableColumn tillDtCol

@FXML private TableColumn interestStCol

@FXML private TableColumn principalModeCol

@FXML private TableColumn intCpTypeCol

@FXML private TableColumn intPenaltyCol

@FXML private TableColumn crtCodeCol

@FXML private Label totalLabel

//поля фильтра

@FXML private ComboBox filterProject

@FXML private ComboBox filterStatus

@FXML private DatePicker filterDtFrom

@FXML private DatePicker filterDtTo

List projectList

List statusList

def paymentId

def dlg

def isInitialize = false

public void initialize() {

idCol.setCellValueFactory(new PropertyValueFactory('id'))

paymentAmountCol.setCellValueFactory(new PropertyValueFactory('paymentAmount'))

dtCreatedCol.setCellValueFactory(new PropertyValueFactory('dtCreated'))

dtPaymentCol.setCellValueFactory(new PropertyValueFactory('dtPayment'))

clientCol.setCellValueFactory(new PropertyValueFactory('client'))

projectCol.setCellValueFactory(new PropertyValueFactory('name'))

contractNumCol.setCellValueFactory(new PropertyValueFactory('contractNumber'))

idForpostVariantCol.setCellValueFactory(new PropertyValueFactory('forpostId'))

statusCol.setCellValueFactory(new PropertyValueFactory('status'))

currencyCol.setCellValueFactory(new PropertyValueFactory('currency'))

plannedDtOpenCol.setCellValueFactory(new PropertyValueFactory('plannedDtOpen'))

plannedDtCloseCol.setCellValueFactory(new PropertyValueFactory('plannedDtClose'))

intSchemeCol.setCellValueFactory(new PropertyValueFactory('interestScheme'))

intRateTypeCol.setCellValueFactory(new PropertyValueFactory('interestRateType'))

dtSaleStartCol.setCellValueFactory(new PropertyValueFactory('dtSaleStart'))

dtSaleEndCol.setCellValueFactory(new PropertyValueFactory('dtSaleEnd'))

parentIdCol.setCellValueFactory(new PropertyValueFactory('parentId'))

paymentIdCol.setCellValueFactory(new PropertyValueFactory('paymentId'))

minSumCol.setCellValueFactory(new PropertyValueFactory('minSum'))

repaymentDtCol.setCellValueFactory(new PropertyValueFactory('repaymentDt'))

tillDtCol.setCellValueFactory(new PropertyValueFactory('tillDt'))

interestStCol.setCellValueFactory(new PropertyValueFactory('interestSt'))

principalModeCol.setCellValueFactory(new PropertyValueFactory('principalMode'))

intCpTypeCol.setCellValueFactory(new PropertyValueFactory('interestCpType'))

intPenaltyCol.setCellValueFactory(new PropertyValueFactory('interestPenalty'))

crtCodeCol.setCellValueFactory(new PropertyValueFactory('crtCode'))

dtPaymentCol.setCellFactory(CellFactoryUtils.dateTimeCellFactory)

plannedDtOpenCol.setCellFactory(CellFactoryUtils.getDateCellFactory())

plannedDtCloseCol.setCellFactory(CellFactoryUtils.getDateCellFactory())

dtSaleStartCol.setCellFactory(CellFactoryUtils.getDateCellFactory())

dtSaleEndCol.setCellFactory(CellFactoryUtils.getDateCellFactory())

repaymentDtCol.setCellFactory(CellFactoryUtils.getDateCellFactory())

tillDtCol.setCellFactory(CellFactoryUtils.getDateCellFactory())

}

void afterRender(){

if ( isInitialize != true ) {

loadComboboxLists()

isInitialize = true

}

}

private void loadComboboxLists() {

def getDimsProjects = {

def wrapper = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.project.getDimsProjects") , DimensionListWrapper.class)

wrapper.dims?.add(0, new Dimension(0, 'ALL', 'Все', 'PROJECT'))

projectList = wrapper.dims

filterProject.setItems( FXCollections.observableArrayList(projectList) )

} as java.lang.Runnable

def getDimsStatus = {

def wrapper = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.getDimEntries", ['DEPOSIT\_STATUS'], null), DimensionListWrapper.class)

wrapper.dims?.add(0, new Dimension(0, 'ALL', 'Все', 'DEPOSIT\_STATUS'))

statusList = wrapper.dims

if ( statusList != null ) filterStatus.setItems(FXCollections.observableArrayList(statusList))

} as java.lang.Runnable

ExecutorService executor = java.util.concurrent.Executors.newFixedThreadPool( 2 )

executor.execute( getDimsProjects )

executor.execute( getDimsStatus )

executor.shutdown()

}

public setMainController(BaseController controller){

mainController = controller;

}

public void fillFilter( def filter) {

onClearFilterAction()

if ( filter.paymentId ) paymentId = filter?.paymentId

}

@FXML

private onFilterAction(ActionEvent event) {

Platform.runLater{ run ->

depositTable.getParent().setDisable( true )

depositTable.getScene().setCursor(Cursor.WAIT)

Thread.start{

try {

def filter = createFilter([

projectId : filterProject.getValue()?.id,

statusId : filterStatus.getValue()?.id,

paymentId : paymentId,

dtFrom : filterDtFrom.getValue() ? filterDtFrom.getValue().toString() : null ,

dtTo : filterDtTo.getValue() ? filterDtTo.getValue().toString() : null

])

def wrapper = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.deposit.getDeposit", filter ), DepositListWrapper.class )

if ( wrapper.code != "000" ){

showAlert( "Ошибка" , wrapper.message, AlertType.INFORMATION )

} else {

wrapper?.deposits ? depositTable.setItems( FXCollections.observableArrayList( wrapper.deposits ) ) : depositTable.getItems().clear()

}

} catch ( Exception e ){

Platform.runLater{ runs ->

showAlertWithException( e, "Ошибка получения данных" )

}

} finally {

Platform.runLater{ runs ->

depositTable.getParent().setDisable( false )

depositTable.getScene().setCursor(Cursor.DEFAULT)

totalLabel.setText( 'Количество: ' + depositTable.getItems().size() )

}

}

}

}

}

@FXML

private void onClearFilterAction() {

filterProject.setValue(null)

filterStatus.setValue(null)

filterDtFrom.setValue(null)

filterDtTo.setValue(null)

paymentId = null

}

@FXML

public showBonusListAction(ActionEvent event) {

def status = depositTable.getSelectionModel().getSelectedItems()[0]?.status

try{

if ( status.name != 'Активный') throw new IllegalArgumentException( "Просмотр бонусов у депозита со статусом '" + status + "' невозможен." )

def filter = createFilter([

id : depositTable.getSelectionModel().getSelectedItems()[0]?.id

])

def bonusList = xmlToObject( ServerScript.run("com.lwo.groovy.wd.gui.deposit.getBonusByDeposit", filter ), DepositBonusListWrapper.class)

if ( bonusList.code == "000" ){

if ( dlg == null ) {

dlg = newDialog('Просмотреть бонусы', 'wikipc.BonusListView')

dlg.controller.setBonusList( bonusList )

}

dlg.controller.initGUI()

dlg.showAndWait()

} else if ( bonusList.code == "100" ){

throw new IllegalArgumentException( bonusList.message )

} else {

throw new Exception( bonusList.message )

}

} catch ( IllegalArgumentException ie ){

showAlert( "Сообщение" , ie.getMessage() , AlertType.INFORMATION )

} catch ( Exception e ){

showAlert( "Ошибка" , e.getMessage() , AlertType.ERROR )

}

dlg = null

}

@FXML

private void onExportAction(){

try{

FileChooser fileChooser = new FileChooser()

fileChooser.setTitle("Сохранить документ")

fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("CSV", "\*.csv"))

File file = fileChooser.showSaveDialog(stage)

String filename = null;

if (!file.toString().toLowerCase().endsWith(".csv")){

filename = file.toString() + ".csv"

}else {

filename = file.toString()

}

if (file != null){

FileWriter writer = new FileWriter(filename);

writer.append("Ид.;Клиент;Проект;Номер договора;Ид. вклада в Forpost;Статус;Валюта;Планируемая дата открытия;Планируемая дата закрытия;Процентная схема;Вид процентной ставки;Дата начала продажи;Дата окончания продажи;Ид. вклада предка;Ид. платежа;Первоначальная сумма;Дата получения выплат;Дата окончания выплат;Процентная ставка;Principal mode;Тип выплат;Штраф;crtCode")

writer.append("\r\n")

def separator = ";"

depositTable.getItems().each{

def result = new StringBuilder()

result << it.id << separator

result << it.client << separator

result << it.name << separator

result << it.contractNumber << separator

result << it.forpostId << separator

result << it.status << separator

result << it.currency << separator

result << it.plannedDtOpen << separator

result << it.plannedDtClose << separator

result << it.interestScheme << separator

result << it.interestRateType << separator

result << it.dtSaleStart << separator

result << it.dtSaleEnd << separator

result << it.parentId << separator

result << it.paymentId << separator

result << it.minSum << separator

result << it.repaymentDt << separator

result << it.tillDt << separator

result << it.interestSt << separator

result << it.principalMode << separator

result << it.interestCpType << separator

result << it.interestPenalty << separator

result << it.crtCode << separator

result << "\r\n"

writer.append( result )

}

writer.close()

showAlert( "Сообщение" , "Экспорт прошел успешно", AlertType.INFORMATION )

}

}catch(Exception ex){

showAlertWithException( e, "Экпорт был прерван " )

}

}

@FXML

public void goToPaymentsAction(ActionEvent event) {

mainController.goToPayments([

paymentId : depositTable.getSelectionModel().getSelectedItems()[0]?.paymentId

])

}

}

package com.lwo.groovy.wikipc

import java.util.concurrent.Executors

import java.util.concurrent.ExecutorService

import javafx.scene.Group;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import groovy.xml.\*

import com.lwo.client.gui.\*

import javafx.fxml.FXML

import javafx.scene.Parent

import javafx.scene.control.\*

import javafx.scene.layout.VBox;

import javafx.geometry.Pos;

import javafx.scene.\*

import javafx.scene.control.cell.PropertyValueFactory

import javafx.scene.layout.AnchorPane

import javafx.scene.layout.Pane

import javafx.scene.input.MouseEvent;

import javafx.event.EventHandler;

import javafx.util.Callback

import com.lwo.client.proxy.ServerScript

import javafx.beans.value.\*

import javafx.collections.\*;

import javafx.event.ActionEvent

import javafx.scene.control.cell.CheckBoxTableCell

import javafx.stage.Modality;

import java.util.Optional;

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.control.ButtonType;

import com.lwo.groovy.core.utils.CellFactoryUtils

import com.lwo.utils.MapWrapper

import com.lwo.groovy.wikipc.bonus.\*

import com.lwo.groovy.wikipc.dimension.\*

import com.lwo.groovy.wikipc.project.\*

import java.time.\*

import java.time.format.\*

import javafx.stage.FileChooser

import javax.xml.parsers.DocumentBuilder

import org.xml.sax.InputSource

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import javafx.application.Platform

import javafx.scene.Cursor

import com.lwo.groovy.we.project.ProjectTmplContentListWrapper

import javafx.application.Platform

/\*\*

\*

\* @author lena

\*/

class ProjectsViewController extends BaseController {

private BaseController mainController

@FXML private TableView projectsTable

@FXML private TableColumn idCol

@FXML private TableColumn nameCol

@FXML private TableColumn categoryCol

@FXML private TableColumn statusCol

@FXML private TableColumn videoLinkCol

@FXML private TableColumn locationCol

@FXML private TableColumn neededSumCol

@FXML private TableColumn latitudeCol

@FXML private TableColumn longitudeCol

@FXML private TableColumn favouriteCol

@FXML private TableColumn expertiseCol

@FXML private ComboBox filterCategory

@FXML private ComboBox filterStatus

@FXML private TextField filterKeywords

@FXML private TextField filterId

@FXML private Label totalLabel

@FXML private VBox loader

def dlg, addDlg, isInitialize

List categoryList

List statusList

List locationList

public void initialize() {

idCol.setCellValueFactory(new PropertyValueFactory('id'))

nameCol.setCellValueFactory(new PropertyValueFactory('name'))

categoryCol.setCellValueFactory(new PropertyValueFactory('category'))

statusCol.setCellValueFactory(new PropertyValueFactory('status'))

videoLinkCol.setCellValueFactory(new PropertyValueFactory('videoLink'))

locationCol.setCellValueFactory(new PropertyValueFactory('location'))

neededSumCol.setCellValueFactory(new PropertyValueFactory('neededSum'))

latitudeCol.setCellValueFactory(new PropertyValueFactory('latitude'))

longitudeCol.setCellValueFactory(new PropertyValueFactory('longitude'))

favouriteCol.setCellValueFactory(new PropertyValueFactory('favouriteFlag'))

favouriteCol.setCellFactory(CheckBoxTableCell.forTableColumn(favouriteCol))

expertiseCol.setCellValueFactory(new PropertyValueFactory('isProjectExpertise'))

expertiseCol.setCellFactory(CheckBoxTableCell.forTableColumn(expertiseCol))

neededSumCol.setCellFactory(CellFactoryUtils.getBigDecimalCellFactory())

projectsTable.setOnMousePressed(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

if (event.isPrimaryButtonDown() && event.getClickCount() == 2) {

def project = projectsTable.getSelectionModel().getSelectedItem()

if ( null != project ) editProject( createFilter([id: project.id]) )

}

}

});

isInitialize = false

}

void afterRender(){

if ( isInitialize != true ) {

Thread.start{

onFilterAction()

}

loadComboboxLists()

isInitialize = true

}

}

private void loadComboboxLists() {

def getDimsCategory = {

def doc = ServerScript.run("com.lwo.groovy.wd.gui.getDimEntries", ['PROJECT\_CATEGORY'], null)

def wrapper = xmlToObject(doc, DimensionListWrapper.class)

categoryList = wrapper.dims

wrapper.dims?.add(0, new Dimension(0, 'ALL', 'Все', 'TYPE\_DOC'))

if (wrapper.dims != null) filterCategory.setItems(FXCollections.observableArrayList(wrapper.dims))

} as java.lang.Runnable

def getDimsStatus = {

def doc = ServerScript.run("com.lwo.groovy.wd.gui.getDimEntries", ['PROJECT\_STATUS'], null)

def wrapper = xmlToObject(doc, DimensionListWrapper.class)

wrapper.dims?.add(0, new Dimension(0, 'ALL', 'Все', 'PROJECT\_STATUS'))

statusList = wrapper.dims

if ( statusList != null ) filterStatus.setItems(FXCollections.observableArrayList(statusList))

} as java.lang.Runnable

def getDimsLocation = {

def doc = ServerScript.run("com.lwo.groovy.wd.gui.getLocationList")

def wrapper = xmlToObject(doc, DimensionListWrapper.class)

locationList = wrapper.dims

loader.setVisible( false )

} as java.lang.Runnable

ExecutorService executor = java.util.concurrent.Executors.newFixedThreadPool( 3 )

executor.execute( getDimsCategory )

executor.execute( getDimsStatus )

executor.execute( getDimsLocation )

executor.shutdown()

}

void setMainController(BaseController controller) {

this.mainController = controller

}

}