Р Е Ф Е Р А Т

Пояснительная записка 104 с., 19 рис., 9 табл., 18 источников
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:

JavaScript, React, React Native, Android, RESTful, командная работа, управление проектами.

Цель работы – разработка программного средства, предназначенного для содействия студентам в процессе поиска управления задачами при выполнения курсовых, дипломных и других видов групповых работ.

Предлагаемое программное средство позволяет студентам объединяться в команды, разграничивать ответственность за отдельные участки проекта, получать опыт работы в команде и управления проектами. Мобильный клиент поспособствует более удобному доступу к данным о проектах и задачах, в условиях, когда полноценный компьютер не доступен.

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.

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 студентам объединяться в группы для совместного написания проектов и упростить работу с управлением проектами, а руководители проекта от университета получат возможность отслеживать прогресс подопечных по выполнению поставленных задач и удельный объем работы каждого участника в конечном результате.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

Аутентификация – проверка подлинности предъявленного пользователем идентификатора.

Задача – (англ. ticket, issue) единица описания заданий, поставленных разработчикам программного обеспечения в системах отслеживания ошибок.

Интроспекция – возможность в некоторых объектно-ориентированных языках определить тип и структуру объекта во время выполнения программы. Эта возможность имеется во всех языках, позволяющих манипулировать типами объектов как объектами первого класса.

Мутации – изменения, применяющиеся к данным.

Нативный компонент – компонент, созданный с использованием встроенных средств платформы.

Облако – сервер (или группа серверов), на котором пользователь, используя Интернет, хранит информацию (облачное хранилище данных) или производит вычисления (облачные вычисления).

Объекты первого класса – элементы, которые могут быть переданы как параметр, возвращены из функции, присвоены переменной.

Полифилл – это библиотека, которая добавляет в устаревшие среды исполнения поддержку возможностей, которые в современных средах являются встроенными.

Поток данных – представление работы приложения в виде схемы, изображающей перемещение данных между структурными элементами.

Рабочий процесс(вокфлоу) – представление потока задач в процессе и связанных с ним подпроцессов, включая специфические работы, информационные зависимости и последовательность решений и работ.

Система отслеживания ошибок – прикладная программа, разработанная с целью помочь разработчикам программного обеспечения учитывать и контролировать ошибки и неполадки, найденные в программах, пожелания пользователей, а также следить за процессом устранения этих ошибок и выполнения или невыполнения пожеланий.

Токен – строка, содержащая информацию по безопасности сеанса и идентифицирующая пользователя, группу пользователей и пользовательские привилегии

Транскомпиляция – компиляция исходного текста одного языка программирования в исходный текст на другом языке.

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

Freemium – бизнес-модель, которая заключается в предложении воспользоваться компьютерной игрой, программным продуктом, онлайн-сервисом или услугой бесплатно, в то время как расширенная (улучшенная, премиум) версия продукта, его дополнительная функциональность, или сервисы, другие продукты, связанные с основным, предлагаются за дополнительную

плату, на основе популярности основного бесплатного продукта.

API – Application Programming Interface – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

URI – унифицированный (единообразный) идентификатор ресурса.

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

ВВЕДЕНИЕ

В связи с усложнением технологий разработки программ и повышением конкурентоспособности 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 ОС Android 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 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

1.1 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).

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.

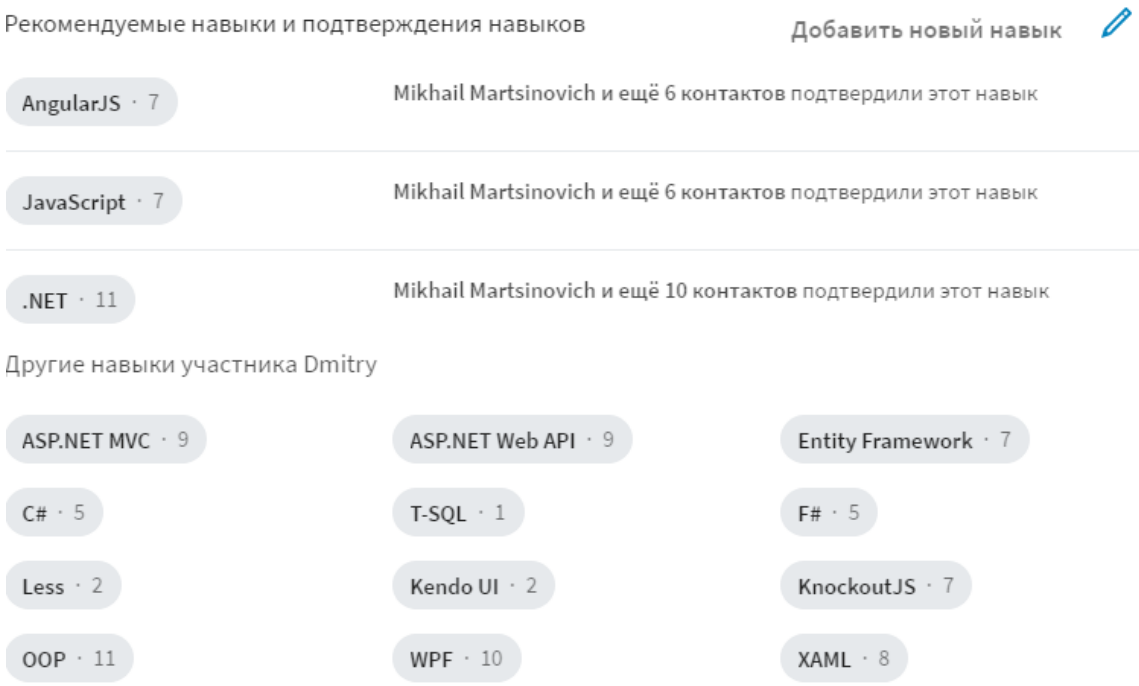

Рисунок 1.1 – Иллюстрация раздела навыков LinkedIn

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].

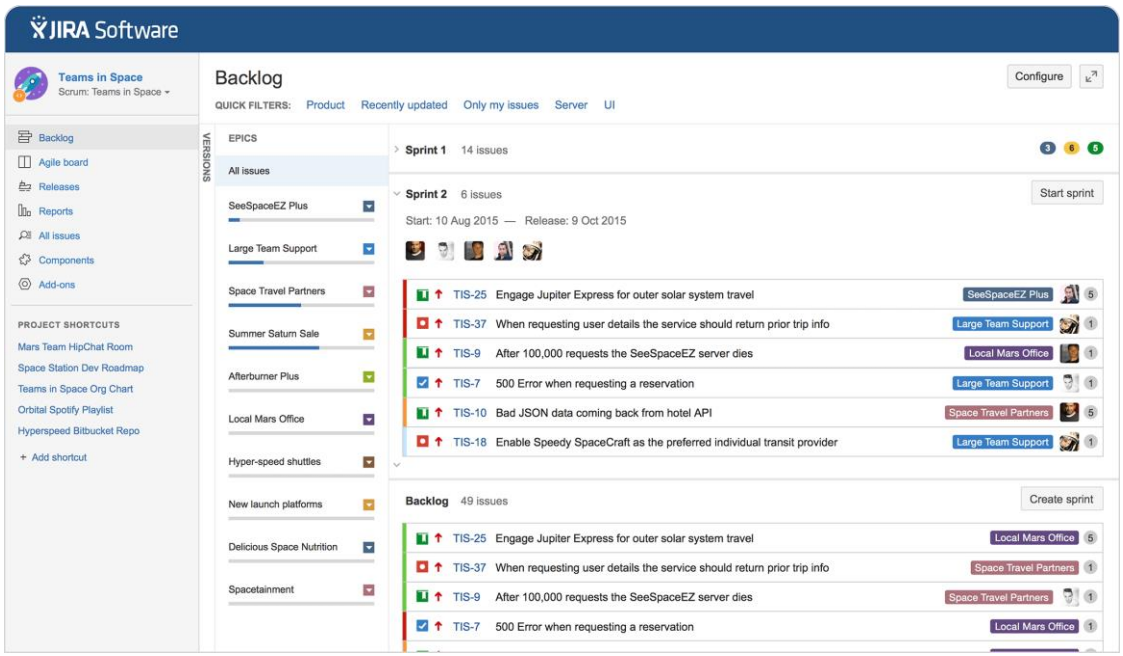

Рисунок 1.2 – Пример интерфейса пользователя JIRA

Задача (англ. ticket или issue) является основным элементом 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, расширяющих функциональные возможности Jira;

– дружественный интерфейс и функциональная система поиска и фильтрации задач 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).

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].

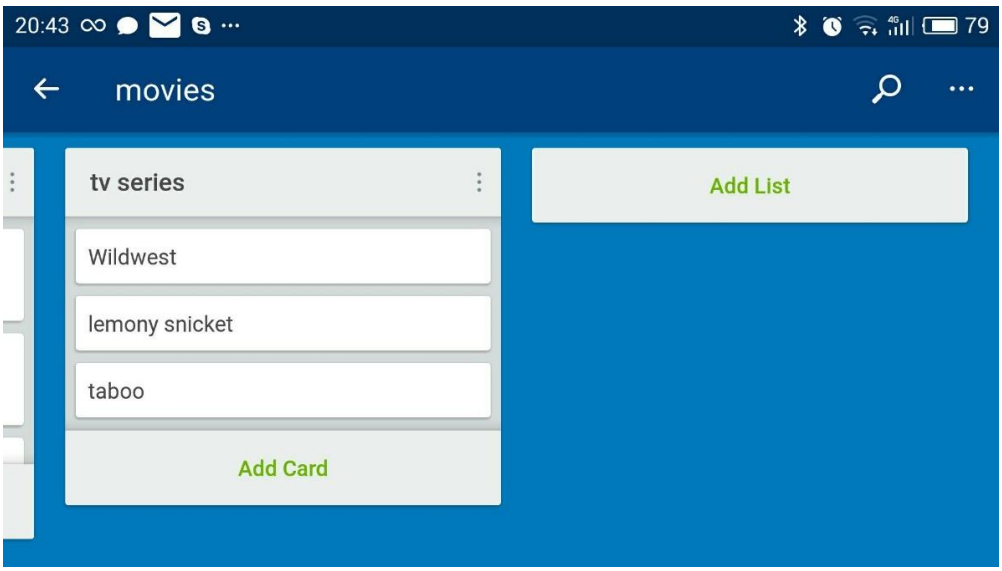

Рисунок 1.3 – Пример интерфейса пользователя мобильного приложения Trello

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 пользователя обязательными полями или определенным рабочим процессом;
– простота и низкий порог входа;
– существует как бесплатная, так и платная версия приложения;
– удобство в планировании задач;
– наличие мобильного клиента.

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).

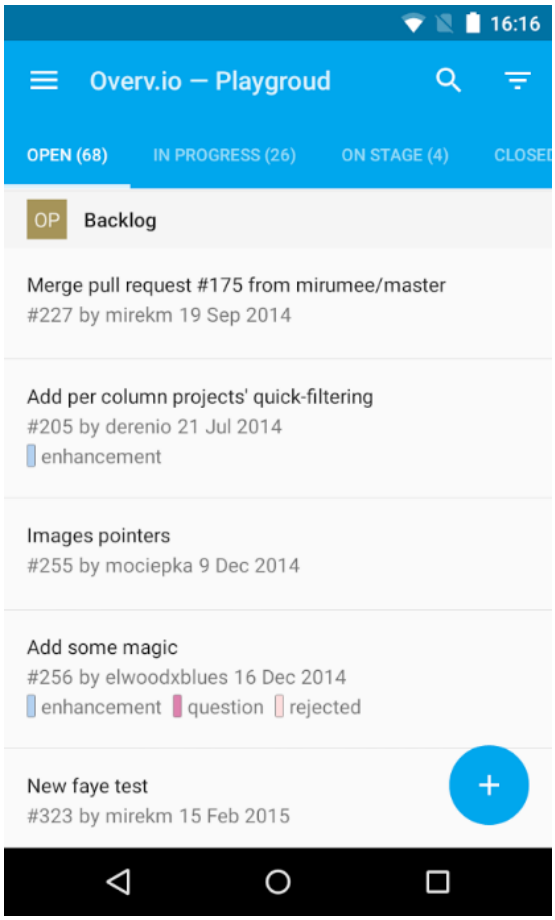

Рисунок 1.4 – Пример интерфейса пользователя мобильного приложения Overvio

Достоинства:

– высокий уровень настраиваемости рабочего процесса в рамках гибкой методологии;
– интеграция с Github;
– бесплатное распространение;
– низкий порог входа;
– наличие мобильного клиента.

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 сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение JavaScript находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

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, который на тот момент работал в компании Netscape. Впервые этот язык появился в браузере именно 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 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 поддержки большинства популярных браузеров необходимо использовать транскомпиляторы, такие, как Babel.

Для написания проекта были рассмотрены основные представители средств разработки кроссплатформенных приложений для мобильных устройств 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;

– сравнительно быстрая разработка за счет одновременного написания кода для двух платформ.

Минусы:

– низкая производительность в сравнении с нативными и гибридными 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 интерфейсами 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;

– сравнительно быстрая разработка;

– использование Angular 2;
– 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[8].

Плюсы React Native:

– кроссплатформенность;
– сравнительно быстрая разработка;
– схожие принципы и совместимость с кодом React приложений;
– использование нативных компонентов;
– высокая производительность приложения;
– поддержка сформировавшимся сообществом и автором фреймворка – технологическим гигантом Facebook.

Минусы:

– ограниченные возможности в стилизации компонентов;
– 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, в качестве фреймворка для написания мобильного приложения был выбрал React Native. Рассмотрим основные концепции React – предшественника и идейного вдохновителя React Native.

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, написанные 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, внешне выглядит схожим с HTML, но им не является. В итоге, он транслируется в цепочку вызовов функции React.createElement(), создающей элементы интерфейса пользователя. С JSX требуется меньше усилий на обучение веб-разработчика клиентской части 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 [8].

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 перемещением про истории состояний приложения. Redux можно использовать вместе с React, React-Native или с любой другой библиотекой представления [9].

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 2015) 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 [10].

1.3 Формирование требований к проектируемому ПС

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 которого 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 проектов и задач;
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 ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Описание функциональности ПС

Для представления функциональной модели была выбрана диаграмма вариантов использования UML [12], 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.

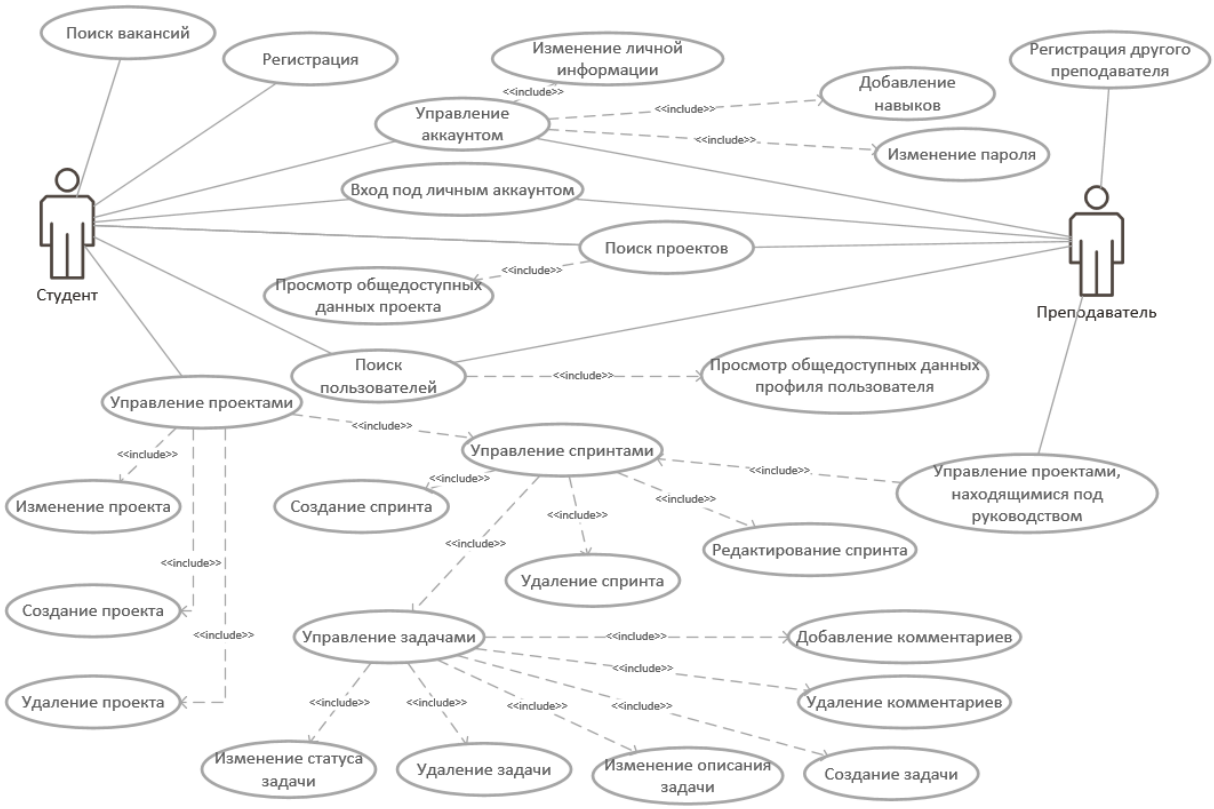

Рисунок 2.1 – Диаграмма вариантов использования

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 актора.

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 программного средства можно выделить следующие задачи, которые поможет решить реализуемое ПС:

– автоматизация организации групповых проектов среди студентов различных групп и специальностей;
– автоматизация управления проектами путем организации процесса разработки и распределения заданий и ролей исполнения между студентами;
– предоставить возможность руководителям проектов отслеживать прогресс выполнения в ходе разработки проекта.

В ходе разработки необходимо реализовать следующие возможности:

– осуществление аутентификации и авторизации пользователей;

– осуществление регистрации пользователей;

– смена пароля пользователя;

– создание проектов;

– редактирование информации о проекте, добавление позиций для участников проекта;

– удаление проектов;

– добавление заявки на участие в проекте;

– подтверждение либо отклонение заявки на участие в проекте со стороны членов команды проекта;

– назначение руководителя проекту;

– управление спринтами проекта;

– создание задач, редактирование информации о задачах и их удаление;

– изменение статуса задачи;

– осуществление поиска по заданиям, пользователей, позиций на проектах и самих проектов;

– просмотр и редактирование профиля пользователя;

– просмотр профилей других пользователей.

Разрабатываемый проект представляет собой клиентскую часть приложения, которая будет отправлять запросы на серверную часть. Для того, чтобы обеспечить корректную работу, требуется включить следующие меры в процесс разработки сервиса:

– все входящие данные должны проверяться на правильность, данные неправильного формата или неполная информация не должна отправляться на сервер для обеспечения более отзывчивого интерфейса пользователя;

– доступ ко всем данным приложения должен быть предоставлен только для авторизованных пользователей, чтобы обеспечить безопасность и приватность данных.

Дополнительно следует включить в ПС следующие вспомогательные функции:

– обеспечение плавности переходов с помощью анимаций;

– обеспечение единообразного интерфейса пользователя для всех частей приложения.

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):

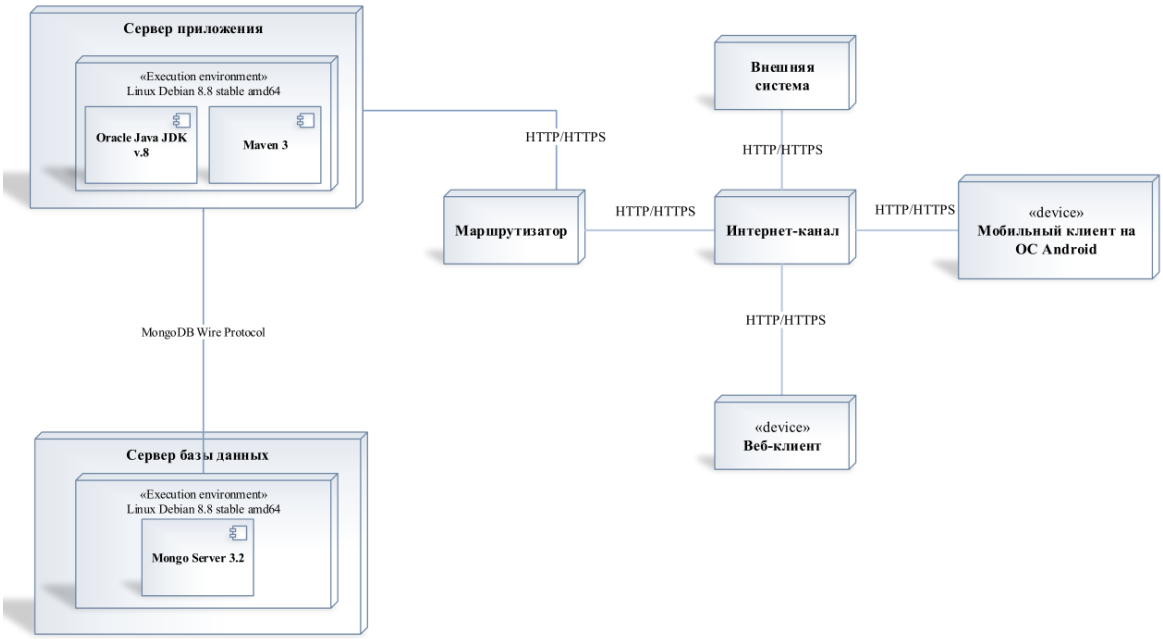

Рисунок 3.1 – Диаграмма развертывания

Данное ПС разрабатывается в виде клиент-серверного приложения. Для запуска веб-сервиса требуется рабочая станция с установленной 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 приложения, 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 1024x824;

– монитор;

– 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 Описание 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) [13].

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 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 иметь код 401 Unauthorised.

3.3 Разработка модели данных, получаемых с API

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

Рисунок 3.2 – Диаграмма классов данных, получаемых с сервера

3.3.1 Сущность «Профиль пользователя»

Данная сущность содержит следующий набор данных:
– идентификатор − идентификатор пользователя (первичный ключ);
– имя;
– фамилия;
– отчество;
– состояние;
– роль;

– 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 (список сущностей «Проект»).

Пользователь может иметь следующие состояния:
– подтвержден;
– не подтвержден.

Пользователь может иметь следующие роли:
– студент;
– преподаватель.

3.3.2 Сущность «Пользователь-уменьшенная»

Данная сущность содержит уменьшенный набор данных о пользователе:
– идентификатор − идентификатор пользователя (первичный ключ);
– имя;
– фамилия;
– ссылка на фото профиля.

3.3.3 Сущность «Пользователь»

Данная сущность содержит данные, необходимы при регистрации пользователя:
– идентификатор − идентификатор пользователя (первичный ключ);
– имя;
– фамилия;
– пароль;
– адрес электронной почты.

3.3.4 Сущность «Проект»

Данная сущность содержит следующий набор данных:
– 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.

3.3.5 Сущность «Задача»

Данная сущность содержит следующий набор данных:
– 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;
– список тегов задачи;
– статус задачи;
– идентификатор спринта задачи;
– идентификатор проекта;
– дата, к которой задача должна быть выполнена.
Тип задачи может иметь следующие значения:
– разработка;
– ошибка;
– косметические исправления;
– не требует исправления;
– предложение.
Приоритет задачи может иметь следующие значения:
– низкий;
– незначительный;
– средний;
– высокие;
– блокирующая задача.

3.3.6 Сущность «Задача с информацией о пользователях»

Данная сущность включает в себя расширенную информацию о авторе задачи и ответственном за выполнение. Содержит следующий набор данных:
– идентификатор задачи – первичный ключ сущности;
– название задачи;
– приоритет задачи;
– тип задачи;
– список идентификаторов подзадач;
– идентификатор родительской задачи;
– список идентификаторов смежных задачи;
– описание задачи;
– автора задачи – сущность «Пользователь-уменьшенная»;

– ответственного за задачу – сущность «Пользователь-уменьшенная»;
– планируемое время на решение задачи;
– уровень задачи;
– список тегов задачи;
– статус задачи;
– идентификатор спринта задачи;
– идентификатор проекта;
– дата, к которой задача должна быть выполнена.

3.3.7 Сущность «Университет»

Данная сущность содержит следующий набор данных:
– название университета – первичный ключ сущности;
– название факультета;
– название специальности;
– номер группы;
– год начала обучения;
– год окончания обучения.

3.3.8 Сущность «Проект-уменьшенная»

Данная сущность содержит следующий уменьшенный набор данных в сравнении с сущностью «Проект»:
– идентификатор проекта – первичный ключ сущности;
– название проекта;
– описание проекта;
– дата старта проекта;
– дата закрытия проекта;
– ссылка на Github репозиторий.

3.3.9 Сущность «Новый проект»

Данная сущность содержит следующий набор данных, необходимый для создания или редактирования проекта:
– идентификатор проекта – первичный ключ сущности;
– название проекта;
– описание проекта;
– ссылка на Github репозиторий;
– список позиций – список сущностей «Позиция»;
– дата старта проекта;
– дата закрытия проекта.

3.3.10 Сущность «Пользователь-Проект»

Данная сущность содержит следующий набор данных:
– идентификатор проекта – первичный ключ сущности;

– информация о позиции - сущность «Позиция-Информация»;
– текущий ли проект;
– дата старта проекта;
– дата закрытия проекта.

3.3.11 Сущность «Позиция»

Данная сущность содержит следующий набор данных:
– идентификатор пользователя – первичный ключ сущности;
– название позиции;
– описание позиции;
– тип позиции;
– идентификатор назначенного пользователя.
Тип позиции может иметь следующие значения:
– фуллстек-программист;
– программист клиентской части;
– программист серверной части;
– мобильный разработчик;
– иной разработчик;
– тестировщик;
– дизайнер;
– бизнес-аналитик;
– инженер;
– лидер проекта.

3.3.12 Сущность «Позиция-Информация»

Данная сущность содержит следующий набор данных:
– название позиции;
– описание позиции;
– тип вакансии;

3.3.13 Сущность «Навык»

Данная сущность содержит следующий набор данных:
– идентификатор навыка – первичный ключ сущности;
– название навыка;
– уровень владения навыком;
– список идентификаторов пользователей, подтвердивших владение навыком;

3.3.14 Сущность «Аккаунт социальной сети»

Данная сущность содержит следующий набор данных:
– идентификатор записи – первичный ключ сущности;

– ссылка;
– тип социальной сети;

Тип социальной сети может иметь следующие значения:

– Вконтакте;
– Facebook;
– Twitter;
– Skype;
– Google +;
– Другая.

3.4 Описание алгоритма работы с программой

Схема алгоритма выполнена на листе формата А1 (см. Графическое приложение). Данная схема отображает алгоритм работы с программным средством. Для входа в приложение необходимо пройти аутентификацию с данными аккаунта пользователя. Далее пользователь имеет ряд путей работы:

– авторизация и регистрация;
– создание и редактирование проектов, просмотр детальной информации о проектах;
– смена пароля;
– просмотр и редактирование данных профиля пользователя;
– создание и удаление спринтов;
– просмотр доски задач проекта, создание и редактирование задач, просмотр полной информации о задаче.

Если пользователь закрывает приложение, его работа с средством управления проектами закончена.

3.5 Описание алгоритма работы с списком проектов пользователя

После входа в приложение под своей учетной записью, пользователь попадает на экран с списком проектов, в которых он принимает, либо принимал участие. Схема алгоритма выполнена на листе формата А1 (см. Графическое приложение), а также на рисунке 3.3.

При загрузке экрана отправляется запрос на сервер на получение списка проектов текущего пользователя. В случае, если проектов нет, то вместо списка проектов отображается предложение создать новый проект. Если же проекты есть, то они фильтруются и отображаются в двух секциях: сначала список активных проектов, затем список завершенных. Далее программа ждет действий пользователя. С экрана просмотра списка проектов есть возможность перейти в следующие экраны:

– экран создания проекта;
– экран просмотра списка задач проекта;
– экран просмотра проекта;
– предыдущий экран из истории навигации (при нажатии на кнопку «Назад»).

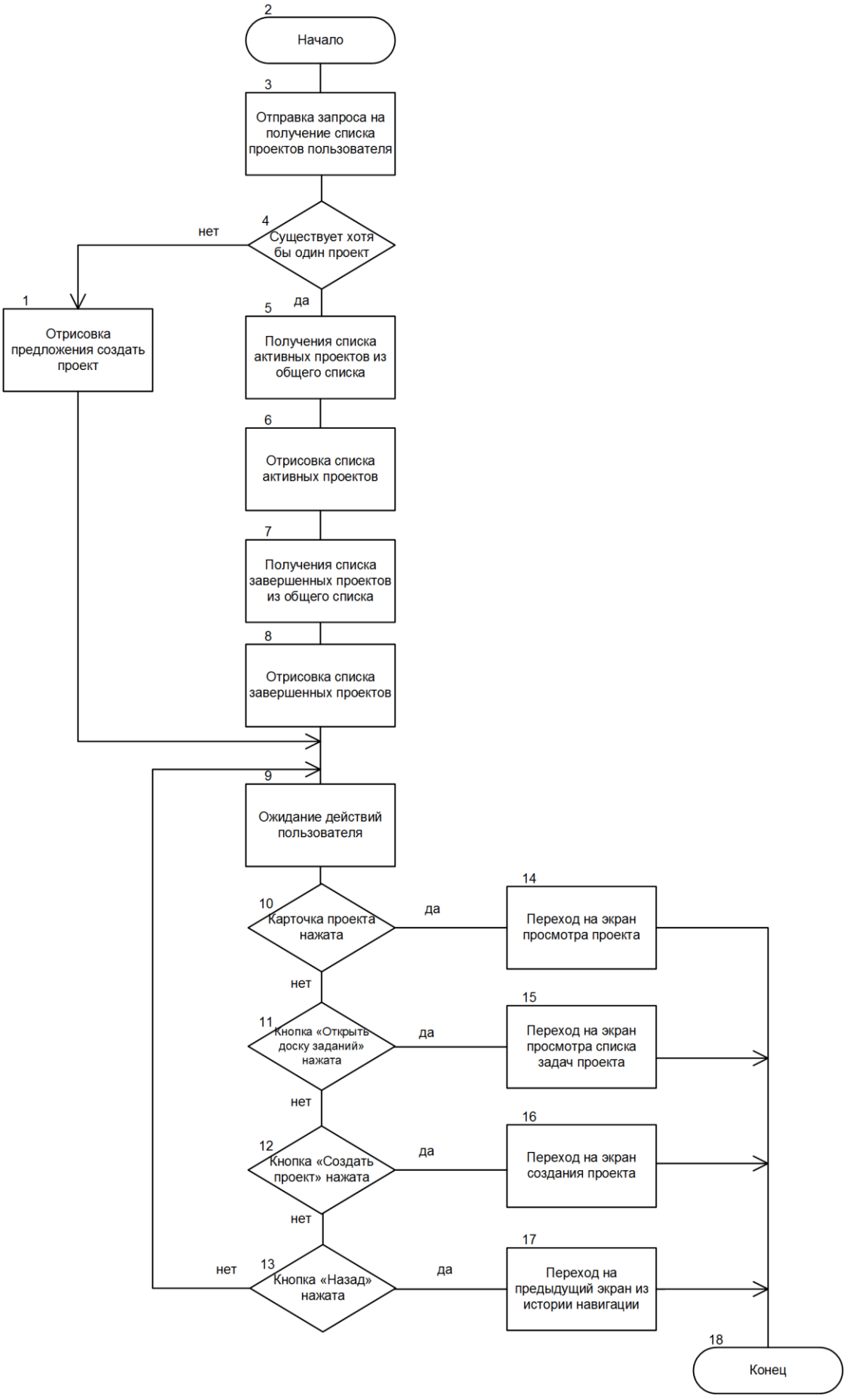

Рисунок 3.3 – Схема алгоритма работы с списком проектов пользователя

3.6 Описание алгоритма работы компонента редактирования профиля

Схема алгоритма работы данного компонента выполнена на листе формата А1 (см. Графическое приложение), а также приведена на рисунке 3.4.

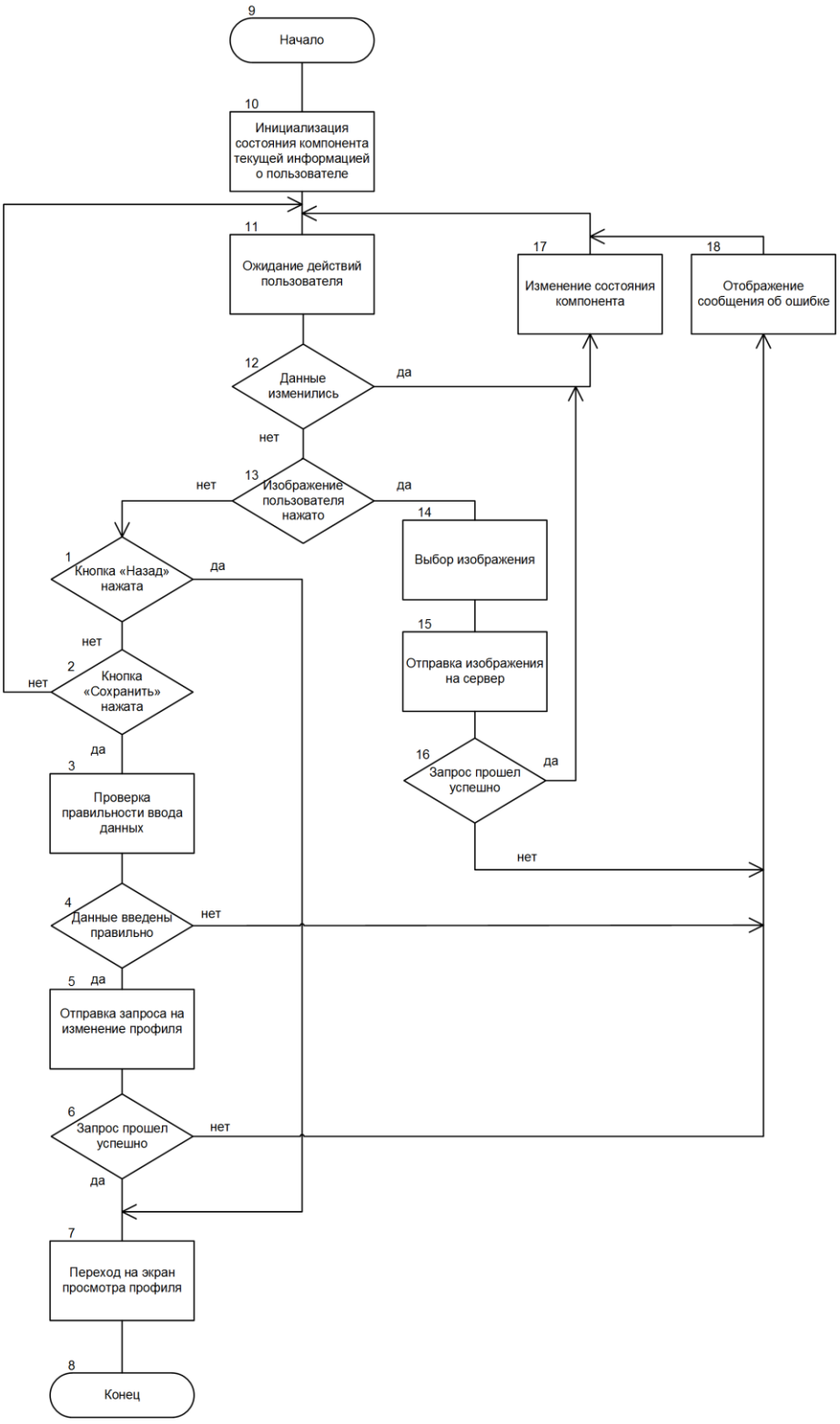

Рисунок 3.4 – Схема алгоритма работы компонента редактирования профиля

Для регистрации в приложении пользователю необходимо ввести только базовую информацию: имя, фамилию, адрес электронной почты. Для того, чтобы расширить объем данных о себе, предоставляемый пользователям приложения существует экран редактирования профиля.

На экране редактирования профиля пользователь может сменить изображение профиля, изменить личные данные, добавить навыки. Для того, чтобы отослать измененные данные профиля на сервер, необходимо нажать кнопку «Сохранить». В случае успешного завершения операции, пользователь перенаправляется на страницу просмотра профиля, иначе увидит сообщение об ошибке.

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, исполняемых 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 (сокр. англ. Representational State Transfer, «передача состояния представления» или «передача репрезентативного состояния») – стиль построения архитектуры распределенных систем [14].

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 их сложности.

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 во избежание устаревания данных. Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями. Наиболее часто используемым протоколом является HTTP [15].

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 составляет всего 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 объект, который описывает факт свершения какого-либо события, которое влечет за собой определённые 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 возвращающая новое состояние. Новое состояние должно быть новым объектом, оно должно быть сформировано 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 взаимодействия происходят с этим объектом.

Таким образом поток данных в redux является однонаправленным. 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 [16].

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 последовательность 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 функции 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 [17]. 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 – библиотека для построения клиентских приложений для ОС 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 в сравнении с стандартным компонентом;

– react-native-image-crop-picker – React Native компонент, предоставляющий возможность использовать стандартный для ОС Android диалог для выбора изображений и в дальнейшем модифицировать изображение перед отправкой;

– 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;

– 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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	Регистрация. Валидация ошибок	1. Запустить приложение. 2. Нажать на кнопку «Зарегистрироваться». 3. Нажать на кнопку «Зарегистрироваться».	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 «Зарегистрироваться». 3. Ввести разные значения в поля «Пароль и «Повторите пароль» 4. Нажать на кнопку «Зарегистрироваться».	Сообщение в всплывающем окне «Пароль и подтверждение не совпадают».	Тест успешно пройден

Продолжение таблицы 5.1

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный 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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	Авторизация. Валидация ошибок	1. Запустить приложение. 2. Нажать на кнопку «Войти».	Сообщение под полем ввода логина «Поле Email обязательно для заполнения».	Тест успешно пройден
2	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
3	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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	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
2	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
3	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
4	Проект. Редактирование 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

Продолжение таблицы 5.3

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
5	Проект. Редактирование 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
6	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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
2	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
3	Пользователь. Редактирование 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
4	Пользователь. Редактирование данных пользователя. Добавление аккаунта социальных сетей.	1. Повторение действий из пункта 14. 2. Внести ссылку на аккаунт в соответствующее 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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
5	Пользователь. Редактирование 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
6	Пользователь. Просмотр данных пользователя	1. Открыть левое боковое меню. 2. Нажать «Профиль»	1. Отображение экрана просмотра профиля.	Тест успешно пройден

Таблица 5.5 – Набор тест-кейсов функциональности «Задачи»

№ тест-кейса	Тестируемая функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
1	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
2	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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
3	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
4	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
5	Задачи. Редактирование 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 функциональность	Последовательность действий	Ожидаемый результат	Полученный результат
6	Задачи. Редактирование 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
7	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
8	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

Положительные результаты при прохождении тестов дают возможность судить о корректности работы программы в условиях работы с реальными данными, а также предполагает соответствие программного средства его функциональным требованиям.

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 устройство 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 либо в настройках приложений. 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 большинстве 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 устройства. Для установки приложения необходимо открыть приложение Google Play Market, с помощью поиска найти приложение «Coworkio» в списке приложений 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 (рисунок 6.1).

После авторизации осуществляется переход на страницу с списком проектов пользователя.

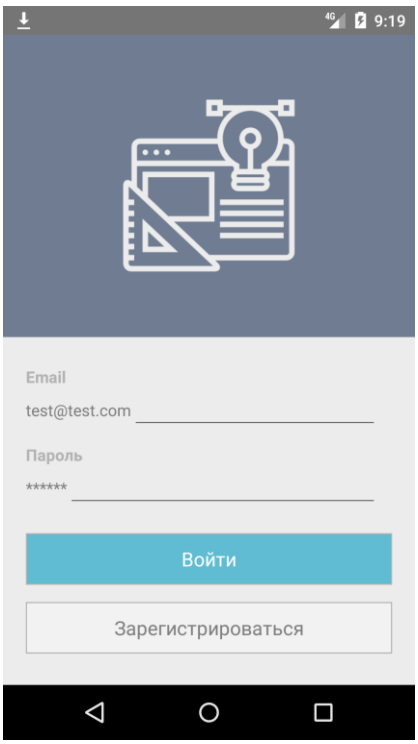

Рисунок 6.1 – Экран входа в систему

6.2.3 Экран списка проектов пользователя

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

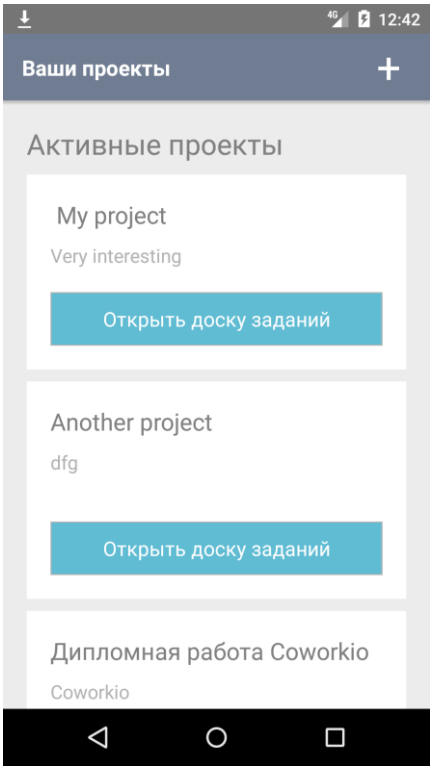

Рисунок 6.2 – Экран списка проектов пользователя

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 экрану создания нового проекта. При протягивании слева направо от левого края экрана открывается боковое меню (рисунок 6.3).

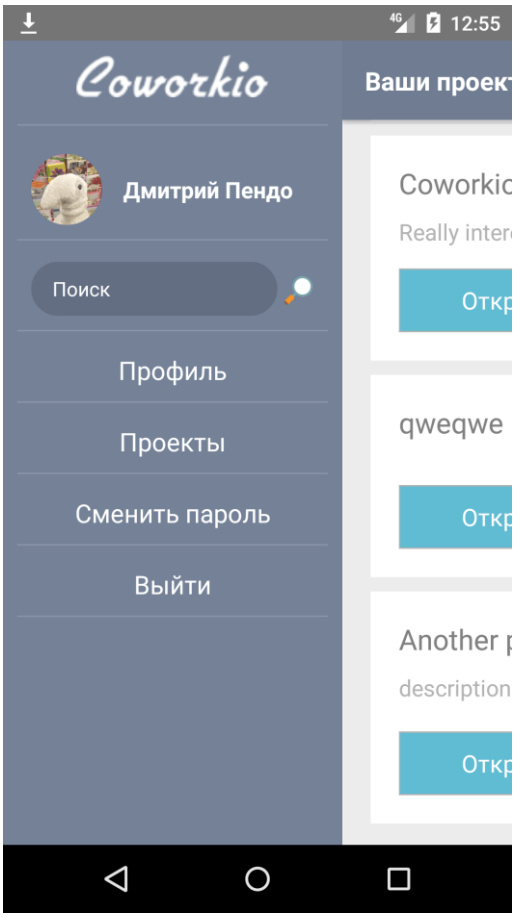

Рисунок 6.3 – Внешний вид открытого бокового меню

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.

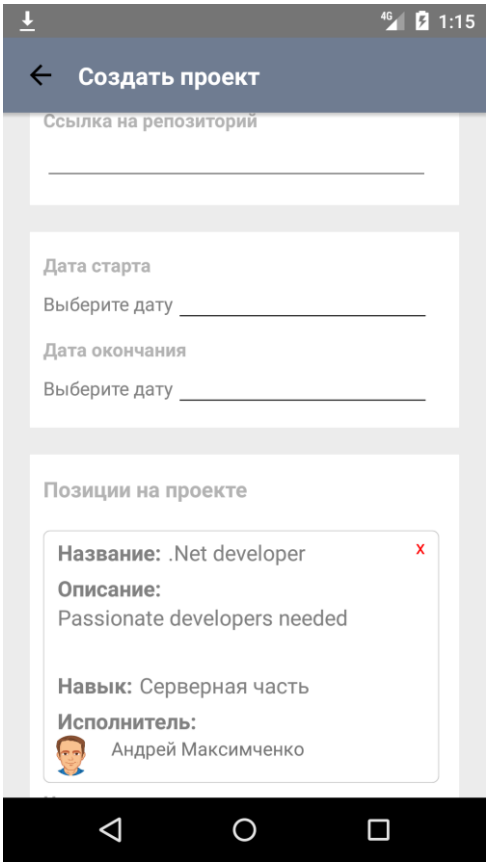

Рисунок 6.4 – Экран создания и редактирования проекта

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.

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.

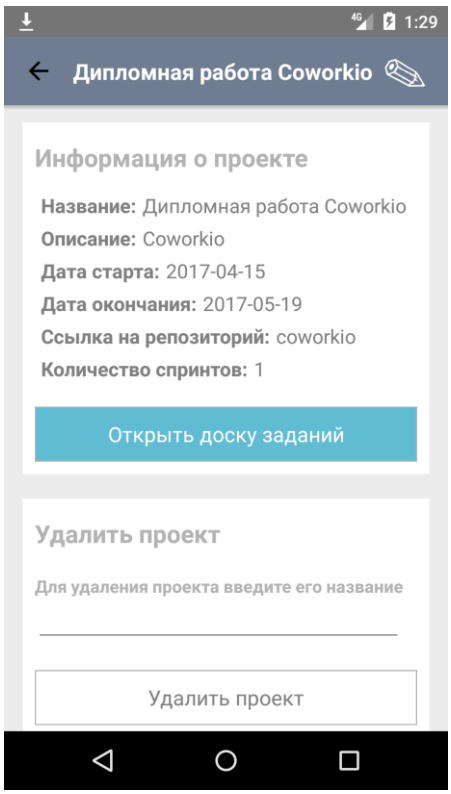

Рисунок 6.5 – Экран просмотра информации о проекте

6.2.7 Экран просмотра доски заданий

Экран просмотра доски заданий (рисунок 6.6) является основным в процессе управления задачами проекта.

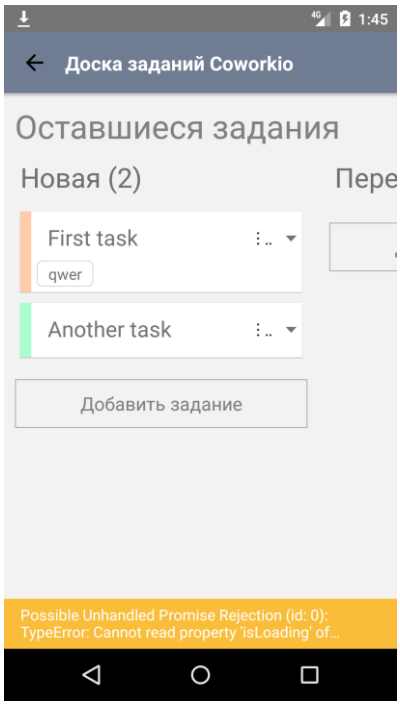

Рисунок 6.6 – Экран просмотра доски заданий

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.

6.2.8 Экран просмотра информации о задаче

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

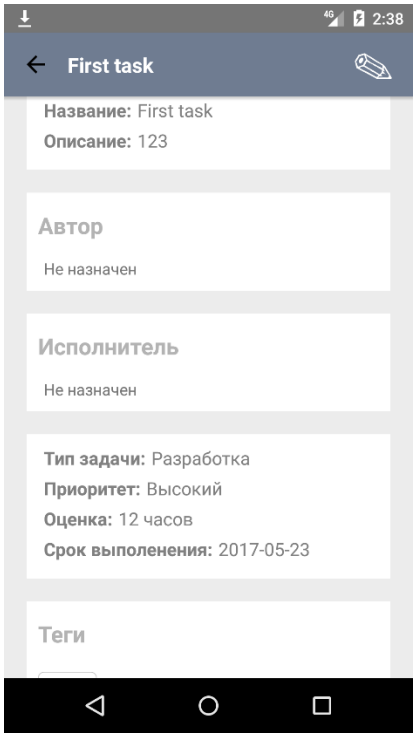

Рисунок 6.7 – Экран просмотра информации о задаче

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.

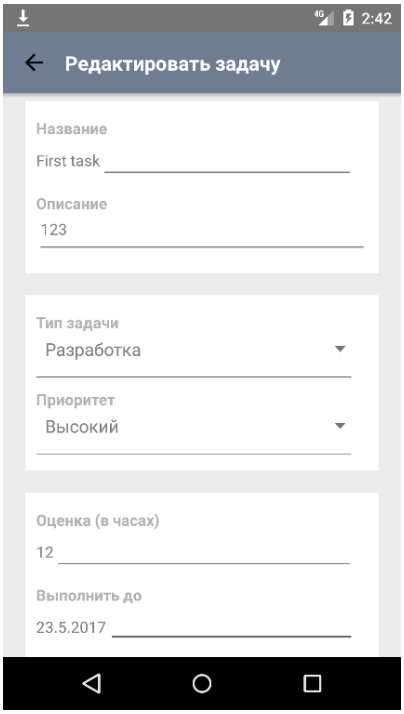

Рисунок 6.8 – Экран просмотра доски заданий

При открытии экрана в режиме редактирования, все имеющиеся 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.

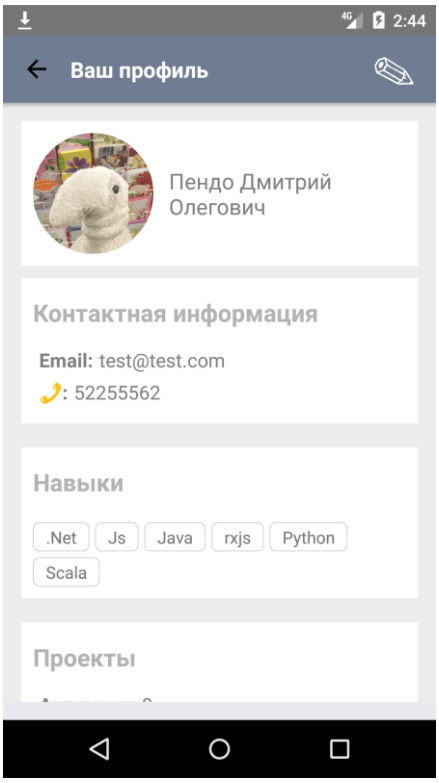

Рисунок 6.9 – Экран просмотра профиля пользователя

6.2.11 Экран редактирования данных профиля

Экран редактирования данных профиля (рисунок 6.10) позволяет вносить изменения в личные данные аккаунта пользователя.

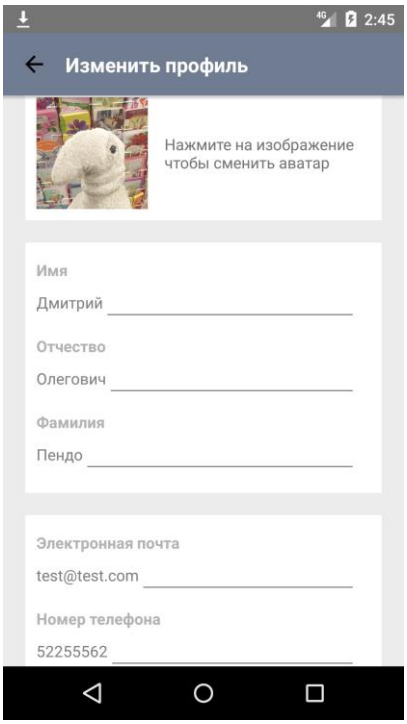

Рисунок 6.10 – Экран редактирования данных профиля

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.

Для смены изображение профиля пользователя нажмите на изображение. Далее откроется диалог выбора. 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. Более того данный проект может дать студентам возможность попробовать себя в роли менеджера проекта, поскольку, несмотря на наличие приложения, координирующего работу команды, всё же потребуются ещё и некоторые людские ресурсы на организацию работы команды. Таким образом, опыт работы в команде и навыки, приобретённые в ходе выполнения обязанностей менеджера проекта, пополнят копилку знаний студентов и будут выгодно смотреться в их резюме.

Воспользоваться программным средством смогут как 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 руководителей от университета, позволяя им легко наблюдать за выполнением заданий в установленные сроки, а руководителям в свою очередь даёт возможность предоставлять помощь студентам, если их настигли какие-либо затруднения.

Расчеты выполнены на основании методического пособия [16].

7.2. Расчет затрат и отпускной цены программного средства

7.2.1 Объем ПО (строки исходного кода, LOC)

$$V_y = \sum_{i=1}^{n} V_{yi}. \qquad (7.1)$$

Таблица 7.1 - Перечень и объем функций программного модуля

№ функции	Наименование (содержание) функции	Объем функции (LOC)	
		По каталогу V_i	Уточненный V_{yi}
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 Code), ПО функционального назначения. Vi = 11730 LOC.

7.2.2 Трудоемкость разработки ПО

Категории сложности ПО – 3-я категория, V_i = 11730 LOC. Нормативная трудоемкость ПО ($T_н$) – 260 чел./дн. Коэффициент сложности ($K_с$) – 0,12. Коэффициент, учитывающий степень использования при разработке ПО стандартных модулей ($K_т$) – 0,6. Коэффициент новизны разрабатываемого ПО ($K_н$) – 0,7.Общая трудоемкость:

$$T_о = T_н \cdot K_с \cdot K_т \cdot K_н = 260 \cdot 1{,}12 \cdot 0{,}6 \cdot 0{,}7 = 122 \text{ чел./дн.}$$

7.2.3 Численность исполнителей проекта ($Ч_р$)

$$Ч_р = \frac{T_о}{T_р \cdot Ф_{эф}}, \qquad (7.2)$$

где $T_р$ – срок разработки проекта (лет).
Эффективный фонд времени работы одного работника ($Ф_{эф}$):

$$Ф_{эф} = Д_г - Д_п - Д_в - Д_о = 236 \text{ дн.}$$

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

$$Ч_р = \frac{122}{0{,}25*236} \approx 2 \ .$$

7.3. Расчет сметы затрат и цены заказного ПО

1. Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле 7.3:

$$З_{oi} = \ \sum_{ni=1} * З_{ci} * Ф_{pi} * К, \qquad (7.3)$$

где n− количество исполнителей, занятых разработкой конкретного ПО; $З_{ci}$ −среднедневная заработная плата i-го исполнителя (*д.е.*); $Ф_{pi}$ − плановый фонд рабочего времени i-го исполнителя (дн.); К − коэффициент премирования (1,2).

Расчет основной заработной платы представлен в табл. 7.2.

Таблица 7.2. Расчет основной заработной платы

Категория исполнителя	Среднедневная заработная плата, руб.	Плановый фонд рабочего времени, дн.	Коэффициент премирования	Основная заработная 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 7.4:

$$З_д = \frac{З_о \cdot Н_д}{100}, \qquad (7.4)$$

где $Н_Д$ – норматив дополнительной заработной платы (20%).

Дополнительная заработная плата составит:

$$З_Д = 5490 \cdot 20/100 = 1098 \text{ руб.}$$

3. Отчисления в фонд социальной защиты населения и на обязательное страхование ($З_с$) определяются в соответствии с действующими законодательными актами по формуле 7.5:

$$З_{сз} = \frac{(З_о + З_д) \cdot Н_{сз}}{100}, \quad (7.5)$$

где $Н_{СЗ}$ – норматив отчислений в фонд социальной защиты населения и на обязательное страхование (34 +0,6%).

$$З_{сз} = (5490 + 1098) \cdot 34{,}6 / 100 = 2279.45 \text{ руб.}$$

4. Расходы по статье «Машинное время» ($Р_м$) включают оплату машинного времени, необходимого для разработки и отладки ПС, и определяются по формуле 7.6:

$$Р_м = Ц_м * Т_ч * С_р, \quad (7.6)$$

где $Ц_м$ – цена одного машино-часа; $Т_ч$ – количество часов работы в день; $С_р$ – длительность проекта.

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

$$Р_м = 1.4 \cdot 8 \cdot 61 = 683{,}2 \text{ руб.}$$

5. Затраты по статье «Накладные расходы» ($Р_н$), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды ($Р_Н$), определяются по формуле 7.7:

$$Р_н = \frac{З_о \cdot Н_{рн}}{100}, \quad (7.7)$$

где $Н_{РН}$ – норматив накладных расходов (50%).

Накладные расходы составят:

$$Р_н = 5490 \cdot 0.5 = 2745 \text{ руб.}$$

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

$$С_п = З_о + З_д + З_{сз} + Р_м + Р_н$$

$$С_п = 5490 + 1098 + 2279.45 + 683{,}2 + 2745 = 12295{,}65 \text{ руб.}$$

Прибыль ПС рассчитывается по формуле 7.8:

$$П_о = \frac{С_п \cdot У_{рп}}{100\%}, \quad (7.8)$$

где $П_{пс}$ – прибыль от реализации ПС заказчику (руб.);
$У_Р$ – уровень рентабельности ПС (25%);
$С_П$ – себестоимость ПС (руб.).

$$П_{пс} = 12295{,}65 \cdot 25/100 = 3073{,}91 \text{ руб.}$$

Прогнозируемая отпускная цена ПС вычисляется по формуле 7.9:

$$Ц_п = С_п + П_с. \quad (7.9)$$

$$Ц_п = 12295{,}65 + 3073{,}91 = 15369{,}56 \text{ руб.}$$

Налог на добавленную стоимость ($НДС_i$):

$$НДС = Ц_П * Н_{ДС} : 100, \quad (7.10)$$

где $Н_{дс}$– норматив НДС (%).

$$НДС = 15369{,}56 * 20\% : 100\% = 3073{,}91 \text{ руб.}$$

Прогнозируемая отпускная цена ($Ц_о$):

$$Ц_о = Ц_п + НДС. \quad (7.11)$$

$$Ц_о = 15369{,}56 + 3073{,}91 = 18443{,}47 \text{ руб.}$$

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

$$Н_с = \frac{Р_с}{С_р} \cdot 100, \quad (7.12)$$

где $Р_с$ – расходы на сопровождение и адаптацию ПС в целом по организации (руб.); $С_Р$ – смета расходов в целом по организации без расходов на сопровождение и адаптацию (руб.).

$$Р_с = 12295{,}65 \cdot 20 / 100 = 2459{,}13 \text{ руб.}$$

7.4. Оценка экономической эффективности применения программного средства у пользователя

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

Таблица 7.3 Исходные данные для расчета экономии ресурсов в связи с применением нового программного средства

Наименование показателей	Обозначения	Единицы измерения	Значение показателя		Наименование источника информации
			в базовом варианте	в новом варианте	
1	2	3	4	5	6
1. Капитальные вложения, включая затраты пользователя на приобретение	Кпр	руб.		18443,47	Договор заказчика с разработчиком
2. Затраты на сопровождение ПО	Кс	руб.		2459,13	Договор заказчика с разработчиком
3. Время простоя сервиса, обусловленное ПО, в день	$П_1$, $П_2$	мин	50	10	Расчетные данные пользователя и паспорт ПО
4. Стоимость одного часа простоя	$С_п$	руб.	30,1	30,1	Расчетные данные пользователя и паспорт ПО
5. Среднемесячная ЗП одного программиста	$З_{см}$	руб.	853,0	853,0	Расчетные данные пользователя
6. Коэффициент начислений на зарплату	$К_{нз}$		1,5	1.5	Рассчитывается по данным пользователя

Продолжение таблицы 7.3

Наименование показателей	Обозначения	Единицы измерения	Значение показателя		Наименование источника информации
			в базовом варианте	в новом варианте	
1	2	3	4	5	6
7. Среднемесячное количество рабочих дней	$Д_р$	день		21.5	Принято для расчета
8. Количество типовых задач, решаемых за год	$З_{т1}$, $З_{т2}$	задача	6000	6000	План пользователя
9. Объем выполняемых работ	$А_1$, $А_2$	задача	6000	6000	План пользователя
10. Средняя трудоемкость работ на задачу	$Т_{с1}$, $Т_{с2}$	Человеко-часов	0,5	0,2	Рассчитывается по данным пользователя
11. Количество часов работы ты в день	$Т_ч$	ч	8	8	Принято для расчета
12. Ставка налога на прибыль	$Н_п$	%		18	

Экономия затрат на заработную плату в расчете на 1 задачу ($С_{зе}$):

$$С_{зе} = \frac{З_{см} \cdot (Т_{с1} - Т_{с2}) : Т_ч}{Д_р}, \qquad (7.13)$$

где $З_{см}$ – среднемесячная заработная плата одного программиста (руб.); $Т_{с1}$, $Т_{с2}$ – снижение трудоемкости работ в расчете на 1 задачу (человеко-часов); $Т_ч$ – количество часов работы в день (ч); $Д_р$ – среднемесячное количество рабочих дней.

$$С_{зе} = 853*(0.5-0.2)/(8*21.5) = 1,49 \text{ руб..}$$

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

$$С_з = С_{зе} А_2, \qquad (7.14)$$

где $С_з$ – экономия заработной платы; $А_2$ – количество типовых задач, решаемых за год (задач).

$$С_з=1,49*6000=8940 \text{ руб.}.$$

Экономия с учетом начисления на зарплату ($С_н$):

$$С_н = С_з \cdot 1,5 = 8940 * 1,5 = 13410 \text{ руб.}.$$

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

$$С_с = \frac{(П_1 - П_2) \cdot Д_{рг} \cdot С_п}{60}, \quad (7.15)$$

где $Д_{рг}$ – плановый фонд работы сервиса (дней).

$$С_с = (180-120)* 21.5*30,1 / 60 = 647,15 \text{ руб.}.$$

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

$$С_о = С_н + С_с, \quad (7.16)$$

$$С_о = 13410 + 647,15 = 14057,15 \text{ тыс руб.}$$

7.4.2. 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 ($\Delta П_ч$), которая определяется по формуле 7.17:

$$\Delta П_ч = С_о - \frac{С_о \cdot Н_п}{100\%}, \quad (7.17)$$

где $Н_п$− ставка налога на прибыль (%).

$$\Delta П_ч = 14057,15 – (14057,15 * 0.18) = 11562,86 \text{ руб.}.$$

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 следующие коэффициенты: 2017 г. – 1, 2018-й – 0,8696, 2019-й – 0,7561, 2020 г. – 0,6575. Все рассчитанные данные экономического эффекта сводятся в таблицу 7.4.

Таблица 7.4. Расчет экономического эффекта от использования нового программного средства

Показатели	Единицы измерения	Годы			
		2017	2018	2019	2020
Результаты					
Прирост прибыли за счет экономии затрат ($П_ч$)	руб.		11562,86	11562,86	11562,86
То же с учетом фактора времени	руб.		10023.76	8715.46	7578.91
Затраты					
Приобретение ПО ($К_{пр}$)	руб.	18443,47			
Сопровождение ($К_с$)	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 данным дипломным проектом были проанализированы системы управления IT-проектами, были исследованы и проработаны различные методы и подходы к решению задач, поставленных с целью разработать программное средство для администрирования IT-проектов с последующим использованием его в образовательной сфере.

В процессе написания работы был выполнен анализ предметной области, также было проведено исследование существующих аналогов разработанного приложения. Результатами такого анализа стали выявление ниши на рынке предложения программных средств управления проектами для использования в образовательной сфере, выявление существующих плюсов и минусов схожих сервисов, а также учёт данных характеристик при разработке функциональных требований к новому программному средству.

На основе функциональных требований было произведено проектирование программного средства. В нем представлены разработка архитектуры ПС, разработка алгоритма программного средства и алгоритмов отдельных модулей. В разделе разработки архитектуры ПС детально рассмотрены схема алгоритма изменения данных пользователя, схема алгоритма работы с программным средством, схема работы функциональности списка проектов. Также в данном разделе представлена диаграмма развертывания приложения.

Согласно сформированным требованиям были составлены тестовые наборы, успешно пройденные в ходе тестовых испытаний программного средства. Положительные результаты при прохождении тестов дают возможность судить о корректности работы программы в условиях работы с реальными данными, а также предполагает соответствие программного средства его функциональным требованиям.

На завершающем этапе выполнения данного дипломного проекта были подробно описаны способы и методы использования разработанного программного средства. Изучение данной документации даёт возможность освоить работу с программой в относительно короткие сроки.

В ходе анализа экономической части проекта были выполнены операции по расчёту экономического эффекта от выпуска программного средства на рынок. В результате таких расчётов целесообразность разработки данного приложения подтвердилась. Также было выявлено, что вложенные в разработку инвестиции окупятся в течение трёх лет.

Среди целей данного проекта была разработка и реализация основного функционала приложения. Данная цель представляется выполненной.

Кроме того, в ходе выполнения задач, поставленных для достижения целей работы, создан простой, минималистичный, удобный и интуитивно понятный интерфейс, основанный на принципах UX, обеспечено высокое быстродействие, осуществлены функции создания и управления проектами, функции поиска исполнителей для проекта. Также создана возможность управления проектами путем управления задачами, осуществлена организация коллективной работы путем распределения задач между участниками, пользователям

предоставлена возможность осуществлять слежение за степенью прогресса проекта, обеспечены защищенность данных пользователей и разграничение прав доступа к различным данным и функциям приложения посредством ролей пользователя. Дизайн в свою очередь не противоречит стилевым рекомендациям мобильной операционной системы. Руководство пользователя также присутствует в наличии.

Так как программа разработана на языке JavaScript с использованием фреймворка React Native, существует возможность портирования данного программного средства на операционную систему iOS при произведении незначительных изменений.

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 данные. – Режим доступа: https://ru.wikipedia.org/wiki/Trello.

[4] Лекция - Возможности языка [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.ronl.ru/lektsii/informatika/873310/.

[5] Стандарт ECMA-262, 3я редакция [Электронный ресурс]. – Электронные данные. – Режим доступа: http://javascript.ru/ecma.

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

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

[8] Learning React Native. Building Native Mobile Apps with JavaScript / B. Eisenman – Sebastopol, CA: O'Reilly Media, Inc. – 272c.

[9] Краткое руководство по React JS [Электронный ресурс]. – Электронные данные. – Режим доступа: https://habrahabr.ru/post/248799/.

[10] Памятка. Redux documentation in russian [Электронный ресурс]. – Электронные данные. – Режим доступа: https://rajdee.gitbooks.io/redux-in-russian/.

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

[12] Хассан, Г. UML-проектирование систем реального времени параллельных и распределенных приложений / Г. Хассан – М.: ДМК Пресс, 2011 – 704с.

[13] The Anatomy of a JSON Web Token – Scotch [Электронный ресурс]. – Электронные данные. – Режим доступа: https://scotch.io/tutorials/the-anatomy-of-a-json-web-token.

[14] Understanding REST [Электронный ресурс]. – Электронные данные. – Режим доступа: https://spring.io/understanding/REST

[15] Representational state transfer – Wikipedia [Электронный ресурс]. – Электронные данные. – Режим доступа: https://en.wikipedia.org/wiki/Representational_state_transfer.

[16] The Complete Redux Book / I. Gelman, B. Dinkevich– NY.: Lean Publishing, 2015 – 179c.

[17] npm [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.npmjs.com/.

[18] Технико-экономическое обоснование дипломных проектов: Методическое пособие для студентов всех специальностей БГУИР. В 4-ч. Ч. 4: Проекты программного обеспечения / В.А. Палицын. – Минск: БГУИР, 2006 г. – 76 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Текст программы

```
import React, { Component } from 'react';
import { AppRegistry, StyleSheet, Text, View, Image, TextInput, Activity-
Indicator, AsyncStorage, Modal, TouchableHighlight, Keyboard, Animated } from
'react-native';
import Button from 'react-native-button';

import colors from '../../../styles/colors';
import formControlStyles from '../../../styles/form-controls';

import { Input } from '../../common/form-controls';

export default class LoginComponent extends Component {
  constructor(props){
    super(props);
    let email = props.navigation.state.params && props.naviga-
tion.state.params.email;
    let password = props.navigation.state.params && props.naviga-
tion.state.params.password;
    this.state = {
      isError: false,
      email: email || 'test@test.com',
      password:password ||'123456',
      imageSize: new Animated.Value(150)
    };
  }

  async componentWillReceiveProps(nextProps) {
    this.setModalVisibility(nextProps.auth.status === 'error')

    if (nextProps.auth.status === 'authenticated' && nextProps.auth.token
&& nextProps.auth.status !== 'error'  && !nextProps.auth.isPending){
      //redirect further

      con-
sole.log('**********************************************************************
********')
      console.log('NAVIGATING: MAIN
**********************************************')
      con-
sole.log('**********************************************************************
********')
```

```
            await AsyncStorage.setItem('token', nextProps.auth.token)
            this.props.navigation.navigate('Main')
        }
    }
    componentWillMount() {
        this.keyboardDidShowListener = Keyboard.addListener('key-
boardDidShow', this.keyboardDidShow.bind(this));
        this.keyboardDidHideListener = Keyboard.addListener('key-
boardDidHide', this.keyboardDidHide.bind(this));
    }

    componentWillUnmount() {
        this.keyboardDidShowListener.remove();
        this.keyboardDidHideListener.remove();
    }

    keyboardDidShow() {
        Animated.timing(
            this.state.imageSize,
            {
              toValue: 50,
              duration: 500,
            },
        ).start();
    }

    keyboardDidHide() {
        Animated.timing(
            this.state.imageSize,
            {
              toValue: 150,
              duration: 500,
            },
        ).start();
    }

    loginPressed() {
        let data = { email: this.state.email, password: this.state.password };
        this.props.onSubmit(data);
    }

    setModalVisibility(value){
        this.setState({isError: value});
    }
```

```
        render() {
            con-
sole.log('**********************************************************************
********')
            console.log('RENDER: LOGIN
**********************************************')
            con-
sole.log('**********************************************************************
********')

            const userData  = this.state;

            return (
                <View style={styles.container}>
                    <Modal
                        animationType={"slide"}
                        transparent={true}
                        visible={ this.state.isError }
                        onRequestClose={()=>{}}>
                        <View style={[styles.modalContainer]}>
                            <View style={[styles.modalInnerContainer]}>
                                <Text>Неправильные email и пароль</Text>
                                <Button
                                    onPress={() => this.setModalVisibility(false)}
                                    style={styles.modalButton}>
                                    Закрыть
                                </Button>
                            </View>
                        </View>
                    </Modal>
                    <View style={styles.header}>
                        <Animated.Image style={{height: this.state.imageSize, width:
this.state.imageSize}} source={require('../../../images/login-icon.png')}/>
                    </View>
                    <View style={styles.content}>
                    {!!userData.name &&
                        <Text style={styles.instructions}>
                            {userData.name}, ваш аккаунт был успешно создан. Вы
можете войти в приложение
                        </Text>
                    }

                    <Input title='Email' value={userData.email}
                        onChangeText={email => this.setState({ email })}/>
```

```
                    <Input title='Пароль' value={userData.password}
                        secureTextEntry={true} onChangeText={password =>
this.setState({password})} />

                    <Button containerStyle={[formControlStyles.buttonContainer,
formControlStyles.submitButtonContainer]}
                            style={[formControlStyles.buttonContent, formCon-
trolStyles.submitButtonContent]} onPress={this.loginPressed.bind(this)}>
                        {this.props.auth.isPending?
                            <ActivityIndicator color="#fff" animat-
ing={this.props.auth.isPending} />:
                            'Войти'
                        }
                    </Button>
                    <Button containerStyle={formControlStyles.buttonContainer}
style={formControlStyles.buttonContent} onPress={() => this.props.naviga-
tion.navigate('Register')}>
                        Зарегистрироваться
                    </Button>
                    </View>
                </View>
            )
        }
    }

    LoginComponent.propTypes = {
        onSubmit: React.PropTypes.func.isRequired
    }

    const styles = StyleSheet.create({
        container: {
            flex: 1,
            justifyContent: 'center',
            backgroundColor: colors.defaultBackground,
        },
        header: {
            backgroundColor: colors.themeBackground,
            flex: 1,
            height: 250,
            justifyContent: 'center',
            alignItems: 'center',
        },
        headerImage: {
            width: 150,
            height: 150,
```

```
    },
    content: {
      flex: 1,
      padding: 20
    },
    welcome: {
      fontSize: 20,
      textAlign: 'center',
      margin: 10,
    },
    instructions: {
      textAlign: 'center',
      color: '#333333',
      marginBottom: 5,
    },
    spinner: {
      alignItems: 'center',
      justifyContent: 'center',
      padding: 8,
    },
    modalContainer: {
      flex: 1,
      justifyContent: 'center',
      padding: 20,
      backgroundColor: 'rgba(0, 0, 0, 0.5)'
    },
    modalInnerContainer: {
      borderRadius: 10,
      alignItems: 'center',
      backgroundColor: '#fff',
      padding: 20
    },
    modalButton: {
      marginTop: 10,
    },
  })

  AppRegistry.registerComponent('LoginComponent', () => LoginCompo-
nent)

  import React from 'react';
  import { connect } from 'react-redux';

  import LoginComponent from './login.component';
  import { loginUser } from '../../../common/actions/auth.actions';
```

```
const mapStateToProps = (state) => {
   return  {
      auth: state.auth
   }
};

const mapDispatchToProps = (dispatch) => {
   return  {
      onSubmit: (credentials) =>  {
         dispatch(loginUser(credentials))
      }
   }
};

export default connect(mapStateToProps, mapDispatchToProps)(Log-
inComponent);

import React, { Component } from 'react';
import { AppRegistry, StyleSheet, Text, View, TextInput, ActivityIndicator,
Modal } from 'react-native';
import Button from 'react-native-button';

import colors from '../../../styles/colors';
import formControlStyles from '../../../styles/form-controls';

import CustomInput from '../../common/form-controls/input.component';

export default class RegisterComponent extends Component {

   constructor(props){
      super(props)
      this.state = {
         firstName: '',
         lastName: '',
         email: '',
         password: '',
         repeatPassword: '',
         isError: false
      };
   }

   componentWillReceiveProps(nextProps){
      if (nextProps.auth.isSuccessful && !nextProps.auth.isPending &&
!nextProps.auth.error ){
```

```
            //redirect further
            this.props.navigation.navigate('Login', { name: this.state.firstName,
email: this.state.email, password: this.state.password })
        }
    }

    registerPressed(){
        if(this.state.password === this.state.repeatPassword){
            let data = {
                firstName: this.state.firstName,
                lastName: this.state.lastName,
                email: this.state.email,
                password: this.state.password,
            }
            this.props.onSubmit(data);
        }
        else{
            this.setModalVisibility(true, 'Пароль и подтверждение не совпа-
дают!')
        }
    }
    setModalVisibility(value, message){
        this.setState({isError: value, errorMessage: message});
    }

    render() {
        return (
            <View style={styles.container}>
                <Modal
                    animationType={"slide"}
                    transparent={true}
                    visible={ this.state.isError }
                    onRequestClose={()=>{}}>
                    <View style={[styles.modalContainer]}>
                        <View style={[styles.modalInnerContainer]}>
                            <Text>{this.state.errorMessage}</Text>
                            <Button
                                onPress={() => this.setModalVisibility(false)}
                                style={styles.modalButton}>
                                Закрыть
                            </Button>
                        </View>
                    </View>
                </Modal>
```

```
                <Text style={styles.header}>
                  Регистрация
                </Text>

                <CustomInput title='Имя' value={this.state.firstName} on-
ChangeText={firstName => this.setState({ firstName })}/>

                <CustomInput title='Фамилия' value={this.state.lastName} on-
ChangeText={lastName => this.setState({ lastName })}/>

                <CustomInput title='Электронная почта' value={this.state.email}
onChangeText={email => this.setState({ email })}/>

                <CustomInput title='Пароль' value={this.state.password} se-
cureTextEntry={true}
                  onChangeText={password => this.setState({ password })}/>

                <CustomInput title='Повторите пароль' value={this.state.re-
peatPassword} secureTextEntry={true}
                  onChangeText={repeatPassword => this.setState({ repeatPass-
word })}/>

                <Button containerStyle={[formControlStyles.buttonContainer,
formControlStyles.submitButtonContainer]}
                    style={[formControlStyles.buttonContent, formCon-
trolStyles.submitButtonContent]} onPress={this.registerPressed.bind(this)}>
                  {this.props.auth.isPending?
                    <ActivityIndicator  color="#fff" animat-
ing={this.props.auth.isPending} />:
                    'Зарегистрироваться'
                  }
                </Button>
                <Button containerStyle={formControlStyles.buttonContainer}
style={formControlStyles.buttonContent} onPress={() => this.props.naviga-
tion.navigate('Login')}>
                  Войти
                </Button>
              </View>
            );
          }
        }

        RegisterComponent.propTypes = {
          onSubmit: React.PropTypes.func.isRequired
        }
```

```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        justifyContent: 'center',
        backgroundColor: colors.defaultBackground,
        padding: 20

    },
    header: {
        fontSize: 20,
        textAlign: 'center',
        margin: 10,
        color: colors.blockContent
    },
    instructions: {
        textAlign: 'center',
        color: '#333333',
        marginBottom: 5,
    },
    login: {
        textAlign: 'center',
        color: '#333333',
        marginBottom: 5,
        fontWeight: 'bold',
        marginTop: 30
    },
    submitButton: {
        color:"#fff",
    },
    spinner: {
        alignItems: 'center',
        justifyContent: 'center',
        padding: 8,
    },
    modalContainer: {
        flex: 1,
        justifyContent: 'center',
        padding: 20,
        backgroundColor: 'rgba(0, 0, 0, 0.5)'
    },
    modalInnerContainer: {
        borderRadius: 10,
        alignItems: 'center',
        backgroundColor: '#fff',
```

```
        padding: 20
      },
      modalButton: {
        marginTop: 10,
      },
    })


    AppRegistry.registerComponent('RegisterComponent', () => RegisterCom-
ponent)

    import React from 'react';
    import { connect } from 'react-redux';

    import RegisterComponent from './register.component';
    import { registerUser } from '../../../common/actions/auth.actions';

    const mapStateToProps = (state) => {
      return  {
        auth: state.auth
      }
    }

    const mapDispatchToProps = (dispatch) => {
      return  {
        onSubmit: (credentials, navigate) =>  {
          dispatch(registerUser(credentials));
        }
      }
    }

    export default connect(mapStateToProps, mapDispatchToProps)(Register-
Component)

    import React, { Component } from 'react';
    import { AppRegistry, StyleSheet, Text, View, TextInput, ActivityIndicator }
from 'react-native';
    import Button from 'react-native-button';

    import Sprint from './sprint/sprint.component';

    import colors from '../../styles/colors';

    export default class BoardComponent extends Component {
      static navigationOptions = ({ navigation, screenProps }) => ({
```

```
        title: `Доска заданий ${navigation.state.params.project.title}`,
      });

      componentWillMount(){
        this.props.getTasks(this.props.navigation.state.params.project.id);
      }

      createTask(sprintId, status){
        const { project } = this.props.navigation.state.params;

        this.props.navigation.navigate('CreateTask',{ sprintId, status, project });
      }

      viewTask(task){
        const { project } = this.props.navigation.state.params;

        this.props.navigation.navigate('ViewTask', { task, project });
      }

      render() {
        const { project } = this.props.navigation.state.params;
        const { tasks } = this.props.tasks;
        return (
          <View style={styles.container}>
            { project.sprints.map((sprint, index) => {
              const sprintTasks = tasks.filter((task) => task.sprintId ==
sprint.id)
              return <Sprint sprint={sprint} key={index} boardConfig={pro-
ject.board}
                             createTask={this.createTask.bind(this)}          view-
Task={this.viewTask.bind(this)}
                             updateTask={this.props.updateTask}
tasks={sprintTasks}></Sprint>
            })
          }
          </View>
        );
      }
    }

    BoardComponent.propTypes = {
      getTasks: React.PropTypes.func.isRequired,
      updateTask: React.PropTypes.func.isRequired,
    }
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'flex-start',
    backgroundColor: colors.buttonBackground,
  },
})


AppRegistry.registerComponent('BoardComponent', () => BoardComponent)

import React from 'react';
import { connect } from 'react-redux';

import BoardComponent from './board.component';

import { fetchTasksByProjectId, updateTask } from '../../common/actions/task.actions';

const mapStateToProps = (state) => {
  return  {
    tasks: state.tasks.tasksList
  }
}

const mapDispatchToProps = (dispatch) => {
  return  {
    getTasks: (projectId) => {
      dispatch(fetchTasksByProjectId(projectId));
    },
    updateTask: (task) => {
      dispatch(updateTask(task));
    }
  }
}

export default connect(mapStateToProps, mapDispatchToProps)(BoardComponent);

import React, { Component } from 'react';
import { AppRegistry, StyleSheet, ScrollView, Dimensions, View, Text } from 'react-native';
import Button from 'react-native-button';
```

```
import BoardColumn from '../column/boardColumn.component';

import colors from '../../../styles/colors';

export default class SprintComponent extends Component {
  scrollView = null;
  isScrolling = false;
  componentWillReceiveProps(nextProps){

  }

  onScroll(event) {

  }

  onEndDrag(event){
    let width  = Dimensions.get('window').width * 0.8;

    let currentOffset = event.nativeEvent.contentOffset.x;
    let isLeftToRight = currentOffset > this.offset;
    let page = Math.floor(currentOffset / width);

    let nextStop = (page + ( isLeftToRight ? 1 : 0)) * width;
    this.scrollView.scrollTo({x: nextStop, animated: true});

    this.offset = currentOffset;

  }

  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.title}>{this.props.sprint.id === 0 ?
'Оставшиеся задания' : this.props.sprint.name}</Text>
          <ScrollView horizontal={true} ref={(scrollView) => {
this.scrollView = scrollView; }}
            onMomentumScrollEnd={this.onEndDrag.bind(this)}
            onScroll={this.onScroll.bind(this)}  automaticallyAdjustConten-
tInsets={false}>
              { this.props.boardConfig.statuses.map((status, index) => {
                const availableStatuses = this.props.boardConfig.statuses;
                const tasks = this.props.tasks.filter((task) => task.status ===
status.order).map(task =>  { return {...task, availableStatuses}});
                return  <BoardColumn  title={status.title}  key={index}
```

```
tasks={tasks} updateTask={this.props.updateTask} viewTask={this.props.view-
Task}
                                createTask={()=>this.props.cre-
ateTask(this.props.sprint.id, status.order)}></BoardColumn> })
                    }
                </ScrollView>
            </View>
        );
    }
}

SprintComponent.propTypes = {
    sprint: React.PropTypes.any.isRequired,
    boardConfig: React.PropTypes.any.isRequired,
    tasks: React.PropTypes.any.isRequired,
    updateTask: React.PropTypes.func.isRequired,
    viewTask: React.PropTypes.func.isRequired,

}

const styles = StyleSheet.create({
    title:{
        paddingLeft: 10,
        marginTop: 10,
        color: colors.darkFontColor,
        fontSize: 30,
    }
})


AppRegistry.registerComponent('SprintComponent', () => SprintCompo-
nent)

import React, { Component } from 'react';
import { AppRegistry, StyleSheet, Text, ScrollView, View, TextInput, Activ-
ityIndicator, Dimensions } from 'react-native';
import Button from 'react-native-button';

import Task from '../../task/board/task.component'

import formControlStyles from '../../../styles/form-controls';

export default class BoardColumnComponent extends Component {
    componentWillReceiveProps(nextProps){
```

```
    }

    render() {
      var { width } = Dimensions.get('window');

      return (
        <ScrollView style={[{ width: width * .8 }]}>
          <View style={[styles.container]}>

          <Text style={styles.projectSectionHeader}>
            {this.props.title} ({this.props.tasks.length})
          </Text>
          { this.props.tasks.map((task, index)=>
            <Task task={task} key={index} updateTask={this.props.up-
dateTask} viewTask={this.props.viewTask}></Task>)
          }
          <Button containerStyle={[formControlStyles.buttonContainer,
{marginBottom: 60}]}
                  style={formControlStyles.buttonContent} on-
Press={this.props.createTask}>
            Добавить задание
          </Button>
          </View>
        </ScrollView>
      );
    }
  }

  BoardColumnComponent.propTypes = {
    title: React.PropTypes.string,
    updateTask: React.PropTypes.func.isRequired,
    viewTask: React.PropTypes.func.isRequired,
  }

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      padding: 10,
    },
    projectSectionHeader: {
      fontSize: 25,
      marginLeft: 5,
      marginBottom: 10,
    },
  })
```

```
AppRegistry.registerComponent('BoardColumnComponent', () => BoardColumnComponent)

import React, { Component } from 'react';
import { AppRegistry, StyleSheet, Text, View } from 'react-native';
import { connect } from 'react-redux';
import SideMenu from 'react-native-side-menu';

import Menu from '../menu';
import Navigation from '../../navigation/main.navigation';
 class MainComponent extends Component {

  constructor(){
    super();
    this.state = { isOpen: false };
  }

  navigateToProfile(profile){
    this.setState({ isOpen: false });
    this.navigator && this.navigator.dispatch({ type: 'Navigation/NAVIGATE', routeName: 'ViewProfile', params: {profile} });
  }

  logOut(){
    this.setState({ isOpen: false });
    this.props.navigation.navigate('Login');
  }

  navigateToDashboard(){
    this.setState({ isOpen: false });
    this.navigator && this.navigator.dispatch({ type: 'Navigation/NAVIGATE', routeName: 'Dashboard'});
  }

  render() {
    console.log('**********************************************************************')
    console.log('RENDER: MAIN ************************************************')
    console.log('**********************************************************************')
```

```
        const menu = <Menu isOpen={this.state.isOpen} navigateToDash-
board={this.navigateToDashboard.bind(this)}
                    navigateToProfile={this.navigateToProfile.bind(this)} log-
Out={this.logOut.bind(this)}/>

        return (
          <SideMenu menu={menu}>
            <Navigation ref={nav => { this.navigator = nav; }}/>
          </SideMenu>
        )
      }
    }

    const styles = StyleSheet.create({
      container: {
        flex: 1,
        alignItems: 'center',
        backgroundColor: '#F5FCFF',
      },
      welcome: {
        fontSize: 20,
        textAlign: 'center',
        margin: 10,
      },
      instructions: {
        textAlign: 'center',
        color: '#333333',
        marginBottom: 5,
      },
    })

    AppRegistry.registerComponent('MainComponent', () => MainComponent);

    export default connect(null, null)(MainComponent)

    import React, { Component } from 'react';
    import { AppRegistry, StyleSheet, Text, View, TouchableOpacity, Image,
TextInput, ActivityIndicator } from 'react-native';

    import colors from '../../styles/colors';

    export default class MenuComponent extends React.Component {
      componentWillMount(){
        this.props.fetchProfile();
```

```
    }

    logOut(){
      this.props.logOut();
      this.props.dispatchLogOut();
    }

    render() {
      const { profile, loading, error } = this.props.profile;
      if(!loading && !error && profile){
        return (
          <View style={styles.container}>
            <View style={styles.logo}>
              <Image source={require('../../images/coworkio.png')} />
            </View>
            <View style={[styles.option, styles.imageSection]}>
              <Image style={styles.image} source={profile.photoUrl? { uri: pro-
file.photoUrl } : require('../../images/placeholder.jpg')}/>
              <Text      style={styles.text}>{profile.firstName}      {profile.last-
Name}</Text>
            </View>
            <View style={[styles.option, styles.search]}>
              <TextInput style={styles.input} placeholder='Поиск' under-
lineColorAndroid='transparent' placeholderTextColor={colors.themeFontColor}/>
              <TouchableOpacity>
                <View>
                  <Text style={styles.optionText}> 🔍</Text>
                </View>
              </TouchableOpacity>
            </View>
            <TouchableOpacity onPress={() => this.props.navigateToProfile(pro-
file)}>
              <View style={styles.option}>
                <Text style={styles.optionText}>Профиль</Text>
              </View>
            </TouchableOpacity>
            <TouchableOpacity onPress={this.props.navigateToDashboard}>
              <View style={styles.option}>
                <Text style={styles.optionText}>Проекты</Text>
              </View>
            </TouchableOpacity>
            <TouchableOpacity>
              <View style={styles.option}>
                <Text style={styles.optionText}>Сменить пароль</Text>
              </View>
```

```
          </TouchableOpacity>
          <TouchableOpacity onPress={this.logOut.bind(this)}>
            <View style={styles.option}>
              <Text style={styles.optionText}>Выйти</Text>
            </View>
          </TouchableOpacity>
        </View>
      )
    }
    return (
      <View style={styles.container}>
        <ActivityIndicator color="#fff" animating={loading} />
      </View>
    )
  }
}

MenuComponent.propTypes = {
  navigateToProfile: React.PropTypes.func.isRequired,
  navigateToDashboard: React.PropTypes.func.isRequired,
  fetchProfile: React.PropTypes.func.isRequired,
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: colors.menuBackground,
  },
  logo: {
    height: 50,
    margin: 10,
    alignItems: 'center',
    justifyContent: 'flex-start',
    borderBottomWidth: 1,
    borderColor: colors.menuBorder
  },
  option: {
    alignItems: 'center',
    marginBottom: 5,
    padding: 10,
    marginLeft: 10,
    marginRight: 10,
    borderBottomWidth: 1,
    borderColor: colors.menuBorder
  },
```

```
  optionText: {
     color: colors.themeFontColor,
     fontSize: 18,
  },
  search: {
    flexDirection: 'row',
  },
  input: {
    flex: 1,
    backgroundColor: '#646E82',
    borderRadius: 50,
    padding: 5,
    paddingLeft: 15,
    paddingRight: 15,
    color: colors.themeFontColor,
  },
  imageSection: {
     flexDirection: 'row'
  },
  image: {
     width: 50,
     borderRadius: 25,
     height: 50,
     marginRight: 10
  },
  text: {
     marginRight: 5,
     marginLeft: 5,
     fontWeight: '500',
     fontSize: 16,
     color: colors.themeFontColor,
  },
})

AppRegistry.registerComponent('MenuComponent', () => MenuComponent)

import React from 'react';
import { connect } from 'react-redux';

import { fetchProfile } from '../../common/actions/profile.actions';
import { logoutUser } from '../../common/actions/auth.actions';

import MenuComponent from './menu.component';

 const mapStateToProps = (state) => {
```

```
    return  {
      profile: state.profile.activeProfile
    }
  }

const mapDispatchToProps = (dispatch) => {
    return  {
      fetchProfile: () => {
        dispatch(fetchProfile())
      },
      dispatchLogOut: () => {
        dispatch(logoutUser())
      }

    }
};

export default connect(mapStateToProps, mapDispatchToProps)(Menu-
Component)

import React, { Component } from 'react';
import { AppRegistry, StyleSheet, Text, View, Image, ScrollView, Linking,
TouchableOpacity } from 'react-native';
import Button from 'react-native-button';

import InfoCard from '../../common/infoCard/infoCard.component';
import  InfoCardSection  from '../../common/infoCard/infoCard-section.com-
ponent';
import { socialAccountTypes } from '../../common/accounts/enums';
import TagList from '../../common/tag/tagList.component';

import colors from '../../../styles/colors';
import formControlStyles from '../../../styles/form-controls';

export default class ViewProfileComponent extends Component {

    constructor(){
      super();
    }

    componentWillMount(){
      this.props.getProfile(this.props.navigation.state.params.profile.id);
    }

    static navigationOptions = ({ navigation, screenProps }) => ({
```

```
        title: `Ваш профиль`,
        headerRight: <Button containerStyle={formControlStyles.navbar-
ButtonContainer} style={formControlStyles.navbarButtonContent}
                    onPress={()=>{navigation.navigate('EditProfile', { profile:
navigation.state.params.profile})}}> ✎ </Button>
      });

      openUrl(url){
        Linking.openURL(url).catch(err => console.error('An error occurred',
err));
      }

      render() {
        const { profile } = this.props.profile;
        con-
sole.log('**********************************************************************
********')
        console.log('RENDER:                VIEW                PROFILE
**************************************************')
        con-
sole.log('**********************************************************************
********')

        return (
          <ScrollView style={styles.container}>
            <View style={[styles.section, styles.imageSection]}>
              <Image style={styles.image} source={profile.photoUrl? { uri:
profile.photoUrl } : require('../../../images/placeholder.jpg')}/>
              <Text style={styles.name}>
                {profile.lastName} {profile.firstName} {profile.middleName}
              </Text>
            </View>

            <InfoCard isVisible={profile.email || profile.phoneNumber} ti-
tle='Контактная информация'>
              <InfoCardSection title='Email' value={profile.email} isVisi-
ble={profile.email}></InfoCardSection>
              <InfoCardSection title='✆' value={profile.phoneNumber} is-
Visible={profile.phoneNumber}></InfoCardSection>
            </InfoCard>

            <InfoCard title='Навыки'>
              <TagList items={profile.skills.map(skill => skill.skillId)} />
            </InfoCard>
```

```
                <InfoCard title='Проекты'>
                  <InfoCardSection title='Активных' value={profile.projects.filter((p) => p.current).length} isVisible={profile.projects}></InfoCardSection>
                  <InfoCardSection title='Завершенных' value={profile.projects.filter((p) => !p.current).length} isVisible={profile.projects}></InfoCardSection>
                </InfoCard>

                <InfoCard title='Аккаунты' isVisible={(profile.github || (profile.accounts && profile.accounts.length))}>
                  <TouchableOpacity onPress={this.openUrl.bind(this,[profile.github])}>
                    <InfoCardSection icon={require('../../../images/icons/github.png')} value={profile.github} isVisible={profile.github}></InfoCardSection>
                  </TouchableOpacity>
                  {profile.accounts.map((item, index)=> {
                    const { icon } = socialAccountTypes.find((account)=> account.value === item.type);
                    return <InfoCardSection key={index} icon={icon} value={item.link}></InfoCardSection>
                  })}
                </InfoCard>

                <InfoCard title='Информация о университете' isVisible={profile.university}>
                  <InfoCardSection title='Университет' value={profile.university.university} isVisible={profile.university.university}></InfoCardSection>
                  <InfoCardSection title='Факультет' value={profile.university.faculty} isVisible={profile.university.faculty}></InfoCardSection>
                  <InfoCardSection title='Специальность' value={profile.university.department} isVisible={profile.university.department}></InfoCardSection>
                  <InfoCardSection title='Номер группы' value={profile.university.group} isVisible={profile.university.group}></InfoCardSection>
                  <InfoCardSection title='Срок обучения' value={(profile.university.startYear || '...') + ' - ' + (profile.university.endYear || '...')}
                    isVisible={profile.university.startYear || profile.university.endYear}></InfoCardSection>
                </InfoCard>
              </ScrollView>
            )
          }
        }
```

```
ViewProfileComponent.propTypes = {
   getProfile: React.PropTypes.func.isRequired
}

const styles = StyleSheet.create({
   container: {
      flex: 1,
      backgroundColor: colors.defaultBackground,
      padding: 15,
      marginBottom:15
   },
   imageSection: {
      backgroundColor: colors.cardBackground,
      padding: 10,
      flexDirection: 'row',
      alignItems: 'center',
   },
   image: {
      width: 100,
      borderRadius: 50,
      height: 100,
      marginRight: 10,
   },
   name:{
      fontSize: 18,
      color: colors.darkFontColor,
      padding: 3,
      flexWrap: 'wrap',
      flex: 1
   },
})


AppRegistry.registerComponent('ViewProfileComponent', () => ViewPro-
fileComponent)

import React from 'react';
import { connect } from 'react-redux';

import { fetchProfile } from '../../../common/actions/profile.actions';

import ViewProfileComponent from './viewProfile.component';

const mapStateToProps = (state) => {
   return  {
```

```
        profile: state.profile.activeProfile
      }
    };

    const mapDispatchToProps = (dispatch) => {
      return  {
        getProfile: () =>  {
          dispatch(fetchProfile())
        }
      }
    };

    export default connect(mapStateToProps, mapDispatchToProps)(ViewProfileComponent);

    import React, { Component } from 'react';
    import { AppRegistry, StyleSheet, Text, View, Image, TouchableOpacity , ActivityIndicator } from 'react-native';
    import Button from 'react-native-button';
    import ImagePicker from 'react-native-image-crop-picker';

    import { socialAccountTypes, universities, faculties } from '../enums';

    import colors from '../../../styles/colors';
    import formControlStyles from '../../../styles/form-controls';

    import { Form, Input, Picker, TagInput, AccountInput, SubmitButton } from '../../common/form-controls';
    import InfoCard from '../../common/infoCard/infoCard.component';

    export default class EditProfileComponent extends Component {
      constructor(props){
        super();
        let { profile } = props.profile;
        this.state = {
          id: profile.id || '',
          firstName: profile.firstName || '',
          middleName: profile.middleName || '',
          lastName: profile.lastName || '',
          email: profile.email || '',
          projects: profile.projects || [],
          phoneNumber: profile.phoneNumber || '',
          accounts: profile.accounts || [],
          github: profile.github || '',
          photoUrl: profile.photoUrl || '',
```

```
            university: {
              university: profile.university.university || universities[0].value,
              faculty: profile.university.faculty || faculties[0].value,
              department: profile.university.department || '',
              group: String(profile.university.group) || '',
              startYear: String(profile.university.startYear) || '',
              endYear: String(profile.university.endYear) || '',
            },
            skills: profile.skills || []
          };
        }

        componentWillReceiveProps(nextProps){
          if (!nextProps.newProfile.isLoading && !nextProps.newProfile.error
&& nextProps.newProfile.profileId){
            this.props.navigation.navigate('ViewProfile', { profile: this.state })
          }
        }

        static navigationOptions = ({ navigation, screenProps }) => ({
          title: 'Изменить профиль'
        });

        changeImage(){
          ImagePicker.openPicker({
            width: 300,
            height: 300,
            cropping: true
          }).then(image => {
            let url = 'https://api.cloudinary.com/v1_1/bookva/image/upload';

            let fd = new FormData();
            fd.append("upload_preset", "coworkio")
            fd.append("file", {
              uri: image.path,
              type: image.mime,
              name: 'ava'
            })

            fetch(url, {
              method: 'post',
              body: fd
            }).then(res => {
              res.json().then((res) => this.setState({photoUrl: res.url}))
            });
```

```
        });
    }

    submit(){
        //validate

        let data = {...this.state};
        this.props.onSubmit(data);
    }

    render() {
        con-
sole.log('**********************************************************************
********')
        console.log('RENDER:                EDIT                PROFILE
**********************************************')
        con-
sole.log('**********************************************************************
********')
        console.log(this.props.profile);

        return (
            <Form>
                <InfoCard style={styles.imagePicker}>
                    <TouchableOpacity onPress={this.changeImage.bind(this)}>
                        <Image style={styles.image} source={this.state.photoUrl?
{uri: this.state.photoUrl } : require('../../../images/placeholder.jpg')}/>
                    </TouchableOpacity>
                    <Text style={styles.imageText}>Нажмите на изображение
чтобы сменить аватар</Text>
                </InfoCard>

                <InfoCard>
                    <Input title='Имя' value={this.state.firstName} on-
ChangeText={firstName => this.setState({ firstName })}/>
                    <Input title='Отчество' value={this.state.middleName} on-
ChangeText={middleName => this.setState({ middleName })}/>
                    <Input title='Фамилия' value={this.state.lastName} on-
ChangeText={lastName => this.setState({ lastName })}/>
                </InfoCard>

                <InfoCard>
                    <Input title='Электронная почта' value={this.state.email} on-
ChangeText={email => this.setState({ email })}/>
                    <Input title='Номер телефона' value={this.state.phoneNumber}
```

```
onChangeText={phoneNumber => this.setState({ phoneNumber })} keyboardType='numeric'/>
                    <Input title='Аккаунт GitHub' value={this.state.github} onChangeText={github => this.setState({ github })}/>
                </InfoCard>

                <InfoCard>
                    <AccountInput title='Аккаунты' items={this.state.accounts} onItemsChange={(accounts)=> this.setState({accounts})}/>
                </InfoCard>

                <InfoCard>
                    <TagInput title='Навыки' hint='Введите навык' items={this.state.skills.map(skill => skill.skillId)}
                        onItemsChange={(skills)=> this.setState({skills: skills.map(skill => {return { skillId: skill }})})}/>
                </InfoCard>

                <InfoCard>
                    <Picker items={universities} value={this.state.university.university} title="Университет" onValueChange={(name) => {
                        let { university } = this.state;
                        university.university = name;
                        this.setState({university})
                    }}/>

                    <Picker items={faculties} value={this.state.university.faculty} title="Факультет" onValueChange={(faculty) => {
                        let { university } = this.state;
                        university.faculty = faculty;
                        this.setState({university})
                    }}/>

                    <Input title='Специальность' value={this.state.university.department} onChangeText={(department)=> {
                        let { university } = this.state;
                            university.department = department;
                            this.setState({university})
                        }}/>

                    <Input title='Номер группы' value={this.state.university.group} onChangeText={(group)=> {
                        let { university } = this.state;
                            university.group = group;
                            this.setState({university})
```

```
                    }}/>

                <Input title='Год начала обучения' value={this.state.university.startYear} keyboardType='numeric' onChangeText={(startYear)=> {
                    let { university } = this.state;
                        university.startYear = startYear;
                        this.setState({university})
                    }}/>

                <Input title='Год окончания' value={this.state.university.endYear} keyboardType='numeric' onChangeText={(endYear)=> {
                        let { university } = this.state;
                        university.endYear = endYear;
                        this.setState({university})
                    }}/>
            </InfoCard>

            <SubmitButton isLoading={this.props.newProfile.isLoading} onPress={this.submit.bind(this)} title='Сохранить' />
        </Form>
      )
    }
  }

  EditProfileComponent.propTypes = {
    onSubmit: React.PropTypes.func.isRequired
  }

  const styles = StyleSheet.create({
    container: {
      flex: 1,
      justifyContent: 'center',
      backgroundColor: colors.defaultBackground,
      padding: 15,
      marginTop: 0
    },
    imagePicker: {
      flexDirection: 'row',
      alignItems: 'center',
      marginBottom: 10
    },
    image: {
      width: 100,
      height: 100,
      marginRight: 15
```

```
    },
    imageText: {
      flex:1
    },
    instructions: {
      textAlign: 'center',
      color: '#333333',
      marginBottom: 5,
    },
    register: {
      textAlign: 'center',
      color: '#333333',
      marginBottom: 5,
      fontWeight: 'bold',
      marginTop: 30
    },
    inputs: {
      color: '#333333',
      marginLeft: 20,
      marginRight: 20
    },
    picker: {
      marginLeft: 25,
      marginRight: 25,
      borderBottomWidth: 0.7,
      borderColor: '#333333',
    },
    submitButton: {
      color:"#fff",
      fontWeight: 'bold'
    },
    spinner: {
      alignItems: 'center',
      justifyContent: 'center',
      padding: 8,
    },

  })


  AppRegistry.registerComponent('EditProfileComponent', () => EditProfile-
Component)

  import React from 'react';
  import { connect } from 'react-redux';
```

```
import EditProfileComponent from './editProfile.component';
import { upsertProfile } from '../../../common/actions/profile.actions';

const mapStateToProps = (state) => {
  return  {
    profile: state.profile.activeProfile,
    newProfile: state.profile.newProfile
  }
};

const mapDispatchToProps = (dispatch) => {
  return  {
    onSubmit: (profile) =>  {
      con-
sole.log('**********************************************************************
********')
      console.log('DISPATCHED:  EDIT  PROFILE
**************************************************')
      con-
sole.log('**********************************************************************
********')
      dispatch(upsertProfile(profile))
    }
  }
};

export default connect(mapStateToProps, mapDispatchToProps)(EditPro-
fileComponent);

import React, { Component } from 'react';
import { AppRegistry, StyleSheet, Text, View, TouchableOpacity, Picker }
from 'react-native';

import colors from '../../styles/colors';

export default class TaskComponent extends Component {
  statusColors = {
    LOW: '#3BFF00',
    MINOR: '#7FFFFF',
    NORMAL: '#FFE97F',
    MAJOR: '#FF7F7F',
    BLOCKER: '#FF0000',
  }
```

```
    viewTask(){

    }

    moveTask(status){
      this.props.updateTask({...this.props.task, status})
    }

    render() {
      const { task } = this.props;

      return (
        <TouchableOpacity onPress={() => this.props.viewTask(task)}>
          <View style={styles.container}>
            <View style={styles.content}>
              <View style={[styles.statusLabel, {backgroundColor: this.statusColors[task.priority]}]}></View>
              <Text style={styles.title} numberOfLines={1} ellipsizeMode='tail'>
                {task.title}
              </Text>
            </View>
            <View style={styles.picker}>
              <Picker prompt='Переместить в...' selectedValue={task.status} onValueChange={(status) => this.moveTask(status)}>
                {task.availableStatuses.map((item, index) =>
                  <Picker.Item key={index} color={task.status === item.order ? colors.defaultFontColor : colors.darkFontColor} label={': '+ item.title} value={item.order} />)
                }
              </Picker>
            </View>
          </View>
        </TouchableOpacity>
      );
    }
  }

  const styles = StyleSheet.create({
    container: {
      flexDirection: 'row',
      backgroundColor: '#FFFFFF',
      height: 50,
      margin: 5,
      borderWidth: .5,
```

```
        borderColor: 'gray',
        justifyContent: 'space-between',
    },
    statusLabel:{
        width: 10,
    },
    content:{
        flexDirection: 'row'
    },
    title: {
        fontSize: 20,
        margin: 10,
    },
    picker: {
        width: 50
    },
})


AppRegistry.registerComponent('TaskComponent', () => TaskComponent)
```