МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к дипломному проекту:**

Мобильное приложение для создания и прохождения мини-опросов

Дипломник ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Цягунович Т. В.

Руководитель проекта ­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бернацкий П. В., асс.

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Смелов В. В., к.т.н., доцент

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соболевский А. С., ст. преп.

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Нистюк О. А., асс.

Дипломный проект защищен с оценкой

Председатель ГЭК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дюбков В. К., к.т.н., доцент

Минск 2023

**Реферат**

ФИО

Подпись

Дата

Лист

1

*3*

*БГТУ 00.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

Реферат

Лит.

Листов

1

*74217096, 2023*

Пояснительная записка дипломного проекта содержит 74 страницы, 35 иллюстраций, 9 таблиц, 11 литературных источников, 12 приложений, 20 листингов.

ОПРОСЫ, JAVA, node.js, JAVASCRIPT, EXPRESS, PostgreSQL, ROOM, Sequelize, ANDROID, RETROFIT2, SQLITE

Целью дипломного проекта было мобильное приложение для создания и прохождения мини-опросов.

Пояснительная записка дипломного проекта состоит из реферата, содержания, введения, семи глав, заключения и списка использованных источников.

В первом разделе описаны аналоги данного программного средства и произведен патентный поиск по теме данного дипломного проекта.

Во втором разделе описан процесс проектирования приложения, описаны диаграммы, блок-схема и объекты базы данных.

В третьем разделе приведено описание процесса разработки мобильного приложения. Приводятся листинги исходного кода разработки базы данных, клиента, а так же описания архитектурных решений.

В четвертом разделе приведен анализ информационной безопасности.

В пятом разделе приведены результаты тестирования приложения.

В шестом разделе приведено руководство пользователя.

В седьмом разделе приводится расчет экономических параметров и себестоимости программного средства.

В заключении приведены результаты проделанной работы, приводятся сооб­ражения насчет использования данного программного средства.

**Abstract**

ФИО

Подпись

Дата

Лист

1

*4*

*БГТУ 00.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

Abstract

Лит.

Листов

1

*74217096, 2023*

The explanatory note of the degree project contains 74 pages, 35 illustrations, 9 tables, 11 literary sources, 12 appendices, 20 listings.

QUESTIONNAIRE, JAVA, node.js, JAVASCRIPT, EXPRESS, PostgreSQL, ROOM, Sequelize, ANDROID, RETROFIT2, SQLITE

The purpose of the graduation project was a mobile application for creating and passing mini-surveys.

The explanatory note of the graduation project consists of an abstract, content, introduction, seven chapters, conclusion and a list of sources used.

The first section describes the analogues of this software tool and conducts a patent search on the topic of this thesis project.

The second section describes the application design process, describes diagrams, flowchart and database objects.

The third section describes the process of developing a mobile application. The listings of the source code of the database development, the client, as well as descriptions of architectural solutions are given.

The fourth section provides an analysis of information security.

The fifth section shows the results of testing the application.

The sixth section contains the user's guide.

In the seventh section, the calculation of economic parameters and the self-cost of the software is given.

In conclusion, the results of the work done are presented, and suggestions are given about the use of this software tool.

Содержание

ФИО

Подпись

Дата

Лист

1

*5*

*БГТУ 00.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

Содержание

Лит.

Листов

2

*74217096, 2023*

[Введение 7](#_Toc136986493)

[1 Постановка задачи и анализ аналогичных решений 8](#_Toc136986494)

[1.1 Обзор аналогов 8](#_Toc136986495)

[1.1.1 Мобильное приложение Pollie 8](#_Toc136986496)

[1.1.2 Мобильное приложение Voliz 11](#_Toc136986497)

[1.1.3 Мобильное приложение Poll for all 12](#_Toc136986498)

[1.2 Патентный поиск 14](#_Toc136986499)

[1.3 Постановка задачи 15](#_Toc136986500)

[1.4 Выводы по разделу 15](#_Toc136986501)

[2 Проектирование приложения 16](#_Toc136986502)

[2.1 Архитектура приложения 16](#_Toc136986503)

[2.1.1 Диаграмма развертывания приложения 17](#_Toc136986504)

[2.2 Проектирование серверной части приложения 19](#_Toc136986505)

[2.2.1 Выбор архитектурного паттерна проектирования 19](#_Toc136986506)

[2.2.2 Проектирование удаленной базы данных 20](#_Toc136986507)

[2.3 Проектирование клиентской части приложения 22](#_Toc136986508)

[2.3.1 Выбор архитектурного паттерна проектирования 22](#_Toc136986509)

[2.3.2 Проектирование локальной базы данных 22](#_Toc136986510)

[2.3.3 Диаграмма вариантов использования 23](#_Toc136986511)

[2.3.4 Схема переходов между фрагментами 25](#_Toc136986512)

[2.3.5 Проектирование макетов экранов 26](#_Toc136986513)

[2.3.6 Блок-схема алгоритма получения достижений за комментарии 29](#_Toc136986514)

[2.4 Выводы по разделу 30](#_Toc136986515)

[3 Реализация приложения 31](#_Toc136986516)

[3.1 Реализация серверной части приложения 31](#_Toc136986517)

[3.1.1 Файловая структура проекта 31](#_Toc136986518)

[3.1.2 Реализация архитектурного паттерна MVC 32](#_Toc136986519)

[3.1.3 Разработка промежуточных программных слоев 37](#_Toc136986520)

[3.1.4 Разработка модуля для работы с почтой 37](#_Toc136986521)

[3.1.5 Работа со стандартом JSON WEB Token 39](#_Toc136986522)

[3.2 Реализация клиентской части приложения 40](#_Toc136986523)

[3.2.1 Файловая структура приложения 40](#_Toc136986524)

[3.2.2 Реализация локальной базы данных 41](#_Toc136986525)

[3.2.3 Работа с сервером с помощью Retrofit2 43](#_Toc136986526)

[3.2.4 Реализация архитектурного паттерна MVVM 45](#_Toc136986527)

[3.3 Выводы по разделу 46](#_Toc136986528)

[4 Анализ информационной безопасности приложения 47](#_Toc136986529)

[4.1 Возможные угрозы безопасности программного средства 47](#_Toc136986530)

[4.2 Реализованные методы защиты программного средства 47](#_Toc136986531)

[4.2.1 Методы защиты передачи данных по сети 47](#_Toc136986532)

[4.2.2 Методы защиты файла с настройками приложения 48](#_Toc136986533)

[4.2.3 Методы защиты баз данных от SQL-инъекций 48](#_Toc136986534)

[4.2.4 Методы защиты исходного кода приложения 48](#_Toc136986535)

[4.2.5 Методы защиты общедоступного API 49](#_Toc136986536)

[4.3 Выводы по разделу 49](#_Toc136986537)

[5 Тестирование приложения 50](#_Toc136986538)

[5.1 Локальное модульное тестирование 50](#_Toc136986539)

[5.2 Функциональное тестирование 51](#_Toc136986540)

[5.3 Выводы по разделу 53](#_Toc136986541)

[6. Руководство пользователя 54](#_Toc136986542)

[6.1 Руководство по развертыванию программного средства 54](#_Toc136986543)

[6.2 Руководство по использованию мобильного приложения 55](#_Toc136986544)

[6.3 Выводы по разделу 61](#_Toc136986545)

[7 Технико-экономическое обоснование проекта 62](#_Toc136986546)

[7.1 Общая характеристика разрабатываемого приложения 62](#_Toc136986547)

[7.2 Исходные данные для проведения расчётов 63](#_Toc136986548)

[7.3 Методика обоснования цены 63](#_Toc136986549)

[7.3.1 Объем программного средства 64](#_Toc136986550)

[7.3.2 Основная заработная плата 65](#_Toc136986551)

[7.3.3 Дополнительная заработная плата 66](#_Toc136986552)

[7.3.4 Отчисления в Фонд социальной защиты населения 66](#_Toc136986553)

[7.3.5 Расходы на материалы 67](#_Toc136986554)

[7.3.6 Расходы на оплату машинного времени 67](#_Toc136986555)

[7.3.7 Прочие прямые затраты 68](#_Toc136986556)

[7.3.8 Накладные расходы 68](#_Toc136986557)

[7.3.9 Сумма расходов на разработку программного средства 68](#_Toc136986558)

[7.3.10 Расходы на сопровождение и адаптацию 69](#_Toc136986559)

[7.3.11 Полная себестоимость 69](#_Toc136986560)

[7.3.12 Определение цены, оценка эффективности 69](#_Toc136986561)

[7.4 Выводы по разделу 72](#_Toc136986562)

[Заключение 73](#_Toc136986563)

[Список использованных источников 74](#_Toc136986564)

[Приложение А. Диаграмма архитектуры приложения 75](#_Toc136986565)

[Приложение Б. Диаграмма развертывания приложения 76](#_Toc136986566)

[Приложение В. Схема удаленной базы данных 77](#_Toc136986567)

[Приложение Г. Диаграмма вариантов использования 78](#_Toc136986568)

[Приложение Д. Схема переходов между фрагментами 79](#_Toc136986569)

[Приложение Е. Блок-схема алгоритма получения достижений за комментарии 80](#_Toc136986570)

[Приложение Ж. Листинг кода реализации моделей для базы данных 81](#_Toc136986571)

[Приложение И. Листинг кода реализации класса ApiError 85](#_Toc136986572)

[Приложение К. Листинг кода реализации программных слоев 86](#_Toc136986573)

[Приложение Л. Листинг кода для реализации класса TokenService 88](#_Toc136986574)

[Приложение М. Листинг класса UserService 90](#_Toc136986575)

[Приложение Н. Таблица экономических показателей 92](#_Toc136986576)

# Введение

Стремительное развитие технологий позволило людям выполнять различные задачи с помощью своих мобильных устройств, начиная от самообразования и заканчивая развлечениями. Существует множество онлайн и офлайн программ развлекательного характера и для интересного времяпрепровождения. Одним из нередко встречающихся развлечений, наблюдаемых чаще всего в социальных сетях, является прохождение небольших опросов, выкладываемых в созданных для этого сообществах их администраторами. Опросы могут быть любого характера: странными и забавными, серьезными и заставляющими задуматься. В большинстве своем в подобных сообществах пользователи могут только проходить опросы, опубликованные администраторами, без возможности создать свои, а так же без возможности их простого поиска, сортировки по интересующим темам или какому-либо рейтингу. Комментирование опросов часто перерастает в бесконечные споры, что вскоре уходят далеко от темы опроса и нередко занимают большую часть комментариев, от чего нельзя прочесть другие мнения. Однако, не смотря на все это, на подобные сообщества могут быть подписаны десятки тысяч пользователей. Можно сделать вывод, что необходимо создание отдельного приложения, позволяющего самим пользователям создавать опросы и без минусов, описанных выше.

Таким образом, целью дипломного проекта является мобильное приложение для создания и прохождения мини-опросов с возможностью оценивания опросов и комментариев к ним, поиска и сортировки.

Для достижения поставленной цели были выделены следующие задачи:

* найти и проанализировать существующие мобильные приложения, которые аналогичны теме дипломного проекта;
* спроектировать мобильное приложение и необходимые ему компоненты, такие как сервер и база данных;
* разработать мобильное приложение с его компонентами, которое будет удовлетворять поставленные цели;
* протестировать разработанное мобильное приложение;
* провести анализ информационной безопасности приложения;
* написать руководство пользователя для мобильного приложения;
* провести технико-экономическое обоснование данной работы.

ФИО

Подпись

Дата

Лист

1

*7*

*БГТУ 00.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

Введение

Лит.

Листов

1

*74217096, 2023*

# 1 Постановка задачи и анализ аналогичных решений

ФИО

Подпись

Дата

Лист

1

*8*

*БГТУ 01.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

1 Постановка задачи и анализ аналогичных решений

Лит.

Листов

8

*74217096, 2023*

## 1.1 Обзор аналогов

В интернете существует множество мобильных приложений, направленных на создание и прохождение опросов. Однако одна их часть направлена на оплачиваемое прохождение опросов без возможности их создания, преимущественно на английском языке. Другая же часть направлена на создание опросов или анкет, ссылку на которые нужно отправить другим пользователям, чтобы те могли пройти их. Иные приложения были не работоспособны или же заброшены.

Однако было подобрано несколько приложений, чья тема была наиболее приближена к теме разрабатываемого дипломного проекта, и проанализировано.

### 1.1.1 Мобильное приложение Pollie

После запуска данного приложения открывается главный экран. Если не было еще создано ни одного опроса, то отображаются надписи на черном фоне, предлагающие создать свой первый опрос. Если же уже были созданы опросы, то они отобразятся на экране, снимок которого предоставлен на рисунке 1.1.

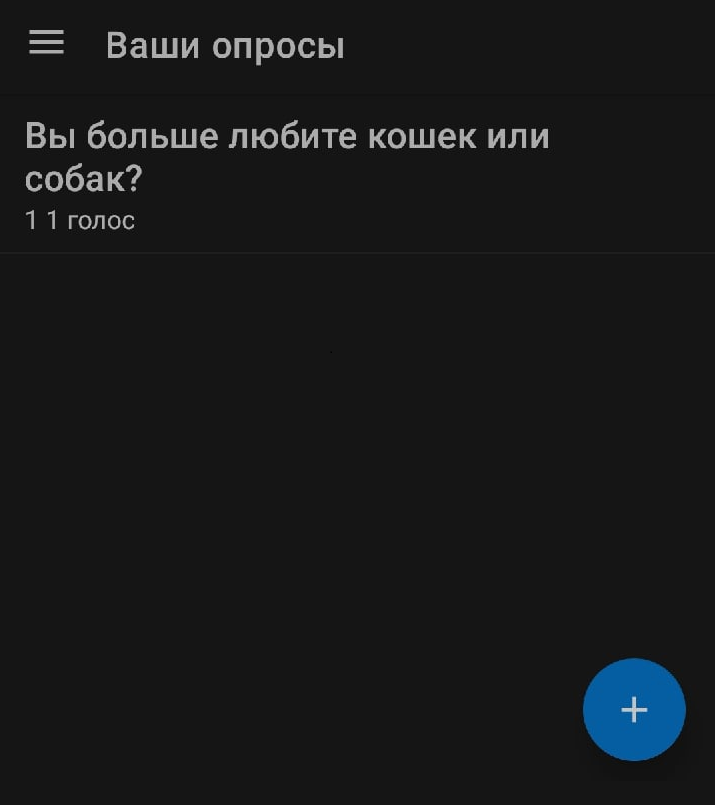


Рисунок 1.1 – Отображение созданных опросов

Так же для создания опроса имеется кнопка, находящаяся на главном экране справа внизу. С ее помощью можно кратчайшим путем перейти к экрану создания опроса, снимок которого изображен на рисунке 1.2.

На данном экране главными элементами являются поле для написания вопроса, добавления вариантов ответа и кнопка для вставки картинки в опрос. Можно так же добавить описание, дополнительно раскрывающее суть данного опроса.

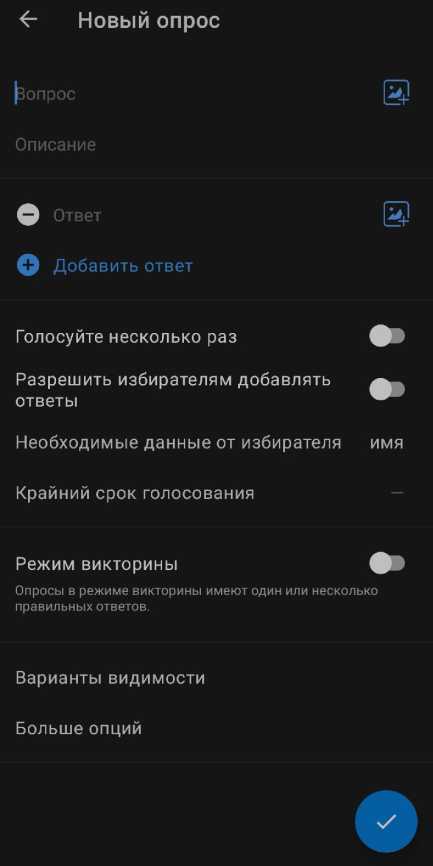


Рисунок 1.2 – Экран создания опроса

При нажатии на опрос в списке опросов на главном экране, он раскрывается на другом экране, изображенным на рисунке 1.3, где сверху размещена картинка, если она есть. После нажатия на нее, она раскрывается на весь экран, чтобы пользователь мог ее хорошо рассмотреть. Повторное нажатие возвращает обратно на данный экран. Ниже картинки размещается крупным шрифтом вопрос, под ним мелким шрифтом его пояснение, которое является необязательным описанием. Отсюда можно получить ссылку на опрос для его прохождения пользователями, нажав на соответствующую кнопку в нижнем правом углу или же посмотреть, как опрос выглядит в браузере, нажав на ссылку под описанием. Здесь можно увидеть количество голосов на тот или иной вариант ответа и их процентное соотношение.

Кнопки «аналитика» и «экспорт» неактивны без платного обновления приложения до версии «Pro» по подписке.

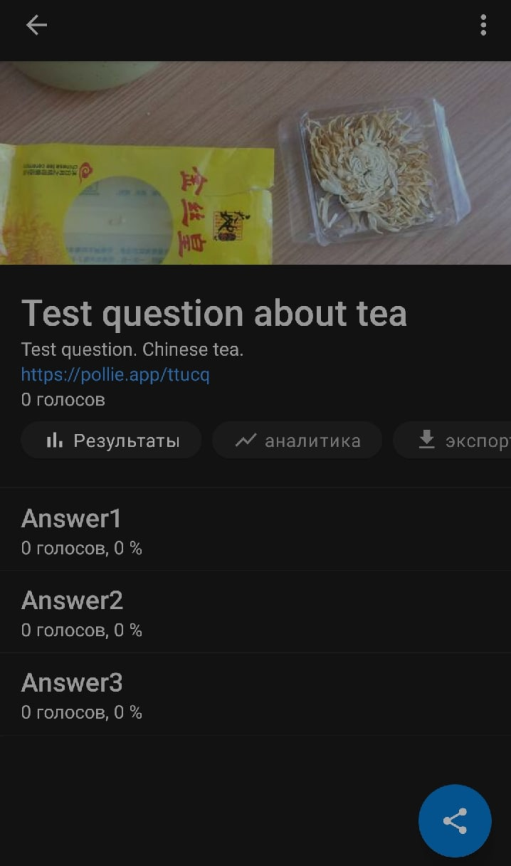


Рисунок 1.3 – Экран для голосования в опросе

Для связи с разработчиками в приложении так же имеется возможность отправить свои пожелания насчет будущего функционала приложения на почту, указав при этом свою почту для обратной связи. Снимок экрана для отправки пожеланий предоставлен на рисунке 1.4.

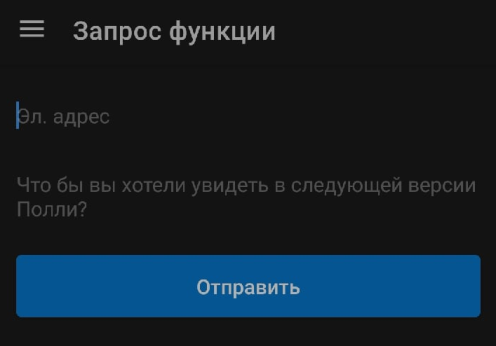


Рисунок 1.4 – Экран для связи с разработчиками

Проанализировав данное мобильное приложение, можно сделать следующие выводы: на главном экране приложения должны быть размещены списки опросов; все необходимые элементы для создания опроса должны быть размещены на одном экране, до которого имеется наикратчайший путь; на экране с самим опросом можно после голосования увидеть количество проголосовавших человек и процентное соотношение голосов; для обратной связи с пользователями можно использовать отправку писем на электронную почту разработчика.

### 1.1.2 Мобильное приложение Voliz

Функционально данное приложение схоже с предыдущим, но имеет меньше опций для создания опроса и может автоматически генерировать текст сообщения для WhatsApp со ссылками для его прохождения. Так же оно имеет еще одну особенность: опросы могут выкладываться публично и пройти их можно без ссылки, наткнувшись на них в общем списке опросов, что представлено на рисунке 1.5.

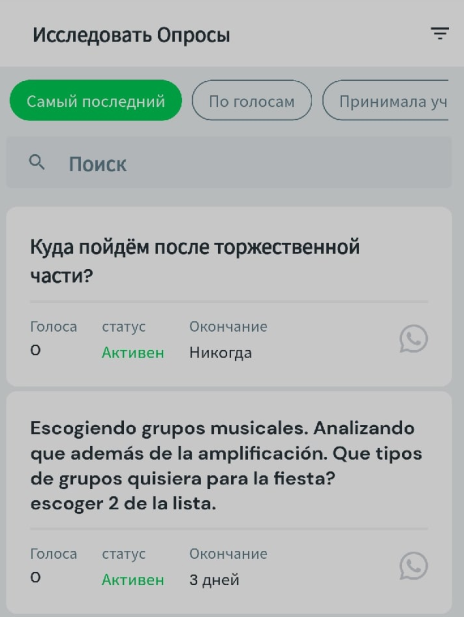


Рисунок 1.5 – Экран со списком публичных опросов

Помимо этого имеется строка поиска опросов по слову и возможность поиска опросов по другим критериям, в данном случае по голосам, по дате, а так же по тому, участвовал пользователь раньше в опросе или нет. При нажатии на опрос происходит переход на экран с выбором варианта ответа на вопрос, а так же с возможностью посмотреть подробности опроса и полученные результаты.

В аналитике опроса указана дата создания опроса, дата окончания, его статус, количество голосов и кем создан опрос. Для создателя опроса так же имеется возможность посмотреть, кто именно проголосовал и за какие варианты ответа.

Снимки экранов с описанными элементами приведены на рисунке 1.6.



Рисунок 1.6 – Вкладки экрана с опросом

После анализа данного мобильного приложения были сделаны следующие выводы: для создателя приложения имеется возможность посмотреть список проголосовавших людей за тот или иной вариант ответа; у опроса имеется возможность посмотреть дату создания и увидеть создателя.

### 1.1.3 Мобильное приложение Poll for all

Функционал данного приложения схож с предыдущими аналогами, однако имеются несколько важных и отличительных особенностей.

Первой отличительной особенностью является возможность регистрации в приложении разными способами, в том числе по почте, при входе в приложение. Снимок экрана регистрации представлен на рисунке 1.7.



Рисунок 1.7 – Регистрация в приложении

Так же вместе с этим имеется профиль и возможность взаимодействия с ним, а именно: редактирование имени пользователя, добавление картинки на аватарку, смена почты и пароля, а так же удаление профиля и выход из него. Описанный выше функционал представлен на рисунке 1.8.

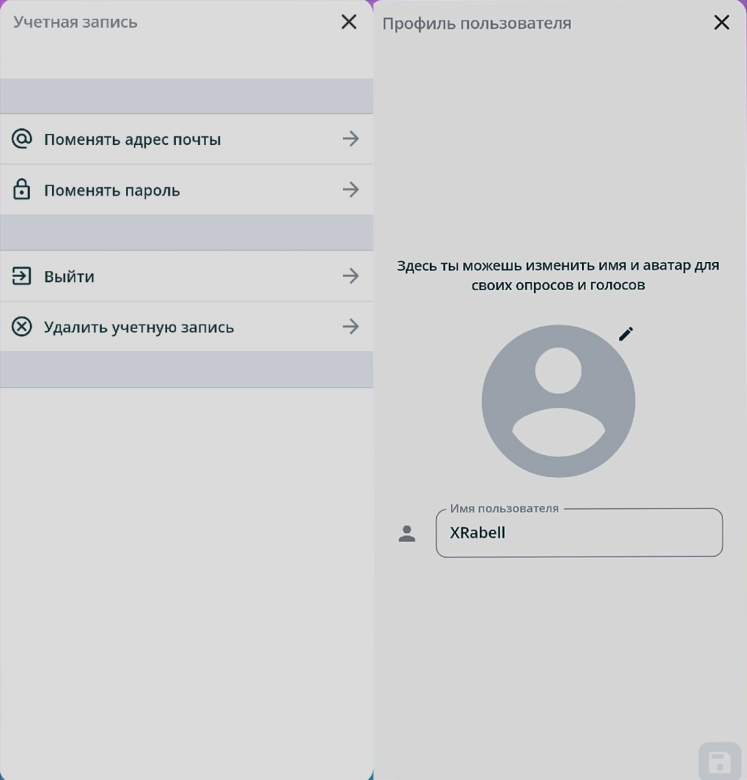


Рисунок 1.8 – Изменение профиля пользователя

Следующей отличительной особенностью приложения является возможность комментирования опросов, как представлено на рисунке 1.9.

Комментарии можно удалить и редактировать при желании.

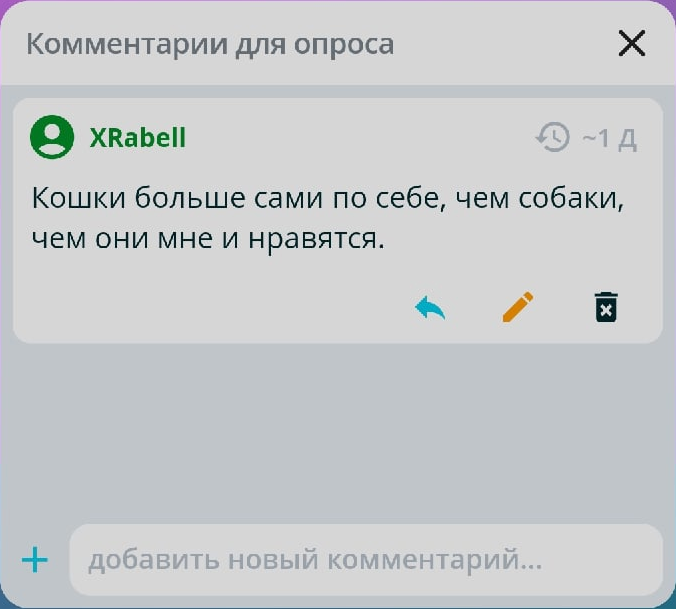


Рисунок 1.9 – Комментирование опроса

Еще одной особенностью является аналитика опроса, представленная на рисунке 1.10, где можно посмотреть диаграмму истории опроса за день, неделю и месяц, а также увидеть количество человек, проголосовавших за тот или иной вариант ответа в количественном или процентном соотношении. Цвета линий на диаграмме представляют собой разные ответы.

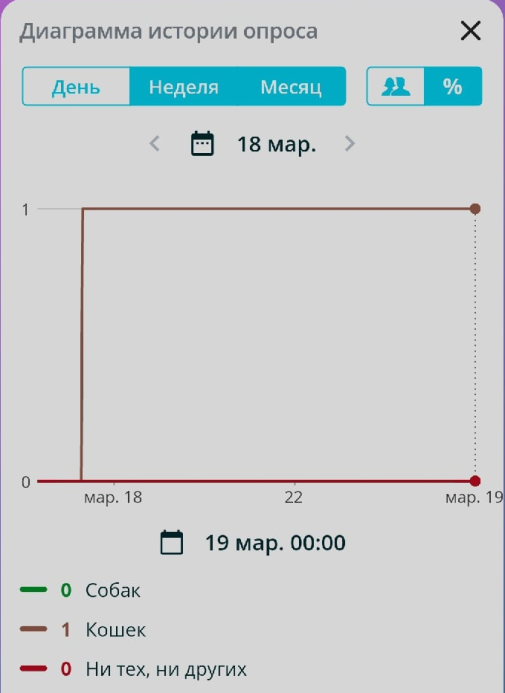


Рисунок 1.10 – Диаграмма истории опроса

В итоге были сделаны следующие выводы: при первом заходе в приложение предлагается пройти регистрацию по почте или же войти, если уже был создан профиль ранее; имеется возможность редактирования имени пользователя в профиле, установки картинки на аватар, смены почты и пароля по желанию; под опросами есть возможность оставлять комментарии; во вкладке «аналитика опроса» представлена диаграмма истории опроса по дате.

## 1.2 Патентный поиск

В данном подразделе представлены результаты патентного исследования. В результате проведенного патентного поиска был найден ряд схожих по тематике патентов, представленных в таблицах 1.1-1.2.

Таблица 1.1 – Описание патента №1

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| Способ тестирования и/или опроса удаленных пользователей | RU0002408925 | 10.01.2011 | Городин Максим Ефимович |

Изобретение относится к способам использования сетей передачи данных для проведения тестирования и/или опроса удаленных пользователей. Техническим результатом является повышение достоверности результатов проведения тестирования. Способ включает создание вопросной базы данных с правильными и неправильными вариантами ответов, автоматическую рассылку с Интернет-сервера компании сигнала удаленным пользователям о том, что они могут принять участие в тестировании и/или опросе. Также имеется получение удаленными пользователями с сервера задания через сети передачи данных, отправление удаленными пользователями результатов ответов, выбор победителей сервером компании и отправление удаленным пользователям сигналов с информацией об итогах тестирования и/или опросов. Перед тестированием и/или опросом проводят оценку пропускной способности канала связи удаленного пользователя. В случае негативных результатов, заключающихся в превышении времени отклика сервера или низкой пропускной способности канала, удаленный пользователь предупреждается о возможных коллизиях при получении результатов тестирования или опроса, а в момент подведения итогов производят поправку по времени, затраченному на прохождение тестирования и/или опроса [1].

Таблица 1.2 – Описание патента №2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Номер патента | Опубликовано | Авторы |
| Система анкетирования в инфокоммуникативных сетях в процессе маркетинговых исследований | U9892 | 28.02.2014 | Ландо Павел Александрович |

Полезная модель относится к маркетинговым исследованиям, может быть реализована с использованием сетевых технологий, связывающих предполагаемого потребителя и производителя нового товара, в отношении которого проводятся маркетинговые исследования. В частности, заявляемая система предназначена для формирования и доставки целевого блока, которая включает в себя анкету и рекламный контент (рекламный образ нового товара) с учетом предполагаемых потребительских предпочтений абонентов инфокоммуникативных сетей [2].

## 1.3 Постановка задачи

Целью дипломного проекта является мобильное приложение для создания и прохождения мини-опросов, основными возможностями которого являются оценивание опросов и комментариев к ним, поиск и сортировка.

Учитывая сделанные выводы в обзоре аналогов выше, мобильное приложение будет содержать следующие функциональные возможности:

* регистрация;
* авторизация;
* создание и удаление опроса;
* участие в опросах от других пользователей;
* оставление комментария под опросом;
* редактирование и удаление профиля пользователя;
* просмотр опросов, на которые был ранее выбран вариант ответа;
* поиск опросов с их сортировкой по темам;
* просмотр статистики по опросам;
* поддерживание ролей администратора и пользователя.

Вместе с этим дополнительными пунктами для функционала приложения будут система достижений, которая будет мотивировать пользователей больше взаимодействовать с приложением и использовать весь его функционал, а так же возможность оставлять «лайки» и «дизлайки».

## 1.4 Выводы по разделу

В результате написания первого раздела дипломного проекта был проведен обзор имеющихся на данный момент аналогов, а также произведен патентный поиск. После патентного поиска и обзора аналогов полного совпадения с темой дипломного проекта обнаружено не было. На основе обзора аналогов был составлен список требований для разрабатываемого мобильного приложения, а так же был выделен дополнительный функционал для добавления большей уникальности приложению в сравнении с другими. После проделанной работы можно преступать к проектированию архитектуры и к разработке дипломного проекта.

# 2 Проектирование приложения

ФИО

Подпись

Дата

Лист

1

*16*

*БГТУ 02.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

2 Проектирование приложения

Лит.

Листов

15

*74217096, 2023*

Перед началом разработки приложения и после составления списка требований и решения, какой именно функционал будет присутствовать в приложении, необходимо спроектировать архитектуру программного средства, схемы базы данных и перехода между фрагментами мобильного приложения, диаграмму вариантов использования, блок-схему алгоритма создания опроса, а так же смоделировать макеты экранов мобильного приложения.

## 2.1 Архитектура приложения

Архитектура приложения – это логическая структура, описывающая отдельные компоненты, их свойства и связи в виде единой системы.

Для реализации программного средства была выбрана клиент-серверная архитектура, позволяющая разделить вычисления между сервером и приложением.

В соответствии с этим архитектуру приложения можно разделить на три отдельных компонента, которые возможно разрабатывать отдельно друг от друга:

* клиентская часть, которая является мобильным приложением, с интерфейсом которого будут взаимодействовать пользователи;
* серверная часть, которая обрабатывает запросы пользователей из мобильного приложения и отвечает на них;
* база данных, которая хранит данные и с которой взаимодействует серверная часть для получения и отправления данных на клиентскую часть для последующего их отображения пользователю.

Диаграмма архитектуры приложения представлена на рисунке 2.1, а так же непосредственно приведена в приложении А.

У подсистемы управления, являющейся сервером, были выделены модули регистрации и авторизации, а так же модуль навигации, с помощью которого происходит взаимодействие со следующими модулями по управлению: пользователями, аккаунтами, опросами, рейтингами, комментариями, достижениями и ролями. Помимо этого были выделены так же следующие модули, не связанные с другими: преобразования данных для ответа на запрос, загрузки и выгрузки графического материала и взаимодействия с почтой.

У подсистемы отображения, являющейся мобильным приложением, были выделены модули для взаимодействия с сервером, упаковки данных для отправления их на сервер, а так же модули для навигации по приложению, отображения данных и для управления локальной базой данных.

Так же были выделены две подсистемы хранения и обработки информации, представляющие собой две базы данных: удаленную и локальную.

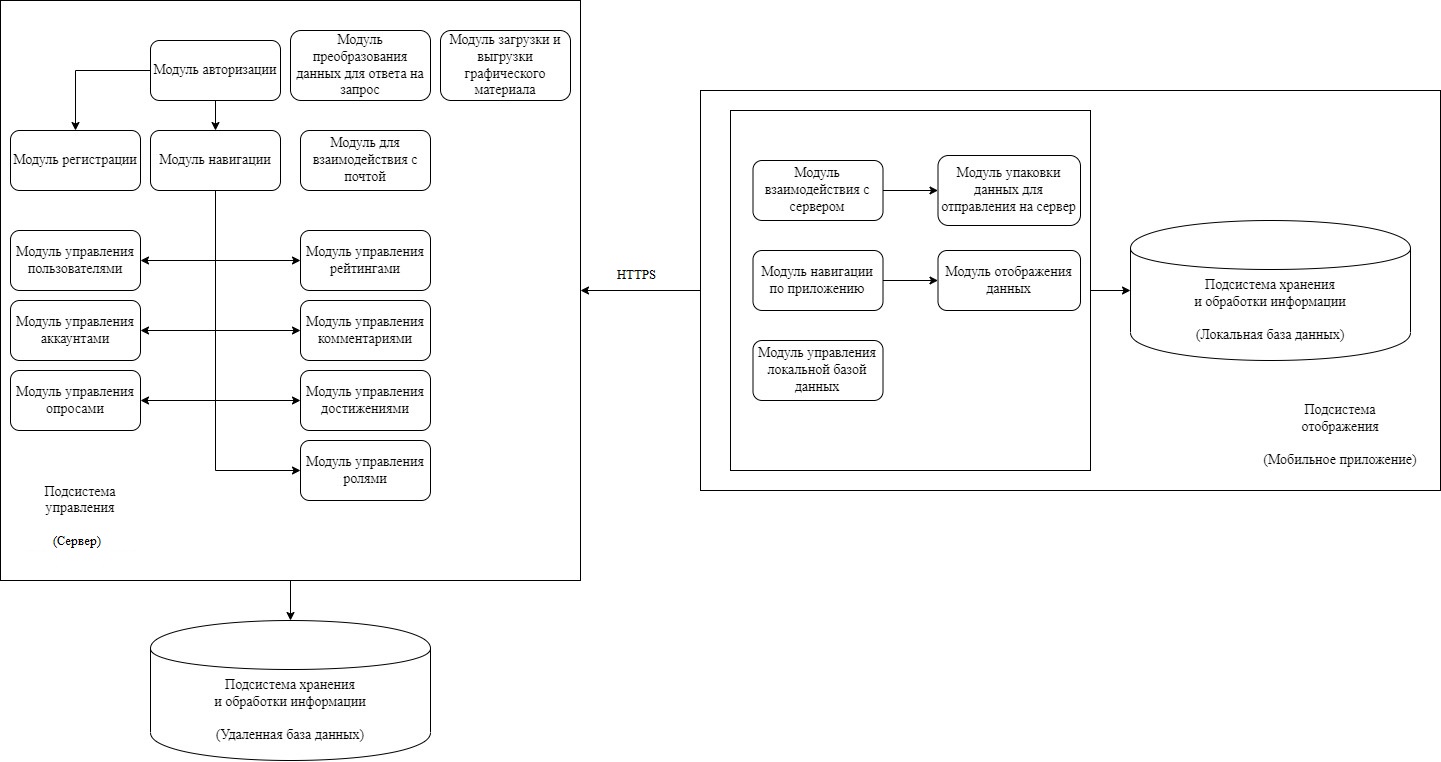


Рисунок 2.1 – Диаграмма архитектуры приложения

Взаимодействие между сервером и мобильным приложением происходит по безопасному протоколу HTTPS.

### 2.1.1 Диаграмма развертывания приложения

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит распределение компонентов по отдельным узлам системы. Диаграмма показывает наличие физических соединений-маршрутов передачи информации между аппаратными устройствами, которые будут задействованы в реализации системы.

Диаграмма развертывания проектируемого приложения представлена на рисунке 2.2, а так же в приложении Б.

Сервер планируется разрабатывать в среде выполнения Node.Js с использованием языка программирования JavaScript. Это позволит легко развернуть сервер на облачной платформе Vercel для возможности получения запросов от мобильного приложения через интернет с последующим отправлением ответов на них. Обмен данными проще и удобнее осуществлять с помощью текстового формата JavaScript Object Notation (JSON), так как для написания сервера будет использоваться язык программирования JavaScript.

Использование Node.Js так же предоставляет возможность воспользоваться менеджером пакетов NPM для скачивания и установки вспомогательных фреймворков и модулей, которые рассмотрены далее.

Для создания API в архитектурном стиле REST на стороне сервера будет использоваться фреймворк Express. Он облегчит создание маршрутизации, привязки к маршрутам обработчиков запросов и к дополнительным функциям, позволяющим выполнять некоторые действия до обработчиков, такие как, например, аутентификация и проверка роли пользователя.



Рисунок 2.2 – Диаграмма развертывания

В качестве инструмента для работы с удаленной базой данных планируется использовать фреймворк Sequlize. Он позволит работать с ней посредством реализации моделей таблиц и вызова их соответствующих методов для создания запросов к базе данных, а так же он поддерживает транзакции, которые позволят сохранить целостность и непротиворечивость данных. Вместе с этим появляется возможность не писать «сырые» запросы на языке SQL напрямую, что кроме того дополнительно дает защиту от SQL-инъекций.

Для реализации аутентификации пользователя будет использоваться открытый стандарт JSON Web Tokens (JWT), основанный на формате JSON. Токены будут создаваться сервером, подписываться секретным ключом и передаваться клиенту, который в дальнейшем будет использовать данный токен для подтверждения подлинности аккаунта при отправке запросов.

С помощью модуля Nodemailer будет осуществляться отправление на почту подтверждающей ссылки для активации личного профиля пользователя.

Мобильное приложение планируется разрабатывать для платформы Android, версия которой будет от 8.0 и выше. Это позволит приложению работать на большом количестве мобильных устройств, процентное количество которых равно 91.84 [3]. Для разработки мобильного приложения будет использоваться язык программирования Java, так как он является официальным для Android.

Поскольку на стороне сервера для обмена данными был выбран формат JSON, то в качестве библиотеки для получения и загрузки данных была выбрана Retrofit2. Она упростит взаимодействие с сервером через REST API, а так же позволит использовать асинхронный режим при отправлении запросов к серверу и получении ответов на них, что исключит возможность провисаний в интерфейсе мобильного приложения при ожидании ответа.

В мобильном приложении будет использоваться база данных SQLite, которая поставляется Android со встроенной реализацией. Для работы с ней была выбрана библиотека ROOM. С ее помощью можно будет создать API-интерфейс поверх встроенного SQLite API для работы с данными. Мобильная база данных будет содержать информацию, которая будет относиться только к конкретному пользователю, что поможет сократить количество запросов к серверу и таким образом улучшить его производительность.

В качестве системы управления базами данных (СУБД) была выбрана PostgreSQL, размещенная на облачном сервисе ElephantSQL. В ней будет размещена удаленная база данных, с которой будет взаимодействовать сервер.

## 2.2 Проектирование серверной части приложения

### 2.2.1 Выбор архитектурного паттерна проектирования

Поскольку при проектировании серверной части приложения была выбрана среда Node.Js, то архитектурным паттерном был выбран Model-View-Controller (MVC), так как он является наиболее используемым при написании сервера, а фреймворк Express позволяет легко разбивать серверные компоненты на слабосвязанные модули, над которыми можно работать раздельно [4].

Паттерн MVC подразумевает под собой разделение данных приложения и управляющей логики на три компонента: модель, представление и контроллер.

Модель будет осуществлять собой интерфейс для взаимодействия с удаленной базой данных с использованием выбранного нами языка программирования, а так же определять структуру и логику используемых данных.

Представление предназначено для показа данных модели в понятном для пользователя виде. Так как пользователем будет являться мобильное приложение, то представление будет структурировано на основе шаблонов. Эти шаблоны в конечном итоге будут заполнены результатами логики контроллера, взаимодействующего с моделью, и после предоставлены конечному пользователю.

Контроллер будет являться компонентом, взаимодействующим с моделью и представлением. С его помощью будут обрабатываться входящие запросы от клиента, с использованием для обработки модели и представления, и отправляться в ответ результаты обработки в формате JSON.

Стоит подметить, что отдельным компонентом, который не часто упоминается в выбранном паттерне, но практически всегда используется, является система маршрутизации, которая будет сопоставлять запросы клиента с маршрутами и выбирать для обработки запросов определенный контроллер.

Схема паттерна изображена на рисунке 2.3.

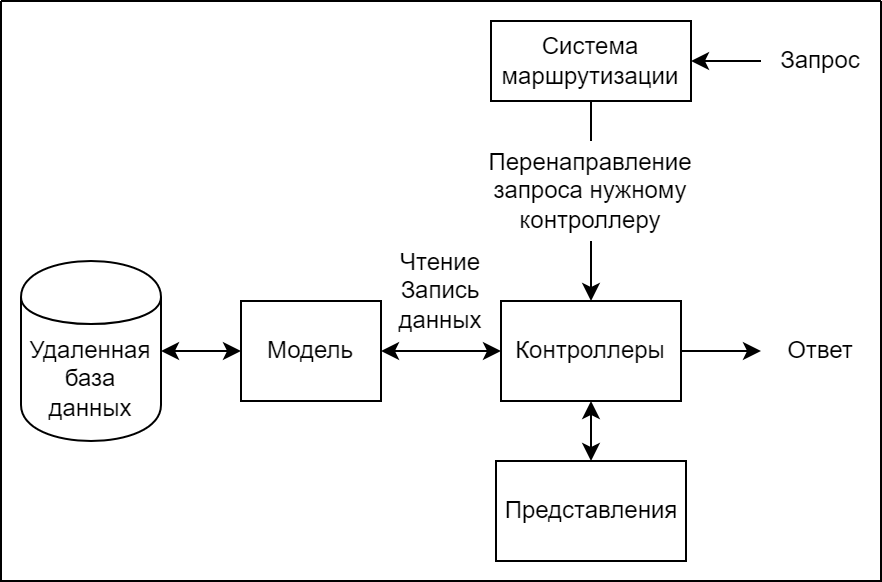


Рисунок 2.3 – Архитектурный паттерн MVC

В общем случае, когда к серверу будет осуществляться запрос, система маршрутизации будет выбирать нужный контроллер для обработки запроса. В процессе обработки он может обращаться к данным через модели и для структуризации ответа использовать представления. Соответствующий результат обработки контроллера будет отправляться в ответ клиенту.

### 2.2.2 Проектирование удаленной базы данных

База данных является упорядоченным набором структурированной информации или данных по определенной предметной области, которые обычно хранятся в электронном виде в компьютерной системе.

Для лучшего понимания структуры разрабатываемой удаленной базы данных была спроектирована ее логическая схема с необходимыми ограничениями целостности, представленная на рисунке 2.4 и в приложении В.

Чтобы реализовать весь функционал приложения, необходимо будет создать четырнадцать таблиц, три из которых существуют для создания связи «многие-ко-многим». Данными таблицами являются:

* «User\_roles»;
* «User\_achievements»;
* «User\_answers».

Они будут по большей части содержать в себе в качестве внешних ключей идентификаторы записей из соответствующих таблиц, указанных в их названиях, а так же собственные идентификаторы записей.



Рисунок 2.4 – Схема удаленной базы данных

Список всех таблиц спроектированной базы данных «Questionnaire» с их описанием представлен ниже:

* «Users» является таблицей, от которой создается больше всего связей и которая содержит данные пользователя для регистрации и авторизации;
* «User\_roles» хранит идентификаторы пользователя и его роли;
* «Roles» отвечает за хранение данных о роли;
* «Accounts» отвечает за хранение данных о профиле, таких как ник, статус, ссылка на картинку для профиля и идентификатор конкретного пользователя, которому непосредственно принадлежит профиль;
* «Tokens» отвечает за хранение токена обновления для токена доступа, который используется для аутентификации пользователя по JWT;
* «Achievements» отвечает за хранение достижений;
* «User\_achievements» отвечает за хранение идентификаторов пользователя и достижения, а так же времени добавления записи в таблицу. Так реализовано хранение полученных пользователем достижений;
* «Questions» отвечает за хранение вопроса опроса, ссылки на картинку на сервере, дополнительного описания, даты создания, рейтинга, а так же идентификатора категории и пользователя, создавшего опрос;
* «Categories» хранит категории, по которым может быть создан опрос;
* «Answers» отвечает за хранение вариантов ответа на опрос;
* «User\_answers» отвечает за хранение идентификаторов пользователя и варианта ответа на опрос, который был им выбран, а так же даты выбора ответа на вопрос для будущей возможности составления статистики по опросу;
* «Rating\_questions» отвечает за хранение оценок, которые были поставлены пользователями опросу, а так же идентификаторов опроса и пользователя;
* «Comments» отвечает за хранение комментариев к опросу, содержит такие данные как время создания опроса, текст комментария, идентификаторы пользователя, создавшего опрос, рейтинг и идентификатор опроса;
* «Rating\_comments» отвечает за хранение оценок, которые были поставлены пользователями комментарию под опросом.

В локальной базе данных в мобильном приложении, которая будет рассмотрена далее в другом подразделе, будут содержаться данные, относящиеся только к конкретному зарегистрированному пользователю.

## 2.3 Проектирование клиентской части приложения

### 2.3.1 Выбор архитектурного паттерна проектирования

Для мобильного приложения был выбран архитектурный паттерн Model-View-ViewModel (MVVM), так как он позволит легко работать с привязками данных к интерфейсу в мобильном приложении [5].

Паттерн MVVM подразумевает под собой разделение данных приложения и управляющей логики на три отдельных компонента:

* модель описывает используемые в приложении данные;
* представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением;
* модель представления представляет собой промежуточный слой между представлением и данными, который обеспечивает их взаимодействие.

Разбиение на отдельные компоненты так же позволит сделать их менее зависимыми друг от друга, что так же облегчит их разработку.

### 2.3.2 Проектирование локальной базы данных

Мобильное приложение будет содержать локальную базу данных, которая будет хранить данные о конкретном пользователе, а так же данные, которые очень редко обновляются, такие как описание достижений.

Схема локальной базы данных представлена на рисунке 2.5.

Локальная база данных спроектирована так, что состоит из четырех таблиц, одна из которых не связана с другими, а именно таблица «Categories». Она будет хранить в себе все возможные категории опросов.

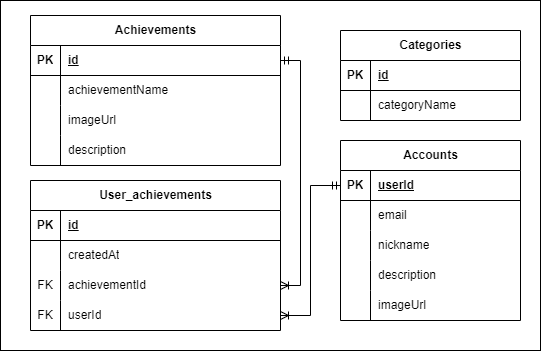


Рисунок 2.5 – Схема локальной базы данных

Список остальных связанных таблиц спроектированной локальной базы данных с их описанием представлен ниже:

* «Accounts» отвечает за хранение данных о профиле пользователя;
* «Achievements» отвечает за хранение всех достижений;
* «User\_achievements» отвечает за хранение идентификаторов пользователя и достижения, а так же времени добавления записи в таблицу.

### 2.3.3 Диаграмма вариантов использования

Диаграмма вариантов использования или же диаграмма прецедентов – это наиболее общее представление функционального назначения системы. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с актером.

На диаграмме использования применяются два типа основных сущностей: варианты использования и действующие лица (актеры), между которыми устанавливаются основные типы отношений. Данными типами отношений являются: ассоциация, обобщение, включение, расширение. Каждый тип отношения обозначается своим видом стрелки между сущностями на диаграмме.

Создание диаграммы вариантов использования имеет следующие цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.

В ходе разработки мобильного приложения будет реализовано два типа пользователей: обычный пользователь и администратор, имеющий расширенный функционал. Администратор помимо дополнительных возможностей может тоже использовать функционал, доступный обычным пользователям.

Функциональные возможности приложения представлены на диаграмме вариантов использования на рисунке 2.4 и в приложении Г.

Как можно увидеть из диаграммы ниже, пользователь сможет взаимодействовать со своим личным профилем, где сможет изменить данные о себе, которыми являются псевдоним пользователя, статус и картинка профиля, а так же сможет посмотреть список полученных им достижений или же удалить свой профиль. Так же он сможет создать опрос по какой-либо конкретной категории, добавив к нему по желанию дополнительное описание и картинку. Обязательным в создании опроса будет являться составление вопроса и добавление вариантов ответа на него, а так же выбор категории из выпадающего списка. В мобильном приложении можно будет добавить минимум два варианта ответа на вопрос. Максимумом являться будет пять вариантов ответов. После этого он может редактировать собственный опрос и просматривать статистику по нему или же удалить его по желанию.

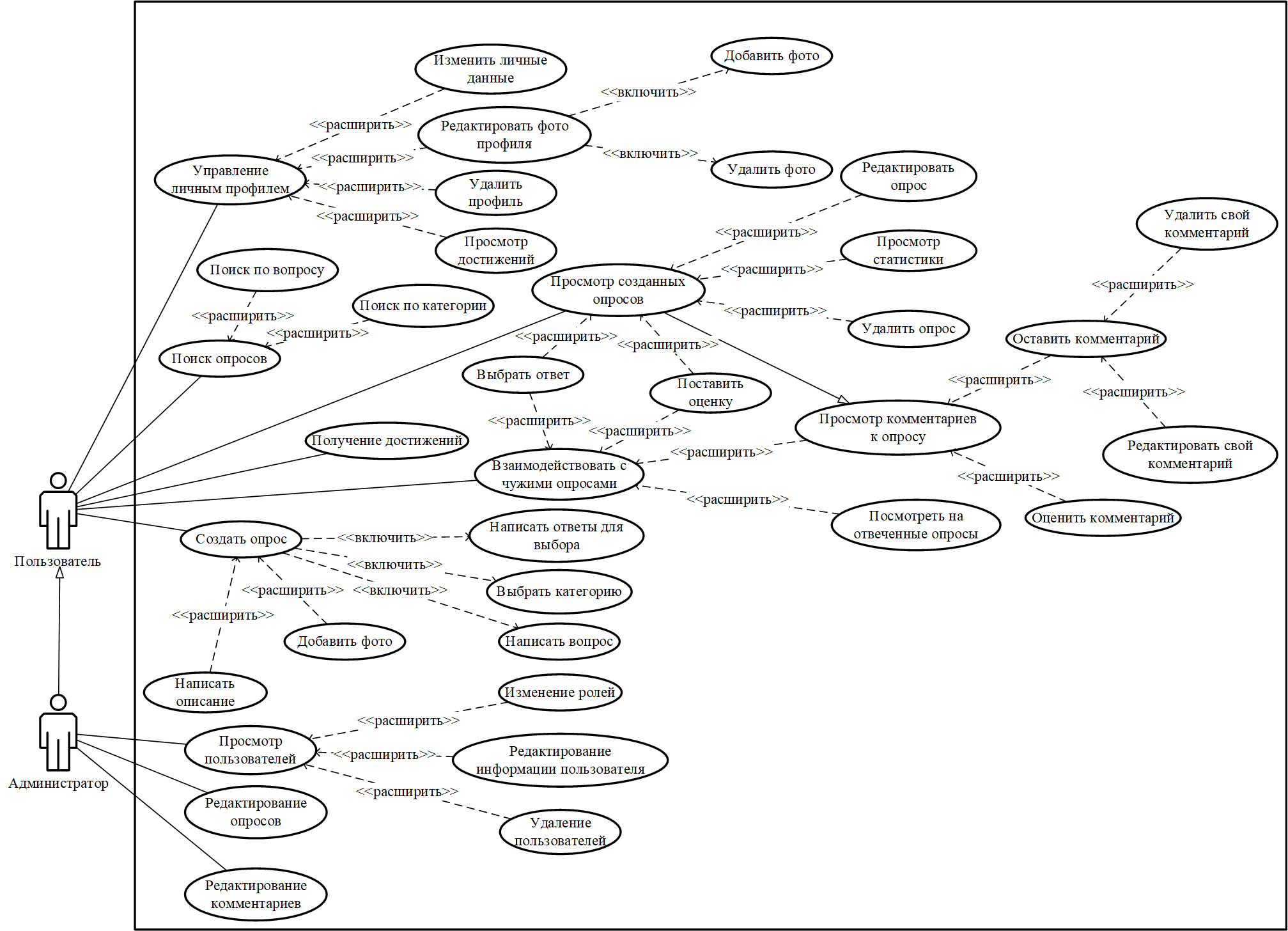


Рисунок 2.4 – Диаграмма вариантов использования

Помимо этого зарегистрированный пользователь сможет взаимодействовать с чужими опросами, совершив их поиск по категории или вопросу, если интересует нечто конкретное. Во взаимодействие со своими и чужими опросами так же входит возможность комментирования, голосования и оценивания опросов и комментариев к ним. Свои комментарии можно будет редактировать или удалять. Будет возможность посмотреть, в каких опросах ранее голосовал пользователь.

Следуя из представленной диаграммы вариантов использования, администратор имеет те же функциональные возможности, какие имеет и обычный пользователь. Однако помимо этого он сможет просматривать списки пользователей и изменять их роли, редактировать информацию о них или же удалять их. Кроме того, он сможет редактировать любые опросы и комментарии.

### 2.3.4 Схема переходов между фрагментами

На данном этапе в виде так называемой схемы переходов между фрагментами отображается концепция возможных сценариев поведения пользователя, показывается связь и взаимодействие между фрагментами. Ключевой задачей является обеспечение интуитивного доступа к искомому содержимому программного средства. Так же это способствует при адекватном решении повышению комфорта и эффективности использования продукта.

Спроектированная схема переходов между фрагментами мобильного приложения представлена на рисунке 2.5 и в приложении Д.

Как видно из схемы, после фрагмента с регистраций и авторизацией следует главный фрагмент со списком всех опросов. С него можно будет перейти на такие фрагменты как «личный профиль», «все опросы», «личные опросы», «отвеченные опросы», «создать опрос». От личного профиля будет осуществляться переход на фрагмент для просмотра достижений, а так же на фрагмент взаимодействия с учетной записью зарегистрированного пользователя.



Рисунок 2.5 – Схема переходов между фрагментами мобильного приложения

С фрагментов, на которых будут определенные списки опросов («все опросы», «личные опросы», «отвеченные опросы») можно будет перейти к фрагменту «выбранный опрос», на котором будет дополнительно отображена информация о конкретном опросе, а так же появится возможность взаимодействовать с ним. С данного фрагмента можно будет перейти на фрагмент с комментариями к опросу и на фрагмент со статистикой опроса, которую будет видеть только создатель.

Фрагмент «список пользователей» будет доступен только для администратора. С его помощью можно будет совершать поиск пользователей, переходить на фрагмент «выбранный пользователь», после чего совершать особые действия к данному пользователю, такие как редактирование и удаление профиля.

### 2.3.5 Проектирование макетов экранов

При анализе продуктов-аналогов и количества их скачиваний было выявлено, что дизайн с закругленными большими элементами, а также с пастельным градиентом привлекает больше пользователей. Помимо этого в дизайне приложения являются важными простота и лаконичность.

Важно уделить особое внимание цветовой палитре, поскольку тщательно подобранные цвета могут улучшить дизайн и, напротив, плохая цветовая палитра может ухудшить впечатление пользователей. В качестве основных цветов по правилу 60-30-10 были выбраны зеленый, желтый и синий соответственно. Правило означает, что 60% пространства должно быть занято преобладающим цветом, 30% – вторичным, а оставшиеся 10% – акцентом [6].

Далее в подразделе представлены главные макеты экранов мобильного приложения с их основными элементами интерфейса.

Макет экрана опросов представлен на рисунке 2.6.



Рисунок 2.6 – Макет экрана со списком опросов

Данный макет будет повторяться на трех экранах: «отвеченные опросы», «все опросы», «мои опросы» в виду однообразия действий на них. Так же стоит отметить наличие вертикального меню слева, которое можно будет открыть, нажав на кнопку в верхнем правом углу, или жестом «вытягивания» из-за края экрана, что удобнее. На макете так же имеются элементы для поиска опросов.

Большую часть экрана будет занимать список с опросами. Каждый элемент списка будет содержать в себе картинку к опросу, вопрос, описание, рейтинг, дату создания и категорию. В зависимости от количества символов в вопросе и описании, элемент списка может быть меньше, чем на макете.

Макет профиля пользователя представлен на рисунке 2.7.



Рисунок 2.7 – Макет экрана профиля пользователя

На данном экране будет отображена информация о пользователе, которую он сможет менять, а так же просматривать свои полученные достижения. Помимо этого он сможет выйти с профиля или удалить его.

Макет экрана создания опроса представлен на рисунке 2.8.

На данном экране будут размещены все элементы для создания опроса, а именно: добавление картинки, добавление вопроса, добавление описание, а так же пять полей для вариантов ответов, два из которых являются обязательными.

Стоит отметить, что экран для редактирования опроса будет иметь ту же структуру, однако отредактировать будет возможно только картинку и описание опроса, а так же добавить дополнительный вариант ответа. Сделано это для того, чтобы не сбивать с толка пользователей, которые вернувшись на данный опрос для чтения комментариев или из интереса, могут в таком случае обнаружить, что проголосовали за иной вариант ответа или же что участвовали в неизвестном для них опросе. Так же данное ограничение будет существовать для того, чтобы создатель опроса не смог фальсифицировать выбор ответа других пользователей.



Рисунок 2.8 – Макет экрана создания опроса

Макет экрана с выбранным опросом представлен на рисунке 2.9.

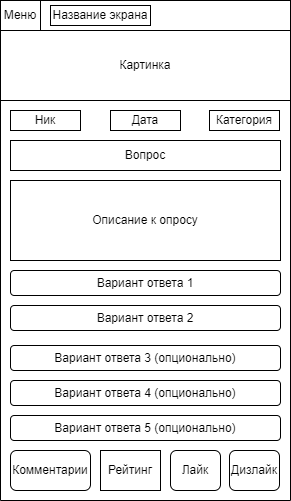


Рисунок 2.9 – Макет экрана с выбранным опросом

На данном экране будет отображаться развернутая информация о выбранном опросе, а так же варианты ответов на него с возможностью после голосования увидеть количество проголосовавших. Так же с этого экрана будет возможно перейти на экран с комментариями к опросу, макет которого представлен на рисунке 2.10.

На представленном экране будут отображаться комментарии пользователей с дополнительной информацией о количестве согласных и несогласных с их комментарием посредством рейтинга, а так же о варианте ответа, который они выбрали. Помимо этого будет выведена дата их комментария и их ник, с возможностью перейти по нему на их профиль для просмотра информации.



Рисунок 2.10 – Макет экрана с комментариями

Помимо рассмотренных экранов в виде макетов будут иметься и другие экраны, однако одна их часть будет предназначена для роли администратора и дизайн не столь важен, а другие не имеют большого количества важных элементов в себе для создания отдельного макета под каждый из них.

### 2.3.6 Блок-схема алгоритма получения достижений за комментарии

Блок-схема – распространённый тип схем, описывающих алгоритмы или процессы, изображающие отдельные шаги в виде блоков различной формы, соединённых между собой линиями, указывающими направление последовательности. Рисунок 2.11 отображает блок-схему алгоритма получения достижений за количество комментариев, так же данная блок-схема представлена в приложении Е.

Алгоритм начинается с проверки доступности сервера, после чего, если тот доступен, на него отправляется запрос на публикацию с данными о комментарии. Клиент от сервера получает ответ, из которого узнает, опубликован комментарий или произошла ошибка. Если произошла ошибка, то выводится сообщение об этом пользователю. При отсутствии ошибки клиент дополнительно узнает, получено достижение или нет. При положительном ответе и при имении достижений в локальной базе данных, он из нее получает дату последнего достижения и отправляет запрос на сервер о получении достижений для конкретного пользователя, которые были получены позже отправленной даты. Полученные достижения добавляются в локальную базу данных в мобильном приложении.

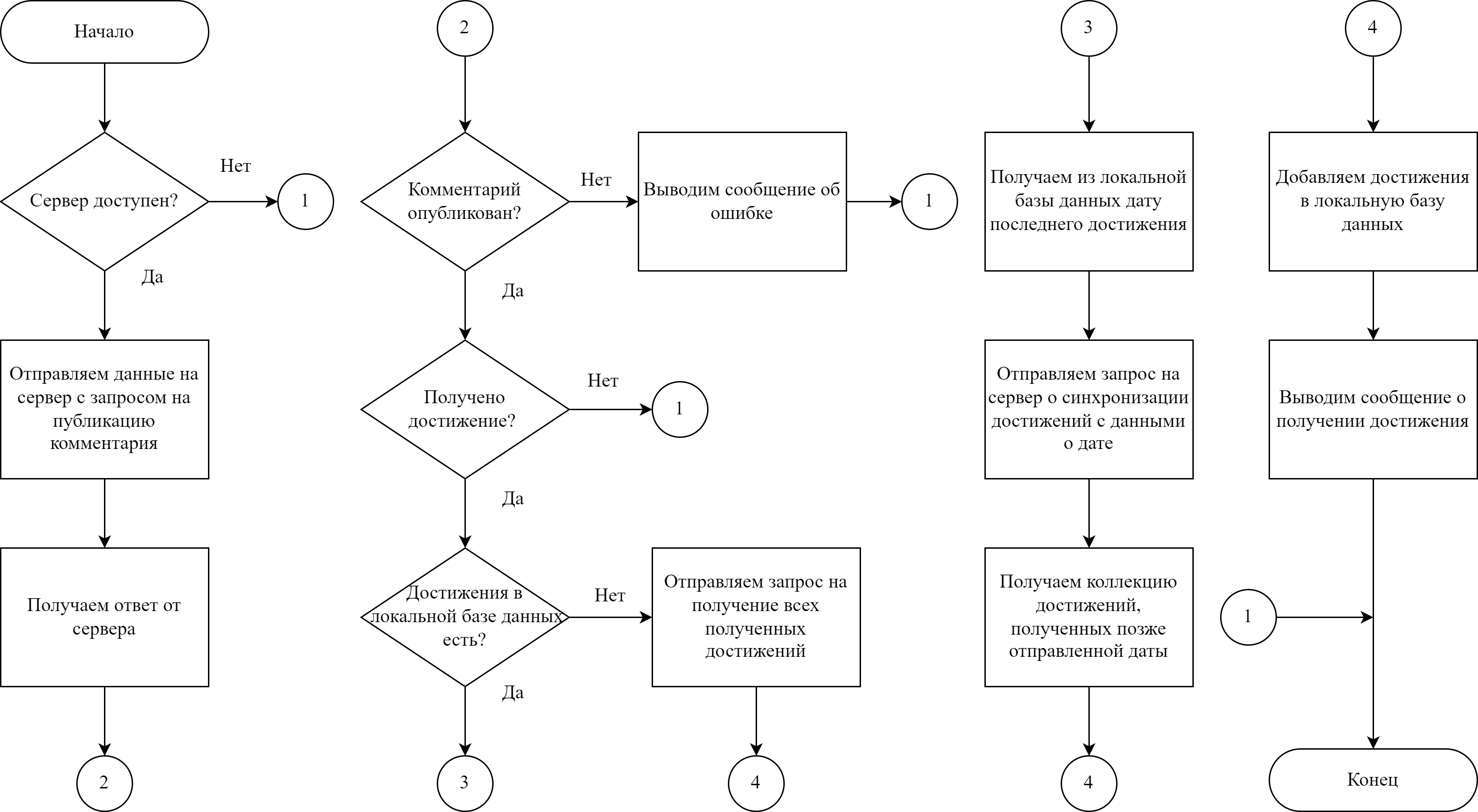


Рисунок 2.11 – Блок-схема алгоритма получения достижений за комментарии

Если достижений в локальной базе данных нет, то клиент посылает запрос на получение всех достижений конкретного пользователя, после чего добавляет их в базу данных на мобильном приложении.

## 2.4 Выводы по разделу

В результате написания второго раздела дипломного проекта для проектирования приложения была выбрана клиент-серверная архитектура. На основе этого была создана диаграмма архитектуры приложения, имеющая в себе такие компоненты как сервер, база данных и мобильное приложение, описаны их модули. Так же была разработана и описана диаграмма развертывания приложения.

Были спроектированы серверная часть и клиентская часть программного средства, выбраны для каждой из них архитектурный паттерн, созданы схема удаленной и локальной базы данных соответственно. Проектирование баз данных осуществлялось при помощи построения модели данных, определения таблиц и связей между ними. Описана каждая таблица.

Помимо этого для клиентской части были разработаны основные макеты интерфейса мобильного приложения, схема переходов между фрагментами, а так же спроектирована блок-схема получения достижений за оставленные пользователем комментарии под опросами.

# 3 Реализация приложения

ФИО

Подпись

Дата

Лист

1

*31*

*БГТУ 03.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

3 Реализация приложения

Лит.

Листов

16

*74217096, 2023*

Разработку программного обеспечения данного дипломного проекта можно условно разделить на две основные части: серверную и клиентскую. В качестве системы контроля версиями использовался сервис GitHub.

## 3.1 Реализация серверной части приложения

### 3.1.1 Файловая структура проекта

Перед началом разработки приложения была продумана файловая структура проекта в соответствии с выбранной архитектурой.

Файловая структура проекта представлена на рисунке 3.1.



Рисунок 3.1 – Файловая структура приложения

На рисунке отображены основные файлы и каталоги.

Проект имеет следующие основные файлы:

* index.js – место, с которого начинается приложение. Оно является точкой

входа в приложение и содержит в себе подключения всех модулей и их запуск;

* .env предназначен для хранения переменных окружения;
* db.js предназначен для подключения к удаленной базе данных;
* vercel.js содержит данные для развертывания приложения на Vercel.

Проект так же имеет следующие основные каталоги:

* routes содержит файлы с маршрутизацией, так же в них прописана связь между маршрутом и контроллером, обрабатывающим его;
* controllers содержит файлы с контроллерами, которые обрабатывают запросы, поступающие со стороны клиента;
* models содержит файлы с объектами, на основе которых создаются таблицы в удаленной базе данных, а так же происходит взаимодействие с ними посредством методов данных объектов;
* middleware содержит файлы с функциями, которые совершают какие-либо действия над запросом до его обработки соответствующим контроллером;
* error содержит файл ApiError.js, в котором находится класс ApiError для обработки ошибок на сервере, наследующий класс Error;
* services содержит файлы для работы с электронной почтой и для реализации авторизации и регистрации с помощью JWT-токенов;
* static содержит загружаемые пользователями картинки;
* dtos содержит шаблоны объектов, в которые преобразовываются полученные данные из базы данных для их отправления клиенту.

Остальные нерассмотренные каталоги и файлы являются автоматически сгенерированными и предназначены для работы модулей и GitHub.

### 3.1.2 Реализация архитектурного паттерна MVC

Как было описано выше, паттерн MVC подразумевает разделение программы на несколько слабо связанных между собой компонентов.

Компонент «модель» в данном проекте представляет собой объекты для взаимодействия с базой данных. Реализация одного из объектов, в данном случае «Question», представлена на листинге 3.1. Все объекты представлены в листинге программного кода в приложении Ж.

Когда сервер запускается, то с помощью автоматически сгенерированных sequlize и отправленных запросов в базу данных будут создаваться таблицы и связи между ними на основе описанных моделей, если таблицы не существуют.

|  |
| --- |
| const Question = sequelize.define('question',  {  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement:true},  question: {type: DataTypes.STRING, allowNull: false},  description: {type: DataTypes.STRING},  imageUrl: {type: DataTypes.STRING},  rating: {type: DataTypes.INTEGER, defaultValue: 0},  }) |

Листинг 3.1 – Объект «Question»

При использовании функции define фреймворка sequlize, в первую очередь указывается название будущей таблицы в базе данных, после чего перечисляются ее поля. В фигурных скобках напротив полей обязательно указываются их типы в соответствующих атрибутах «type», при этом используются значения объектов DataTypes. Так же могут быть использованы дополнительные атрибуты, такие как «primaryKey», «autoIncrement», «defaultValue», «unique», «allowNull» и другие, которые соответствуют схожим по названию ограничениям целостности в используемой базе данных на облачном сервисе.

Для создания трех стандартных связей между таблицами используются комбинации четырех функций созданных ранее объектов: «hasOne», «hasMany», «belongsTo» и «belongsToMany» [7].

Реализация всех связей с объектом «Question» представлена в листинге 3.2.

|  |
| --- |
| …//связи с другими объектами  Question.hasMany(Comment,  {onDelete: 'CASCADE'})  Comment.belongsTo(Question)  Question.hasMany(Rating\_question,  {onDelete: 'CASCADE'})  Rating\_question.belongsTo(Question)  Question.hasMany(Answer,  {onDelete: 'CASCADE'})  Answer.belongsTo(Question)  User.hasMany(Question,  {onDelete: 'SET NULL'})  Question.belongsTo(User)  Category.hasMany(Question,  {onDelete: 'RESTRICT'})  Question.belongsTo(Category)  …//связи с другими объектами |

Листинг 3.2 – Связи объекта «Question»

При создании связей так же указываются действия, которые должны происходить при удалении записи, на которую ссылается внешний ключ, такие как: установить в поле значение «null», удалить все связанные записи, не позволить удалять записи и установить значение по умолчанию.

Контроллеры представляют собой классы, которые подключаются как стандартные модули, и затем асинхронные функции контроллеров используются для обработки маршрутов [8].

Часть кода контроллера для обработки запросов, связанных с комментариями, представлена в листинге 3.3.

В представленном далее контроллере в функции «create» из тела запроса инициируются переменные. После этого выполняется запрос в базу данных с помощью модели «Comment» и ее функции «create», в которой указаны необходимые переменные для создания записи в базе данных. Результат выполнения запроса конвертируется в формат JSON и отправляется клиенту. В функции «update» таким же образом происходит запрос в базу данных на обновление данных, где в соответствующей функции модели указывается новое значение и id обновляемой записи.

Результатом выполнения данного запроса является значение «1», если запрос прошел успешно и «0», если нет.

|  |
| --- |
| class CommentController  {  async create(req, res, next) {  try {  let {userId, questionId, text} = req.body  const comment = await Comment.create({userId, questionId, text})  return res.json(comment)  }  catch (e) {  next(ApiError.badRequest(e.message))  }  }  async update(req, res, next) {  try {  let {id} = req.params  let {text} = req.body  const comment = await Comment.update({text}, {  where: {id}  })  return res.json(comment)  }  catch (e) {  next(ApiError.badRequest(e.message))  }  }  …//другие методы  } |

Листинг 3.3 – Контроллер для обработки комментариев

Для обработки ошибок и предотвращения экстренного завершения работы сервера во всех функциях используется конструкция «try-catch». В блоке «catch» используется общий класс для обработки ошибок ApiError. Реализация класса ApiError представлена в приложении И.

Большие и сложные объекты, возвращаемые из базы данных на сервер при запросах с включением значений других связанных внешними ключами моделей, инкапсулируются в соответствующие более упрощенные и легкие объекты для их последующей передачи клиенту. Данные объекты являются Data Transfer Object (DTO) и не содержат ничего, кроме конструктора и параметров, содержащих данные, нужные для передачи клиентской части.

Реализация DTO для полученных данных, являющихся элементами передаваемого списка опросов клиенту, представлена в листинге 3.4.

В данном случае внутрь конструктора передается объект класса Question, после чего необходимыми данными инициализируются переменные класса ListQuestionDto. Таким образом клиент не получит опциональные данные, присутствующие в объектах, предоставляемых базой данных в ответ на некоторые сложные запросы, и уменьшится размер пересылаемых данных.

|  |
| --- |
| Module.exports = class ListQuestionDto  {  id;  question;  description;  imageUrl;  rating;  createdAt;  category;  constructor(model)  {  this.id = model.id;  this.question = model.question;  this.description = model.description;  this.imageUrl = model.imageUrl;  this.rating = model.rating;  this.createdAt = model.createdAt;  this.category = model.category.categoryName;  }  } |

Листинг 3.4 – DTO элемента списка опросов

Маршрутизация реализована с использованием модуля Router. С его помощью происходит привязка путей к обрабатывающим их функциям специальных контроллеров. Так же возможно простое разделение маршрутов по отдельным файлам и подключением их к Router. По необходимости перед обработчиками маршрутов располагаются дополнительные функции, предшествующие по выполнению перед обработчиками. С их помощью можно провести дополнительные проверки по поводу правильности данных из запроса клиента. После успешной проверки запрос обрабатывается соответствующим маршруту контроллером, а после неудачной – прекращается выполнение дальнейших функций в цепочке и выдается ошибка. Файл со всей маршрутизацией представлен в листинге 3.5.

|  |
| --- |
| …//подключение используемых модулей  router.use('/user', userRouter)  router.use('/category', categoryRouter)  router.use('/question', questionRouter)  router.use('/rating', ratingRouter)  router.use('/comment', commentRouter)  router.use('/achievement', achievementRouter)  router.use('/account', accountRouter) |

Листинг 3.5 – Содержимое файла с маршрутизацией

В данном файле происходит привязка дочерних роутеров к главному роутеру. В дочерних роутерах располагаются маршруты с привязанными к ним контроллерами, соответствующие их названию. Такое разделение кода позволяет легко подключать новые маршруты и разрабатывать их отдельно от остальных.

Реализация дочернего роутера для выполнения запросов по поводу действий с комментариями представлена в листинге 3.6.

|  |
| --- |
| …//подключение используемых модулей  router.post('/', authMiddleware, commentController.create)  router.put('/:id', authMiddleware, commentController.update  router.get('/', authMiddleware, commentController.getAll)  router.get('/:questionId',authMiddleware, commentController.getAllByQuestion) |

Листинг 3.6 – Дочерний роутер commentRouter

В данном роутере, как и во всех других, указывается метод HTTP-запроса, его URI, дополнительные функции и функция контроллера, которая будет обрабатывать запрос клиента. Каждый метод HTTP-запроса предназначен для определенного типа взаимодействия с ресурсом или же данными:

* GET – получение ресурса;
* POST – создание ресурса;
* PUT – обновление ресурса;
* DELETE – удаление ресурса.

Контроллер commentController, используемый в данном роутере, с некоторыми его функциями был рассмотрен выше в листинге 3.3.

Все основные маршруты и привязанные к ним обработчики запросов с их описанием представлены в таблице 3.1.

Таблица 3.1 – Основные маршруты, обрабатываемые сервером.

|  |  |  |
| --- | --- | --- |
| URI | Метод HTTP | Предназначение |
| 1 | 2 | 3 |
| api/user/registration | POST | Метод для регистрации пользователя |
| api/user/login | POST | Метод для авторизации пользователя |
| api/user/logout | POST | Метод для выхода из учетной записи пользователя |
| api/user/activate/:link | GET | Метод для подтверждения почты пользователя по ссылке |
| api/user/refresh | GET | Метод для обновления Access-токена |
| api/category/ | GET | Метод для получения всех категорий |
| api/question/ | POST | Метод для создания опроса |
| api/question/vote | POST | Метод для голосования в опросе |
| api/question/:id | PUT | Метод для обновления опроса |
| api/question/vote | DELETE | Метод для удаления голоса пользователя в опросе |
| api/question/:id | DELETE | Метод для удаления опроса |
| api/question/ | GET | Метод для получения всех опросов |
| api/question/:id | GET | Метод для получения одного опроса со всей информацией по нему |
| api/rating/question/ | POST | Метод для добавления оценки опросу |
| api/rating/question/ | PUT | Метод для обновления оценки опроса |
| api/rating/question/ | DELETE | Метод для удаления оценки опроса |

Продолжение таблицы 3.1.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| api/comment/ | POST | Метод для создания комментария |
| api/comment/:id | PUT | Метод для обновления комментария |
| api/comment/:qid | GET | Метод для получения всех комментариев под опросом |
| api/achievement/ | GET | Метод для получения всех возможных достижений пользователя |
| api/achievement/ :userId | GET | Метод для получения списка достижений, полученных пользователем |
| api/account/:id | PUT | Метод для обновления данных о профиле пользователя |
| api/account/:userId | GET | Метод для получения данных о профиле пользователя |

Таким образом, было разработано двадцать три основных маршрута.

Дополнительные функции, которые обрабатывают запрос до функций контроллера, будут описаны в следующем подразделе.

### 3.1.3 Разработка промежуточных программных слоев

Middleware (промежуточный программный слой) – это фрагмент кода в конвейере приложения, используемый для обработки запросов и ответов.

На сервере присутствует три функции, являющиеся Middleware:

* AuthMiddleware – это функция, которая предназначена для проверки того, авторизован пользователь или же нет.
* CheckActivationAccountMiddleware – это функция, которая предназначена для проверки того, подтверждена почта пользователем или нет.
* CheckRoleMiddleware – это функция для проверки роли пользователя.

Функция CheckActivationAccountMiddleware используется только при авторизации в приложении. Функция CheckRoleMiddleware используется в маршрутах, которые может использовать только администратор.

Реализация данных функций представлена в приложении К.

### 3.1.4 Разработка модуля для работы с почтой

Класс MailService предназначен для работы с почтой, т.е. для отправления письма с подтверждающей ссылкой пользователю при регистрации.

Перед написанием класса, была создана специальная электронная почта для отправки писем [questionnaireappmail@gmail.com](mailto:questionnaireappmail@gmail.com) в почтовом сервисе Gmail.

Для того чтобы позволить серверу взаимодействовать с данной электронной почтой, необходимо включить двухфакторную идентификацию и включить в настройках доступ по протоколу IMAP. После этого нужно настроить «пароли приложений» в соответствующей вкладке в разделе «Безопасность». Пароль автоматически сгенерируется сервисом и после его можно использовать для подключения сервера к данной почте.

Данные, такие как используемый порт и хост, находятся в разделе «справка», представленного на рисунке 3.2.

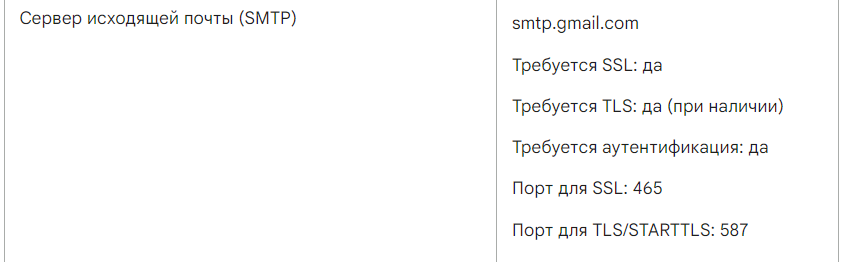


Рисунок 3.2 – Информация для подключения к Gmail-почте

Реализация класса MailService представлена в листинге 3.7.

|  |
| --- |
| class MailService  {  constructor() {  this.transporter = nodemailer.createTransport({  host: process.env.SMTP\_HOST,  port: process.env.SMTP\_PORT,  secure: false,  auth: {  user: process.env.SMTP\_USER,  pass: process.env.SMTP\_PASSWORD  }  })  }  async sendActivationMail(to, link) {  await this.transporter.sendMail({  from: process.env.SMTP\_USER,  to,  subject: 'Активация профиля в приложении Questionnaire',  text: '',  html:  <div>  <h1>Для активации перейдите по ссылке</h1>  <a href="${link}">${link}</a>  </div>  })  }  } |

Листинг 3.7 – Класс MailService

Класс имеет конструктор и функцию отправки письма. В конструкторе при помощи модуля nodemailer происходит создание подключения к почте по протоколу SMTP, с указанием хоста, порта и данных для авторизации. Функция sendActivationMail принимает в параметре «to» почту пользователя, которую он указал при регистрации, а в параметре «link» ссылку для подтверждения того, что почта принадлежит пользователю. Ссылка автоматически генерируется при регистрации пользователя и записывается в базе данных. При переходе по ней профиль будет активирован и пользователь сможет авторизоваться в приложении.

### 3.1.5 Работа со стандартом JSON WEB Token

JSON Web Token (JWT) – открытый стандарт для создания токенов доступа, основанный на формате JSON. Он используется для передачи данных для аутентификации в клиент-серверных приложениях [9].

Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения подлинности профиля. Работа с токенами реализована в классе TokenService, который представлен в приложении Л.

Функция для генерации токенов представлена в листинге 3.8.

|  |
| --- |
| generateTokens(payload) {  const accessToken = jwt.sign(payload, process.env.JWT\_ACCESS\_SECRET, {expiresIn: '15m'})  const refreshToken = jwt.sign(payload, process.env.JWT\_REFRESH\_SECRET, {expiresIn: '15d'})  return {  accessToken,  refreshToken  }  } |

Листинг 3.8 – Функция создания токенов

При авторизации пользователя будут сгенерированы access и refresh токены и отправлены клиенту, где будут храниться в SharedPreferences.

Access токен предназначен для авторизации пользователя и будет передаваться вместе с запросом в заголовке Authorization. Срок его жизни равен 15 минутам. Refresh токен предназначен для обновления access токена после окончания его срока жизни, храниться в удаленной базе данных для каждого пользователя, срок его жизни равен 15 дням. После окончания срока жизни refresh токена пользователю будет необходимо снова пройти авторизацию в приложении, чтобы получить новые refresh и access токены.

Функция для проверки действительности refresh токена в листинге 3.9.

|  |
| --- |
| validateAccessToken(token) {  try {  const userData = jwt.verify(token, process.env.JWT\_ACCESS\_SECRET);  return userData;  }catch (e) {  return null;  }} |

Листинг 3.9 – Функция проверки действительности refresh токена

Функция для проверки действительности access токена в листинге 3.10.

|  |
| --- |
| validateRefreshToken(token)  {  try {  const userData = jwt.verify(token, process.env.JWT\_REFRESH\_SECRET);  return userData;  }  catch (e)  {  return null;  }  } |

Листинг 3.10 – Функция проверки действительности access токена

Данные функции получают в параметре соответствующие токены и верифицируют их с помощью секретных ключей, хранящихся только на сервере.

При получении запроса сервер будет извлекать из заголовка access токен и проверять его на действительность, после чего, в случае успеха, обрабатывать полученный запрос. Иначе же будет отправлена ошибка с кодом «401», которую клиент будет перехватывать, а после отправлять запрос на обновление вместе с refresh токеном в cookie. Если отправленный токен окажется действителен, то сервер отправит новые refresh и access токены, в случае же недействительности токена пользователю придется авторизоваться в мобильном приложении вновь.

Эти и другие методы для работы с токенами используются в классе UserService, реализация которого представлена в приложении М.

## 3.2 Реализация клиентской части приложения

### 3.2.1 Файловая структура приложения

Перед началом разработки приложения была продумана файловая структура проекта в соответствии с выбранной архитектурой, а так же для группировки файлов по их значению для мобильного приложения.

Файловая структура проекта представлена на рисунке 3.3. На рисунке показаны основные пакеты и файлы.

Проект имеет следующие основные файлы:

* SplashScreen.java – activity, запускаемая при старте приложения и показывает экран-заставку, пока идет загрузка другой activity.
* MainActivity.java – главная activity приложения, с фрагментами которой в основном будет взаимодействовать пользователь.
* ActivityRegLog.java – activity, запускаемая, когда пользователь запускает приложение в первый раз или если он не авторизовался. Содержит в себе фрагменты для регистрации и авторизации в приложении.
* App.java – синглтон класс используемый для работы с одним и тем же экземпляром базы данных, что позволяет не нагружать мобильное устройство пользователя «тяжелыми» объектами.

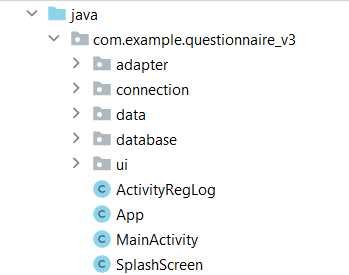


Рисунок 3.3 – Файловая структура мобильного приложения.

Проект имеет следующие основные пакеты:

* «ui» содержит в себе пакеты с фрагментами и их ViewModel.
* «database» содержит пакеты Entity и Dao, где описаны в качестве объектов таблицы базы данных и методы для взаимодействия с ними соответственно, а так же в данном пакете находится файл AppDatabase для работы с базой данных.
* «data» содержит пакеты Model и POJO, в которых хранятся классы для работы с мобильным приложением и для взаимодействия с сервером.
* «connection» содержит необходимые файлы и интерфейсы для взаимодействия с сервером с помощью библиотеки Retrofit2.
* «adapter» содержит классы для реализации списков.

Правильная файловая структура позволяет быстрее и легче разрабатывать множество программных компонентов.

### 3.2.2 Реализация локальной базы данных

Для взаимодействия с локальной базой данных используется библиотека ROOM. Она предоставляет аннотации для того, чтобы указать какие классы и как должны быть использованы для реализации базы данных [10].

Первым делом были созданы классы Category, Account, Achievement, UserAchievement и отмечены аннотацией @Entity для объявления сущности базы данных. Поля, являющиеся первичными ключами, отмечены соответственно аннотацией @PrimaryKey. Аннотация @ColumnInfo явно указывает, как должны быть названы поля в таблицах на основе данных классов.

Внешние ключи для таблицы UserAchievements указываются в аннотации @Entity с указанием дополнительных параметров в соответствующих атрибутах, что представлено в листинге 3.11. Кроме того, в атрибутах может быть явно указано название создаваемой таблицы.

|  |
| --- |
| @Entity(tableName = "userAchievement",  foreignKeys = {  @ForeignKey  (entity = Account.class, parentColumns = "userId", childColumns = "userId", onDelete = CASCADE),  @ForeignKey  (entity = Achievement.class, parentColumns = "id", childColumns = "achievementId", onDelete = CASCADE)})  public class UserAchievement {  @PrimaryKey  public long id;  public Date createdAt;  @ColumnInfo(name = "userId")  public long userId;  @ColumnInfo(name = "achievementId")  public long achievementId;  } |

Листинг 3.11 – Класс UserAchievement

Для взаимодействия с сущностями базы данных используются специальные классы с аннотацией @Dao. Данная аннотация предназначена для объявления интерфейса, который будет заниматься манипулированием данными базы данных. Внутри классов описываются стандартные методы для взаимодействия с данными в таблицах, отмеченные соответствующими аннотациями, а также прописываются специфические sql-запросы, если те необходимы.

Реализация интерфейса для взаимодействия с сущностями achievement и userAchievement представлена в листинге 3.12.

|  |
| --- |
| @Dao  public interface AchievementDao {  @Query  ("SELECT \* FROM achievement")  List<Achievement> getAll();  @Query  ("SELECT achievement.id, achievement.achievementName, achievement.description, achievement.imageUrl" +  " FROM achievement join userAchievement on achievement.id == userAchievement.achievementId" +  " where userAchievement.userId == :userId")  List<Achievement> getAllByUser(long userId);  @Insert  void insert(Achievement achievement);  …//другие методы  } |

Листинг 3.12 – Интерфейс AchievementDao

После создания классов сущностей и интерфейсов для взаимодействия с ними, был создан основной класс по работе с базой данных, который помечен аннотацией @Database. Этот класс является абстрактным и наследует RoomDatabase. Этот класс представлен в листинге 3.13.

|  |
| --- |
| @Database(entities = {Account.class, Category.class, Achievement.class, UserAchievement.class}, version = 1)  public abstract class AppDatabase extends RoomDatabase {  public abstract AccountDao userDao();  public abstract CategoryDao categoryDao();  public abstract AchievementDao achievementDao();  } |

Листинг 3.13 – Класс AppDatabase

В параметрах аннотации @Database указывается, какие @Entity будут использоваться и версия базы. Для каждого @Entity класса из списка entities будет создана таблица. В @Database классе описываются абстрактные методы для получения @Dao объектов, которые будут использоваться для взаимодействия с данными таблицами из локальной базы данных.

Поскольку класс AppDatabase является «тяжелым», то используется только один его экземпляр для всех операцией, что позволяет реализовать singleton класс App, представленный в листинге 3.14.

|  |
| --- |
| public class App extends Application  {  public static App instance;  private AppDatabase database;  @Override  public void onCreate()  {  super.onCreate();  instance = this;  database = Room.databaseBuilder(this, AppDatabase.class, "database").build();  }  public static App getInstance() {  return instance;  }  public AppDatabase getDatabase() {  return database;  }  } |

Листинг 3.14 – Класс App

Таким образом, для получения базы данных в коде используется строка «AppDatabase db = App.getInstance().getDatabase();», после чего из данного объекта получаются нужные @Dao классы и используются их методы.

### 3.2.3 Работа с сервером с помощью Retrofit2

Библиотека Retrofit2 предоставляет REST-клиент для взаимодействия с сервером. Для работы с Retrofit понадобятся следующие три вида классов:

* model классы, которые используются как модели JSON;
* интерфейсы, которые определяют возможные HTTP-операции;
* класс Retrofit.Builder – экземпляр, использующий интерфейс и API Builder ради определения конечной точки URL для операций HTTP.

Первым делом был реализован основной класс, представленный в листинге 3.15, с помощью экземпляра которого будет происходить взаимодействие с созданным ранее сервером на облачной платформе.

|  |
| --- |
| public class RetrofitClient {  private static Retrofit retrofit = null;  public static Retrofit getClient(String baseUrl) {  if (retrofit==null){  retrofit = new Retrofit.Builder()  .baseUrl(baseUrl)  .addConverterFactory(GsonConverterFactory.create())  .build();  }  return retrofit;  }  } |

Листинг 3.15 – Класс RetrofitClient

Этот класс создает синглтон Retrofit. Для модификации необходим базовый URL-адрес для создания своего экземпляра, поэтому он будет передаваться при вызове «RetrofitClient.getClient(String baseUrl)». Затем этот URL будет использоваться для построения экземпляра класса.

Далее был создан интерфейс ServerAPI, содержащий методы, которые будут использоваться для выполнения HTTP-запросов, таких как GET, POST, PUT и DELETE. Каждый метод отмечен соответствующей аннотацией, указывающей на тип запроса и имеющей URL-адрес. Возвращаемые данные сразу инкапсулируются в специально созданные для этого классы, чтобы с ними можно было работать как с обычными объектами Java. Пример реализации интерфейса с несколькими методами отображен в листинге 3.16.

|  |
| --- |
| public interface ServerAPI {  @GET("user/users")  Call<List<User>> getAllUsers();  @POST("user/registration")  Call<RegistrationInfoResponse> registration(@Body RegistrationInfo registrationInfo);  @POST("user/login")  Call<RegistrationInfoResponse> login(@Body LoginInfo loginInfo);  …//другие методы  } |

Листинг 3.16 – Интерфейс ServerAPI

В параметрах методов для отправки данных на сервер так же размещены классы, основанные на JSON. Данные, хранящиеся в экземплярах этих классов, сериализуются в формат JSON и отправляются вместе с запросом. Пример модели для сериализиации представлен в листинге 3.17.

|  |
| --- |
| public class LoginInfo {  @SerializedName("email")  @Expose  private String email;  @SerializedName("password")  @Expose  private String password;  public LoginInfo(String email, String password)  {  super();  this.email = email;  this.password = password;  }  …//остальная часть класса  } |

Листинг 3.17 – Модель для сериализации LoginInfo

Был разработан вспомогательный класс ApiUtils, который в листинге 3.18. Этот класс имеет базовый URL-адрес в качестве статической переменной, а также предоставляет интерфейс ServerAPI приложению через static-метод getServerApi().

|  |
| --- |
| public class ApiUtils  {  private static final String BASE\_URL = "https://server-l22apf5e0-xrabell13.vercel.app/api/";  public static ServerAPI getServerApi() {  return RetrofitClient.getClient(BASE\_URL)  .create(ServerAPI.class);  }  } |

Листинг 3.18 – Класс ApiUtils

Таким образом, для получения клиента в коде будет использоваться строка «ServerAPI mService = ApiUtils.getServerApi();», после чего из данного объекта будут доступны методы для взаимодействия с сервером.

### 3.2.4 Реализация архитектурного паттерна MVVM

Для реализации паттерна MVVM стандартные компоненты для разработки мобильного приложения были разделены следующим образом:

* связка XML-файл и связанный с ним Java-класс являются компонентом представления. Данный Java-класс отвечает только за ту логику, которая относится к разметке интерфейса, но никак не касается его поведения;
* компонент модели представления является отдельным Java-классом, к публичным методам и полям которого происходит привязка из представления с помощью технологии Android Data Binding;
* компонент модели представляет собой данные, с которыми будет взаимодействовать модель представления.

В мобильном приложении используется шаблон NavigationView для реализации меню с использованием фрагментов для более быстрой загрузки, размещенный в главной активности MainActivity. Помимо данной активности существуют еще две: SplashScreen и ActivityRegLog.

Активность SplashScreen отображается пока идет загрузка главной активности или же активности, на которой размещены фрагменты авторизации и регистрации. Во время ее работы проверяется значение «userRegistered», после чего, в случае получения значения «true», происходит проверка валидности access и refresh токенов. В зависимости от их валидности запускается либо главная активность, либо активность для отображения фрагментов регистрации и авторизации. При получении значения «false» будет сразу запущена активность ActivityRegLog.

У каждого фрагмента имеется класс ViewModel, реализующий всю бизнес-логику и предоставляющий данные для пользования представлению. В данных классах так же используются асинхронные запросы к серверу на получение данных с помощью описанных ранее методов.

## 3.3 Выводы по разделу

В данном разделе была рассмотрена реализация серверной и клиентской части разработанного программного средства.

В серверной части приложения были созданы модели для взаимодействия с данными в удаленной базе данных и для автоматической генерации ее структуры, система маршрутизации, контроллеры для обработки запросов и DTO для организации отправляемых данных клиенту. Были описаны основные маршруты, обрабатывающие их контроллеры и описание методов. Для реализации аутентификации и возможности пользования приложением только авторизованными пользователями был использован открытый стандарт JWT. Для совершения дополнительных действий до передачи запроса соответствующим контроллерам был разработан дополнительный программный слой, состоящий из функций.

В клиентской части приложения была разработана локальная база данных с использованием классов, которые обозначены соответствующими аннотациями для последующего создания таблиц на их основе и использования методов классов как запросов для взаимодействия с ними. Были разработаны классы и интерфейсы для взаимодействия с сервером, разработаны сериализуемые классы для получения и отправления данных. Реализован выбранный архитектурный паттерн.

# 4 Анализ информационной безопасности приложения

ФИО

Подпись

Дата

Лист

1

*47*

*БГТУ 04.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

4 Анализ информационной

безопасности приложения

Лит.

Листов

3

*74217096, 2023*

Разработанное программное средство состоит из мобильного приложения, сервера, удаленной и локальной базы данных. Каждый из компонентов приложения может иметь уязвимости, которые необходимо ликвидировать.

## 4.1 Возможные угрозы безопасности программного средства

В ходе анализа программного средства было выявлено несколько потенциальных угроз, которые описаны далее.

В серверной части программного средства потенциальной угрозой является общедоступное API, а также база данных, из которой с помощью API злоумышленники могут извлекать данные.

В мобильной части программного средства потенциальными угрозами являются возможность пользователя с правами ROOT просмотреть файл с настройками приложения SharedPreferences, использование SQL-инъекций для взаимодействия с локальной или удаленной базой данных и получение исходного кода путем декомпиляции байткода dex-файлов собранного apk приложения.

Помимо прочего потенциальной угрозой в обоих случаях является перехват трафика по незащищенному соединению.

Далее описаны методы для устранения найденных угроз.

## 4.2 Реализованные методы защиты программного средства

### 4.2.1 Методы защиты передачи данных по сети

Для защиты сети передачи данных между клиентом и сервером, в приложении используется протокол HTTPS – расширение протокола HTTP для поддержки шифрования в целях повышения безопасности.

Защиту данных в HTTPS обеспечивает криптографический протокол SSL/TLS, который шифрует передаваемую информацию. Таким образом, этот протокол является обёрткой для HTTP. Он обеспечивает шифрование данных и делает их недоступными для посторонних.

Протокол SSL/TLS позволяет двум незнакомым между собой участникам сети установить защищённое соединение через незащищённый канал посредством использования асимметричной криптографии и симметричное шифрования.

Данный протокол используется в облачной платформе Vercel, на которой и развернута серверная часть приложения для возможности взаимодействия мобильного приложения с сервером через интернет.

### 4.2.2 Методы защиты файла с настройками приложения

Некоторые данные, такие как токены и значение, зарегистрирован пользователь или нет, хранятся в файле настроек SharedPreferences. Чтобы исключить возможность просмотра данных этого файла был использован EncryptedSharedPreferences – реализация SharedPreferences, которая шифрует ключи и значения при записи информации, является официальным решением от Android. Для использования шифрования был создан или извлечен, если он уже существует, мастер-ключ из хранилища ключей Android. После этого он используется для инициализации или открытия экземпляра EncryptedSharedPreferences. Так же зашифрованный файл будет меняться каждый раз, когда происходит изменение данных в нем или их получение, что затруднит доступ к нему у злоумышленников.

Небольшим минусом является существование значительной разницы в производительности между SharedPreferences и EncryptedSharedPreferences, однако при хранении небольшого количества часто извлекаемых данных это не сильно отобразится на работоспособности приложения, но значительно усилит безопасность хранимых данных, поэтому этим минусом в данном случае можно пренебречь.

### 4.2.3 Методы защиты баз данных от SQL-инъекций

SQL-инъекция является распространенной уязвимостью, которая позволяет атакующему использовать фрагмент вредоносного кода на языке структурированных запросов (SQL) для манипулирования базой данных и получения доступа к потенциально ценной информации.

Для предотвращения данной уязвимости при обращении к локальной или удаленной базе данных передаваемые запросы с потенциальными SQL-инъекциями используются как prepared statement, то есть как функции, используемые для предварительной компиляции кода SQL, отделяя его от данных.

Использование prepared statement так же являются более эффективными, чем «сырой» запрос на SQL к базе данных, потому что их можно использовать далее без повторной компиляции.

### 4.2.4 Методы защиты исходного кода приложения

Для защиты исходного кода мобильного приложения используется утилита ProGuard, которая удалит из готового кода неиспользуемые фрагменты и изменит имена переменных и методов для усложнения реверс-инжиниринга приложения. Кроме того, это позволит уменьшить размер загружаемых на устройство файлов.

Во время своей работы утилита совершает несколько последовательных шагов. Первым шагом является рекурсивная проверка того, какие именно классы и члены классов (переменные, методы, константы) используются. Другие классы или члены классов, неиспользуемые в мобильном приложении, будут удалены. Вторым шагом является запутывание, в которое входит изменение названий классов и членов классов, которые не являются точками входа, случайными строками, состоящими из цифр и букв. Точки входа сохранят свое оригинальное название.

Все это затрудняет декомпиляцию и исследование работы приложения.

### 4.2.5 Методы защиты общедоступного API

Для защиты от несанкционированного доступа незарегистрированного пользователя к общедоступному api, был использован JWT Token. При первом запуске мобильного приложения пользователю необходимо зарегистрироваться и после пройти авторизацию. Запрос с данными о почте и пароле передается на сервер. Если все введено верно, то пользователю выдаются два токена: обновления и доступа, которые сохранятся в зашифрованном «SharedPreferences». Когда пользователь попытается обратиться к серверу, то в заголовок запроса «Authorization» включается токен доступа, а затем сервер решает предоставлять или запрещать доступ на основе этого токена. Время жизни токенов определяется в конфигурации сервера.

Токены создаются на сервере на основе некоторой информации о пользователе и заголовка с помощью секретных ключей, хранящихся на сервере. Таким образом, без знания данных ключей невозможно подделать токены.

Для защиты удаленной базы данных используется разграничение ролей на сервере. С помощью дополнительного программного слоя и токенов определяется id пользователя, после чего выясняются его роли. На основе этих данных решается, может он получить доступ к тем или иным функциями или же нет.

## 4.3 Выводы по разделу

В данном разделе были рассмотрены потенциальные угрозы разработанного дипломного проекта, такие как: сеть передачи данных, хранение данных в файле настроек приложения, SQL-инъекции для доступа к локальной или удаленной базе данных, получение исходного кода мобильного приложения, общедоступное API с возможностью доступа к базе данных.

Были описаны реализованные методы защиты приложения от несанкционированного доступа злоумышленниками, утечки данных.

К основным методам защиты приложения относится использование: безопасного протокола HTTPS для передачи данных по сети, шифрования данных из файла настроек с помощью EncryptedSharedPreferences на основе специальных мастер-ключей из хранилища Android, заранее скомпилированных SQL-запросов перед вставкой в них введенных пользователем данных, генерируемых на сервере с помощью секретных ключей JWT-токенов, утилиты ProGuard для оптимизации и запутывания исходного кода, разграничения ролей пользователя.

# 5 Тестирование приложения

ФИО

Подпись

Дата

Лист

1

*50*

*БГТУ 05.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

5 Тестирование приложения

Лит.

Листов

4

*74217096, 2023*

Тестирование приложения является важным этапом при разработке программного средства, чтобы еще на начальных этапах выявлять ошибки и исправлять их до того, как конечный пользователь начнет пользоваться приложением.

Существует огромное количество вариантов тестирования приложения, однако будут выделены те, что использовались непосредственно в данной работе.

Тестирование данного мобильного приложения включает в себя описанные далее варианты тестирования.

Локальное модульное тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки. Данное тестирование позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Функциональное тестирование – процесс проверки приложения на соответствия системы или компонента заданным функциональным требованиям. Оно является тестированием черного ящика, что предоставляет возможность протестировать функциональность программного средства, не зная его внутренней работы.

## 5.1 Локальное модульное тестирование

Для реализации локального модульного тестирования были созданы и после использованы методы в классе ExampleUnitTest для тестирования основных функций мобильного приложения.

Реализация одного из тестов, в данном случае теста авторизации пользователя, представлена в листинге 5.1.

|  |
| --- |
| …//другие методы для тестирования  @Test  public void login\_isCorrect()  {  LoginViewModel lgvm = new LoginViewModel();  boolean result = lgvm.Login("XRabell13@gmail.com", "1234");  boolean expected = true;  assertEquals(expected, result);  }  …//другие методы для тестирования |

Листинг 5.1 – Модульное тестирование авторизации

Суть модульных тестов заключена в том, что в специальных методах отмеченных аннотацией @Test создаются тестируемые экземпляры классов. Вызываются их методы, в параметры которых подставляются определенные значения, от чего заранее известно, каким именно должен быть желаемый результат. Далее полученный результат от метода сравнивается с желаемым с помощью специального статического метода assertEquals. Еще один из реализованных тестов представлен в листинге 5.2, а именно, тест регистрации пользователя.

|  |
| --- |
| …//другие методы для тестирования  @Test  public void registration\_isCorrect()  {  RegistrationViewModel rvm = new RegistrationViewModel();  boolean result = rvm.Registration ("Tim","XRabell2020@gmail.com", "1234567");  boolean expected = true;  assertEquals(expected, result);  }  …//другие методы для тестирования |

Листинг 5.2 – Модульное тестирование регистрации

Стоит также отметить, что использование архитектурного шаблона MVVM облегчило модульное тестирование, так как бизнес-логика и взаимодействие с интерфейсом и данными слабо зависят друг от друга.

Результат работы представленного метода приведен на рисунке 5.1.

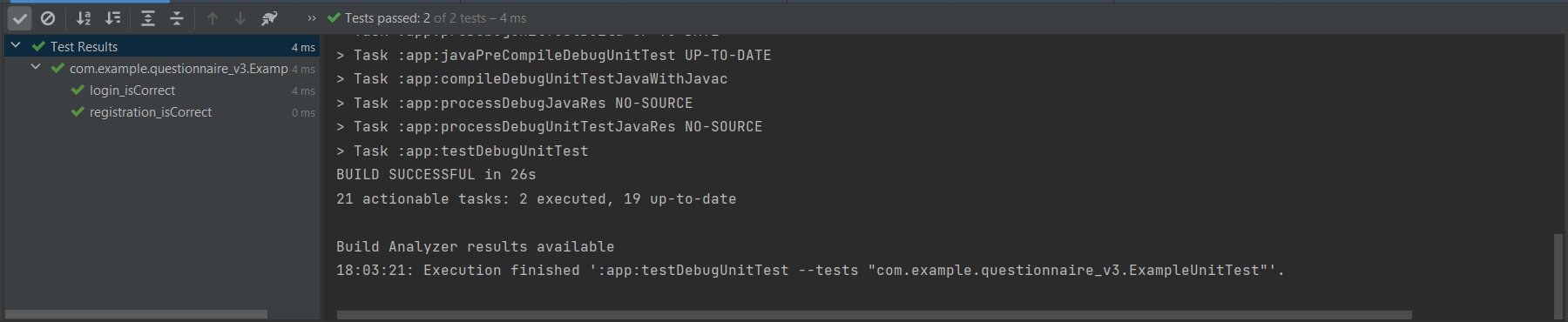


Рисунок 5.1 – Результат работы теста login\_isCorrect()

Подобным образом были модульно протестированы все основные функции разработанного мобильного приложения.

## 5.2 Функциональное тестирование

Основной задачей функционального тестирования является проверка того, соответствует ли разработанная система требованиям к ней, а так же являются ли действия системы при взаимодействии с ней предсказуемыми.

Для функционального тестирования были разработаны тестовые кейсы – наборы входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанных для определённой цели или тестового условия, таких как выполнения определённого пути программы или же для проверки соответствия определённому требованию.

Могут быть как позитивные тестовые кейсы, так и негативные. Позитивные тестовые кейсы предполагают под собой проверку того, что система адекватно отреагирует на корректные данные. Негативные же тестовые кейсы показывают, как система работает с некорректными данными.

Пример некоторых тестовых кейсов приложения представлен в таблице 5.1.

Таблица 5.1 – Основные тестовые кейсы мобильного приложения

|  |  |  |
| --- | --- | --- |
| Описание теста | Шаги воспроизведения | Результат |
| 1 | 2 | 3 |
| Регистрация (позитивный) | 1. Перейти на экран регистрации;  2. Ввести в поля «ник», «email», «пароль» данные «Снежана», «snezhanaorlenkovich@gmail.com», «1234567» соответственно;  3. Нажать на кнопку «Зарегистрироваться». | Ожидаемый результат: на указанную пользователем почту приходит письмо с подтверждающей ссылкой, в приложении выводится надпись «Проверьте электронную почту». |
| Фактический результат: соответствует ожидаемому результату. |
| Регистрация (негативный) | 1. Перейти на экран регистрации;  2. Ввести в поля «Ник», «Email», «Пароль» данные «Снежана», «snezhanagmail.com», «1234567» соответственно;  3. Нажать на кнопку «Зарегистрироваться». | Ожидаемый результат: в приложении выводится надпись «Неверный email». |
| Фактический результат: соответствует ожидаемому результату. |
| Авторизация (позитивный) | 1. Перейти на экран авторизации;  2. Ввести в поля «Email» и «Пароль» данные уже имеющейся активной учетной записи, «snezhanaorlenkovich@gmail.com» и «1234567» соответственно;  3. Нажать на кнопку «войти». | Ожидаемый результат: произведен вход в приложение, открывается страница с профилем. |
| Фактический результат: соответствует ожидаемому результату. |
| Авторизация (негативный) | 1. Перейти на экран авторизации;  2. Ввести в поля «Email» и «Пароль» данные несуществующей учетной записи, «test1» и «test1@gmail.com».  3. Нажать на кнопку «войти». | Ожидаемый результат: в приложении выводится надпись «Неверный email или пароль». |
| Фактический результат: соответствует ожидаемому результату. |

Продолжение таблицы 5.1.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Создание опроса (позитивный) | 1. Перейти на экран создания опроса;  2. Ввести в поля «вопрос», «ответ 1» и «ответ 2» данные «Test2», «answer1», «answer2» соответственно;  3. Нажать на кнопку «опубликовать». | Ожидаемый результат: опрос успешно публикуется приложением. |
| Фактический результат: соответствует ожидаемому результату. |
| Создание опроса (негативный) | 1. Перейти на страницу создания опроса;  2. Ввести в поля «вопрос» и «ответ 1» данные «Test3» и «answer1» соответственно;  3. Нажать на кнопку «опубликовать». | Ожидаемый результат: в приложении выводится надпись «количество вариантов ответов не может быть меньше двух». |
| Фактический результат: соответствует ожидаемому результату. |
| Создание комментария под опросом  (позитивный) | 1. Перейти на страницу со всеми опросами;  2. Выбрать один из опросов;  3. Нажать на кнопку «комментарии»;  4. Ввести в поле «комментарий» какой-либо текст;  5. Нажать на кнопку «добавить». | Ожидаемый результат: к опросу добавлен новый комментарий. |
| Фактический результат: соответствует ожидаемому результату. |

Подобным образом было проведено тестирование оставшегося функционала мобильного приложения. Данное тестирование проводилось как при непосредственной разработке интерфейса мобильного приложения, чтобы исправлять возникшие ошибки при первом же их появлении, так и после завершения разработки с помощью иных лиц – добровольцев, не причастных к созданию приложения.

## 5.3 Выводы по разделу

В данном разделе были рассмотрены и описаны использовавшиеся виды тестирования программного средства: локальное модульное и функциональное.

Для локального модульного тестирования в среде разработки Android Studio были разработаны специальные методы для тестов в специально сгенерированном для этого классе. Некоторые из методов описаны и показаны их результаты.

Для функционального тестирования использовались тестовые кейсы, которые разделены на два вида: негативные и позитивные. В таблицах было описано несколько основных позитивных и негативных тестовых кейсов, расписаны их названия, тип, шаги воспроизведения, а так же ожидаемые и фактические результаты.

Некоторые ошибки выявлялись в ходе тестирования еще на этапе разработки, другие же незначительные несоответствия ожидаемым результатам, выявленные позже, были также исправлены.

# 6. Руководство пользователя

ФИО

Подпись

Дата

Лист

1

*54*

*БГТУ 06.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

6 Руководство пользователя

Лит.

Листов

8

*74217096, 2023*

## 6.1 Руководство по развертыванию программного средства

Для развертывания программного средства необходимо развернуть серверную и клиентскую части приложения.

Для развертывания серверной части требуется иметь:

* Visual Studio Code 1.78;
* Node.Js 18.13.0;
* PostgreSQL13.9;
* операционную систему Windows 10 версии 19045 или выше;
* процессор Intel(R) Core(TM) i5-8265U;
* CPU @ 1.60GHz, 1800 МГц;
* 8 логических процессоров, 4 ядра;
* видеокарта, которая поддерживает минимальное разрешение экрана WXGA (1366 на 768).

Для начала следует загрузить проект в Visual Studio Code. После необходимо открыть терминал в среде разработке и выполнить команду «npm init -y». Данная команда установит все необходимые модули, опираясь на информацию о них в файле «package.json». В файле «db.js» указывается либо строка подключения к базе данных PostgreSQL, либо иные имеющиеся данные для подключения к ней.

Далее для запуска сервера локально на компьютере необходимо ввести команду «npm run start». Запущенный сервер будет работать по адресу «http://localhost:7000», что можно использовать для тестирования сервера без интернета. Для изменения порта, на котором запустится сервер, в файле .env необходимо изменить параметр PORT.

Для запуска сервера глобально необходимо загрузить проект на GitHub, после чего зарегистрироваться на платформе Vercel. Проект из GitHub можно использовать для развертывания на платформе, указывая, какую именно ветвь для этого брать.

Минимальные настройки для развертывания проекта на платформе Vercel находятся в файле «vercel.json».

Для развертывания клиентской части требуется иметь:

* Android Studio Flamingo, 2022.2.1;
* Java 8;
* мобильное устройство с Android 8.0 и выше, эмулятор с Android 8 и выше;
* операционную систему Windows 10 версии 19045 или выше.

Кроме того, требуются те же технические требования для компьютера, что были указаны в списке выше для развертывания серверной части.

Для начала следует загрузить проект в среде разработки Android Studio, после чего дождаться автоматической установки всех необходимых независимо компилируемых программных модулей, динамически подключаемых к основной программе и предназначенных для расширения и использования её возможностей. Далее для проверки работы проекта необходимо либо создать эмулятор с операционной версией Android 8 и выше, или подключить с помощью USB кабеля или Wi-Fi реальное мобильное устройство с теми же требованиями по операционной системе, после чего совершить сборку приложения и его запуск.

## 6.2 Руководство по использованию мобильного приложения

При первом запуске приложения пользователь попадает на экран регистрации, снимок которого представлен на рисунке 6.1.

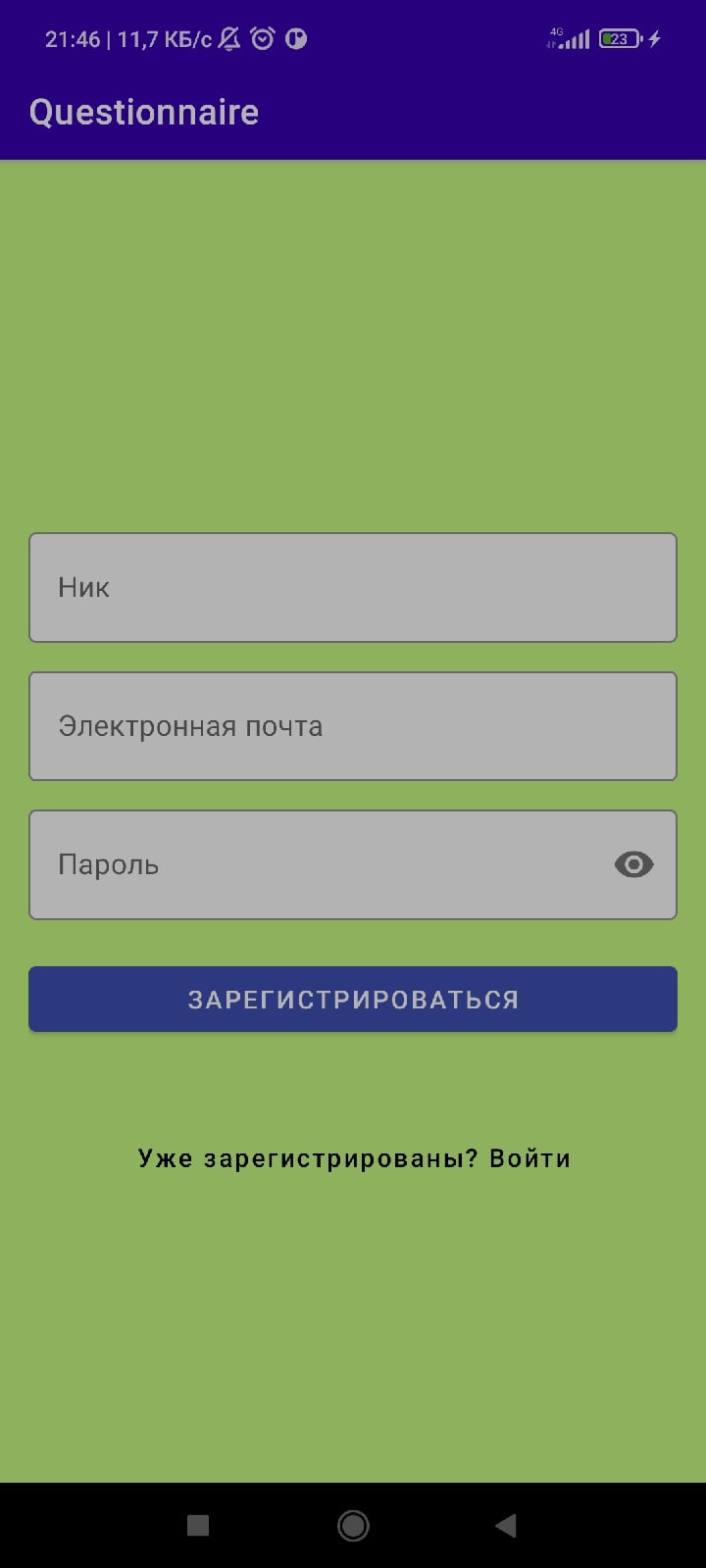


Рисунок 6.1 – Экран регистрации

Для регистрации пользователю необходимо указать ник, электронную почту и пароль, после чего нажать на кнопку «Регистрация». Далее ему придет на указанную электронную почту письмо со ссылкой, перейдя по которой он активирует свой профиль. После этого он сможет авторизоваться в приложении.

Чтобы перейти на страницу с авторизацией необходимо нажать на надпись под кнопкой «Регистрация», которая гласит «Уже зарегистрированы? Войти», после чего откроется экран авторизации, представленный на снимке 6.2.

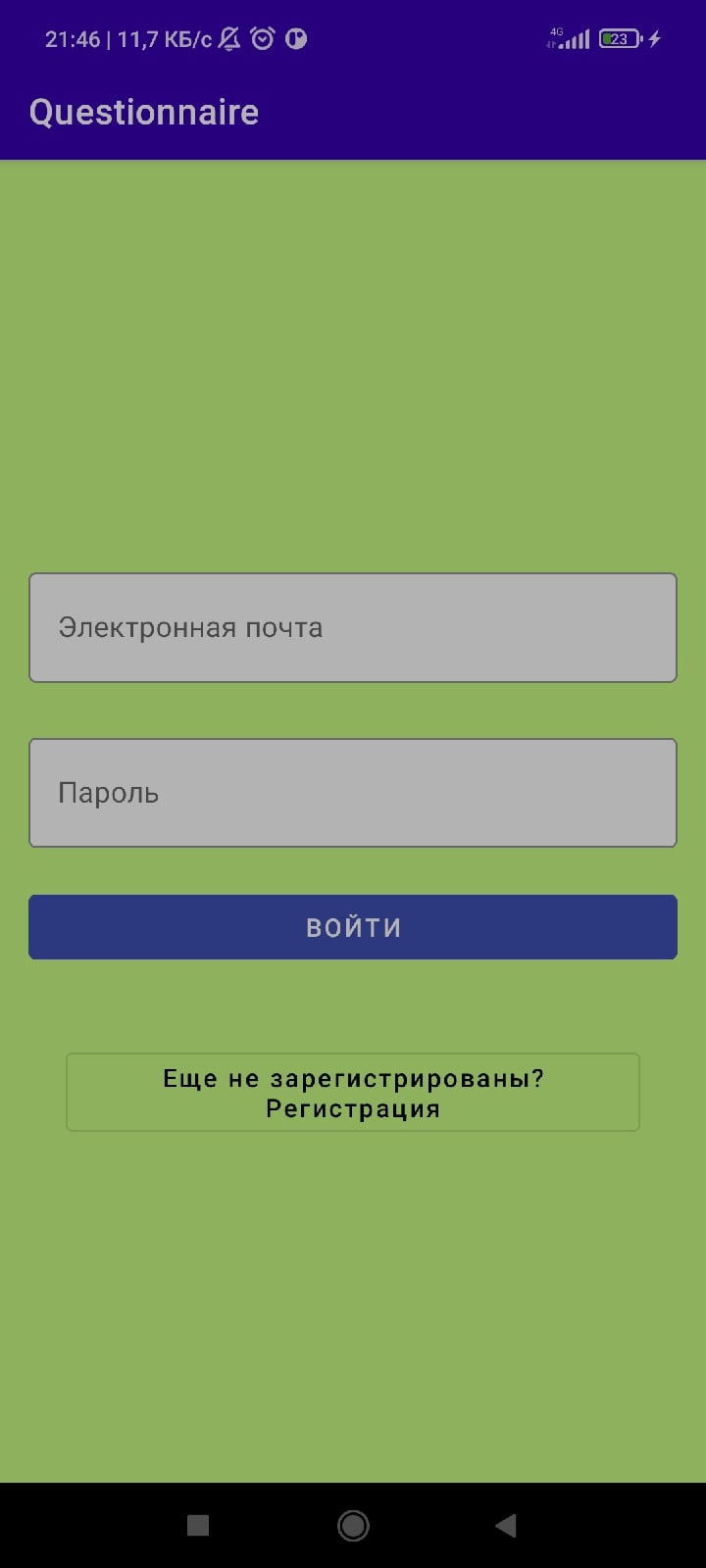


Рисунок 6.2 – Экран авторизации

На данном экране необходимо ввести требуемые значения, такие как электронная почта и пароль, после чего нажать на кнопку «Войти». Таким образом, пользователь зайдет в свою учетную запись и откроется экран со списком всех опросов в приложении, представленный на снимке 6.3.

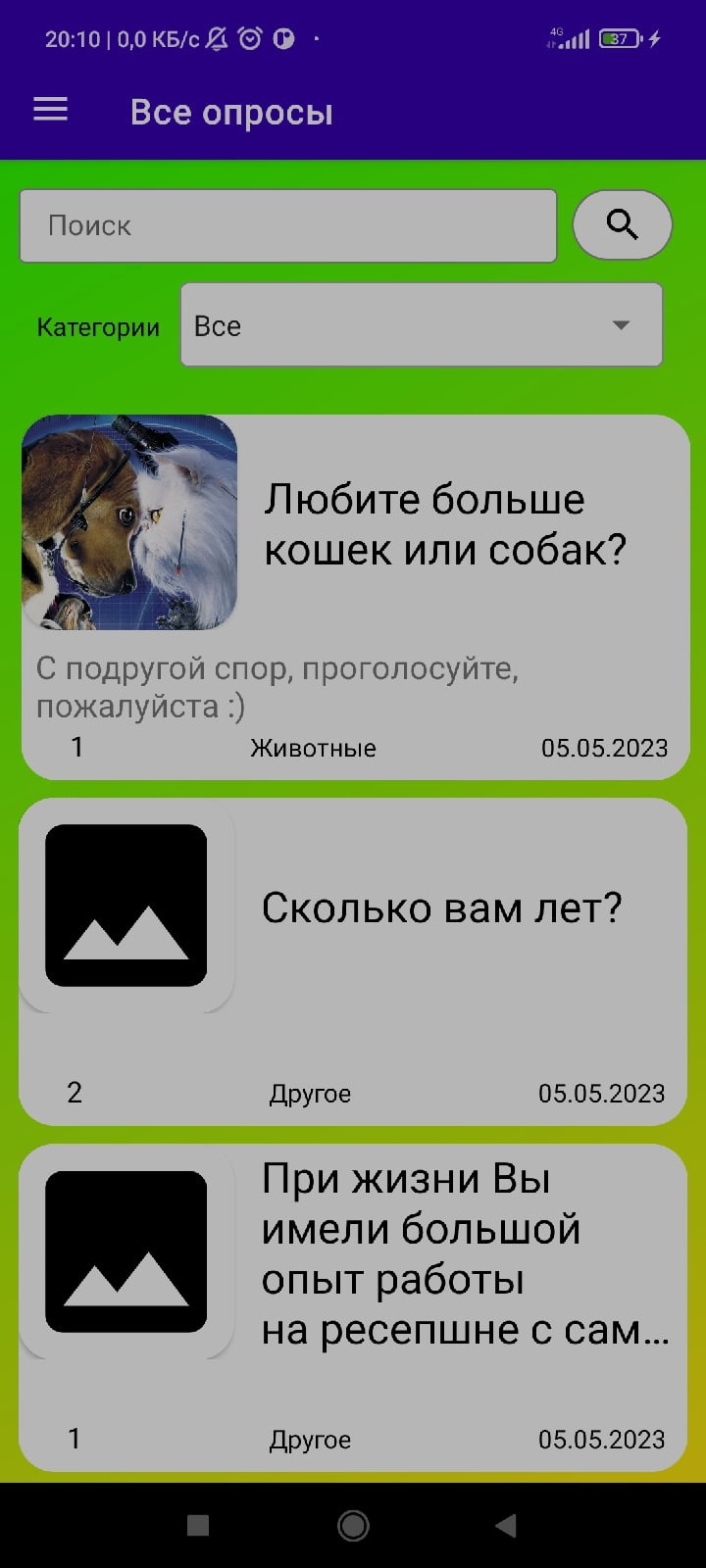


Рисунок 6.3 – Экран со списком всех опросов

На данном экране расположена строка для поиска опросов по слову, а так же предоставлена возможность поиска опросов и по категориям. Опросы представляют собой элементы списка, на которых можно увидеть вопрос опроса, рейтинг, дату создания и его категорию. Так же, если имеется, можно увидеть картинку к опросу и частичное его описание.

Если нажать на опрос из списка, то открывается экран с данным опросом, представленный на снимке 6.4, где можно лучше увидеть картинку и полное описание к опросу, а так же варианты ответов на опрос. Если опрос открыл пользователь, который его создал, то для него есть дополнительные кнопки, такие как: редактирование, удаление или просмотр статистики.

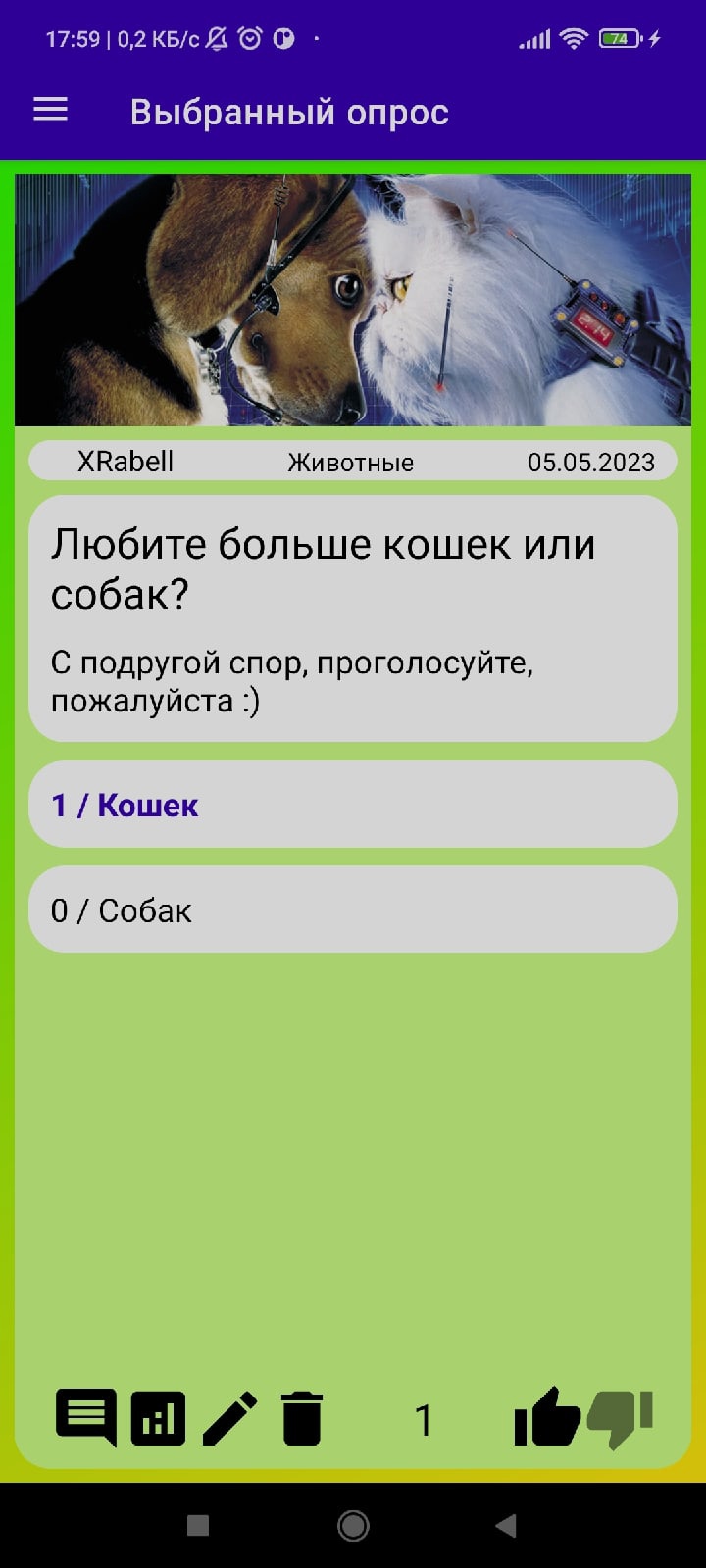


Рисунок 6.4 – Экран с выбранным опросом

После нажатия на один из вариантов ответа на опрос становится видно, сколько человек проголосовало за тот или иной ответ. Помимо прочего есть возможность поставить «лайк» или «дизлайк» опросу, повлияв на его рейтинг.

При нажатии на кнопку «комментарии», расположенную в левом нижнем углу, открывается экран с комментариями к данному опросу, снимок которого представлен на рисунке 6.5. Для ввода текста комментария имеется строка в низу экрана. При нажатии на кнопку «добавить» комментарий будет опубликован. Кроме того, имеется возможность ставить «лайк» и «дизлайк» комментарию так же, как и опросам. Свой комментарий можно удалить или отредактировать, зажав его и выбрав соответствующее действие.

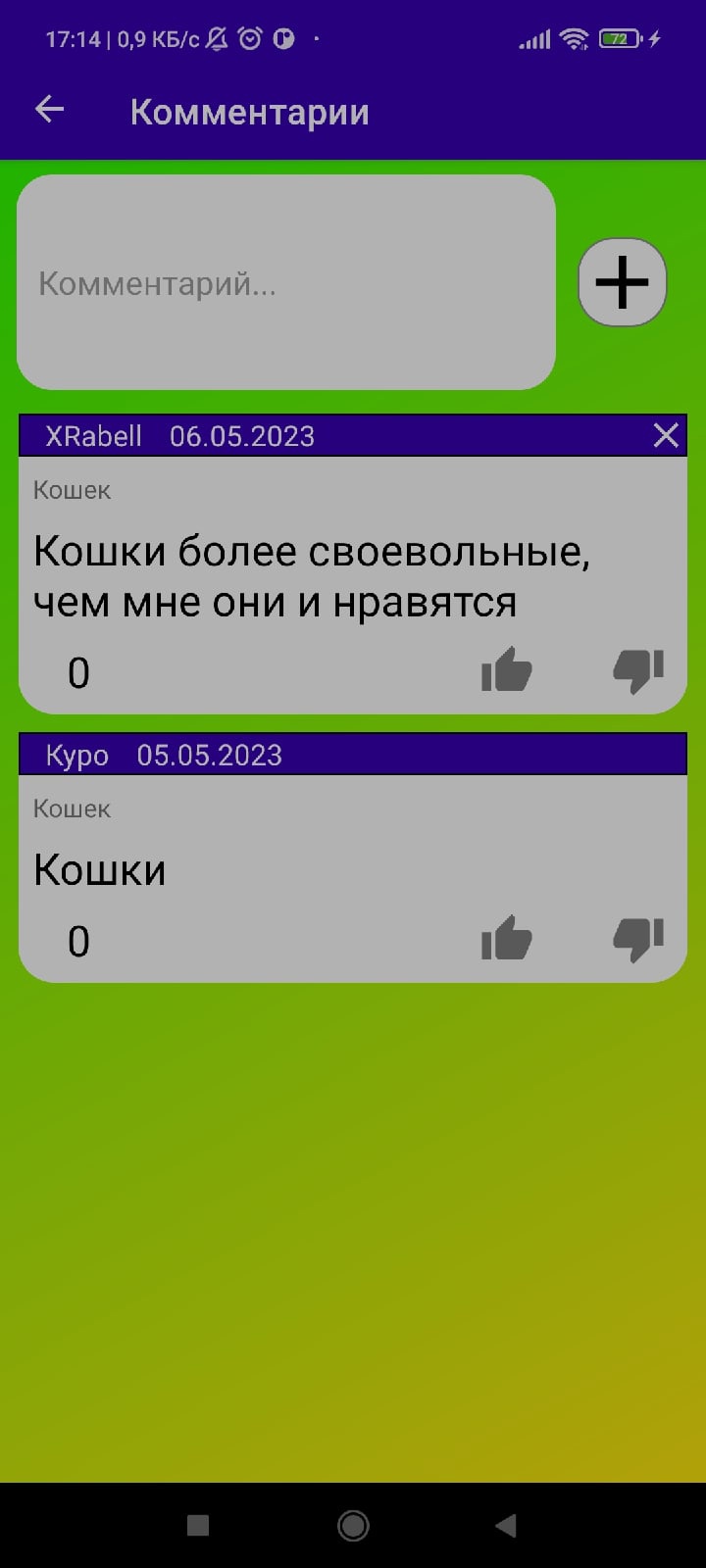


Рисунок 6.5 – Экран с комментариями к опросу

В мобильном приложении так же имеется меню, представленное на рисунке 6.6, имеющие такие пункты как: «Профиль», «Все опросы», «Мои опросы», «Отвеченные опросы» и «Создать опрос».

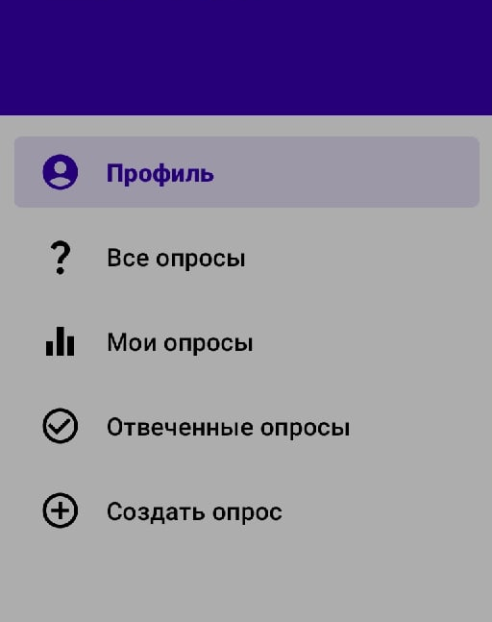


Рисунок 6.6 – Меню мобильного приложения

При нажатии на пункт меню «Профиль» открывается экран с профилем пользователя, снимок которого представлен на рисунке 6.7. Профиль можно изменить путем редактирования описания и ника, а так же добавления картинки на аватар. Кроме того, имеется возможность посмотреть полученные достижения, выйти из профиля и удалить учетную запись.

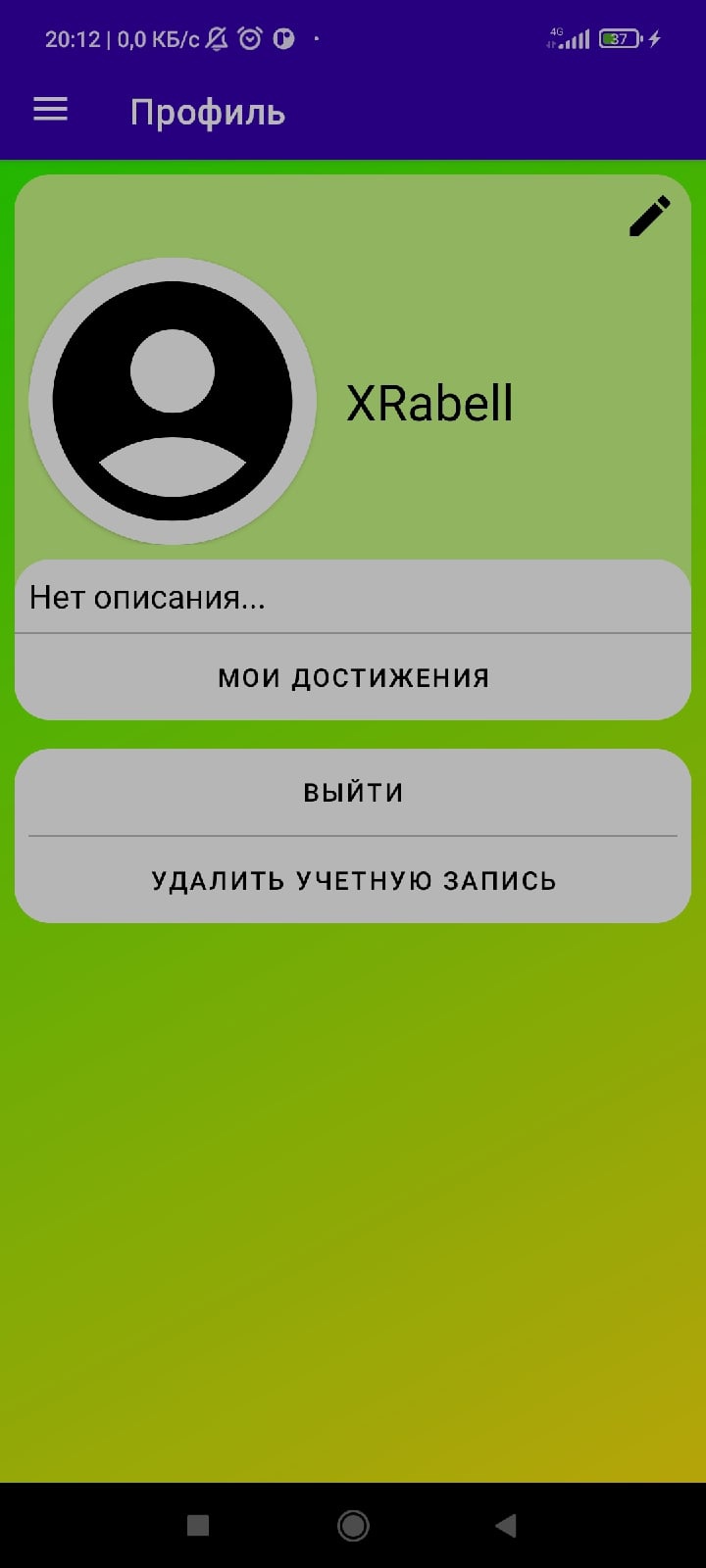


Рисунок 6.7 – Экран с профилем пользователя

При нажатии на кнопку «Мои достижения» откроется экран со всеми достижениями, имеющимися в приложении. Снимок экрана представлен на рисунке 6.8. Достижения, полученные пользователем, будут отображаться серым цветом.

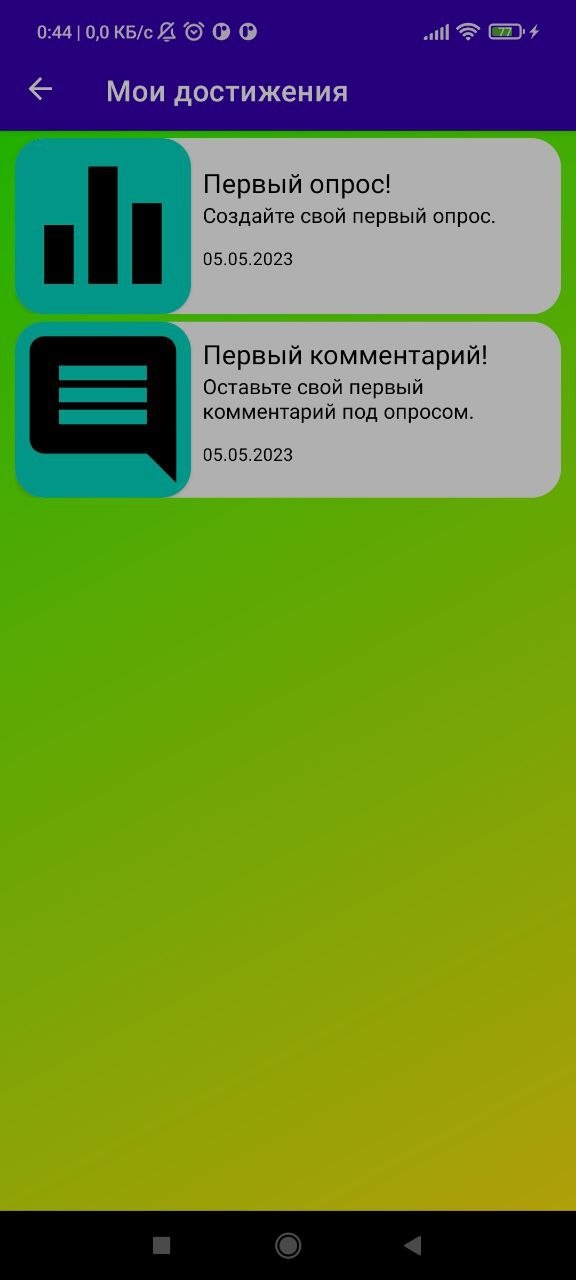


Рисунок 6.8 – Экран с достижениями

Пункт «Мои опросы» ведет на экран с опросами, созданными пользователем. По структуре экран подобен экрану со всеми опросами. Свои опросы пользователь может частично отредактировать или удалить, а так же посмотреть их статистику, которая представлена на рисунке 6.9.

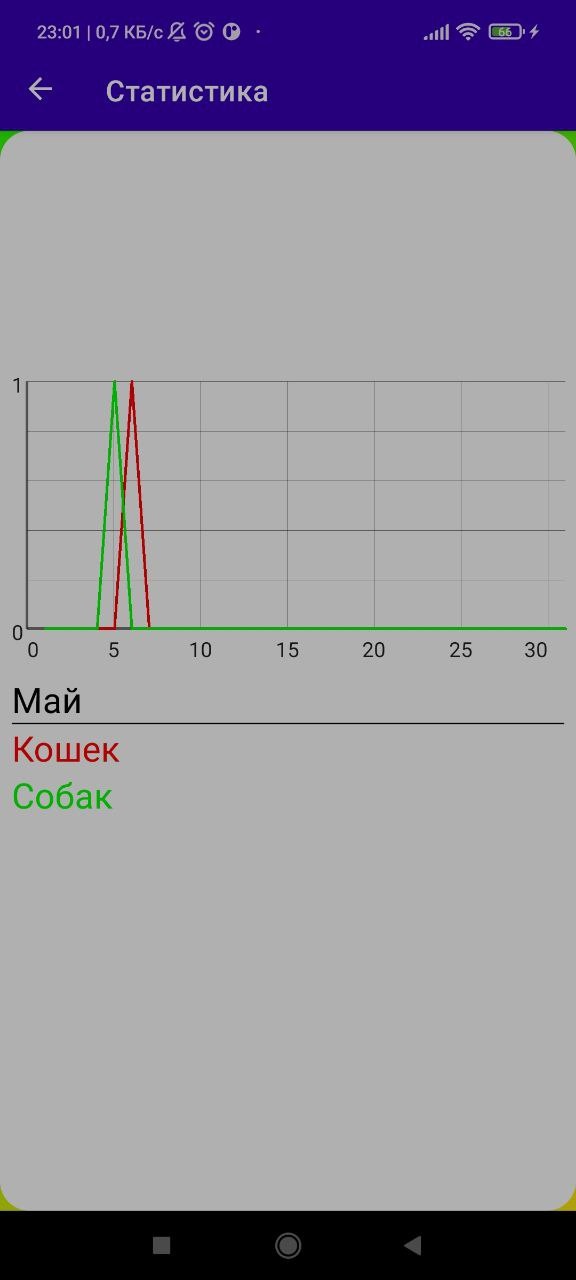


Рисунок 6.9 – Статистика по опросу

Пункт «Отвеченные опросы» ведет на экран, содержащий в себе опросы, на которые ранее был дан ответ пользователем. Данный экран по структуре такой же, как и «Все опросы» или «Мои опросы».

Пункт «Создать опрос» ведет на экран для создания опроса, снимок которого представлен на рисунке 6.10. На данном экране необходимо написать сам вопрос и два обязательных варианта ответа на него. Описание, выбор картинки и добавление еще трех вариантов ответа на опрос являются опциональными и не препятствуют публикации опроса.

Подобным образом выглядит и экран с редактированием опроса, за тем исключением, что поля при открытии экрана уже будут заполнены изначальными данными редактируемого опроса. Для редактирования будут доступны только изображение опроса, описание, а также поля для добавления новых вариантов ответа, если изначально было создано меньше пяти вариантов ответа в опросе.

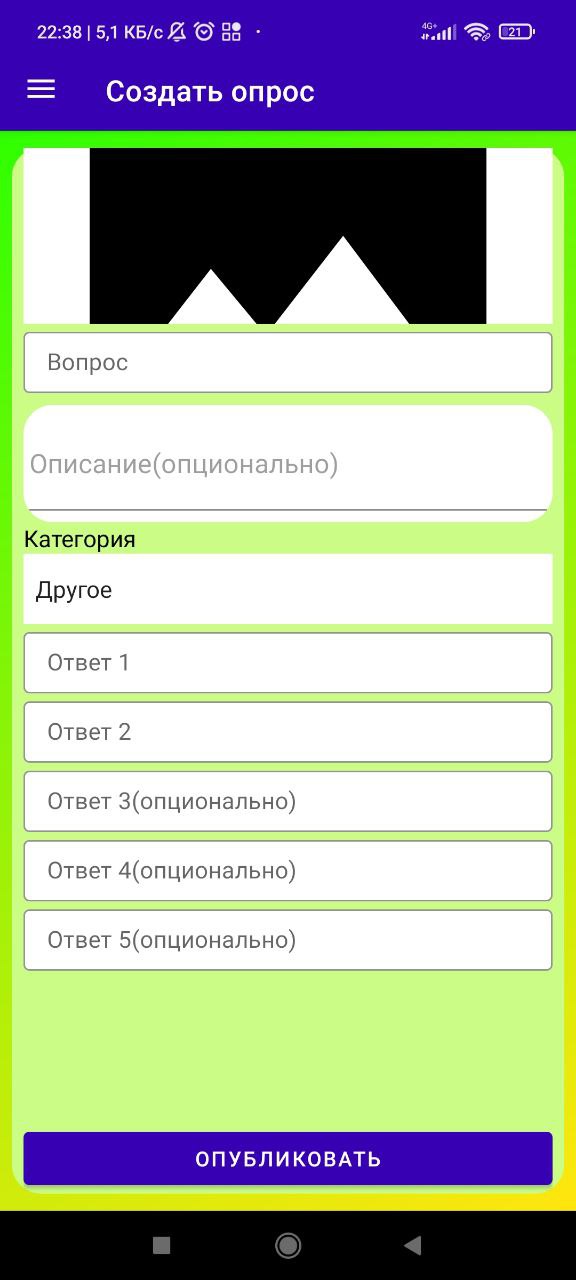


Рисунок 6.10 – Экран создания опроса

После нажатия на кнопку «Опубликовать» опрос станет доступен для всех пользователей в приложении.

## 6.3 Выводы по разделу

В данном разделе было описано руководство по развертыванию серверной и клиентской части разработанного программного средства. Кроме этого было описано возможное взаимодействие с приложением, представлены снимки его интерфейса и их подробное описание. Разработанный интерфейс является интуитивно понятным и не должен предоставлять проблем конечным пользователям.

# 7 Технико-экономическое обоснование проекта

ФИО

Подпись

Дата

Лист

1

*62*

*БГТУ 07.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В.*

Консульт.

*Соболевский А.С. .*

Н. контр.

Нистюк О.А.

Утв.

*Смелов В.В.*

7 Технико-экономическое обоснование проекта

Лит.

Листов

11

*74217096, 2023*

Основной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства, представленного в дипломном проекте. В рамках данного раздела необходимо определить затраты, произведенные на всех стадиях разработки описанного программного модуля, а также полную стоимость программного средства.

## 7.1 Общая характеристика разрабатываемого приложения

При выполнении данного дипломного проекта было разработано мобильное приложение, предназначенное для взаимодействия с мини-опросами.

Целью приложения было связывать любителей проходить опросы с возможностью их создания, комментирования и оценивания, а так же поиска по интересующим их категориям и просмотра статистики.

Любой пользователь может зарегистрироваться в приложении, для этого надо заполнить форму с личными данными, для чего необходимо придумать ник и пароль, подтвердить почту. После регистрации, пользователь, который уже считается зарегистрированным пользователем, попадает на главную страницу и ему становятся доступны все имеющиеся функции.

Пользователям приложения предлагается создавать свои опросы, принимать участие в чужих опросах, комментировать их и ставить лайки или дизлайки, производить поиск интересующего опроса по категории или словам. Имеется так же возможность оценивать комментарии под опросами. Вместе с этим пользователь может просматривать опросы, на которые были даны ответы.

Так же имеется возможность просматривать статистику по своим опросам и получать достижения за количество созданных опросов по категории, комментариев и оставленных лайков или дизлайков.

Во время разработки дипломного проекта использовалась технология Java 8 для разработки мобильного приложения на Android.

Разработанное программное решение имеет следующие преимущества перед рассмотренными в разделе 1 аналогичными образцами:

− интуитивно понятное использование приложения;

− минималистичный и удобный интерфейс;

− отсутствие больших нагрузок на сеть;

− полнота необходимого функционала;

− дополнительный функционал, которого не имеется в аналогах;

− обеспечение безопасности данных пользователей.

Стратегия монетизации предполагает платную месячную подписку на дополнительный функционал в приложении.

В рамках данного раздела необходимо определить затраты на всех стадиях разработки программного средства. Также необходимо выполнить расчет экономии основных видов ресурсов в связи с использованием разработанного приложения на основе успеха и монетизации приложений-аналогов.

## 7.2 Исходные данные для проведения расчётов

Исходные данные для расчета приведены в таблице 7.1.

Таблица 7.1 – Исходные данные для расчета

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные обозначения | Норматив |
| Численность разработчиков, чел | Чр | 1 |
| Коэффициент изменения скорости обработки информации, ед. | Кск | 0,6 |
| Норматив дополнительной заработной платы, % | Ндз | 12,5 |
| Норматив общепроизводственных и общехозяйственных расходов, % | Нобп, обх | 20 |
| Ставка отчислений в Фонд социальной защиты населения, % | Нфсзн | 34 |
| Ставка отчислений по обязательному страхованию в БРУСП «Белгосстрах», % | Нбгс | 0,6 |
| Цена одного машино–часа, руб. | Смч | 0,06 |
| Норматив прочих прямых затрат, % | Нпз | 20 |
| Норматив расходов на сопровождение и адаптацию, % | Нрса | 17,0 |
| Ставка НДС, % | ННДС | 0 |
| Налог на прибыль, % | Нп | 0 |

Источниками исходных данных для данных расчетов выступают действующие законы и нормативно-правовые акты. Для ПВТ НДС и налог на прибыль 0%.

## 7.3 Методика обоснования цены

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования программных средств. Выбор эффективных проектов программных средств связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у обычного пользователя системы.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации программных средств, которые остаются в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования, в том числе и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на оплату машинного времени;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена конечного программного средства [11].

### 7.3.1 Объем программного средства

В таблице 7.2 указаны в укрупнённом виде все работы, реально выполненные для создания, указанного в дипломной работе программного средства и количество рабочих дней, реально потраченных для выполнения работ.

Таблица 7.2 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, дней |
| Построение диаграмм для проектирования дипломного проекта | 6 |
| Создание базы данных | 6 |
| Разработка серверной части программного средства | 12 |
| Разработка клиентской части программного средства | 21 |
| Тестирование интерфейса мобильного приложения | 1 |
| Тестирование бизнес-логики | 3 |
| Написание руководства пользователя | 2 |
| Всего | 51 |

Результат по данной таблице будет использовано далее для расчётов.

Для оценки объёма программного средства, все его функции классифицируются с использованием специального каталога функций, который определяет их объем. Общий объем программного средства *V*o, вычисляется как сумма объёмов *Vi* каждой из *n* его функций по формуле 7.1

|  |  |  |
| --- | --- | --- |
|  | , | (7.1) |

где – объем *i*-ой функции ПС, условных машинных команд;

*n* – общее число функций.

В таблице 7.3 представлены функции, присутствующие в рассматриваемом приложении и соответствующий им объем в условных машино-командах.

Таблица 7.3 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| Номер функции | Содержание функции | Объем, условных  машино-команд |
| 101 | Организация ввода информации | 150 |
| 102 | Контроль, предварительная обработка | 450 |
| 111 | Управление вводом/выводом | 2400 |
| 204 | Обработка наборов и записей базы данных | 2670 |
| 207 | Манипулирование данными | 9550 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 |
| 707 | Графический вывод результатов | 480 |

Опираясь на данные таблицы 7.3, можно определить объем программного средства, разработанного в ходе дипломного проектирования:

*Vo* = 150+450+2400+2670+9550+410+480= 16 110 (маш. команд).

Уточнённый объем программного средства *V*o*/* в формуле 7.2

|  |  |  |
| --- | --- | --- |
|  | , | (7.2) |

где *Vo* – объем программного средства, усл. машино-команд;

Кск – коэффициент изменения скорости обработки информации.

Исходя из вычисленного объёма программного средства, можно определить уточненный объем программного средства:

*V*o*/* = 16 110 ⋅ 0,6 = 9 666 (маш. команд).

### 7.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в программировании на Java. В итоге было установлено, что средняя месячная заработная плата на позиции разработчика junior составляет 1900 рублей. Согласно таблице 7.1, проект разрабатывался одним человеком на протяжении 51 дня, что с расчетом на то, что рабочих дней в месяце 21, означает, что разработка происходила в течение 2,4 месяцев.

Таким образом, основная заработная плата рассчитывается по формуле 7.3

,

|  |  |  |
| --- | --- | --- |
|  | , | (7.3) |

где Соз– основная заработная плата, руб.;

Траз1 – время раработки Java разработчика, месяцев;

Сзп1 – средняя месячная заработная плата Java разработчика.

руб

В дальнейшем для других расчётов используется основная заработная плата, рассчитанная по указанной выше методике.

### 7.3.3 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (7.4)

|  |  |  |
| --- | --- | --- |
|  | , | (7.4) |

где Соз– основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб.

### 7.3.4 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаем на производстве и профессиональных заболеваний определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей. Отчисления в ФСЗН вычисляются по формуле 7.5

|  |  |  |
| --- | --- | --- |
|  |  | (7.5) |

где – основная заработная плата, руб.;

– дополнительная заработная плата на конкретное ПС, руб.;

– норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 7.6

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили руб., а в фонд социальной защиты населения – руб.

### 7.3.5 Расходы на материалы

Сумма расходов на материалы СМ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода НМ на уточненный объем программного средства *Vo/* , по формуле 7.7

|  |  |  |
| --- | --- | --- |
|  | . | (7.7) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равен 0,46 руб. (по данным, приведенным в приложении 2 таблице П 2.10 «Оценка значений среднего расхода материалов на разработку и отладку 100 строк кода применения ПС» методического пособия), можно определить сумму расходов на материалы.

СМ = 0,46 ⋅ 9 666 / 100 = 44,46 руб.

Сумма расходов на материалы была вычислена на основе данных, приведенных в таблице 7.1 данного дипломного проектирования.

### 7.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *Vo/* и на норматив расхода машинного времени на отладку ста строк исходного кода НМВ, по формуле 7.8.

|  |  |  |
| --- | --- | --- |
|  | . | (7.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 12 (по данным, приведенным в приложении 2 таблице П 2.11 «Оценка значений среднего машинного времени на отладку 100 строк исходного кода без применения ПС» методического пособия), можно определить сумму расходов на оплату машинного времени.

Смв = 0,06 ⋅ 9 666 ⋅ 12 / 100 = 69,6 руб.

Сумма расходов на оплату машинного времени была вычислена на основе данных, приведенных в таблице 7.1 данного дипломного проектирования.

### 7.3.7 Прочие прямые затраты

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз, и находится по формуле 7.9.

|  |  |  |
| --- | --- | --- |
|  |  | (7.9) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму прочих затрат, что и вычислено далее.

Спз = ⋅ 20 / 100 = 912 руб.

### 7.3.8 Накладные расходы

Сумма накладныхрасходовСобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, по формуле 7.10.

|  |  |  |
| --- | --- | --- |
|  |  | (7.10) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов.

= ⋅ 20 / 100 = 912 руб.

### 7.3.9 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов, по формуле 7.11.

|  |  |  |
| --- | --- | --- |
|  | Ср = Соз + Сдз + Сфсзн + Сбгс + См + Смв + Спз + Собп,обх. | (7.11) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

Ср = +++ 30,78 + 44,46 + 69,6 + 912 + 912 = 8 843,04 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе, и составила 8 843,04 рублей.

### 7.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса, и находится по формуле 7.12.

|  |  |  |
| --- | --- | --- |
|  |  | (7.12) |

= 8 843,04 ⋅ 17 / 100 = 1 503,32 руб.

Сумма расходов на сопровождение и адаптацию была вычислена на основе данных, рассчитанных ранее в данном разделе.

Все проведенные выше расчеты необходимы для вычисления полной себестоимости разработанного дипломного проекта.

### 7.3.11 Полная себестоимость

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса.

Полная себестоимость Сп вычисляется по формуле 7.13

|  |  |  |
| --- | --- | --- |
|  |  | (7.13) |

Сп = 8 843,04 + 1503,32 = 10 346,36 руб.

Полная себестоимость программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

### 7.3.12 Определение цены, оценка эффективности

Рассматриваемое программное средство разрабатывается для потребления другими лицами. После анализа способа монетизации продуктов-аналогов, был выбран способ монетизации «расширенная подписка на месяц».

Продукты-аналоги, наиболее близкие теме дипломного проекта:

* «Voliz» – мобильное приложение от разработчика 7Span для создания опросов, которые можно пройти в приложении и поделиться ими.
* «Pollie» – мобильное приложение от разработчика Pollie B.V. для создания опросов, в котором пользователям предоставляется ссылка для прохождения.
* «Poll For All» – мобильное приложение от разработчика Softry OU для создания опросов и анкет, для прохождения их по ссылке.

По балловому методу цены рассчитываются на основе оценок значимости различных параметров качества программного продукта для потребителей. Сначала каждому из тех параметров, по которым судят о качестве продукции, присваивается значение удельного веса, которое характеризует, насколько данный параметр важен для потребителя по сравнению с другими. Далее каждому параметру базового программного продукта присваивается определенное число баллов, суммирование которых с учетом весовых коэффициентов дает интегральный показатель качества продукции

|  |  |  |
| --- | --- | --- |
|  |  | (7.14) |

где К*i* − коэффициент, отражающий значимость *i-*го показателя;

ПК*i* − число баллов у *i-*го показателя продукта конкурента.

Показатель качества рассматриваемого продукта ИР:

|  |  |  |
| --- | --- | --- |
|  |  | (7.15) |

где ПР*i* − число баллов, присвоенное *i-*му показателю качества рассматриваемого в расчетах продукта.

Были выбраны следующие характеристики для показателей качества рассматриваемого программного продукта и программного продукта конкурента:

* дизайн – это то, как приложение выглядит, что очень важно для любого современного мобильного приложения;
* юзабилити – насколько приложение удобно в использовании;
* функциональность – количество инноваций, внедряемых в приложение;
* отсутствие багов – наличие несущественных ошибок в работе.

Расчет показателей качества базового и нового продуктов, согласно балловому методу, приводится в таблицах 7.4

Таблица 7.4 – Показатели качества рассматриваемого программного продукта и программного продукта конкурента

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатель качества | Весовой коэффициент | Моё приложение | Voliz | Pollie | Poll For All |
| Дизайн | 0,3 | 6 | 5 | 4 | 8 |
| Юзабилити | 0,4 | 7 | 6 | 5 | 7 |
| Функциональность | 0,2 | 7 | 6 | 7 | 8 |
| Отсутствие багов | 0,1 | 8 | 5 | 9 | 8 |
| Всего | 1 | 6,8 | 5,6 | 5,5 | 7,6 |

Расчёт прогнозного количества установок программного средства Кi при монетизации методом расширенной месячный подписки на дополнительный функционал приложения, рассчитывается по формуле

|  |  |
| --- | --- |
|  | (7.16) |

где К0 − количество установок ПС конкурента;

T0 – количество лет существования приложения;

ИР – показатель рассматриваемого программного продукта;

ИК – показатель программного продукта конкурента.

Далее представлены расчеты количества установок:

К1 = (100 000 / 4 6,8) / 5,6 = 30 357,14 (установок в год),

К2 = (200 000 / 4 6,8) / 5,5 = 61 818,18 (установок в год),

К3 = (100 000 / 4 6,8) / 7,6 = 22 368,42 (установок в год),

К = (30 357,14 + 61 818,18 + 22 368,42) / 3 = 38 181,25 (установок в год).

Определение цены месячной расширенной подписки нового продукта Ц1 будет осуществляться по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (7.17) |

где Ц0 − цена подписки программного продукта конкурента,

ИР – показатель рассматриваемого программного продукта,

ИК – показатель программного продукта конкурента.

Ц1 = (28,3 6,8) / 5,6 = 34,36 рублей,

Ц2 = ( 6,8) / 5,5 = 30,40 рублей,

Ц3 = ( 6,8) / 7,6 = 17,74 рублей,

Ц = (34,36 + 30,40 + 17,74) / 3 = 27,5 рублей.

Пользователи получают базовый функционал бесплатно, а за дополнительные опции платят. При таком раскладе обычно только 0,5 процент пользователей покупает расширенную подписку. В таком случае при среднем количестве установок в год, равном 38 181,25, денежные поступления от покупки подписки будут составлять Пост.в год = 5 249,92 рублей за год.

Количество покупателей продукта необходимых для окупаемости расширения Пп вычисляется по формуле 7.16:

|  |  |  |
| --- | --- | --- |
|  |  | (7.18) |

где Сп – полная себестоимость;

Пост.в год – денежные поступления от продажи подписки за год.

Ткп = 10346,36 / 27,5 = 377 покупателя.

Срок окупаемости приложения Tок вычисляется по формуле 7.15

|  |  |  |
| --- | --- | --- |
|  |  | (7.19) |

где – полная себестоимость;

Пост.в год – денежные поступления от рекламы в приложении за год.

Ток = 10346,36 / 5 249,92 = 1,97 года.

## 7.4 Выводы по разделу

В таблице 7.6 представлены результаты расчётов для основных показателей данной главы в краткой форме. С ними можно ознакомиться и в приложении Н.

Таблица 7.6 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 2,4 |
| Количество программистов, чел. | 1 |
| Основная заработная плата, руб. |  |
| Дополнительная заработная плата, руб. |  |
| Отчисления в Фонд социальной защиты населения и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний, руб. | 1774,98 |
| Расходы на материалы, руб. | 44,46 |
| Расходы на оплату машинного времени, руб. | 69,6 |
| Прочие прямые затраты, руб. | 912 |
| Накладные расходы, руб. | 912 |
| Себестоимость разработки программного средства, руб. | 8 843,04 |
| Расходы на сопровождение и адаптацию, руб. | 1503,32 |
| Полная себестоимость, руб. | 10346,36 |
| Годовые денежные поступления от продажи подписки, руб. | 5 249,92 |
| Срок окупаемости, лет | 1,97 |
| Количество покупателей для окупаемости, чел. | 377 |

Разработка программного средства, осуществляемая одним программистом в течение 2,4 месяцев, при заданных условиях обойдется в 10346,36 руб. Реализации данного программного средства будет приносить годовые денежные поступления в размере 5249,92 рублей и окупиться при покупке 377 подписок через 1,97 года.

Необходимость разработки программного средства обусловлена немалой заинтересованностью людей в прохождении опросов, однако на данный момент существует крайне мало приложений, непосредственно в которых происходило бы взаимодействие между людьми и их опросами.

# Заключение

ФИО

Подпись

Дата

Лист

1

*73*

*БГТУ 00.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. Контр.

Нистюк О.А.

Утверд.

*Смелов В.В.*

Заключение

Лит.

Листов

1

*74217096, 2023*

Целью дипломного проекта является мобильное приложение для создания и прохождения мини-опросов.

В результате анализа продуктов-аналогов было рассмотрено три приложения, сделаны выводы о них, а также произведён патентный поиск. Эта информация была использована для улучшения функциональности мобильного приложения и пользовательского интерфейса.

Были построены диаграммы архитектуры, развертывания и вариантов использования программного средства. Разработаны схемы переходов между фрагментами, удаленной и локальной базы данных, а так же блок-схема алгоритма получения достижений за комментарии. Кроме того, были рассмотрены технологии, которые используются в клиентской и серверной части, определены архитектурные стили проектирования для серверной и клиентской части.

Приведены реализации выбранных архитектурных стилей для сервера и клиента, а так же этапы разработки баз данных и интерфейса взаимодействия клиента с сервером. Помимо этого приведены листинги разработанных модулей для работы с почтой и JWT-токенами.

Были рассмотрены потенциальные угрозы информационной безопасности программного средства, методы защиты сети передачи данных, а также клиентской и серверной части приложения.

Было проведено тестирование основных частей программного средства, произведен обзор возможных ошибок. Анализ полученных результатов показал, что система работает корректно и соответствует поставленным требованиям

Разработано руководство пользователя, предоставляющее пользователям ознакомление со всеми возможностями функциональности приложения.

В экономическом разделе приведено технико-экономическое обоснование проекта, рассчитана себестоимость и прибыль от реализации.

В соответствии с полученным результатом, можно сделать вывод, что дипломный проект выполнен в полном объеме, поставленные целы были достигнуты.

# Список использованных источников

ФИО

Подпись

Дата

Лист

1

*74*

*БГТУ 00.00.ПЗ*

Разраб.

*Цягунович Т.В.*

Провер.

*.*

*Бернацкий П.В. .*

Н. Контр.

Нистюк О.А.

Утверд.

*Смелов В.В.*

Список использованных

источников

Лит.

Листов

1

*74217096, 2023*

1. RU0002408925 - способ тестирования и/или опроса удаленных пользователей [Электронный ресурс]. – Режим доступа: http://allpatents.ru/patent/2408925.html. ­– Дата доступа: 15.03.2023.
2. U9892 - система анкетирования в инфокоммуникативных сетях в процессе маркетинговых исследований [Электронный ресурс]. – Режим доступа: https://bypatents.com/5-u9892-sistema-anketirovaniya-v-infokommunikativnyh-setyah-v-processe-marketingovyh-issledovanijj.html. ­– Дата доступа: 15.03.2023.
3. Доля рынка мобильных и планшетных версий Android по всему миру [Электронный ресурс]. – Режим доступа: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide/#monthly-202204-202304-bar. ­– Дата доступа: 29.03.2023.
4. Паттерн MVC [Электронный ресурс]. – Режим доступа: <https://metanit.com/web/nodejs/7.1.php>. – Дата доступа: 04.04.2023.
5. Реализация MVVM в Android [Электронный ресурс]. – Режим доступа: https://stfalcon.com/ru/blog/post/android-mvvm. – Дата доступа: 06.04.2023.
6. Психология цвета в UX [Электронный ресурс]. – Режим доступа: <https://www.uprock.ru/articles/psihologiya-cveta-v-ux>. – Дата доступа: 10.04.2023.
7. Руководство по Sequelize [Электронный ресурс]. – Режим доступа: https://my-js.org/docs/guide/sequelize/. – Дата доступа: 16.04.2023.
8. Фреймворк Express – руководство пользователя [Электронный ресурс]. – Режим доступа: <https://huncode.com/js/express>. – Дата доступа: 17.04.2023.
9. Введение в веб-токен JSON [Электронный ресурс]. – Режим доступа: <https://jwt.io/introduction>. – Дата доступа: 17.04.2023.
10. Документация androidx.ROOM [Электронный ресурс]. – Режим доступа: https://developer.android.com/reference/androidx/room/package-summary. – Дата доступа: 20.04.2023.
11. Технико-экономическое обоснование дипломных проектов [Электронный ресурс]. – Режим доступа: <https://elib.belstu.by/handle/123456789/14185>. ­– Дата доступа: 10.05.2023.

# Приложение А. Диаграмма архитектуры приложения

# Приложение Б. Диаграмма развертывания приложения

# Приложение В. Схема удаленной базы данных

# Приложение Г. Диаграмма вариантов использования

# Приложение Д. Схема переходов между фрагментами

# Приложение Е. Блок-схема алгоритма получения достижений за комментарии

# Приложение Ж. Листинг кода реализации моделей для базы данных

const sequelize = require('../db')

const {DataTypes} = require('sequelize')

const queryInterface = sequelize.getQueryInterface()

const Role = sequelize.define('role', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

roleName: {type: DataTypes.STRING(20), unique: true, allowNull: false},

}, {

timestamps: false

})

const Role\_user = sequelize.define('role\_user', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

}, {

timestamps: false

})

const Category = sequelize.define('category', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

categoryName: {type: DataTypes.STRING, unique: true, allowNull: false},

}, {

timestamps: false

})

const Account = sequelize.define('account', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

nickname: {type: DataTypes.STRING(20), allowNull: false},

description: {type: DataTypes.STRING(255)},

imageUrl: {type: DataTypes.STRING},

}, {

timestamps: false

})

const User = sequelize.define('user', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

email: {type: DataTypes.STRING, unique: true, allowNull: false},

password: {type: DataTypes.STRING, allowNull: false},

isActivated: {type: DataTypes.BOOLEAN, defaultValue: false},

activationLink: {type: DataTypes.STRING}

}, {

timestamps: false

})

const Token = sequelize.define('token', {

refreshToken: {type: DataTypes.STRING, allowNull: false},

}, {

timestamps: false

})

const User\_achievement = sequelize.define('user\_achievement', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

createdAt: {type: DataTypes.DATE, allowNull: false}

}, {

timestamps: false

})

const Achievement = sequelize.define('achievement', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

achievementName: {type: DataTypes.STRING, unique: true, allowNull: false},

imageUrl: {type: DataTypes.STRING, allowNull: false},

description: {type: DataTypes.STRING, allowNull: false},

}, {

timestamps: false

})

const Question = sequelize.define('question', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

question: {type: DataTypes.STRING, allowNull: false},

description: {type: DataTypes.STRING},

imageUrl: {type: DataTypes.STRING},

rating: {type: DataTypes.INTEGER, defaultValue: 0},

})

const Answer = sequelize.define('answer', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

answer: {type: DataTypes.STRING, allowNull: false},

}, {

timestamps: false

})

const User\_answer = sequelize.define('user\_answer', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

})

const Rating\_question = sequelize.define('rating\_question', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

questionRate: {type: DataTypes.INTEGER, allowNull: false},

}, {

timestamps: false

})

const Comment = sequelize.define('comment', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

text: {type: DataTypes.STRING, allowNull: false},

rating: {type: DataTypes.INTEGER, defaultValue: 0}

})

const Rating\_comment = sequelize.define('rating\_comment', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

commentRate: {type: DataTypes.INTEGER, allowNull: false},

}, {

timestamps: false

})

// связи таблиц

Role.belongsToMany(User, {through: Role\_user, onDelete: 'SET NULL'})

User.belongsToMany(Role, {through: Role\_user, onDelete: 'SET NULL'})

User.hasOne(Account, {onDelete: 'CASCADE'})

Account.belongsTo(User)

User.belongsToMany(Achievement, {through: User\_achievement, onDelete: 'SET NULL'})

Achievement.belongsToMany(User, {through: User\_achievement, onDelete: 'CASCADE'})

User.hasMany(Comment, {onDelete: 'CASCADE'})

Comment.belongsTo(User)

User.hasMany(Rating\_comment, {onDelete: 'CASCADE'})

Rating\_comment.belongsTo(User)

User.hasMany(Question, {onDelete: 'SET NULL'})

Question.belongsTo(User)

User.belongsToMany(Answer, {through: User\_answer, onDelete: 'CASCADE'})

Answer.belongsToMany(User, {through: User\_answer, onDelete: 'CASCADE'})

User.hasMany(Rating\_question, {onDelete: 'CASCADE'})

Rating\_question.belongsTo(User)

User.hasOne(Token, {onDelete: 'CASCADE'})

Token.belongsTo(User)

Question.hasMany(Comment, {onDelete: 'CASCADE'})

Comment.belongsTo(Question)

Question.hasMany(Rating\_question, {onDelete: 'CASCADE'})

Rating\_question.belongsTo(Question)

Question.hasMany(Answer, {onDelete: 'CASCADE'})

Answer.belongsTo(Question)

Category.hasMany(Question, {onDelete: 'RESTRICT'})

Question.belongsTo(Category)

Comment.hasMany(Rating\_comment, {onDelete: 'CASCADE'})

Rating\_comment.belongsTo(Comment)

// уникальные пары значений

queryInterface.addConstraint('rating\_questions', {

fields: ['userId', 'questionId'],

type: 'unique',

name: 'unique\_constraint\_rating\_question'

});

queryInterface.addConstraint('rating\_comments', {

fields: ['userId', 'commentId'],

type: 'unique',

name: 'unique\_constraint\_rating\_comment'

});

queryInterface.addConstraint('comments', {

fields: ['userId', 'questionId'],

type: 'unique',

name: 'unique\_constraint\_comment'

});

module.exports = {

Role,

Role\_user,

Category,

Account,

User,

User\_achievement,

Achievement,

Question,

Answer,

User\_answer,

Rating\_question,

Comment,

Rating\_comment,

Token

}

# Приложение И. Листинг кода реализации класса ApiError

class ApiError extends Error{

status;

errors;

constructor(status, message, errors = []) {

super(message);

this.status = status;

this.errors = errors;

}

static badRequest(message) {

return new ApiError(404, message)

}

static internal(message) {

return new ApiError(500, message)

}

static forbidden(message) {

return new ApiError(403, message)

}

static noAccessError() {

return new ApiError(403, 'Нет доступа')

}

static unauthorizedError() {

return new ApiError(401, 'Пользователь не авторизован')

}

static badRequestErrors(message, errors = []) {

return new ApiError(400, message, errors);

}

}

module.exports = ApiError

# Приложение К. Листинг кода реализации программных слоев

// файл AuthMiddleware.js

const ApiError = require('../error/ApiError');

const tokenService = require('../services/tokenService');

module.exports = function (req, res, next) {

try {

const authorizationHeader = req.headers.authorization;

if (!authorizationHeader) {

return next(ApiError.unauthorizedError());

}

const accessToken = authorizationHeader.split(' ')[1];

if (!accessToken) {

return next(ApiError.unauthorizedError());

}

const userData = tokenService.validateAccessToken(accessToken);

if (!userData) {

return next(ApiError.unauthorizedError());

}

req.user = userData;

next();

} catch (e) {

return next(ApiError.unauthorizedError());

}

};

// файл CheckActivationAccountMiddleware.js

const ApiError = require('../error/ApiError');

const {User} = require('../models/models');

module.exports = async function (req, res, next) {

try {

const {email} = req.body

const u = await User.findOne({where: {email: email.toLowerCase()}})

if (u.isActivated==false) {

return next(ApiError.badRequest("Подтвердите почту"));

}

next();

} catch (e) {

return next(ApiError.unauthorizedError());

}

};

// файл CheckRoleMiddleware.js

const jwt = require('jsonwebtoken');

const ApiError = require('../error/ApiError');

const sequelize = require('../db')

const { QueryTypes } = require('sequelize')

module.exports = function(role) {

return async function (req, res, next) {

if (req.method === "OPTIONS") {

next()

}

try {

const decoded = req.user;

await sequelize.query('select GetUserRoles(:\_id) as "roleName";', {replacements: { \_id: decoded.id }, type: QueryTypes.SELECT}).then(result => {

if (result)

{

if(JSON.stringify(result).indexOf(role)<0) {

return next(ApiError.noAccessError());

}

}})

req.user = decoded;

next()

} catch (e) {

return next(ApiError.unauthorizedError());

}

};

}

# Приложение Л. Листинг кода для реализации класса TokenService

const jwt = require('jsonwebtoken');

const {Token} = require('../models/models.js');

class TokenService {

generateTokens(payload) {

const accessToken = jwt.sign(payload, process.env.JWT\_ACCESS\_SECRET, {expiresIn: '15m'})

const refreshToken = jwt.sign(payload, process.env.JWT\_REFRESH\_SECRET, {expiresIn: '15d'})

return {

accessToken,

refreshToken

}

}

validateAccessToken(token) {

try {

const userData = jwt.verify(token, process.env.JWT\_ACCESS\_SECRET);

return userData;

} catch (e) {

return null;

}

}

validateRefreshToken(token) {

try {

const userData = jwt.verify(token, process.env.JWT\_REFRESH\_SECRET);

return userData;

} catch (e) {

return null;

}

}

async saveToken(userId, refreshToken) {

const tokenData = await Token.findOne({user: userId})

if (tokenData) {

tokenData.refreshToken = refreshToken;

return tokenData.save();

}

const token = await Token.create({userId: userId, refreshToken})

return token;

}

async removeToken(refreshToken) {

const tokenData = await Token.destroy({

where: {

refreshToken: refreshToken

}

})

return tokenData;

}

async findToken(refreshToken) {

const tokenData = await Token.findOne({

where: {

refreshToken: refreshToken

}

})

return tokenData;

}

}

module.exports = new TokenService();

# Приложение М. Листинг класса UserService

const {User, Role\_user} = require('../models/models')

const bcrypt = require('bcrypt');

const uuid = require('uuid');

const mailService = require('./mailService');

const tokenService = require('./tokenService');

const UserDto = require('../dtos/user-dto');

const ApiError = require('../error/ApiError');

class UserService {

async registration(email, password) {

const candidate = await User.findOne({

where: {

email: email

}

})

if (candidate) {

throw ApiError.badRequest(`Пользователь с почтовым адресом ${email} уже существует`)

}

const hashPassword = await bcrypt.hash(password, 3);

const activationLink = uuid.v4();

const user = await User.create({email, password: hashPassword, activationLink})

Role\_user.create({userId: user.id, roleId: 2})

await mailService.sendActivationMail(email, `${process.env.API\_URL}/api/user/activate/${activationLink}`);

const userDto = new UserDto(user); // id, email, isActivated

const tokens = tokenService.generateTokens({...userDto});

await tokenService.saveToken(userDto.id, tokens.refreshToken);

return {...tokens, user: userDto}

}

async activate(activationLink) {

const user = await User.findOne({

where: {

activationLink: activationLink

}

})

if (!user) {

throw ApiError.badRequest('Неккоректная ссылка активации')

}

user.isActivated = true;

await user.save();

}

async login(email, password) {

const user = await User.findOne({

where: {

email: email.toLowerCase()

}

})

if (!user) {

throw ApiError.badRequest('Пользователь с таким email не найден')

}

const isPassEquals = await bcrypt.compare(password, user.password);

if (!isPassEquals) {

throw ApiError.badRequest('Неверный пароль');

}

const userDto = new UserDto(user);

const tokens = tokenService.generateTokens({...userDto});

await tokenService.saveToken(userDto.id, tokens.refreshToken);

return {...tokens, user: userDto}

}

async logout(refreshToken) {

const token = await tokenService.removeToken(refreshToken);

return token;

}

async refresh(refreshToken) {

if (!refreshToken) {

throw ApiError.unauthorizedError();

}

const userData = tokenService.validateRefreshToken(refreshToken);

const tokenFromDb = await tokenService.findToken(refreshToken);

if (!userData || !tokenFromDb) {

throw ApiError.unauthorizedError();

}

const user = await User.findByPk(userData.id)

const userDto = new UserDto(user);

const tokens = tokenService.generateTokens({...userDto});

await tokenService.saveToken(userDto.id, tokens.refreshToken);

return {...tokens, user: userDto}

}

}

module.exports = new UserService()

# Приложение Н. Таблица экономических показателей

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 2,4 |
| Количество программистов, чел. | 1 |
| Основная заработная плата, руб. | 4560 |
| Дополнительная заработная плата, руб. | 570 |
| Отчисления в Фонд социальной защиты населения и по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний, руб. | 1774,98 |
| Расходы на материалы, руб. | 44,46 |
| Расходы на оплату машинного времени, руб. | 69,6 |
| Прочие прямые затраты, руб. | 912 |
| Накладные расходы, руб. | 912 |
| Себестоимость разработки программного средства, руб. | 8 843,04 |
| Расходы на сопровождение и адаптацию, руб. | 1503,32 |
| Полная себестоимость, руб. | 10346,36 |
| Годовые денежные поступления от продажи подписки, руб. | 5 249,92 |
| Срок окупаемости, лет | 1,97 |
| Количество покупателей для окупаемости, чел. | 377 |