|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение   высшего образования  **«МИРЭА– Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий (ИТ) |
|  |
| Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ) |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ИТОГОВОМУ ПРОЕКТУ** | |
| **по дисциплине** | |
| **«Технология разработки программных проектов»**  **Тема: «Мобильное приложение – VisPlanner»** | |
|  | |
| Выполнил студент группы ИКБО 33-21 | Горбан В.А |
| Принял | Миронов А.И. |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2023г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2023г. | *(подпись руководителя)* |

Москва 2023

**СОДЕРЖАНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ТЕРМИНОВ 3](#_Toc135318604)

[ВВЕДЕНИЕ 4](#_Toc135318605)

[1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 6](#_Toc135318606)

[1.1 Анализ предметной области 6](#_Toc135318607)

[1.2 Обзор аналогичных проектов 6](#_Toc135318608)

[1.3 Выбор инструментальных средств моделирования 6](#_Toc135318609)

[2 ПРОЕКТНАЯ ЧАСТЬ 10](#_Toc135318612)

[2.1 Определение пользовательских требований 10](#_Toc135318613)

[2.2 Определение функциональных требований 12](#_Toc135318614)

[2.3 Выбор технологий разработки 12](#_Toc135318615)

[2.4 Архитектура программной системы 12](#_Toc135318616)

[2.4.1 Пользовательский интерфейс………………………………………..12](#_Toc135318617)

[2.4.2 Базы данных…………………………………………………………..13](#_Toc135318618)

[3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 14](#_Toc135318619)

[3.1 Описание клиентской части разрабатываемого приложения 14](#_Toc135318620)

4 СИСТЕМА СБОРКИ ПРОЕКТА……………………………………………...25

[4.1 Описание сборки проекта 25](#_Toc135318620)

5 ЗАГРУЗКА ПРОЕКТА НА ГИТХАБ………………………………………...26

[5.1 Описание процесса работы гит репозитория 26](#_Toc135318620)

6 АРХИТЕКТУРА ПРИЛОЖЕНИЯ С УКАЗАНИЕМ СТЕКА ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ…………………………………………...30

[6.1 Обосновать выбранный стек технологий 30](#_Toc135318620)

[ЗАКЛЮЧЕНИЕ 31](#_Toc135318621)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc135318622)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ТЕРМИНОВ

В настоящем отчете применяют следующие сокращения:

|  |  |
| --- | --- |
| ПК | – Персональный компьютер |
| Java | – строго типизированный объектно-ориентированный язык программирования общего назначения |
| ЯП | – язык программирования |
| IDE | – это приложение, которое объединяет такие возможности, как редактирование, создание, тестирование и упаковка программного обеспечения |
| Wireframe  СУБД | – это схематичное изображение, набросок пользовательского интерфейса  – это набор программ, которые управляют структурой БД и контролируют доступ к данным, хранящимся в БД. |

# ВВЕДЕНИЕ

Многие студенты сталкиваются с тем, что им трудно распределять свое время для успешного выполнения всех поставленных учебным планом задач. В таких случаях они прибегают к использованию различных способов планирования, например: использование мобильных приложений планировщиков, выписывание всех задач в ежедневник и т.д.

Планирование – очень важный этап для правильного распределения сил и времени студента, так как большая нагрузка может приводить к переутомлению, выгоранию и потере интереса к тому, чем они занимаются. Первостепенная задача планирования заключается в том, чтобы оптимизировать время. Оно также делает нас более дисциплинированными и эффективными.

На данный момент представлено много возможностей, начиная от бумажных ежедневников, заканчивая полноценными приложения для планирования, но мало из них подходит именно для студентов.

Цель курсовой работы: разработать мобильное приложение – для планирования средствами языка Java.

Для достижения поставленной цели необходимо в заданной последовательности решить определённый перечень задач.

Задачи курсовой работы:

1. провести анализ предметной области разрабатываемого мобильного приложения;
2. проанализировать существующие аналогичные проекты разрабатываемого приложения;
3. обосновать выбор технологий разработки мобильного приложения;
4. создать страницы мобильного приложения и наполненить их тематической информацией;
5. описать архитектуру программной системы (пользовательский интерфейс, базы данных);
6. привести описание клиентской части разрабатываемого приложения;
7. иллюстрирование основных разделов.

Структура пояснительной записки: введение, 10 пунктов, заключение, список из 10 используемых источников.

# 1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

## 1.1 Анализ предметной области

Целью данной работы является разработка мобильного приложения для планирования, в частности для удобного использования студентами. В рамках текущей работы предлагается следующее решение поставленной задачи: приложение, в функционал которого входит учет не только возможность выписывать важные для выполнения задачи, но и внесение туда такой информации как: расписание сессии, список дисциплин и список всех практических работ по каждой дисциплине со всей необходимой информацией. Также было важно подобрать правильное оформление, которое настраивало бы на учебу и не перегружала бы визуальное восприятие, т.е. была бы достаточно минималистичной.

## 1.2 Обзор аналогичных проектов

Проведен анализ уже используемых приложений для платформы Android, созданных для планирования. Для анализа было выбрано несколько наиболее популярных приложений: Microsoft To do: Lists&Tasks (рисунок 1), Ежедневник, список дел, планер(рисунок 2), Список дел: планер, напоминания(рисунок 3)].

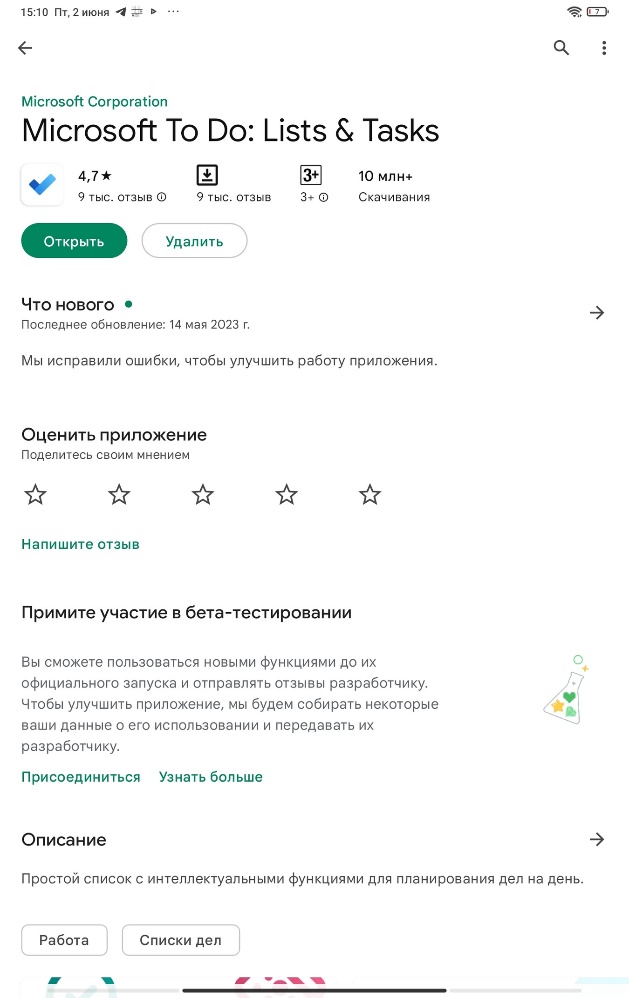


Рисунок 1 – Страница Microsoft To Do: Lists&Tasks

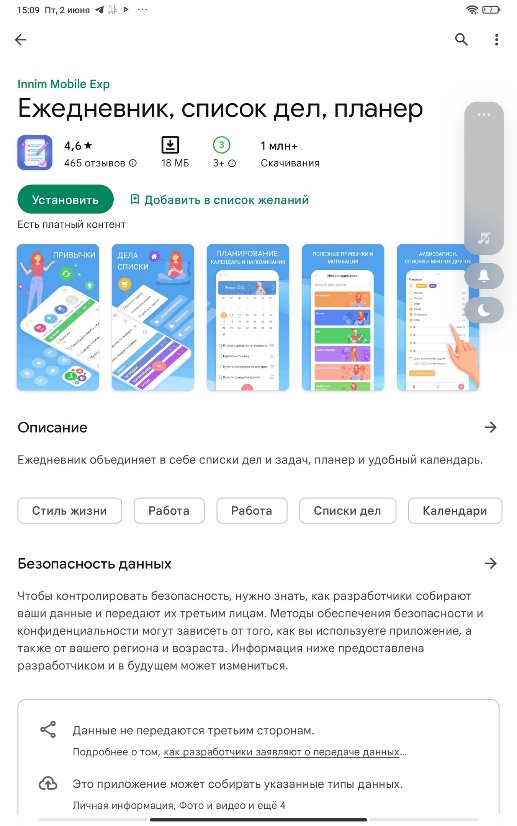


Рисунок 2 – Страница Ежедневник, список дел, планер

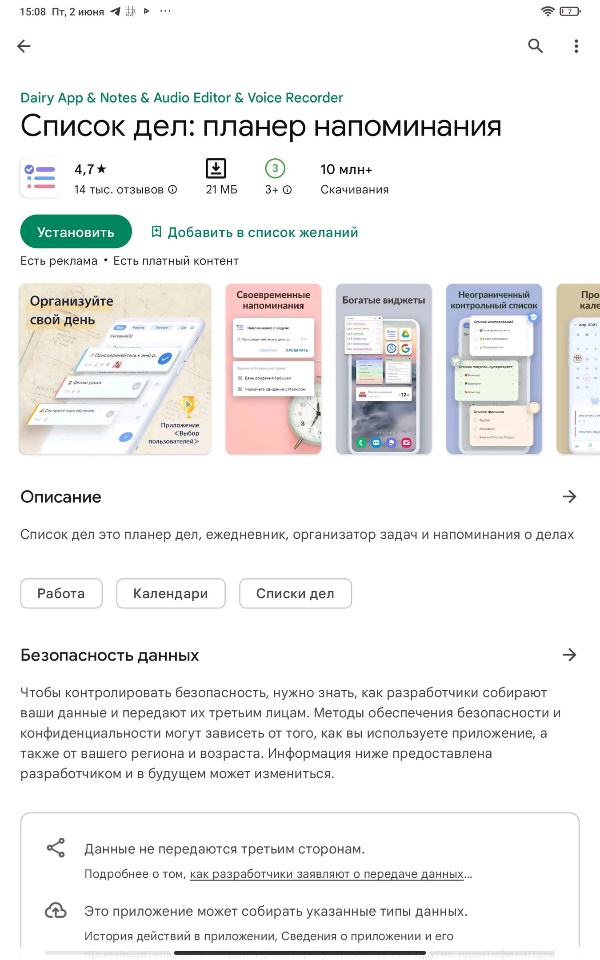


Рисунок 3 – Страница Список дел: планер напоминания

Сравнительный анализ проводился по следующим основным критериям:

- дизайн,

- удобство использования,

- функциональность,

- производительность и стабильность работы,

- достаточность характеристик комплектующих.

Результаты сравнения аналогов по 10-бальной шкале приведены в таблице 1.

Таблица 1 – Характеристики аналогов по 10-бальной системе

|  |  |  |  |
| --- | --- | --- | --- |
| Сравнительные показатели | Microsoft To Do: Lists&Tasks | Ежедневник, список дел, планер | Список дел: планер напоминания |
| Дизайн | 7 | 6 | 9 |
| Удобство использования | 6 | 6 | 7 |
| Функциональность | 7 | 7 | 8 |
| Содержание | 8 | 10 | 9 |
| Структура и навигация | 8 | 6 | 10 |

Основываясь на приведенных данных, в разработке приложения будет больший упор на хороший дизайн и удобное взаимодействие с приложением, на наличие всех необходимых списков для планирования.

# 2 ПРОЕКТНАЯ ЧАСТЬ

## 2.1 Определение пользовательских требований

Основные виды представления пользовательских требований: пользовательские истории и диаграмма вариантов использования.

Пользовательские истории представлены в таблице 2.

Таблица 2 – Пользовательские истории

|  |  |  |  |
| --- | --- | --- | --- |
| *As a/an* | *I want to* | *So that* | *To Do* |
| Студент | Иметь возможность видеть все практические работы и дедлайны к ним | Могу ввести их все в одну табллицу | Реализация удобной таблицы\списка для ввода дисциплины, названии работы и дедлайна к ней |
| Студент | Видеть расписание в удобном в виде, а не в гугл таблице со всеми другими группами | Вводить расписание в специальный блок в планнере | Реализация ввода предметов и их корректное отображение |
| Студент иностранец | Иметь возможность использовать другой язык | Могу переключить выбранный язык | Добавление внескольких языков в настройках |
| Студент | Правильно распределять свое время на работы | Получать уведомления о приближающихся дедлайнах | Реализация уведомлений от приложения |
| Студент | Заниматься дополнитеьно в свободное время и не забывать о том, что нужно взять с собой | Получать уведомления со списком нужных вещей | Релоизация уведомлений, при приближении занятия, куда нужно что-то пинести(Напнример: спортивная форма), со списком нужных вещей |
| Тех. Поддержка | Получать обратную связь или сведения о ошибках | Иметь возможность связи пользователь-поддержка | Реализация чата оперативной помощи и функция отправки сообщений и отзывов |

## 

## 2.2 Определение функциональных требований

Функциональные требования для удобства использования приложения: создать категории списков.

Функциональные требования для использования списков: создать карточку ввода нужной информации, добавить возможность отметить выполненные задачи.

Функциональные требования для взаимодействия со списками: просмотреть имеющиеся списки, установить уведомления, удалить список.

## 2.3 Выбор технологий разработки

Для разработки мобильного приложения на Android потребуется использование ЯП Java или Kotlin. Так как в цели курсовой работы прописан язык Java, то все приложение будет написано на этом языке.

Для написания приложения необходима IDE. В данной работе будет использоваться Android Studio, так как является официальной IDE для разработки Android приложений.

## 2.4 Архитектура программной системы

### 2.4.1 Пользовательский интерфейс

Первым этапом разработки дизайна является создание ментальной карты приложения, которая представлена на рисунке 4. Ментальная карта разработана в приложение XMIND.

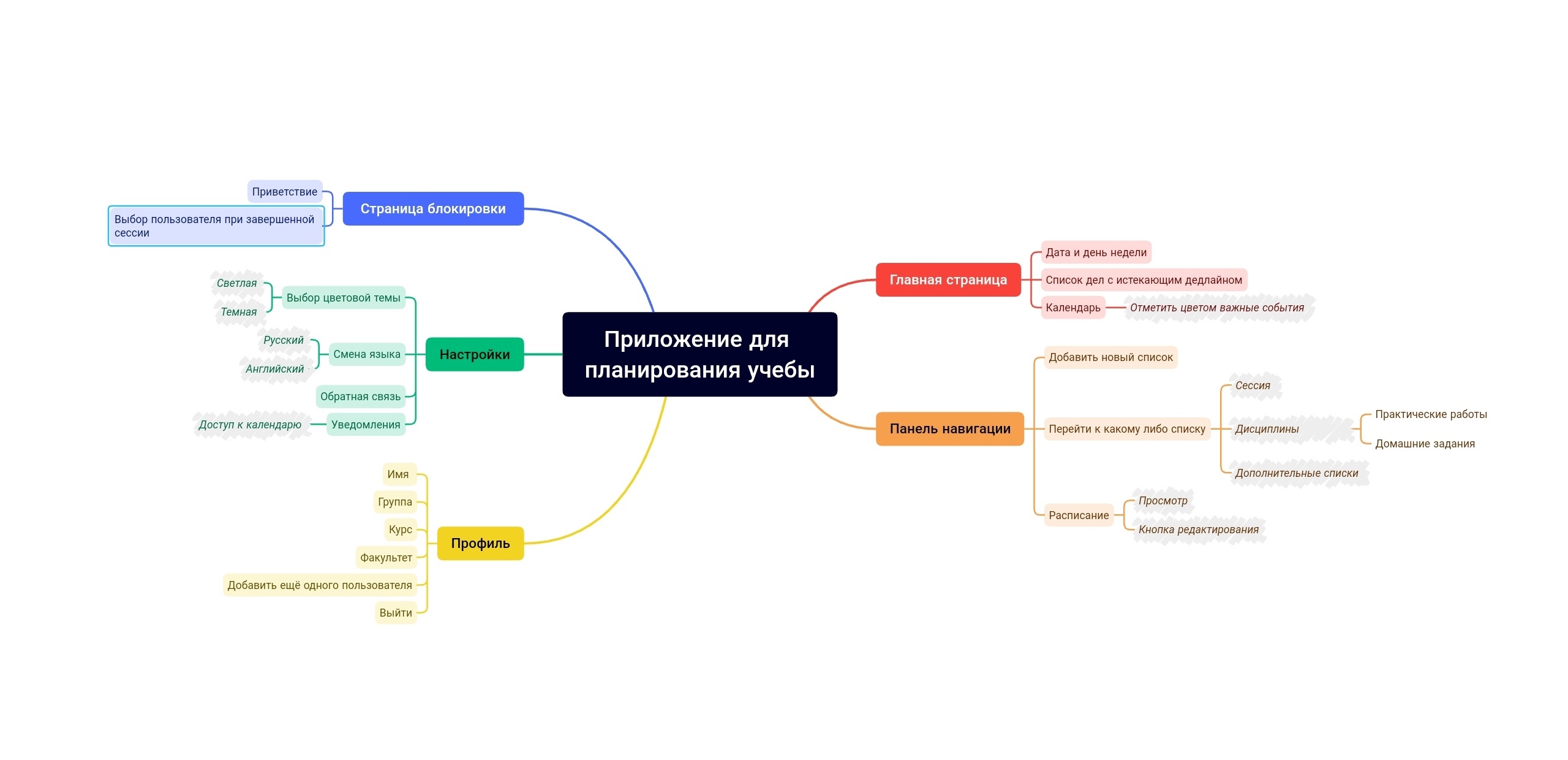


Рисунок 4 – Ментальная карта приложения

Вторым этапом является сама разработка дизайна. Для разработки пользовательского интерфейса будет использоваться бесплатный графический онлайн-редактор Figma. Макет приложения представлен на рисунке 5.

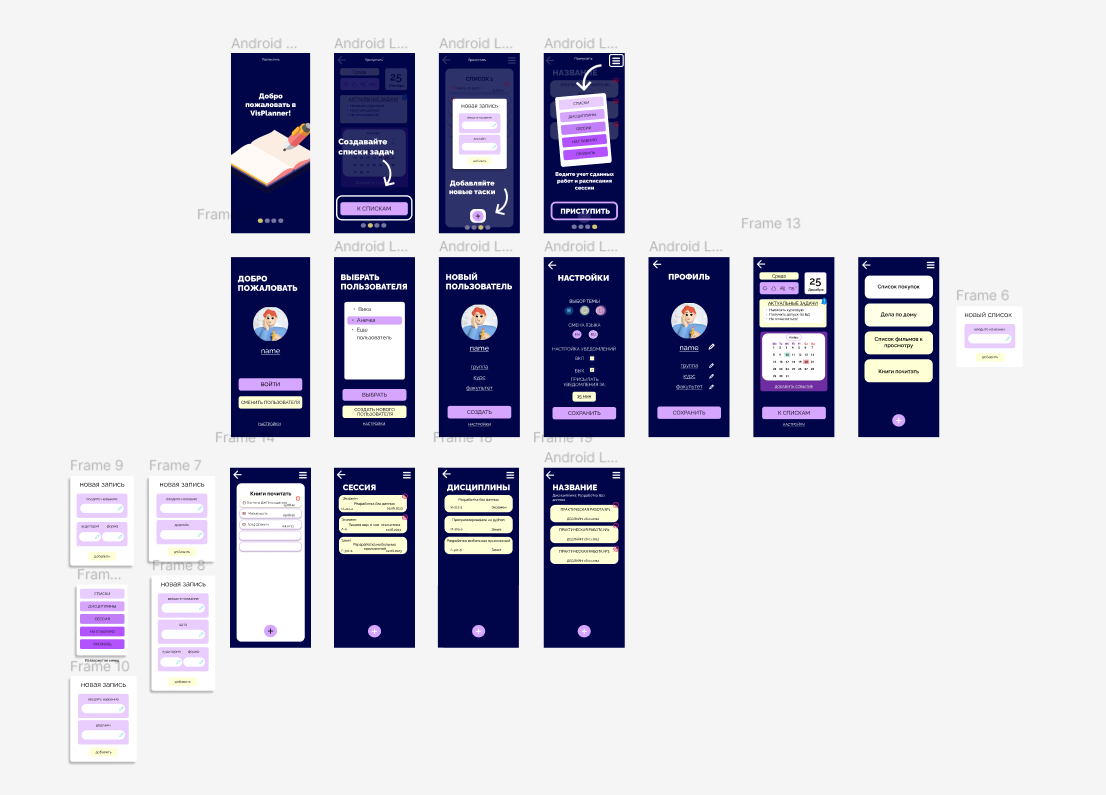


Рисунок 5 – Макет приложения

# 3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 3.1 Описание клиентской части разрабатываемого приложения

При запуске приложения пользователь попадает на стартовый экран, в котором расположены следующие элементы (рисунок 7):

1. кнопка входа в приложение;
2. кнопка перехода в раздел настроек;

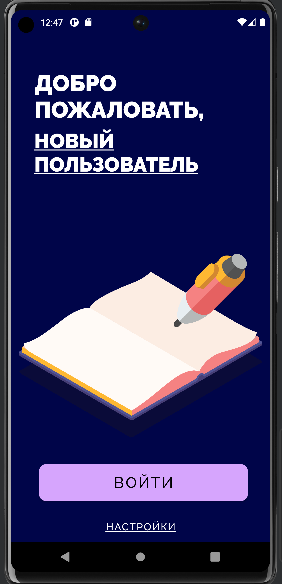


Рисунок 7 – Вход в приложение

Для перехода в приложение необходимо нажать кнопку 1, а для перехода в настройки необходимо нажать на кнопку 2. После нажатия мы переходим на главную страницу, которая содержит данные о погоде, дате, актуальных задачах и календарь (рисунок 8):



Рисунок 8 – Главная страница

Далее мы можем перейти на страницу с списками, не относящимися к учебе (рисунок 9).

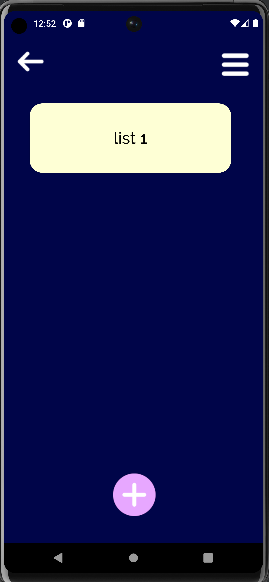


Рисунок 9 – Страница со списками

По нажатию на кнопку, появляется окно ввода для добавления нового списка (рисунок 10). После добавления списка, мы можем перейти в него по нажатию на элемент (рисунок 11).

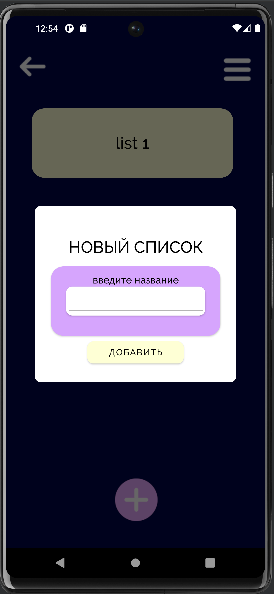


Рисунок 10 – Добавление нового списка



Рисунок 11 – Экран со списком дел

Добавление новых элементов происходит по нажатию на кнопку с изображением плюса в нижней части экрана. При нажатии на нее, появляется форма ввода (рисунок 12). Выход с данного экрана происходит по нажатию кнопки со стрелкой.

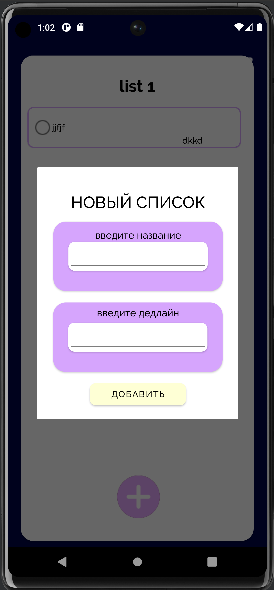


Рисунок 12 – Форма ввода новой записи

При нажатии на иконку меню в верхнем правом углу, появляется само меню, откуда мы можем перейти в любой раздел приложения (рисунок 13).



Рисунок 13 – Меню приложения

Перейдем в раздел “Дисциплины” по нажатию на соответсвующую кнопку. Здесь мы также можем добавить новую запись и перейти в список работ по дисциплине по нажатию на элемент списка (рисунки 14-15).

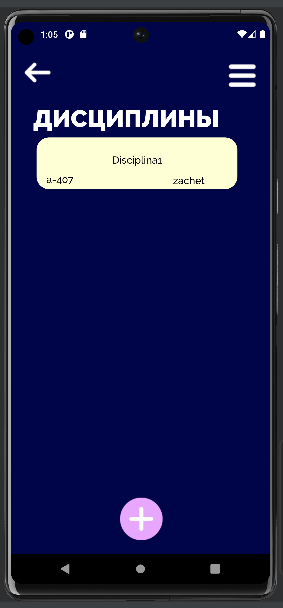


Рисунок 14 – Экран дисциплин

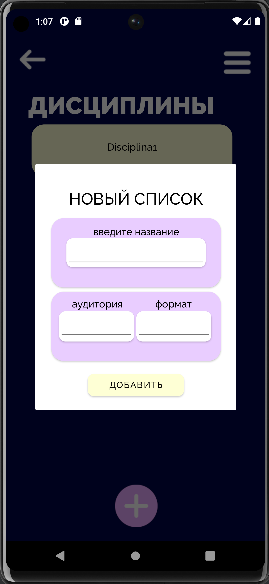


Рисунок 15 – Форма ввода новой дисциплины

Перейдя на экран с практическими работами (рисунок 16), мы видим название дисциплины и список практических работ по ней, добавление новой работы происходит по нажатию иконки плюса в нижней части экрана. Форма ввода аналогично представленным ранее.

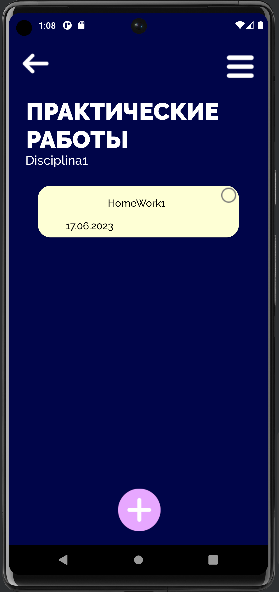


Рисунок 16 – Экран со списком практических работ

После мы можем перейти на страницу с расписанием сессии (рисунок 17). Добавление нового элемента списка происходит по нажатию на иконку плюса в нижней части экрана (рисунок 18).



Рисунок 17 – Расписание сессии

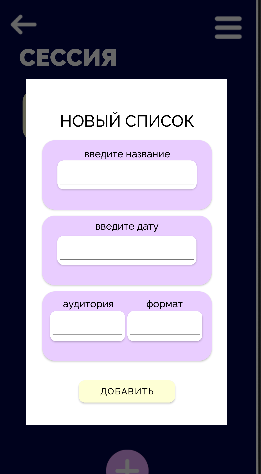


Рисунок 18 – Форма ввода новой записи

При переходе на каждую из карточек мы переходим на страницу с медитацией (рисунки 15-16).

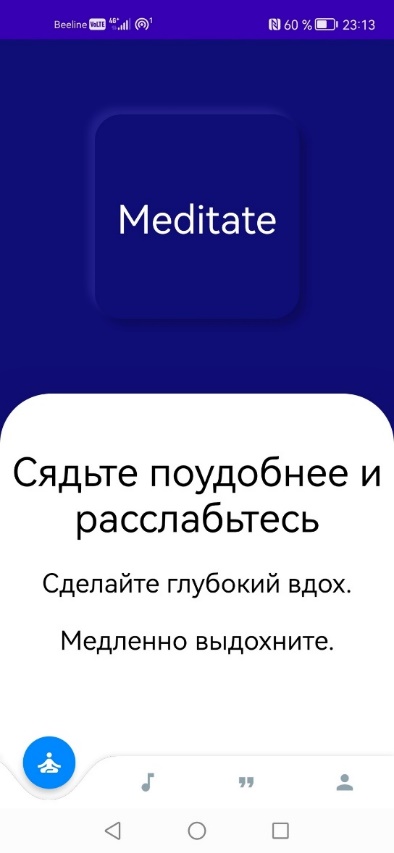


Рисунок 15 – Страница с медитацией для прослушивания

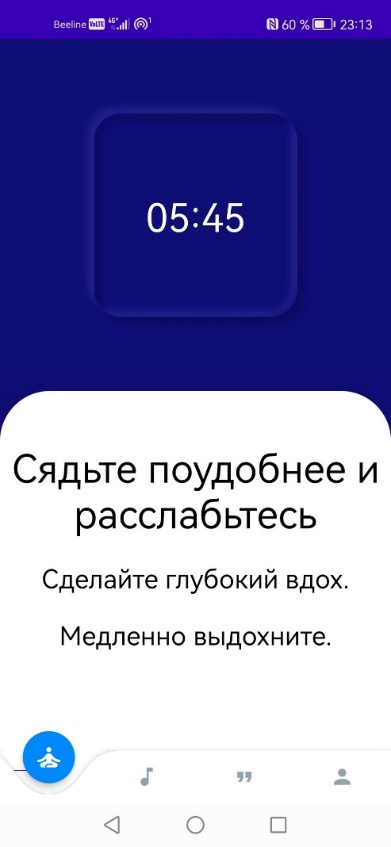


Рисунок 16 – Запуск аудиозаписи

Далее переходим в следующий раздел: “Блокнот”. Блокнот содержит в себе Quotes Generator Api. API — это заранее подготовленный шаблон инструкций по кодированию, который стороннее приложение может использовать для выполнения какой-либо функции. В частности, API-интерфейс цитат представляет собой обширную базу данных цитат. API позволяет интегрировать общую архитектуру стороннего приложения для доступа к этим строкам текста (кавычкам).

Таким образом, разработчики, которые размещают API-интерфейсы цитат в своих приложениях, позволяют своим пользователям искать и извлекать любые цитаты, которые им нравятся.

API-интерфейс генератора случайных цитат может выбирать цитаты случайным образом в попытке удивить пользователя чем-то интересным.

API-интерфейсы генератора случайных цитат и другие API, связанные с цитатами, выполняют описанные действия, а затем представляют свои выходные данные на языке, подобном JSON, который приложения могут считывать и использовать в качестве входных данных в дальнейших процессах.

Более того, генерируемые цитаты можно копировать или отправлять.

Результат работы программы представлен на рисунках 17-20.

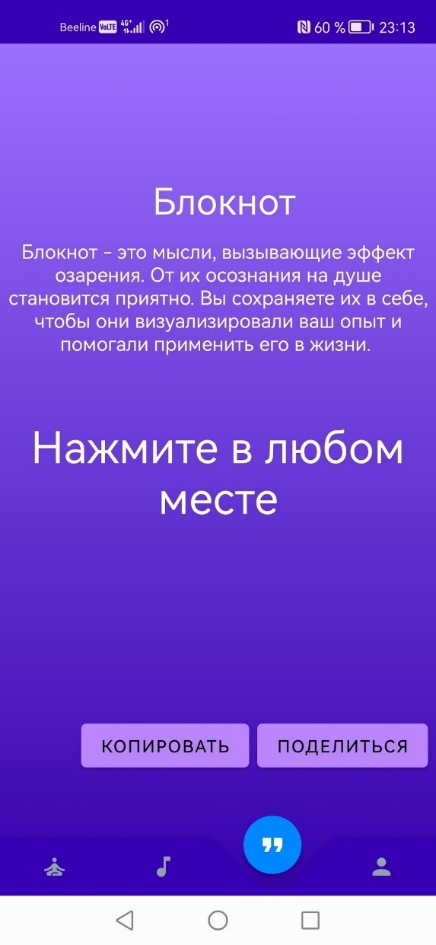


Рисунок 17 – Страница “Блокнот”



Рисунок 18 – Первая генерируемая цитата



Рисунок 19 – Копирование цитаты

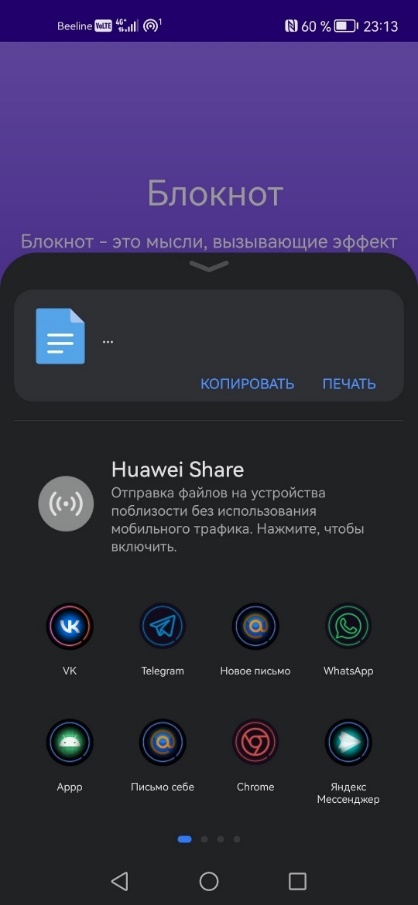


Рисунок 20 – Поделиться цитатой

Далее, переходим на страницу профиля (рисунок 21).

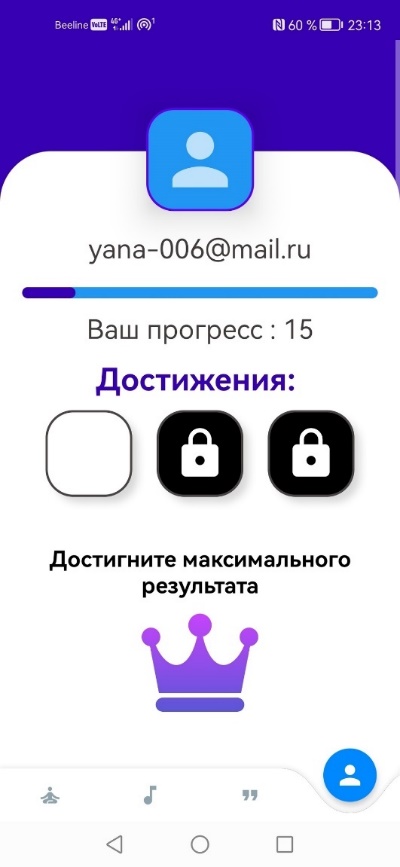


Рисунок 21 – Страница профиля

Страница профиля содержит имя пользователя и учитывает прогресс. За каждый достигнутый результат открывается достижение (рисунки 22-24).



Рисунок 22 – Достижение 1

# 4 СИСТЕМА СБОРКИ ПРОЕКТА

## 4.1 Описание сборки проекта

## Gradle — система сборки пакетов для программных продуктов, разработанных, преимущественно, на Java, Groovy и Scala. По назначению и принципу работы похож на Maven, но с более простым конфигурационным файлом (более лаконичный код Groovy или Kotlin против большого и неудобного XML).

## Проект содержит все необходимые зависимости в блоке в build.gradle, чтобы проект собирался и работал корректно. Например, implementation 'com.google.firebase:firebase-messaging:20.1.0' отвечает за обмен сообщениями, а implementation 'com.google.firebase:firebase-auth:21.3.0' необходима для аутентификации (рисунок 25).

## 

## Рисунок 25 – Зависимости в блоке в build.gradle

## Результат вывода работы проекта представлен на рисунке 26.

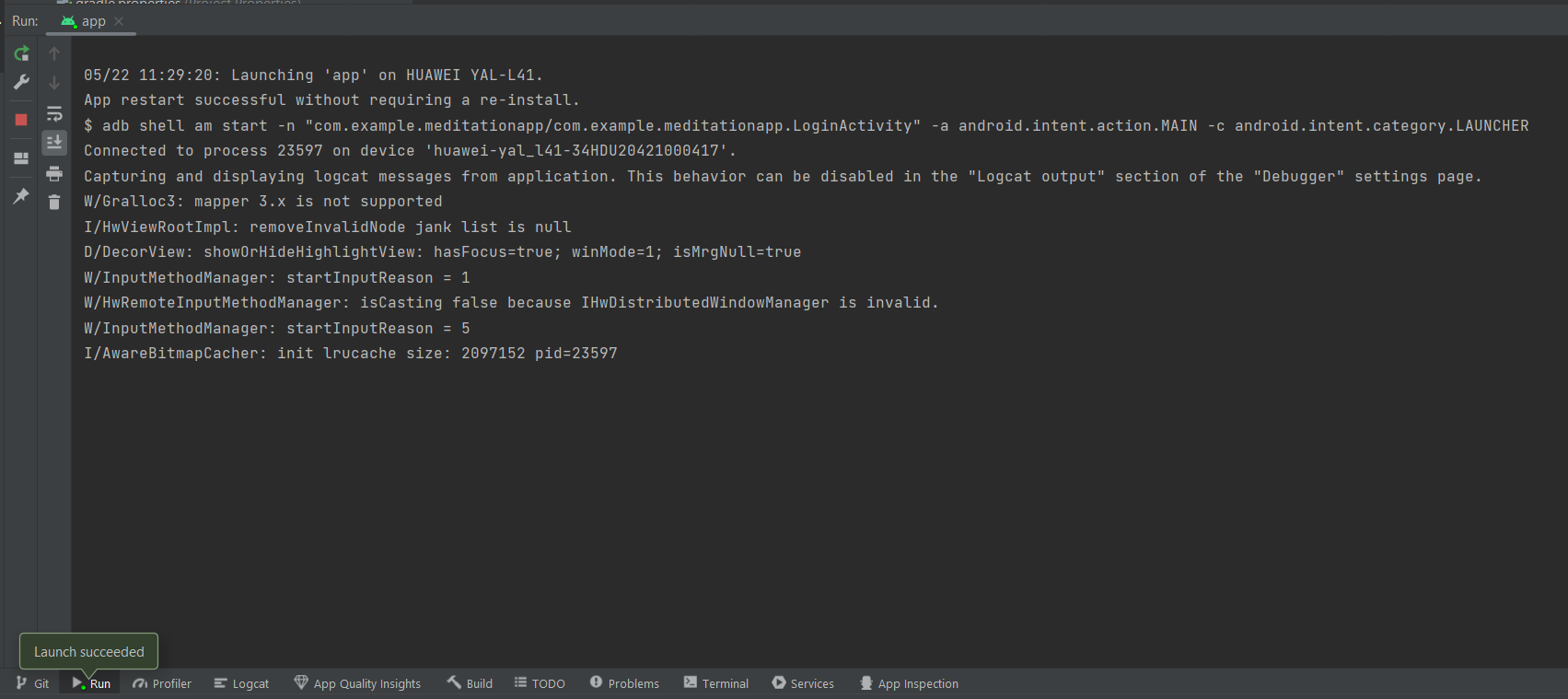


Рисунок 26 – Результат вывода правильности работы проекта

# 5 ЗАГРУЗКА ПРОЕКТА НА ГИТХАБ

## 5.1 Описание процесса работы гит репозитория

GitHub — это служба размещения в Интернете репозиториев Git, которые используются для хранения содержимого docs.microsoft.com. В GitHub размещается основной репозиторий всех проектов. Из этого репозитория участники копируют свою работу.

Для начала необходимо завести аккаунт на гитхаб. У меня аккаунт уже есть, поэтому я нажимаю New Repository. И указываю имя HeadSpace. И нажимаю Create repository (рисунок 27).

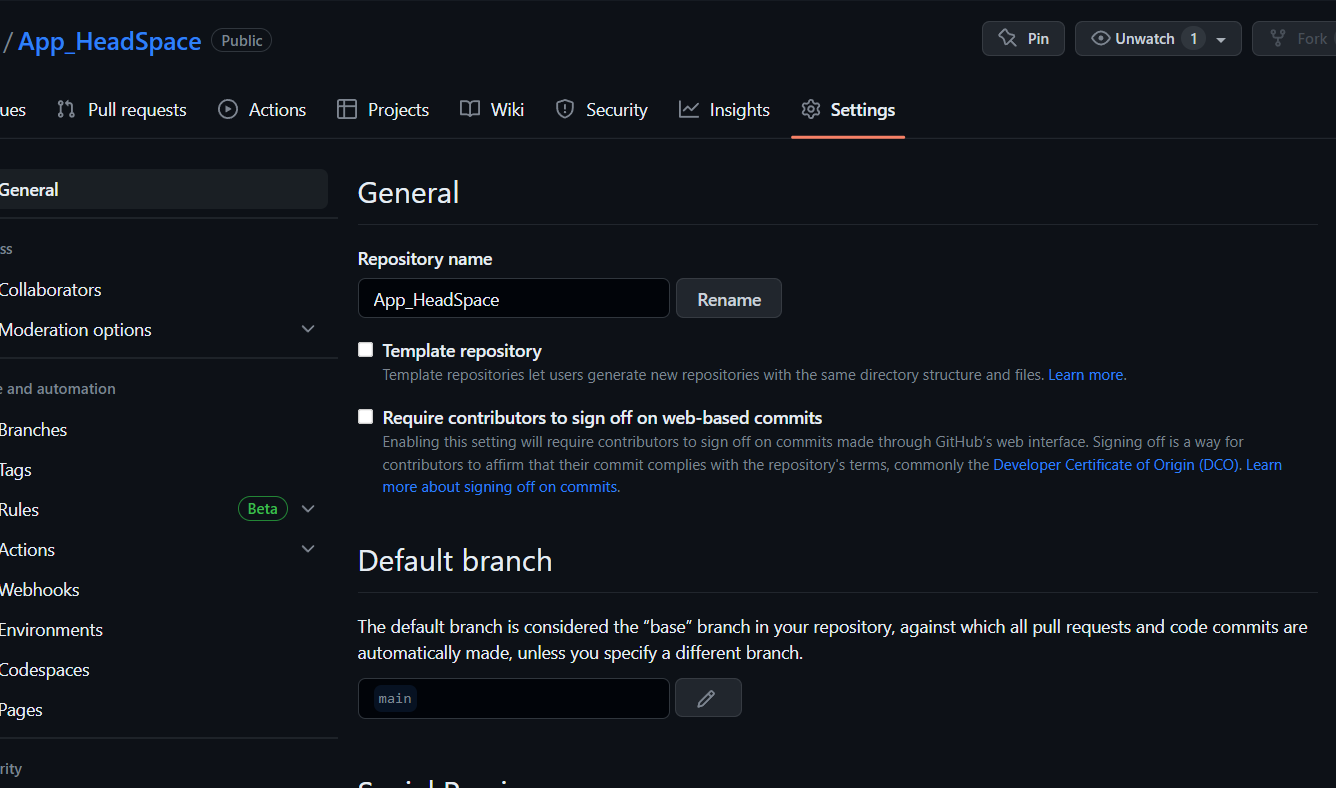


Рисунок 27 – Создание репозитория

В нашем случае настройка должна стоять на HTTS. Для этого прописываем первые строчки в консоль проекта (рисунок 28).

1. Создаем новый репозиторий Git с помощью команды git init.
2. Прописываем команду для записи индексированных изменений в репозиторий Git – git commit -m “first commit”, ult Флаг -m указывает команде git commit, что вы добавляете полезный комментарий о коде.
3. Работаем с веткой main
4. Далее добавляем по URL (назван origin) удаленный репозиторий
5. С помощью последней команды git push мы выполняем отправку недавних коммитов c компьютера локального репозитория на сервер с удаленным репозиторием.

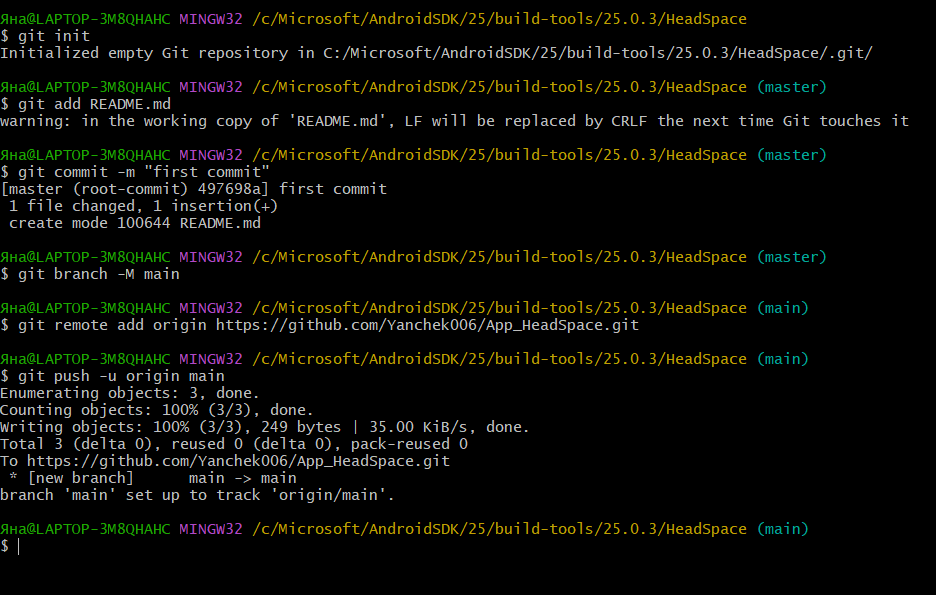


Рисунок 28 – Работа с репозиторием проекта

Чтобы работать со своего компьютера с GitHub, иметь доступ к проектам, хранящимся на сервисе, выполнять команды в консоли без постоянного подтверждения пароля, нужно пройти авторизацию у сервера. В этом помогают SSH-ключи (рисунки 29-30). Чтобы пройти авторизацию по SSH-ключу, его надо сгенерировать или найти уже ранее созданный ключ на своём компьютере.



Рисунок 29 - Создание SSH-ключа для авторизации

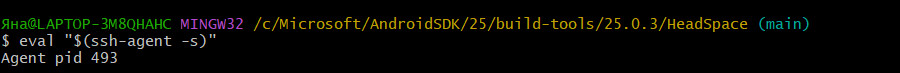


Рисунок 30 – Проверяем доступность ключа

Далее, указываем верный путь до файла с ключом и его имя (рисунок 31).

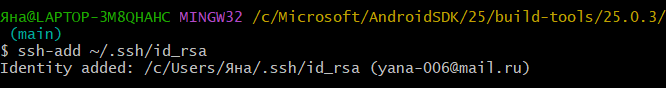


Рисунок 31 - Путь до файла с ключом и его имя

Чтобы связать свой локальный и удаленный репозиторий, выведите в консоль содержимое файла с помощью команды cat (рисунок 32):

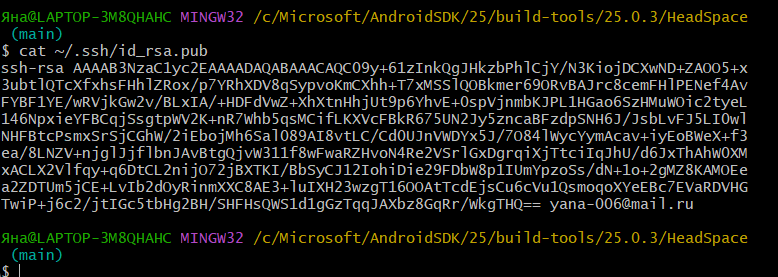


Рисунок 32 – Связываем локальный и удаленный репозиторий

Затем переходим на страницу для работы с ключами: <https://github.com/settings/keys> и добавляем ключ, который мы получили выше (рисунок 33-34).

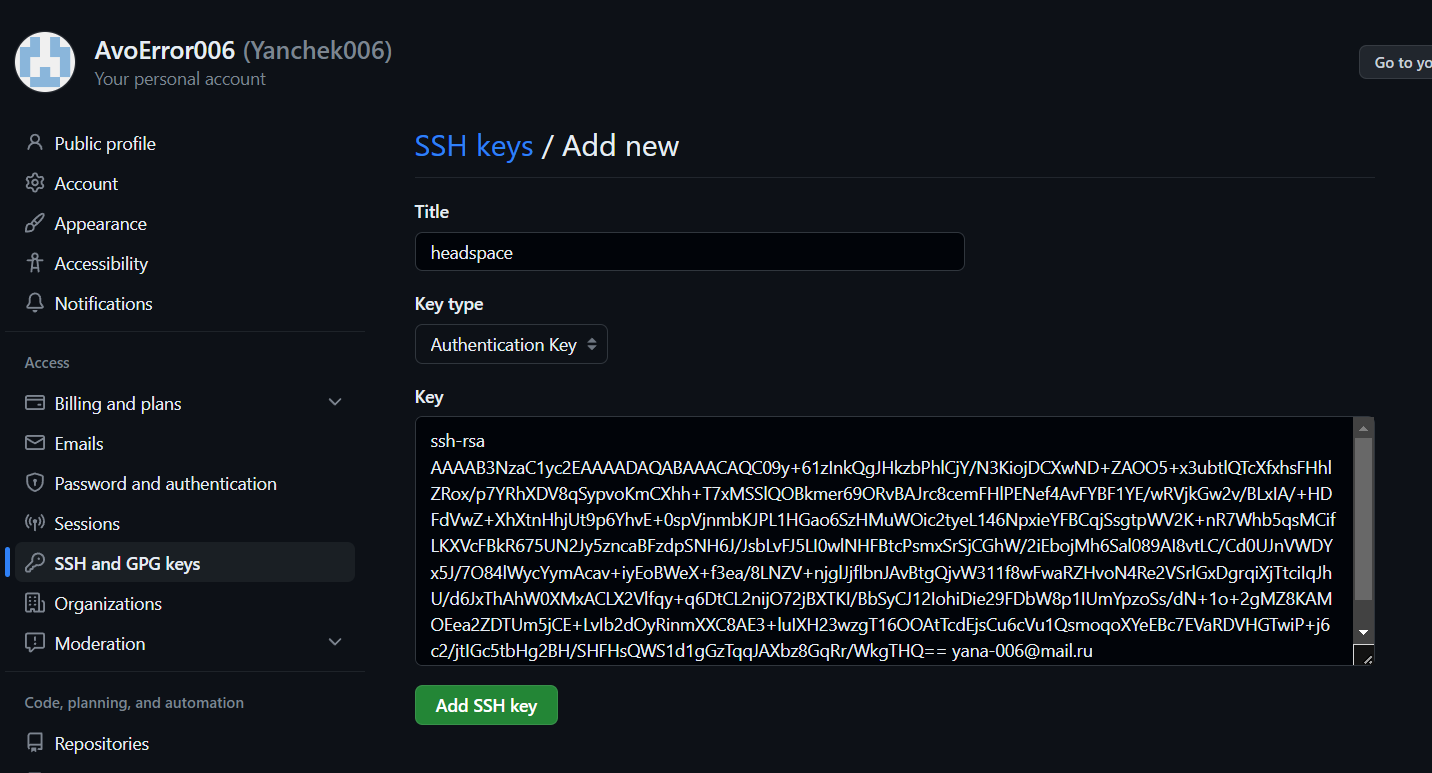


Рисунок 33 - Добавляем новый ключ

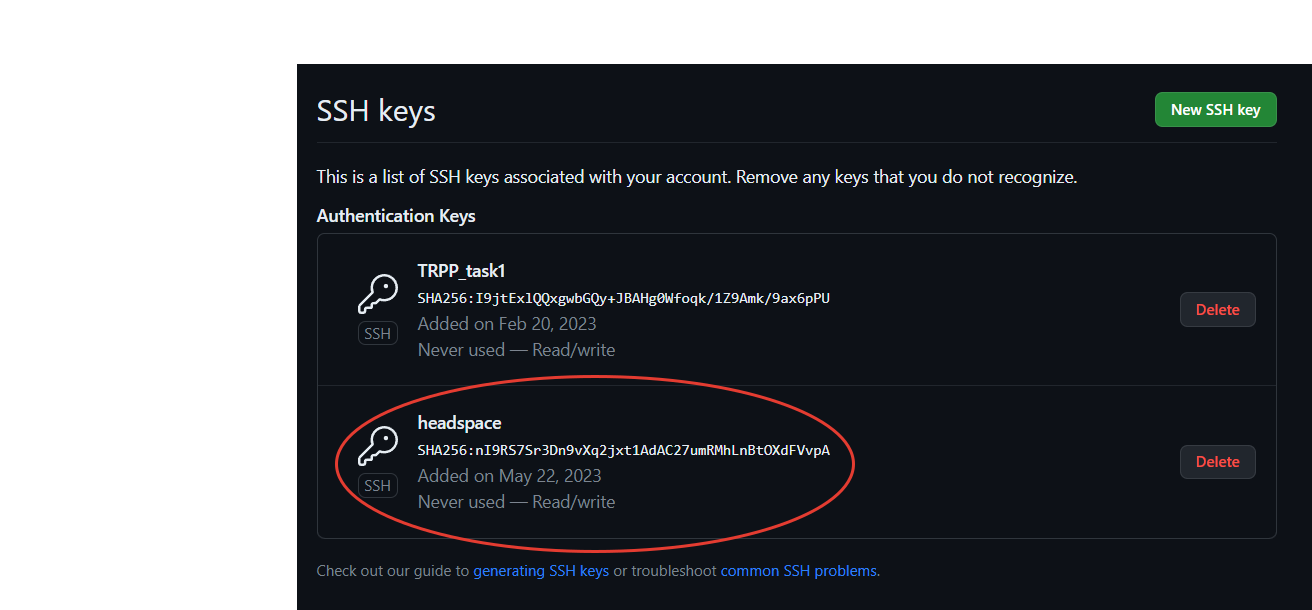


Рисунок 34 – Результат добавления SSH-ключа

Проверим наличие ключа в папке (рисунок 35).

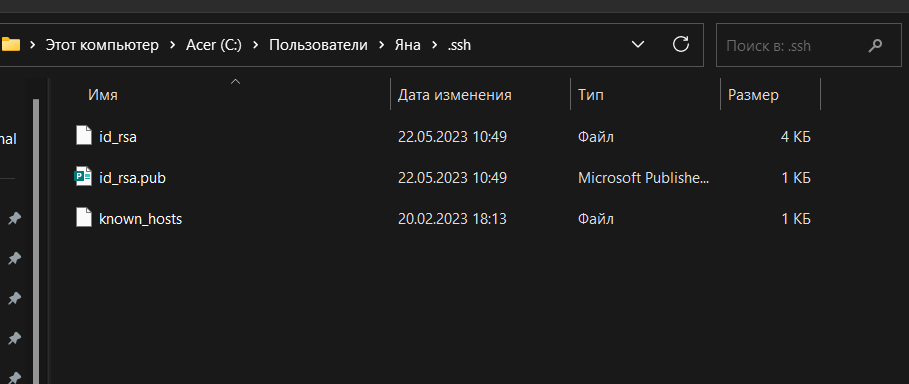


Рисунок 35 – SSH-ключ добавлен корректно

# 6 АРХИТЕКТУРА ПРИЛОЖЕНИЯ С УКАЗАНИЕМ СТЕКА ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

## 6.1 Обосновать выбранный стек технологий

## Была разработана схема архитектуры приложения с указанием стека используемых технологий (рисунок 36). В качестве системы управления базами данных была использована SQLite, поскольку она входит в состав стандартной библиотеки Android.

## В качестве языка программирования для back-end севера была использована java.

## Что касается front-end, то для него был применен XML – это язык разметки подобный HTML.

## 

# Рисунок 36 - Архитектуры приложения с указанием стека используемых технологий

# ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта были получены следующие результаты: проанализирована предметная область; выявлены плюсы и минусы существующих аналогов приложения; описан и обоснован выбор используемых технологий; приведено описание создания пользовательского интерфейса и баз данных. Была произведена сборка проекта и добавлены все необходимые зависимости. Проект был загружен на гитхаб с помощью гита, а также был добавлен SSH-ключ, чтобы иметь доступ к проектам, хранящимся на сервисе и выполнять команды в консоли без постоянного подтверждения пароля. Также была описана клиентская часть разрабатываемого приложения.

Так как все задачи решены, то была достигнута поставленная цель – разработано мобильное приложение для духовного развития, в частности для медитаций средствами языка Java.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Приложения для медитации. URL: <https://vsvoemdome.ru/it-i-tehnologii/luchshie-prilozheniya-dlya-meditacii> (дата обращения 21.05.2023).

2. Дизайн приложения для медитации: советы и примеры. – URL: <https://www.purrweb.com/ru/blog/dizajn-prilozheniya-dlya-meditacii/> (дата обращения 21.05.2023).

3. Prana Breath: Дыши и Медитируй. – URL: <https://play.google.com/store/apps/details?id=app.meditasyon> (дата обращения 21.05.2023).

4. Meditopia: Сон, Медитация. – URL: <https://play.google.com/store/apps/details?id=com.abdula.pranabreath> (дата обращения 21.05.2023).

5. Prosto: Медитация и Сон. – URL: <https://play.google.com/store/apps/details?id=com.prostoapp> (дата обращения 21.05.2023).

7. Android Studio. – URL: <https://developer.android.com/studio> (дата обращения 21.05.2023).

8. XMIND. – URL: <https://xmind.app/> (дата обращения 08.05.2023).

9. Figma. – URL: <https://www.figma.com/> (дата обращения 08.05.2023).

10. DB Browser for SQLite. – URL: <https://sqlitebrowser.org/dl/> (дата обращения 21.05.2023).