МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка Android-приложения для подготовки к ЕГЭ по информатике на Kotlin

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-363.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Н.С. Силкина

Автор работы,
студент группы КЭ-403
_____ Н.А. Мишунин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студенту группы КЭ-403
Мишунину Никите Александровичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка Android-приложения для подготовки к ЕГЭ по информатике на Kotlin.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Открытый банк заданий ФИПИ по ЕГЭ. [Электронный ресурс] URL: https://fipi.ru/ege/otkrytyy-bank-zadaniy-ege (дата обращения: 24.02.2024 г.).
3.2. Демонстрационная версия ЕГЭ 2023 по информатике. [Электронный ресурс] URL: https://inf-ege.sdamgia.ru/test?id=11304444 (дата обращения: 24.02.2024 г.).
3.3. Android Developers. [Электронный ресурс] URL: https://developer.android.com (дата обращения: 24.02.2024 г.).
3.4. Сомон П.И. Волшебство Kotlin: руководство / П.И. Сомон; перевод с английского А.Н. Киселева. // Москва: ДМК Пресс, 2020. – 536 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области и сделать обзор существующих приложений по данной тематике.
4.2. Спроектировать и реализовать базу данных для хранения информации приложения.

4.3. Спроектировать мобильное приложение.

4.4. Реализовать мобильное приложение.

4.5. Протестировать работу реализованного приложения.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП Н.С. Силкина

Задание принял к исполнению Н.А. Мишунин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Мобильные приложения в настоящее время становятся неотъемлемой частью жизни большинства людей. На ноябрь 2023 года мобильным телефоном пользуются 5,87 миллиарда человек – 73 % мирового населения [1]. С января 2022 года количество уникальных мобильных пользователей выросло на 1,8 % (95 миллионов), в то время как общее количество мобильных подключений увеличилось на 80 миллионов (1 %) и достигло 8,2 миллиарда к началу 2023 года [2].

В это же время растет и популярность обучения и подготовки к единому государственному экзамену онлайн. Согласно предоставленным данным сайта Учи.Ру по итогам 2023 года 22% школьников готовятся к экзаменам в онлайн формате [3].

В связи с ростом этих двух направлений, спрос на мобильные приложения для подготовки к ЕГЭ достаточно высок, так как они полезны как детям, которые учатся в школе и хотят сдать ЕГЭ, для подготовки к экзамену, так и преподавателям, которые занимаются подготовкой детей к единому государственному экзамену.

Информатика является одним из самых рациональных предметов, формат ЕГЭ по которому легко сделать в виде приложения.

Постановка задачи

Задачей данной работы является разработка мобильного приложения по подготовке к ЕГЭ для платформы Android.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) проанализировать предметную область;
2) спроектировать и реализовать мобильное приложение;
3) спроектировать базу данных;
4) протестировать работу реализованного приложения.

Структура и содержание работы

Работа состоит из введения, шести глав, заключения и списка литературы. Объем работы составляет 69 страниц, объем списка литературы – 24 источника.

В первой главе описывается анализ предметной области, какие мобильные приложения для подготовки к ЕГЭ по информатике уже существуют на рынке. Также анализируются функции этих приложений, что есть в большинстве аналогов, и какого полезного функционала нет у конкурентов.

Вторая глава посвящена анализу требований к будущему приложению. В этой главе решается вопрос, каким обязательным набором функций должно обладать современное приложение для подготовки к ЕГЭ, а также составляется диаграмма вариантов использования такого продукта.

Третья глава описывает процесс проектирования мобильного приложения. Этот этап представляет собой создание дизайна мобильного приложения и проектирование структуры хранения данных.

В четвертой главе говорится о процессе реализации Android-приложения. В нем проведен обзор технологий, представлена реализация облачной базы данных для приложения и подробно описаны некоторые функции.

Пятая глава полностью отражает в себе процесс тестирования реализованного мобильного приложения. В ней подробно описывается функциональное тестирование приложения и юзабилити-тестирование, которое осуществляли потенциальные пользователи приложения.

Шестая глава описывает процесс проектирования, реализации и тестирования веб-интерфейса администратора для добавления новых и редактирования старых заданий в мобильном приложении.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

На данный момент существует небольшое количество приложений для Android-устройств, которые позволяют проходить подготовку к ЕГЭ. Функционал приложений такого вида обычно состоит из экзаменационных тестов, вопросов по отдельным тематикам и дополнительных учебных материалов.

ОГЭ, ЕГЭ информатика – программирование на Python

Мобильное приложение «ОГЭ, ЕГЭ информатика – программирование на Python» разработано Виктором Трофимовым, который является преподавателем и разработчиком на языке Python. Скриншоты приложения представлены на рисунке 1.

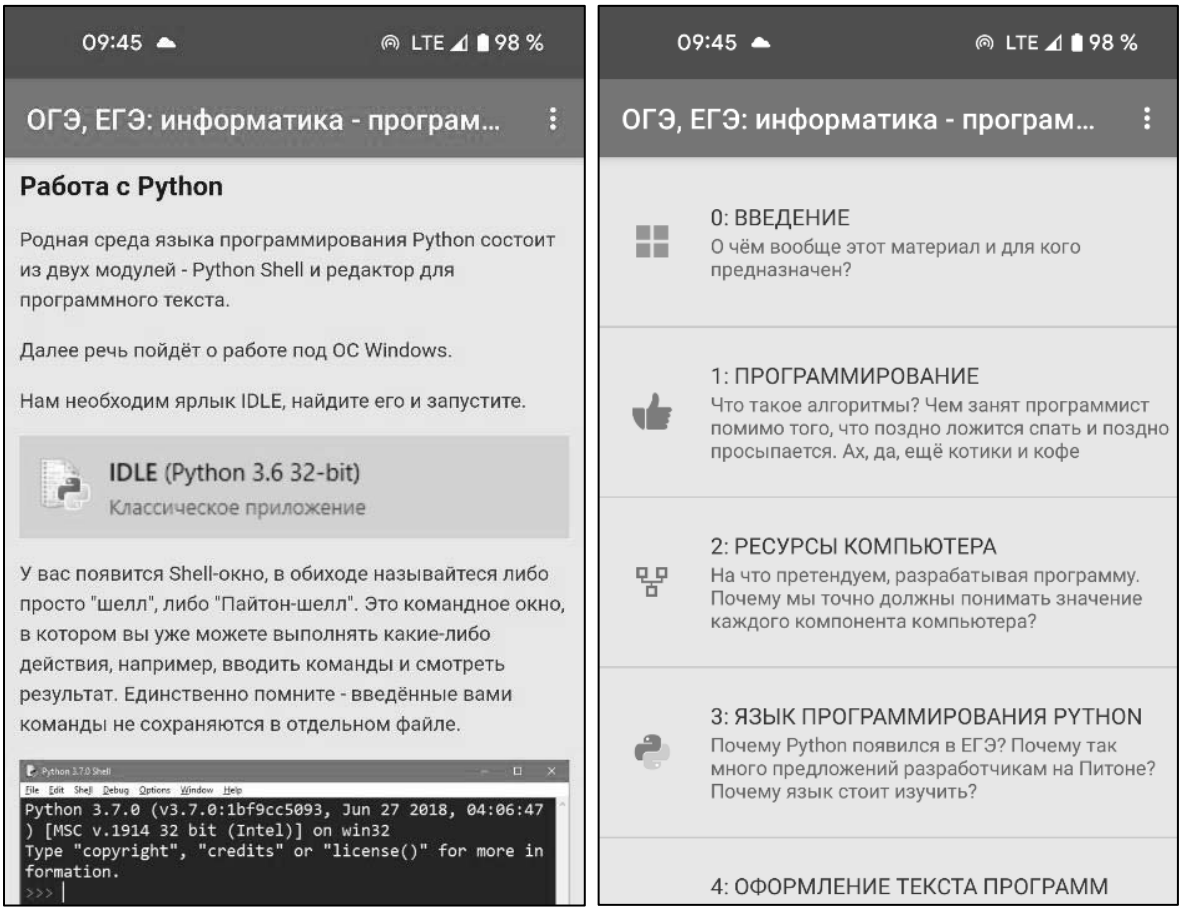

Рисунок 1 – Скриншоты приложения
«ОГЭ, ЕГЭ информатика – программирование на Python»

Данное приложение выполняет только информационную функцию. Приложение состоит из информации для подготовки к ЕГЭ и задач, но без

проверки правильных ответов. Однако сразу после раздела с задачами есть раздел с правильными ответами к ним [4]. Минимальная операционная система для установки приложения: Android 4.4.

Из недостатков можно выделить, что в приложении нет функции мгновенной проверки задач с помощью тестов, как по отдельным заданиям, так и по вариантам экзамена, из-за чего невозможно следить за статистикой подготовки. Еще минусом приложения является то, что большая часть обучающего контента заблокирована до приобретения платной версии приложения.

Информатика ЕГЭ

На рисунке 2 представлены скриншоты приложения «Информатика ЕГЭ», которое охватывает практически все ключевые функции, характерные для приложений, помогающих в подготовке к экзаменам. Для установки требуется минимальная операционная система Android 5.0. В Пользователям предоставляется возможность изучения теории к экзаменационным заданиям, а также прохождение тестов по конкретным номерам ЕГЭ. Замечательной особенностью приложения является его интуитивно понятный интерфейс. Кроме того, пользователи могут воспользоваться функцией записи на платные курсы по подготовке к ЕГЭ по информатике [5].

Из положительных особенностей данного приложения следует выделить четкое и понятное объяснение теории по каждому из экзаменационных заданий, а также полное отсутствие платного контента и раздражающей рекламы.

Однако, среди негативных моментов приложения можно выделить отсутствие возможности полного решения экзамена по информатике и отсутствие статистической информации, что может ограничить пользователей в оценке своего прогресса и эффективности подготовки. Улучшение этих аспектов могло бы значительно повысить ценность приложения для пользователей.

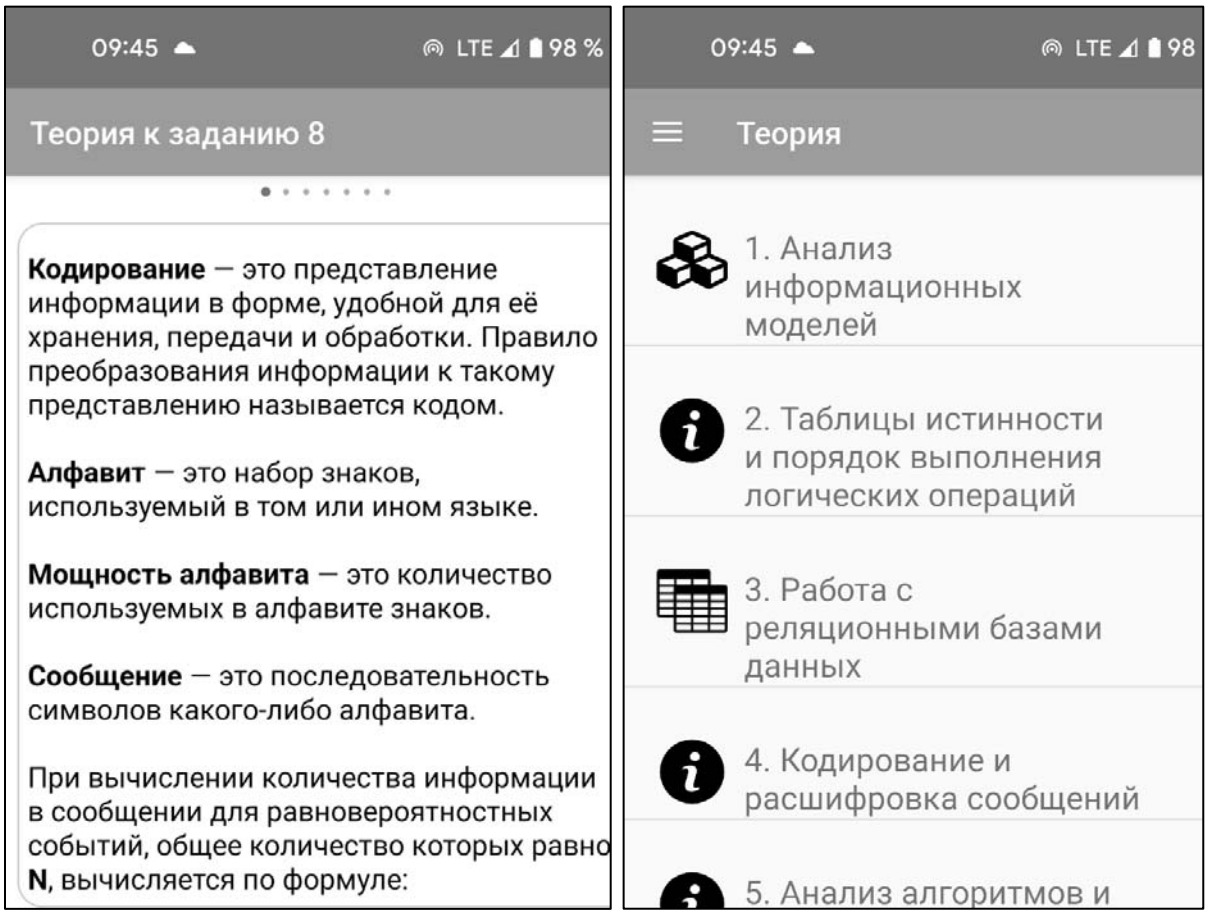

Рисунок 2 – Скриншоты приложения «Информатика ЕГЭ»

ЕГЭ по информатике 2023

Еще одно приложение для подготовки к экзамену по информатике «Подготовка к ЕГЭ по информатике 2023». Скриншоты приложения представлены на рисунке 3. Минимальная операционная система, которая нужна, чтобы установить его – Android 5.1. Приложение имеет следующие функции: прохождение неполного экзамена, а также набор тестов по определенному номеру задания в экзамене и игра, которая никак не влияет на обучение, но помогает отвлечься от подготовки на пару минут. Приложение полностью бесплатно, а также в нем отсутствует реклама. Разработчики отметили, что это лишь бета версия приложения, в дальнейшем планируется добавлять больше заданий и функций [6].

Приложение обладает рядом недостатков, которые стоит отметить. Во-первых, отсутствует достаточное количество теоретического материала

для подготовки к экзамену. Во-вторых, наблюдается избыточное использование цветов на главной странице. Восемь различных цветов, применяемых в дизайне, могут негативно влиять на общую визуальную структуру приложения. Третьим недостатком является отсутствие статистики по пройденным тестам. Наличие такой статистики предоставило бы пользователям более детальную обратную связь об их успехах и слабостях. Наконец, учитывая, что приложение все еще находится в бета-версии, обнаружено значительное количество ошибок в его реализации. Эти проблемы требуют дополнительного внимания и устранения со стороны разработчиков, потому что наличие ошибок способствует негативному пользовательскому опыту по отношению к мобильному приложению.

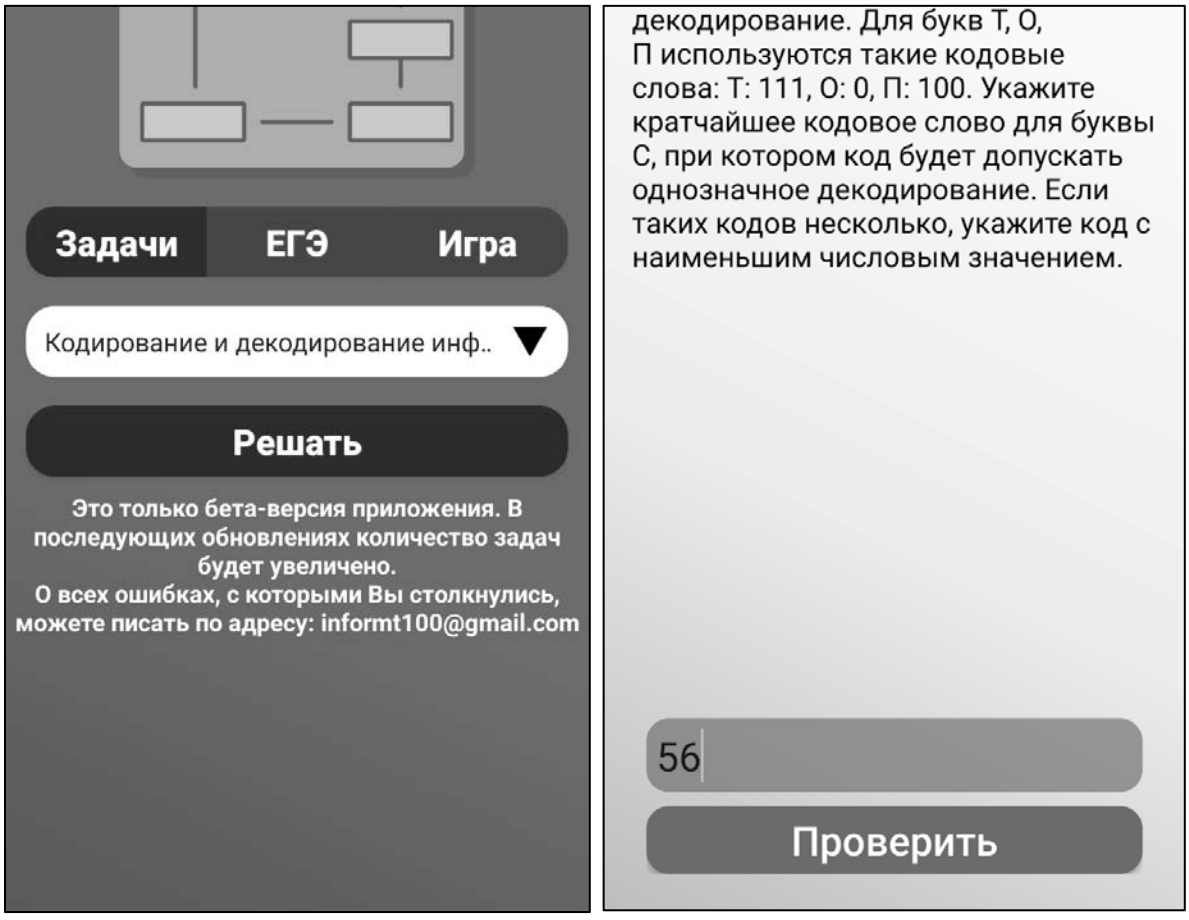

Рисунок 3 – Скриншоты приложения «ЕГЭ по информатике 2023»

Таким образом рассмотренные в анализе приложения обладают определенными аспектами, которые представлены и детально разобраны в таблице 1.

Таблица 1 – Функции приложений

Критерии	**ОГЭ, ЕГЭ информатика – программирование на Python**	**Информатика ЕГЭ**	**ЕГЭ по информатике 2023**
Минимальная операционная система	Android 4.4	Android 5.0	Android 5.1
Дополнительные материалы для подготовки	+	–	–
Тесты по заданиям	–	+	+
Тест полного экзамена	–	–	+
Статистика	–	–	–

При проведении анализа аналогичных приложений, становится ясным, что в разрабатываемом приложении следует уделить внимание расширению образовательных ресурсов для подготовки к ЕГЭ по информатике. На основе опыта предыдущих платформ, важным дополнением будет наличие разнообразных материалов, охватывающих различные аспекты предмета.

Кроме того, разрабатываемое приложение планируется обогатить тестами, охватывающими как отдельные типы заданий, так и полные экзаменационные тесты. Это позволит пользователям систематически оценивать свои знания и подготовленность, а также целенаправленно тренироваться в решении конкретных типов задач.

Дополнительным функционалом, который станет важным дополнением, будет возможность просмотра статистики. Пользователи смогут отслеживать свой прогресс в изучении материалов, анализировать результаты тестов, выявлять слабые места и фокусироваться на них в дальнейшей подготовке. Такой подход обеспечит более эффективное использование приложения в образовательных целях.

Выводы по первой главе

В данной главе был проведен анализ предметной области. Также были выделены функции, которыми должно обладать будущее Android-приложение.

2. АНАЛИЗ ТРЕБОВАНИЙ

В ходе анализа требований к мобильному приложения по подготовке к ЕГЭ по информатике были определены функциональные и нефункциональные требования к разрабатываемой системе, сформированы варианты использования системы.

К функциональным требованиям относятся те, которые описывают требуемое поведения системы в определенных условиях [7].

Нефункциональные требования являются ограничениями, которые налагают на систему и определяют ее атрибуты качества [7].

2.1. Определение требований

В результате анализа предметной области и обзора существующих решений были сформированы следующие функциональные требования:

1) система должна обеспечивать возможность прохождения пользователем экзамена, состоящих из заданий открытого банка ФИПИ;

2) система должна обеспечивать возможность прохождения тестов по конкретным заданиям из ЕГЭ по информатике;

3) система должна обеспечивать возможность просмотра пользователем материалов для подготовки к ЕГЭ по информатике;

4) система должна обеспечивать возможность пользователю просматривать правильный ответ и содержать разбор решения задания;

5) система должна сохранять результаты пользователя и вести статистику по подготовке к ЕГЭ.

В процессе проектирования системы было акцентировано внимание на определении нефункциональных требований, которые представляют собой важный элемент общей функциональности Android-приложения для подготовки к ЕГЭ по информатике. Разработка этих нефункциональных требований направлена на обеспечение высокой производительности, безопасности и удобства использования системы. Сформированы следующие нефункциональные требования для Android-приложения:

1) система должна обладать важным аспектом, который необходимо учесть в разрабатываемой системе – способность адаптировать пользовательский интерфейс приложения под разнообразные размеры экранов;

2) система должна использовать облачную базу данных Firebase [8] для хранения информации;

3) система должна поддерживать операционную систему Android не ниже версии 5.1.

2.2. Варианты использования системы

Для проектирования мобильного приложения был использован язык графического описания для объектного моделирования UML [9]. На рисунке 4 представлена диаграмма вариантов использования приложения для обучения подготовке к ЕГЭ по информатике. С приложением взаимодействуют два актера: авторизованный пользователь и не авторизованный пользователь.

Для взаимодействия с функциями приложения, пользователю необходимо авторизоваться. Рассмотрим варианты использования.

1. Пройти симуляцию ЕГЭ по информатике – авторизованный пользователь может решить экзамен по информатике; после ответа на последний вопрос пользователь получает результат прохождения экзамена, где будет указано количество первичных баллов, которое он получил за решение теста ЕГЭ.

2. Посмотреть статистику подготовки к экзамену – авторизованный пользователь может посмотреть статистику прохождения тестов ЕГЭ по информатике, в статистике отражаются баллы и время, которые пользователь получил за попытки прохождения тестов ЕГЭ по информатике в приложении.

3. Посмотреть теоретические материалы – авторизованный пользователь может просмотреть дополнительные теоретические материалы для подготовки к заданиям ЕГЭ по информатике.

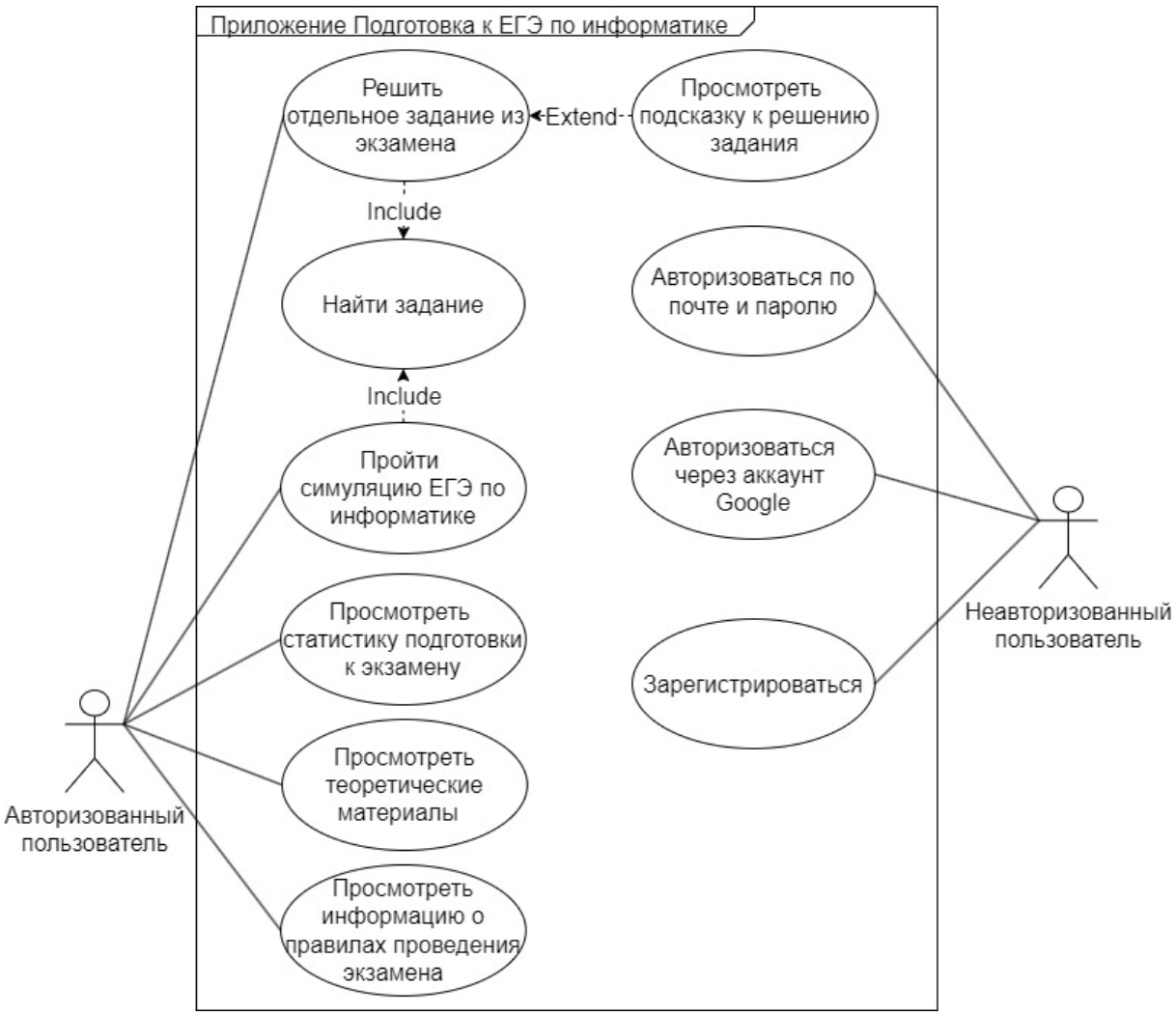

Рисунок 4 – Диаграмма вариантов использования

4. Просмотреть подсказку к решению задания – во время решения отдельных заданий, пользователь может перейти в специальную активность, где показана подсказка к текущему заданию, информация для которой взята с сайта «Решу ЕГЭ» [10].

5. Решить отдельное задание экзамена – пользователь может решить каждое задание из ЕГЭ по отдельности. Выбрать номер задачи и количество заданий с этим номером.

6. Найти задание – вспомогательная функция приложения, которая позволяет загружать задание из базы данных и отображать его перед авторизованным пользователем во время решения экзамена или отдельного задания ЕГЭ по информатике.

7. Просмотреть информацию о правилах проведения экзамена – авторизованный пользователь может увидеть таблицу для перевода первичных баллов во вторичные по предмету информатика, а также прочитать правила проведения экзамена по информатике.

8. Авторизоваться по почте и паролю – неавторизованный пользователь может пройти процедуру аутентификации по почте и паролю. Аутентификацию можно пройти только, если у неавторизованного пользователя имеется зарегистрированный аккаунт в приложении. После прохождения процедуры аутентификации пользователю становятся доступный основные функции приложения.

9. Авторизоваться через аккаунт Google – неавторизованный пользователь может пройти процедуру аутентификации при помощи аккаунта в сервисе Google. Аутентификацию можно пройти только, если у неавторизованного пользователя имеется зарегистрированный аккаунт Google. После прохождения процедуры аутентификации пользователю становятся доступный основные функции приложения.

10. Зарегистрироваться – неавторизованный пользователь может зарегистрироваться новый аккаунт в приложении. После регистрации нового аккаунта пользователь становится авторизованным и может пользоваться основными функциями приложения.

Выводы по второй главе

В данной главе сформулированы функциональные и нефункциональные требования к разрабатываемому Android-приложению. Были описаны функции, которые будут предоставлены пользователям в процессе использования приложения. Это позволяет задать вектор для дальнейшей разработки приложения.

3. ПРОЕКТИРОВАНИЕ

3.1. Дизайн приложения

При проектировании дизайна приложения необходимо учитывать следующие моменты.

1. Цель приложения заключается в том, чтобы предоставить пользователям инструмент для эффективной подготовки к ЕГЭ по информатике. Необходимо учитывать, какие именно задачи пользователи стремятся решить при изучении материала и решении тестов, и обеспечить им удобство использования приложения в этом процессе. Дизайн приложения должен способствовать быстрому и легкому достижению целей пользователей в подготовке к экзамену.

2. Простота и удобство использования играют ключевую роль. Чем более простое и интуитивно понятное приложение, тем выше вероятность, что пользователи будут готовиться к экзаменам именно с ним. Также стоит отметить важность удобного расположения всех элементов приложения, их хорошей видимости и простоты доступа для пользователей.

3. Палитра цветов. Оттенки могут вызывать разнообразные эмоциональные реакции у пользователей. Важно подбирать цвета таким образом, чтобы они соответствовали желаемой атмосфере и настроению приложения для пользователей.

4. Форма элементов. Внешний вид элементов должен соответствовать их функциональности. Например, кнопки навигации или вызова должны быть явно выделены среди прочих элементов, а текст должно быть хорошо виден и читаться.

5. Ясность и информативность. Информация должна быть доступной и наглядной. Важно не сокращать информацию важных элементов, так как это станет ключевым успехом интуитивно понятного интерфейса для приложения.

Поэтому при разработке дизайна приложения важно учитывать не только его внешний вид, но и функциональные аспекты, чтобы создать приложение, которое не только эстетично, но и удобно и функционально для пользователей.

Проектирование интерфейса мобильного приложения было выполнено с использованием средств разработки в Figma [11], поскольку оно предоставляет все необходимые инструменты для создания качественного дизайна интерфейса.

Цветовая гамма была выбрана в теплых пастельных тонах зеленого и бежевого. Выбор теплых пастельных оттенков зеленого и бежевого создает атмосферу уюта и спокойствия. Эти цвета хорошо сочетаются между собой, придавая интерьеру нежность и гармонию. В такой цветовой гамме помещение выглядит свежо и естественно, что способствует созданию приятной атмосферы для отдыха и работы. Вся палитра цветов приложения представлена в цветовой модели HEX на рисунке 5.

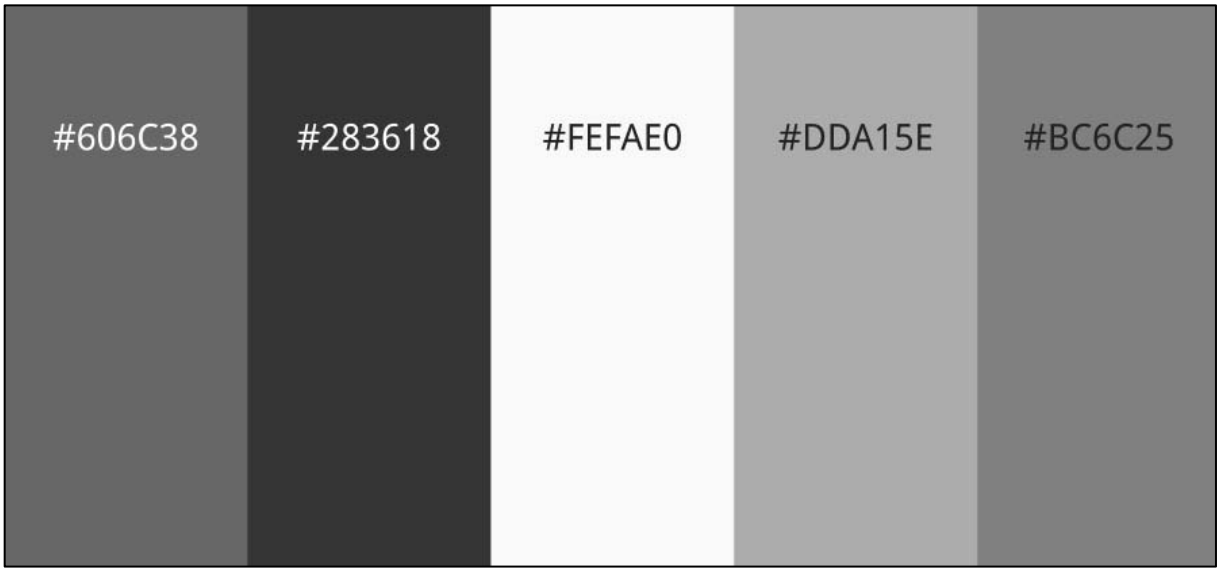

Рисунок 5 – Палитра цветов приложения

При открытии приложения пользователь первым делом попадает на страницу авторизации, где пользователь указывает E-mail и пароль, чтобы попасть в свой профиль, либо проходит авторизацию через аккаунт Google.

Также пользователь может создать новый аккаунт на странице регистрации. Главное меню будет отображаться после прохождения авторизации.

Из главного меню мы можем попасть во все остальные разделы приложения. На рисунке 6 представлены макеты экрана авторизации, регистрации и главного экрана приложения.

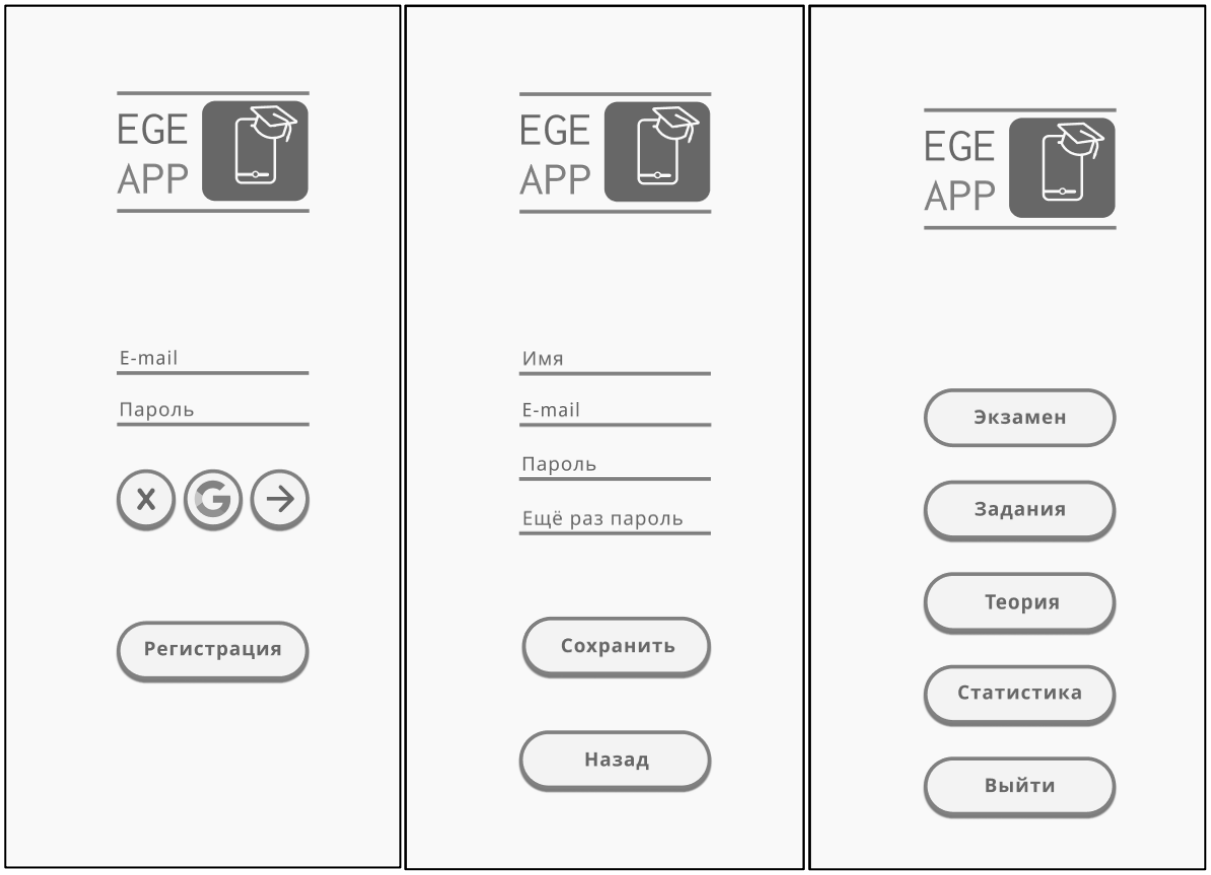

Рисунок 6 – Макеты экранов авторизации (слева),
регистрации (по середине) и главный экран (справа)

После успешной авторизации пользователь получает доступ к различным функциям приложения, направленным на подготовку к экзамену. Он может приступить к прохождению тестов, изучению теоретического материала и просмотру статистики своего прогресса. Визуальное представление статистики помогает пользователю отслеживать свой прогресс и оценивать свою подготовку, что помогает более эффективно распределить время. А выбор конкретных задач позволяет лучше подготовиться к определенному типу заданий, что повышает эффективность обучения. Макеты экранов со

статистикой, выбора заданий и тестовой составляющей представлены на рисунке 7.

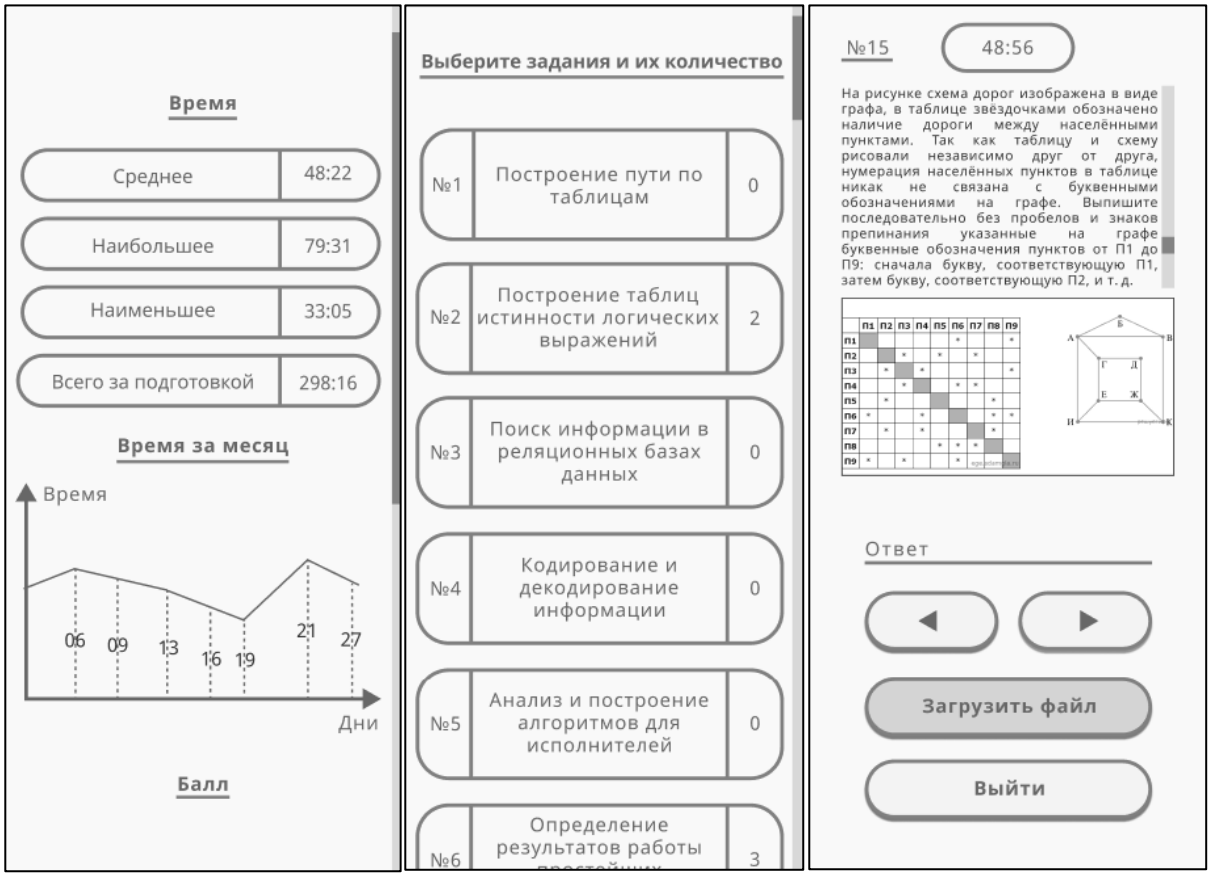

Рисунок 7 – Макеты экранов статистики (слева),
выбора заданий (по середине) и теста (справа)

3.2. Выбор СУБД

База данных (БД) – структурированное именованное хранилище информации [12]. На сегодняшний день большинство приложений используют для хранения информации системы управления базами данных (СУБД). СУБД представляет собой программное обеспечение, с помощью которого можно создавать базы данных и проводить над ними различные операции: обновлять, удалять, выбирать, редактировать и т. д. СУБД гарантирует сохранность, целостность, безопасность хранения данных и позволяет выдавать доступ к администрированию базы данных [12].

СУБД обеспечивает:

1) работу с данными, размещенными на внешних накопителях;

2) работу с данными, находящимися в оперативном запоминающем устройстве с применением дискового кэша;

3) ведение отчетности резервирования, редактирования, восстановление данных и т. д.;

4) поддержку различных форматов данных [13].

На данный момент существует множество СУБД для платформы Android. В обзоре рассматриваются некоторые из них. На основе проведенного обзора, выберем одну из СУБД для работы с приложением «EGEApp».

SQLite

SQLite [14] – компактная встраиваемая реляционная база данных. Исходный код библиотеки находится в общем доступе, что является плюсом для СУБД. Она является реляционной базой данных. Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер. Т.е. движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы.

Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных в единственном стандартном файле на том устройстве, на котором исполняется программа. К недостаткам данной СУБД можно отнести:

1) отсутствие системы пользователей – более крупные СУБД включают в свой состав системы управления правами доступа пользователей;

2) отсутствие возможности увеличения производительности;

3) данная СУБД входит в состав приложения, тем самым приложение становится объемнее и занимает больше памяти в устройствах.

Однако, у SQLite можно отметить важную особенность: база данных состоит из одного файла, поэтому ее очень легко переносить на разные устройства.

Realm

Realm [15] – это система управления базами данных для мобильных устройств. Она написана на C++, что делает ее одной из самых быстрых мобильных баз данных. Разработчики изначально ставили перед собой цель создать СУБД, которая должна отвечать следующим требованиям:

1) простота использования;
2) высокая скорость работы;
3) реляционность;
4) кроссплатформенность (Java, Swift & Objective C);
5) поддержка современных функций (шифрование, миграции, поддержка расширений и т. д.) [16].

Преимуществом данной СУБД является широкое сообщество разработчиков на GitHub и Stackoverflow, в котором можно найти ответы на многие интересующие вопросы.

Firebase

Firebase [8] – это облачная база данных, которая позволяет пользователям хранить и получать сохраненную информацию, а также имеет удобные средства и методы взаимодействия с ней. Firebase хранит текстовые данные в JSON формате и предоставляет удобные методы для чтения, обновления и извлечения данных. Также, Firebase может помочь с регистрацией и авторизацией пользователей, хранением сессий (авторизованные пользователи), медиафайлов к которым с легкостью предоставляет доступ благодаря Cloud Storage. Firebase не является полностью бесплатной. Часть функционала остается недоступным для пользователей, которые не оплатили использование данной СУБД. Но основные востребованные функции регистрации, авторизации и хранения текста доступны всем после регистрации в системе. Огромными и главными плюсами Firebase является гибкость и скорость загрузки в проект. Firebase позволяет сохранять скорость работы приложения. Не нужно отвлекаться на лишние вещи (создание базы данных,

написание API передачи и получения данных). Вся серверная нагрузка ложится на этот сервис.

Таким образом, системой для хранения данных приложения «Подготовка к ЕГЭ» была выбрана СУБД Firebase. Она обладает всеми нужными качествами для использования ее в разработке приложения, а также довольно простым и понятным функционалом.

3.3. Модель базы данных

Для эффективного хранения данных, принятых в приложении, было принято решение использовать облачную NoSQL базу данных – Firebase Realtime Database. Этот выбор обусловлен рядом преимуществ, которые вносят важный вклад в общую функциональность системы.

Прежде всего, Firebase Realtime Database является облачной, что предоставляет высокую степень мобильности данных. Это обеспечивает мгновенные обновления в режиме реального времени для всех подключенных устройств, создавая единое и актуальное представление данных. Такой подход особенно важен для приложения, ориентированного на образовательные задачи, где оперативность и актуальность информации играют ключевую роль.

Кроме того, преимуществом Firebase Realtime Database является его способность оперативно обновлять задания и результаты без необходимости перезаписи всего объема данных. Это существенно повышает эффективность обновлений и минимизирует временные задержки при обработке информации.

Важно отметить, что данная база данных использует JSON формат для хранения данных, что открывает широкие возможности для их структурирования. В данном случае, представление данных в виде дерева соответствует принципам организации базы данных, что обеспечивает более эффективный доступ и манипуляции с информацией.

Таким образом, использование Firebase Realtime Database не только обеспечивает надежное хранение данных, но и вносит важные аспекты оперативности и гибкости в работу системы.

На рисунке 8 изображено представление схемы базы данных экзаменационных вариантов и дополнительных материалов для подготовки к ЕГЭ по информатике в виде графа.

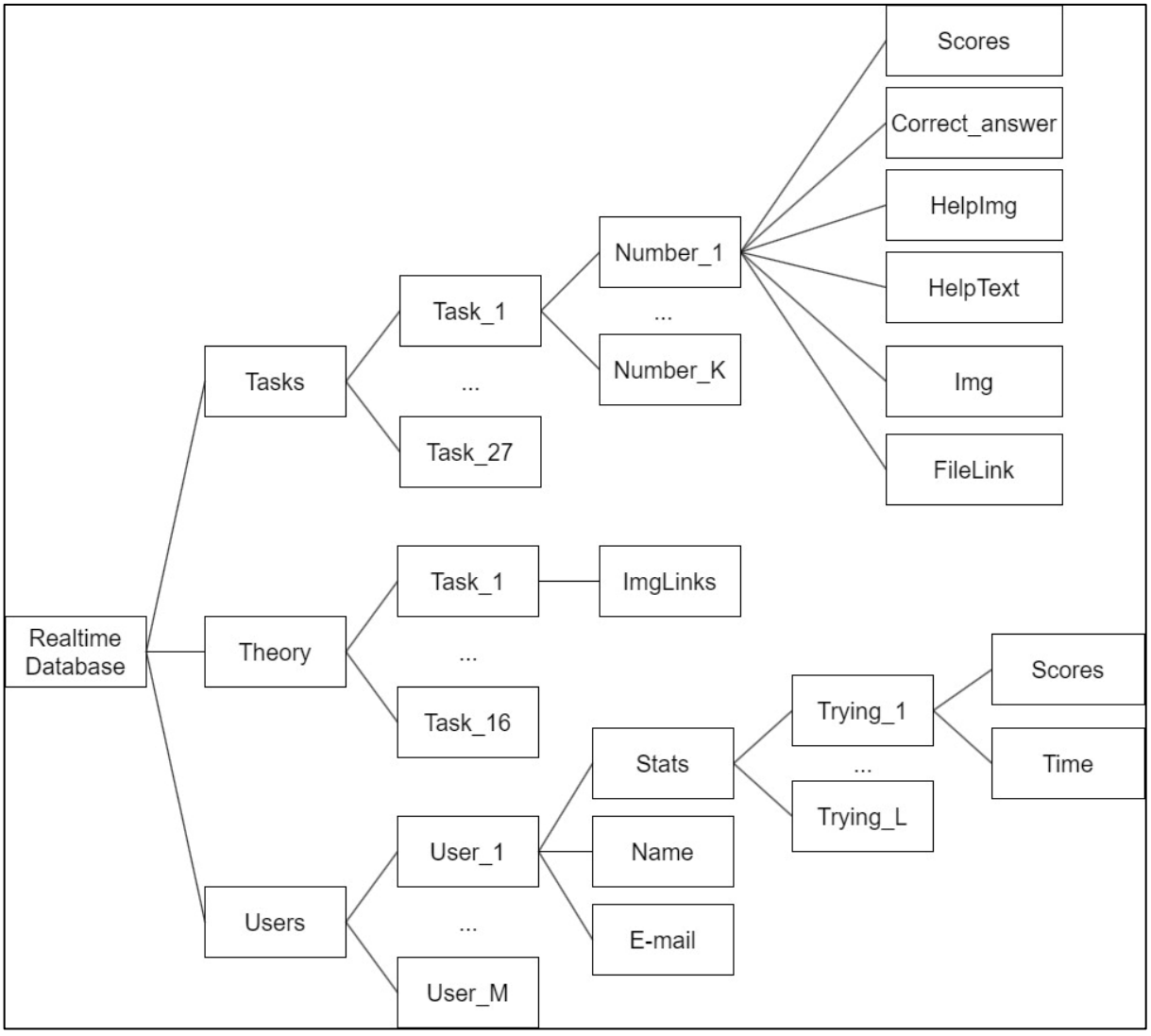

Рисунок 8 – Схема базы данных

Исходя из хранимой информации, база данных делится на три основных ветви: Theory (содержит ссылки на дополнительные материалы), `Tasks` (содержит экзаменационные задания) и `Users` (содержит информацию о пользователях). Таким образом, схема базы данных представляет собой иерархическую структуру, которая позволяет эффективно организовывать и

систематизировать большой объем информации, а также предоставляет удобный интерфейс для поиска и изучения необходимых материалов для подготовки к ЕГЭ по информатике.

`Tasks` содержит 27 экзаменационных заданий ЕГЭ по информатике. Каждое экзаменационное задание состоит из определенного числа вариантов этих заданий. В свою очередь каждый вариант имеет следующие поля: `Correct_answer` (правильный ответ на задание), `Task` (текст вопроса), `Scores` (количество баллов за задние), `HelpImg` (картинка для подсказки), `HelpText` (текст подсказки) и `FileLink` (ссылка для скачивания файла, который необходим для решения задания).

`Theory` содержит некоторые задания из ЕГЭ по информатике, для которых существуют дополнительные теоретические материалы. Каждое задание имеет ссылку на картинку, которые содержит в себе вспомогательные материалы, помогающие решать задания более успешно.

`Users` содержит информацию о зарегистрированных пользователях. Каждая запись в этой таблице содержит уникальный идентификатор пользователя, его электронный адрес (E-mail), имя и данные о статистике, связанной с решением задач и экзаменов, а также затраченного на подготовку времени.

3.4. Архитектура системы

Мобильное приложение построено на базе архитектуры MVC (Model-View-Controller). MVC состоит из объектов трех видов. Модель – это объект приложения, содержащий данные приложения и «бизнес-логику». Представление отображает данные на экран пользователя. Контроллер реагирует на действия пользователя, оповещая модель о необходимости изменений. каждый из этих элементов играет свою роль в обеспечении функциональности приложения, и вместе они образуют полноценное и эффективное решение в разработке Android-приложения. Общий принцип работы архитектуры MVC изображен на рисунке 9.

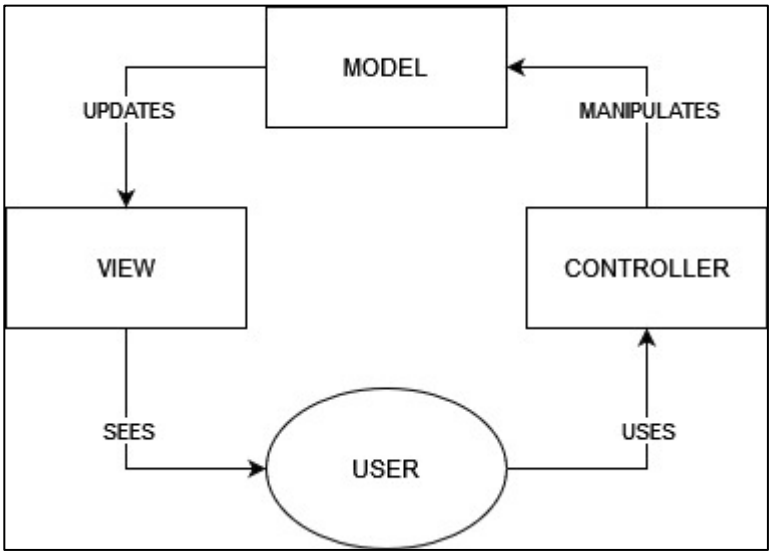

Рисунок 9 – Диаграмма архитектуры MVC

Архитектурный шаблон MVC играет значительную роль в упрощении разработки мобильного приложения. Данный шаблон обеспечивает четкое разделение обязанностей, ведь при добавлении в приложение нового функционала разработчику проще думать на уровне модели, чем о корректной работе всего приложения в целом. Также MVC облегчает повторное использование кода, так как классы с ограниченным функционалом легче адаптировать и использовать в других частях приложения или даже в других проектах, чем классы, которые пытаются выполнять множество функций, что может привести к сложности и непредсказуемым результатам.

Android-приложение по подготовке к ЕГЭ по информатике построено на основе схемы MVC. Помимо элементов данной схемы, приложение содержит облачную базу данных. В контексте мобильного приложения использование архитектурного шаблона MVC позволяет создать структурированное, модульное и легко поддерживаемое решение. На рисунке 10 представлена диаграмма основных компонентов мобильного приложения по подготовке к ЕГЭ по информатике. Рассмотрим более подробно элементы приложения.

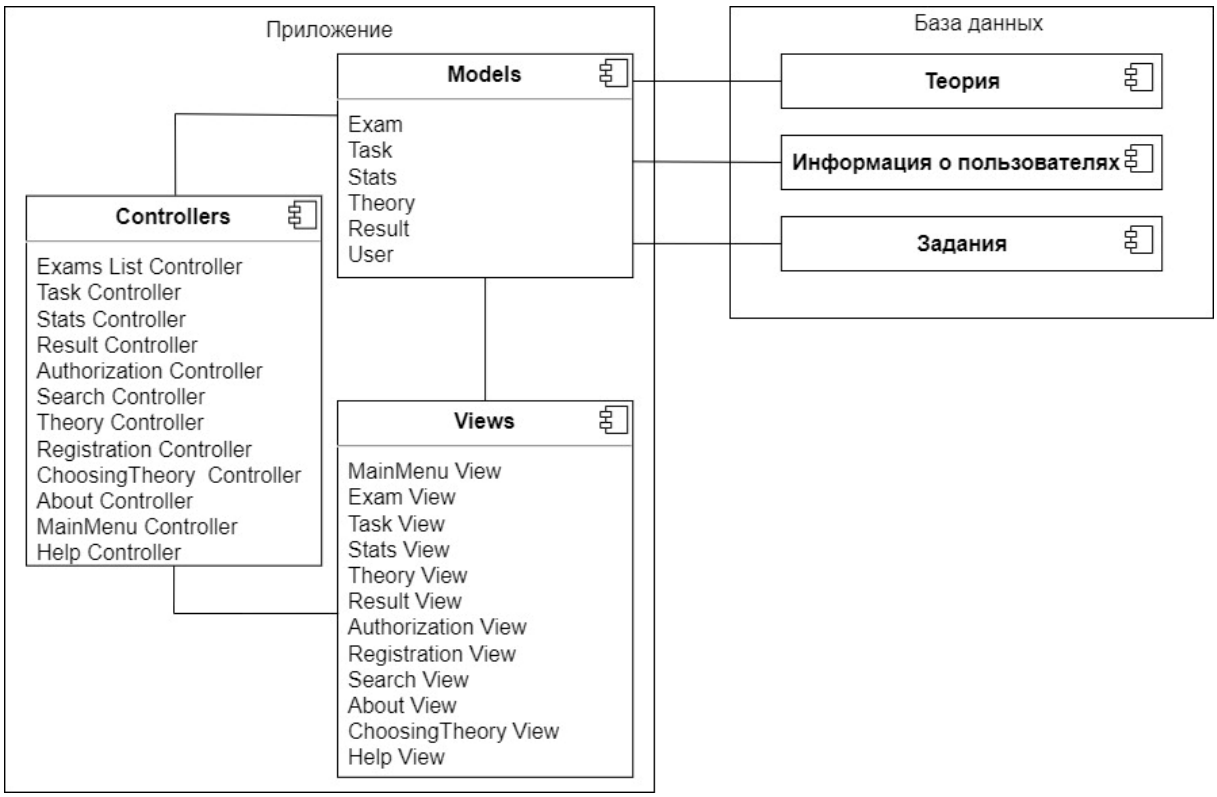

Рисунок 10 – Диаграмма основных компонентов

Модели (Models)

Были выделены 6 основных моделей для Android-приложения.

1. `User` – модель для хранения и обработки информации о пользователях.

2. `Stats` – модель для хранения и обработки информации о статистике пользователя.

3. `Exam` – модель для хранения и обработки информации об экзаменационных вариантах ЕГЭ по информатике.

4. `Task` – модель для хранения и обработки информации о заданиях и подсказок к ним в экзаменационных вариантах.

5. `Theory` – модель для хранения и обработки дополнительных материалов для подготовки к экзамену.

6. `Result` – модель для хранения и обработки информации о решении пользователем экзаменационного варианта или отдельного задания.

Представления (Views)

Были выделены 12 представлений для Andorid-приложения для подготовки к ЕГЭ по информатике.

1. `MainMenu View` – экран главного меню приложения.
2. `Exam View` – экран приложения, содержащий список вариантов.
3. `About View` – экран приложения, отвечающий за отображение информации об особенностях проведения экзамена по информатике.
4. `Theory View` – экран приложения, отвечающий за дополнительные материалы для подготовки к ЕГЭ по информатике.
5. `Stats View` – экран приложения, отвечающий за информацию о статистике пользователей.
6. `Task View` – экран приложения, отвечающий за прохождение вопроса пользователем.
7. `Result View` – экран приложения, отвечающий за результат прохождения экзаменационного варианта или отдельного задания пользователем.
8. `Registration View` – экран приложения, отвечающий за регистрацию пользователей.
9. `Authorization View` – экран приложения, отвечающий за авторизацию пользователей.
10. `Search View` – экран приложения, который отвечает за отображения списка заданий, где пользователь может составить себе тест самостоятельно.
11. `ChoosingTheory View` – экран приложения, который отвечает за отображения списка теории, в котором пользователь может выбрать интересующий его раздел.
12. `Help View` – экран приложения, который отвечает за отображения подсказки к заданию.

Контроллеры (Controllers)

Были выделены 12 контроллеров для Android-приложения для подготовки к ЕГЭ по информатике.

1. `Exams List Controller` – контроллер, отвечающий за работу пользователя со списком экзаменационных вариантов к ЕГЭ по информатике.

2. `ChoosingTheory Controller` – контроллер, отвечающий отображение списка с дополнительными материалами для подготовки к ЕГЭ по информатике, и обработку выбора пользователем раздела с теоретическими материалами.

3. `Theory Controller` – контроллер, отвечающий за отображения дополнительных материалов для подготовки к ЕГЭ по информатике перед пользователем на экране.

4. `Stats Controller` – контроллер, отвечающий за выгрузку из базы данных и отображение информации о статистике пользователя по подготовке к экзаменам.

5. `Task Controller` – контроллер, отвечающий за работу пользователя при решении отдельных заданий из экзаменационных вариантов ЕГЭ по информатике.

6. `Result Controller` – контроллер, который отвечает за отображение результатов прохождения экзамена или отдельных заданий, а также отправку результатов в базу данных Android-приложения.

7. `Registration Controller` – контроллер, который отвечает за регистрацию и проверку данных нового пользователя в базе данных Android-приложения.

8. `Authorization Controller` – контроллер, который отвечает за процедуру аутентификации пользователя в приложении по почте и паролю или через аккаунт Google.

9. `About Controller` – контроллер, отвечающий за загрузку информации об особенностях экзамена по информатике и таблицы перевода первичных баллов во вторичные.

10. `Search Controller` – контроллер, который отвечает за поиск заданий для теста и экзамена из базы данных.

11. `MainMenu Controller` – контроллер, который отвечает за обработку выбора пользователем и последующую загрузку раздела в главном меню Android-приложения.

12. `Help Controller` – контроллер, который отвечает за загрузку подсказки к заданию из базы данных Android-приложения.

Выводы по третьей главе

В процессе проектирования мобильного приложения был проведен детальный анализ, который включал: разработку дизайна, создание модели хранения данных и построение архитектуры мобильного приложения.

Во время проектирования дизайна Android-приложения было уделено особое внимание современным тенденциям в области пользовательского интерфейса и пользовательского опыта (UI/UX). Это позволило создать интуитивно понятный и привлекательный дизайн.

Также было проведено проектирование базы данных, которое включало в себя определение структуры данных, которые приложение будет использовать для хранения информации, и создание эффективной модели данных. Была выбрана среда для реализации базы данных, которая обеспечивает высокую производительность и надежность, а также поддерживает нашу модель данных.

Еще была спроектирована архитектура мобильного приложения на основе шаблона MVC. Архитектура состоит из 6 моделей, 12 контроллеров и 12 представлений. При реализации это обеспечит четкую структуру и разделение обязанностей, что упрощает процесс разработки и облегчает поддержку Android-приложения.

4. РЕАЛИЗАЦИЯ ANDROID-ПРИЛОЖЕНИЯ

4.1. Технологии для реализации приложения

На данный момент существует несколько Android IDE, которые используются для создания Andoid-приложений. Ниже рассмотрим некоторые из них.

Android Studio

Бесплатная среда Android Studio [17] (созданная на базе IntelliJ IDEA Community Edition) в настоящее время является основной интегрированной средой, рекомендуемой для разработки приложений Android (исходные средства разработки Android работали на базе Eclipse IDE) [17]. Среда Android Studio в сочетании с бесплатным пакетом Android Software Development Kit (SDK) и бесплатным пакетом Java Development Kit (JDK) предоставляет все необходимое для создания, запуска и отладки приложений Android и поддержки их распространения.

Eclipse

Eclipse – свободная интегрированная среда разработки кроссплатформенных приложений. Android Development Tools (ADT) – это плагин для Eclipse IDE, позволяющий разрабатывать приложения для платформы Android. Но Android Studio является более релевантной средой разработки, так как она была специально создана для разработки приложений для Android ОС, поэтому она позволяет проще и быстрее создавать приложения по сравнению с Eclipse [18].

React Native

React Native – это JS-фреймворк для создания нативно отображаемых iOS и Android-приложений [19]. В его основе лежит разработанная в Facebook JS-библиотека React, предназначенная для создания пользовательских интерфейсов. Но вместо браузеров она ориентирована на мобильные платформы. При разработке с помощью React Native нужно использовать OS X. Это неизбежное ограничение для большинства разработчиков.

В ходе анализа IDE, предназначенных для разработки приложений на Android устройства, было выявлено, что на текущий момент среда разработки Android Studio является наиболее удобной и многофункциональной. Кроме того, Android Studio считается официальным средством разработки Android приложений.

4.2. Реализация облачной базы данных

В ходе обзора СУБД для платформы Android в пункте 3.1 для хранения данных приложения была выбрана облачная СУБД Firebase. Для хранения текстовых данных был использован сервис Realtime Database. Данный режим позволяет обновлять данные приложения в реальном времени на всех устройствах, которые будут использовать приложение.

Также был использован сервис Authentication для хранения данных пользователей, который авторизовались или зарегистрировались в приложении. Firebase Authentication позволяет организовывать быструю авторизацию через сервисы Google.

Примеры хранения данных

Хранение данных об экзаменационных заданиях ЕГЭ по информатике с помощью Firebase Realtime Database представлено на рисунке 11. Информация о заданиях хранится в виде JSON-объектов. Это обеспечивает гибкость, масштабируемость и безопасность данных, а также позволяет быстро вносить изменения и обновлять информацию с помощью различных инструментов и API.

Хранение данных об авторизованных через Google и зарегистрированных пользователях, с помощью Firebase Authentication. На рисунке 12 же представлена ветка с данными пользователя в Firebase Realtime Database, где каждый пользователь имеет уникальный идентификатор, а также раздел со статистикой, где хранятся попытки решения экзаменационных вариантов ЕГЭ по информатике, которые также имеют уникальные идентификаторы.

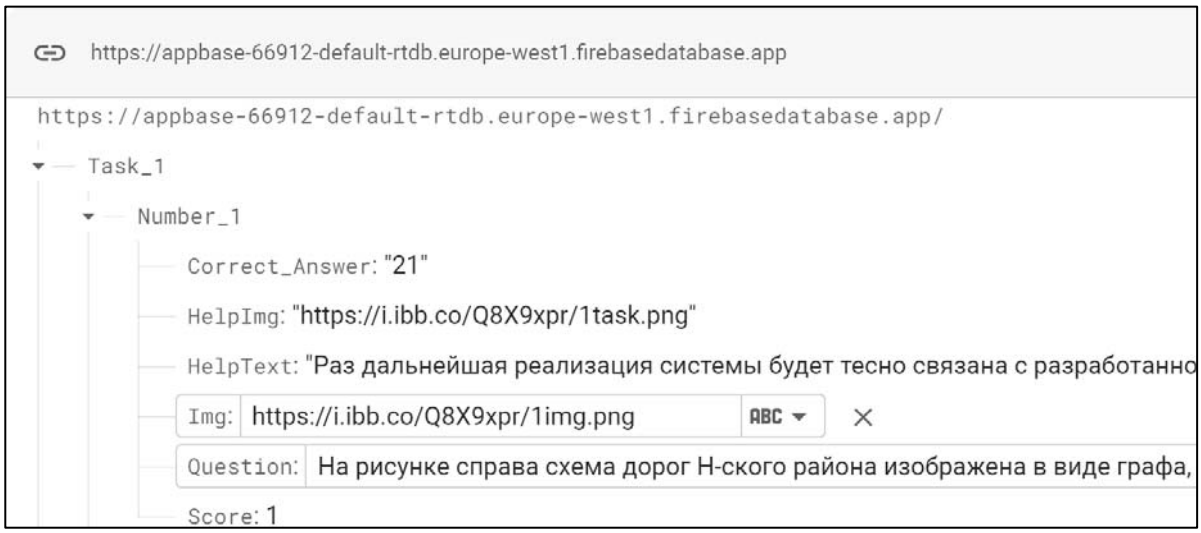

Рисунок 11 – Хранение данных об экзаменационных заданиях ЕГЭ по информатике

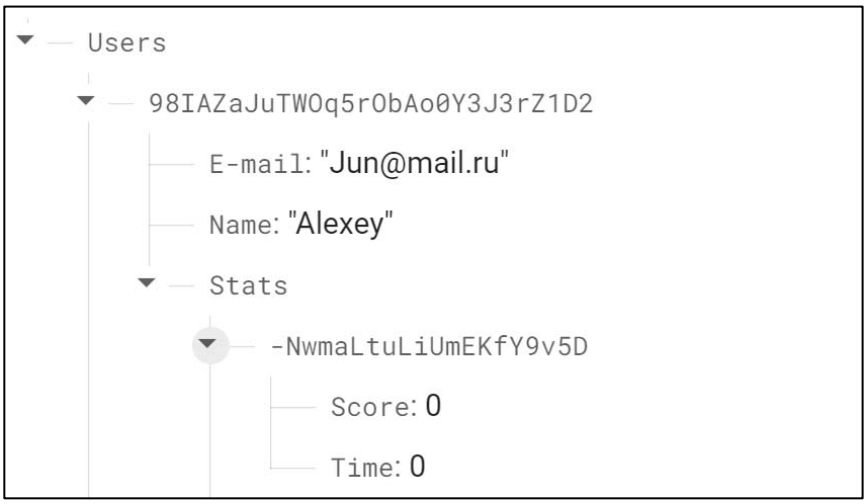

Рисунок 12 – Хранение данных пользователей

Облачная база данных содержит ссылки на теоретические материалы к ЕГЭ по информатике, информацию об особенностях экзамена, а также тексты и ссылки на картинки самих заданий, объяснения решений и правильные ответы. Таким образом база данных содержит большое количество практических упражнений и полезной информации для подготовки к экзамену. Не менее важно то, что база данных также хранит информацию о пользователях приложения, включая их профили и результаты тестирова-

ния (количество баллов и время прохождения теста), что позволяет персонализировать опыт использования и лучше следить за прогрессом подготовки к экзамену.

В свою очередь, с технической точки зрения использование облачной базы данных позволяет значительно снизить размер приложения при его загрузке на мобильное устройство, поскольку основная часть данных хранится удаленно и загружается по мере необходимости.

4.3. Реализация мобильного приложения

Мобильное приложение обеспечивает возможность использования только при подключении к интернету, так как данные подгружаются из облачной базы данных Firebase.

Одним из главным компонентом приложения Android является activity (активность). Обычно activity ассоциируется с отдельным экраном приложения, а переключение между экранами приложения происходит как перемещение от одной activity к другой.

Авторизация и регистрация в приложении

В листинге 1 представлена функция `loginUser` класса `EmailPasswordActivity`, которая отвечает за авторизацию. Функция `loginUser` получает доступ к элементам пользовательского интерфейса `emailEditText` и `passwordEditText`, соответствующим полям ввода для электронной почты и пароля. Затем функция проверяет, являются ли поля электронной почты и пароля пустыми с помощью метода `TextUtils.isEmpty()`. Если поле электронной почты или пароля пустое, выводится короткое уведомление (`Toast.makeText()`), и функция прекращает выполнение. Если поля электронной почты, имени и пароля заполнены, а поле пароля совпадает с полем повтора пароля, то функция вызывает метод `signInWithEmailAndPassword(email, password)` объекта `firebaseAuth`, чтобы выполнить процедуру аутентификации пользователя с использованием электронной почты и пароля.

Листинг 1 – Фрагмент функции авторизации пользователя

```
fun loginUser(view: View?) {
        val emailEditText = findViewById<EditText>(R.id.emailLog)
        val passwordEditText = findViewById<EditText>(R.id.passwordLog)
        val email = emailEditText.text.toString().trim { it <= ' ' }
        val password = passwordEditText.text.toString().trim { it <= ' ' }
        if (TextUtils.isEmpty(email)) {
            Toast.makeText(this, "Пожалуйста, введите e-mail для
авторизации", Toast.LENGTH_SHORT).show()
            return
        }
        if (TextUtils.isEmpty(password)) {
            Toast.makeText(this, "Пожалуйста, введите пароль для
авторизации", Toast.LENGTH_SHORT).show()
            return
        }
        firebaseAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    // Авторизация успешна, выполняем дальнейшие действия
                    Toast.makeText(this@LogingActivity, "Авторизация прошла
успешно", Toast.LENGTH_SHORT).show()
                    val intent = Intent(this@LogingActivity, MainActiv-
ity::class.java)
                    startActivity(intent)
                    // Переход на другой экран или выполнение других дей-
ствий
                } else {
                    // Авторизация провалилась
                    Toast.makeText(this@LogingActivity, "Ошибка авториза-
ции: " + task.exception!!.message, Toast.LENGTH_LONG).show()
                }
            }

    }
```

В листинге 2 представлена функция `registerUser`, которая отвечает за регистрацию нового аккаунта пользователя. Сначала функция `registerUser(View view)` получает доступ к элементам пользовательского интерфейса `emailEditText`, `passwordEditText` и `nameEditText` соответствующим полям ввода для электронной почты, пароля и имени пользователя и получает их значения, проверив перед этим не являются ли поля пустыми. Далее создается новый экземпляр `FirebaseAuth` с помощью `FirebaseAuth.getInstance()`. Потом вызывается метод `createUserWithEmailAndPassword(email, password)` объекта `firebaseAuth`, чтобы зарегистрировать нового пользователя с использованием введенной электронной почты и пароля. Затем с помощью метода `currentUser` объекта `fire-`

baseAuth в Realtime Database создается подузел с уникальным идентификатором пользователя, для дальнейшего ведения статистики. Имя пользователя также закрепляется за ним в Realtime Database.

Листинг 2 – Функция регистрации пользователя

```
fun registerUser(view: View?) {
        val emailEditText = findViewById<EditText>(R.id.background)
        val nameEditText = findViewById<EditText>(R.id.Name)
        val passwordEditText = findViewById<EditText>(R.id.passwordEdit-
Text)
        val passwordEditText2 = findViewById<EditText>(R.id.passwordEdit-
Text2)
        val email = emailEditText.text.toString().trim { it <= ' ' }
        val password = passwordEditText.text.toString().trim { it <= ' ' }
        val password2 = passwordEditText2.text.toString().trim { it <= ' '}
        val name = nameEditText.text.toString().trim { it <= ' ' }
        if (TextUtils.isEmpty(email)) {
            Toast.makeText(this, "Пожалуйста, введите почту",
Toast.LENGTH_SHORT).show()return}
        if (TextUtils.isEmpty(password)) {
            Toast.makeText(this, "Пожалуйста, введите пароль",
Toast.LENGTH_SHORT).show()return}
        if ((TextUtils.isEmpty(password2)) || (password != password2)) {
            Toast.makeText(this, "Пожалуйста, повторите тот же пароль",
Toast.LENGTH_SHORT).show()return}
        if (TextUtils.isEmpty(name)) {
            Toast.makeText(this, "Пожалуйста, введите ваше имя",
Toast.LENGTH_SHORT).show()return}
        val firebaseAuth = FirebaseAuth.getInstance()
        firebaseAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    val currentUser = firebaseAuth.currentUser
                    val userId = currentUser?.uid
                    val database = FirebaseDatabase.getInstance()
                    val usersRef = database.getReference("Users")
                    val currentUserRef = usersRef.child(userId ?: "")
                    currentUserRef.child("E-mail").setValue(email)
                    currentUserRef.child("Name").setValue(name)
                    currentUserRef.child("Stats").setValue("")
                    Toast.makeText(this@RegActivity, "Регистрация завер-
шена", Toast.LENGTH_SHORT).show()} else {
                    Toast.makeText(this@RegActivity, "Ошибка регистрации
данных: " + Objects.requireNonNull(task.exception)!!.message,
Toast.LENGTH_LONG).show()}}
```

На рисунке 13 представлен экран «Авторизация», который позволяет войти в аккаунт пользователя. Для входа в аккаунт, пользователь должен ввести свой логин (электронную почту) и пароль, после чего нажать кнопку «Войти». Если данные введены верно, пользователь будет перенаправлен на главный экран приложения. В случае неверного ввода данных, система оповестит пользователя об ошибке.

Также, для повышения безопасности, система предлагает сохранить данные учетной записи на устройстве, чтобы в следующий раз вход осуществлялся автоматически.

Рисунок 13 – Экран «Авторизация»

Главный экран

После авторизации пользователь попадает на экран «Главный», представленный на рисунке 14, с которого пользователь может перемещаться по приложению. Реализация класса «Главный» представлена в листинге 3.

Листинг 3 – Класс `MainActivity`

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        supportActionBar?.setBackgroundDrawable(ColorDrawable(Color.par-
seColor("#FEFAE0")))}
    fun logOut(view: View?) {
        signOut()
        val intent = Intent(this, LogingActivity::class.java)
        intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIV-
ITY_CLEAR_TASK
```

```
        startActivity(intent)}
    fun onClickTest(view: View?) {
        val intent = Intent(this, TestActivity::class.java)
        startActivity(intent)}
    fun onClickEx(view: View?) {
        val intent = Intent(this, ExActivity::class.java)
        startActivity(intent)}
    fun onClickChoiceTheory(view: View?) {
        val intent = Intent(this, ChoosingTheoryActivity::class.java)
        startActivity(intent)}
    fun onClickStat(view: View?) {
        val intent = Intent(this, StatActivity::class.java)
        startActivity(intent)}
    fun onClickAbout(view: View?){
        val intent = Intent(this, AboutActivity::class.java)
        startActivity(intent)}
}
```

Рисунок 14 – Экран «Главный»

Класс `MainActivity` определяет основную активность приложения, устанавливает макет для этой активности и содержит методы для обработки событий, таких как выход из аккаунта и переход на другие экраны Android-приложения.

Для возможности перехода и логаута класс `MainActivity` содержит три публичных метода.

1. Метод `logOut(View view)` вызывается при клике на кнопку выхода (`logOut`) в пользовательском интерфейсе. Внутри метода вызывается статический метод `signOut()` класса `LogingActivity`, который отвечает за выход из аккаунта.

2. Метод `onClickTest(View view)` вызывается при клике на определенную кнопку `buttonTest` в пользовательском интерфейсе. Внутри метода создается интент для перехода на активность `TestActivity`.

3. Метод `onClickChoiceTheory(View view)` вызывается при клике на определенную кнопку `buttinTheory` в пользовательском интерфейсе. Внутри метода создается интент для перехода на активность `ChoosingTheoryActivity`.

4. Метод `onClickEx(View view)` вызывается при клике на кнопку `buttonEx` в пользовательском интерфейсе. Внутри функции создается интент для перехода на активность `ActivityEx`.

5. Метод `onClickStat(View view)` вызывается при клике на кнопку `buttonStat` в пользовательском интерфейсе. Внутри функции создается интент для перехода на активность `ActivityStat`.

6. Метод `onClickAbout(View view)` вызывается при клике на кнопку `About` в пользовательском интерфейсе. Внутри функции создается интент для перехода на активность `ActivityAbout`.

Режим экзамена

При переходе по кнопке с текстом «Решить ЕГЭ по информатике» пользователь попадает на экран «Тест», на котором пользователь может пройти симуляцию ЕГЭ по информатике. Для загрузки заданий из базы данных Firebase используется функция `loadTaskFromFirebase`, реализация которой представлена в листинге 4.

Листинг 4 – Функция загрузки данных

```
private fun loadTaskFromFirebase(taskKey: String, numberKey: String?) {
        val taskRef = FirebaseDatabase.getInstance("
https://appbase-66912-default-rtdb.europe-west1.firebasedatabase.app").ref-
erence.child(taskKey).child(numberKey!!)
        taskRef.addListenerForSingleValueEvent(object : ValueEventListener{
            override fun onDataChange(dataSnapshot: DataSnapshot) {
                if (dataSnapshot.exists()) {
                    textForTask = dataSnapshot.child("Question").get-
Value(String::class.java)
                    imageUrl = dataSnapshot.child("Img").get-
Value(String::class.java)
                    downloadUrl = dataSnapshot.child("FileLink").get-
Value(String::class.java)
       rightAnswer = dataSnapshot.child("Correct_Answer").get-
Value(String::class.java)
       scoreForTask = dataSnapshot.child("Score").getValue(Int::class.java)
                    TaskText!!.text = textForTask
                    if (imageUrl == null) {
                        imageBox!!.visibility = View.GONE} else {
                        imageBox!!.visibility = View.VISIBLE
                        Picasso.get().load(imageUrl).into(imageBox)}
                    if (downloadUrl != null) {
                        buttonDownload!!.isEnabled = true
                        buttonDownload?.setBackgroundColor(Color.par-
seColor("#F7F4E0"))} else { buttonDownload!!.isEnabled = false
                        buttonDownload?.setBackgroundColor(Color.par-
seColor("#DAD6BC"))}}}})}
```

Для некоторых заданий прилагаются файл. Реализация загрузки файлов через функцию `startDownload` представлена в листинге 5.

Листинг 5 – Функция загрузки файлов

```
private fun startDownload() {
        val request = DownloadManager.Request(Uri.parse(downloadUrl))
        request.setAllowedNetworkTypes(DownloadManager.Request.NETWORK_WIFI
or DownloadManager.Request.NETWORK_MOBILE)
        request.setTitle("Загрузка файла")
        request.setDescription("Загрузка файла...")
        request.setNotificationVisibility(DownloadManager.Request.VISIBIL-
ITY_VISIBLE_NOTIFY_COMPLETED)
        request.setDestinationInExternalPublicDir(Environment.DIREC-
TORY_DOWNLOADS, "file")
        val downloadManager = getSystemService(DOWNLOAD_SERVICE) as Down-
loadManager
        downloadManager.enqueue(request)}
```

Функция `startDownload()` выполняет запуск загрузки файла с использованием `DownloadManager` в Android. Она создает объект запроса `DownloadManager.Request` с указанием URL загрузки. Затем устанавливает типы сети, которые разрешены для загрузки (Wi-Fi и мобильные данные).

Задает заголовок и описание запроса загрузки. Также устанавливает видимость уведомления о завершении загрузки файла. Затем указывает путь для сохранения загруженного файла в публичной директории загрузок на внешнем хранилище устройства. Наконец, получает экземпляр `DownloadManager` из службы загрузки (`DOWNLOAD_SERVICE`) и добавляет запрос в очередь загрузок для выполнения.

Активность «Тест» представляет собой повторно используемую часть пользовательского интерфейса приложения. В тесте присутствуют кнопки перемещения по заданиям и кнопка загрузки файлов для заданий, которая появляется только тогда, когда присутствует ссылка для скачивания. Блок-схема алгоритма загрузки заданий представлена на рисунке 15.

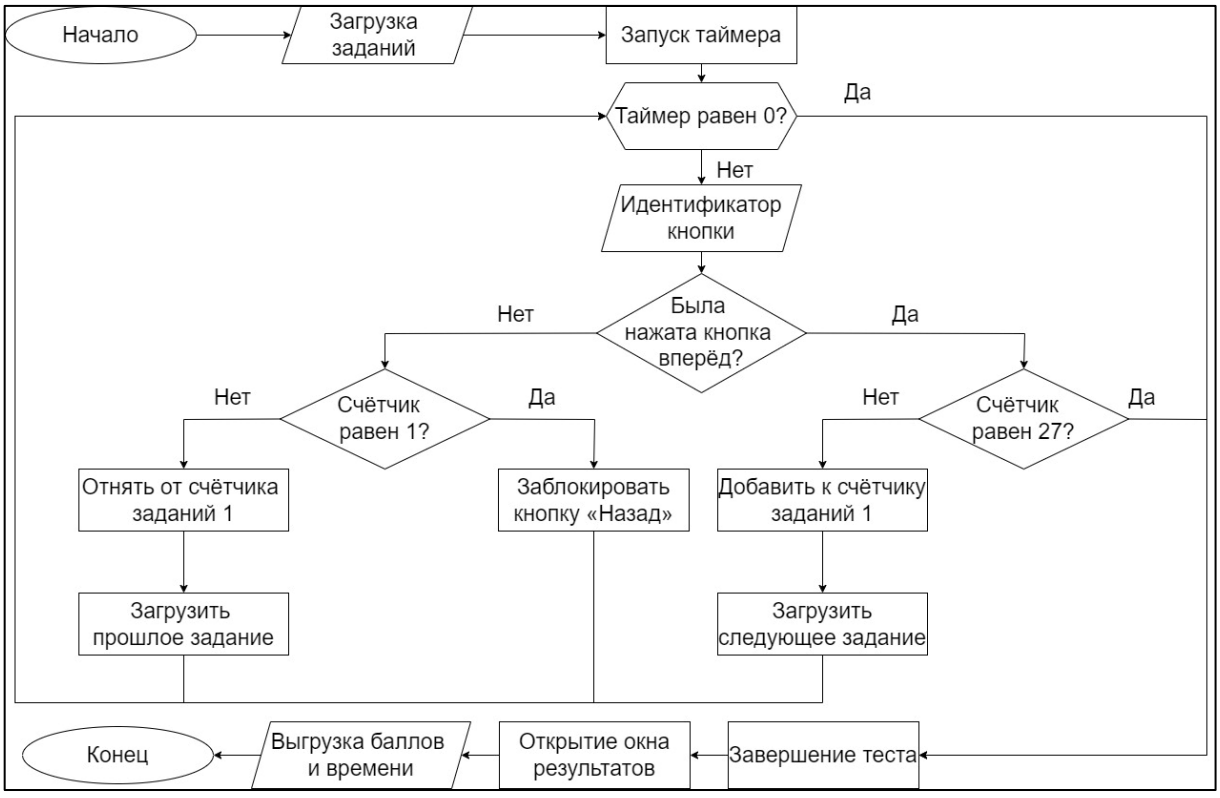

Рисунок 15 – Блок-схема алгоритма загрузки заданий

В классе `TestResult` реализован метод `onCreate`, который выводит результат прохождения теста на экран. При прохождении теста выше порогового значения будет выдана оценка «Неплохо!», если же количество баллов выше среднего значения на экзаменах, то будет оценка «Поздравляем!»,

иначе будет выдана оценка «К сожалению, у вас 0 баллов!». Реализация метода `onCreate` класса `TestResult` представлена в листинге 6.

Листинг 6 – Метод оценки и вывода результатов на экран

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        supportActionBar?.setBackgroundDrawable(ColorDrawable(Color.par-
seColor("#FEFAE0")))
        setContentView(R.layout.activity_test_result)
        val ResultTextView = findViewById<TextView>(R.id.ResultTextView)
        val CongratulationsTextView = findViewById<TextView>(R.id.Congratu-
lationTextView)
        val ScoreTextView = findViewById<TextView>(R.id.ScoreTextView)
        var secondaryScore: String? = null
        val globalScore = intent.getIntExtra("Score", 0)
        secondaryScore = convertPrimaryToSecondary(globalScore)
        var timeInMilleseconds: Long = intent.getLongExtra("timeInMillisec-
onds", 0)
        if (globalScore == 1 || globalScore == 21) {
        ScoreTextView.text = "балл"}
        else if (globalScore > 1 && globalScore < 5) {
            ScoreTextView.text = "балла"}
        else if (globalScore > 21 && globalScore < 25) {
            ScoreTextView.text = "балла"
        }
        else {ScoreTextView.text = "баллов"}
        if (globalScore > 20) {
            CongratulationsTextView.text = "Поздравляем!"
        }
        else if (globalScore <= 20 && globalScore > 7) {
            CongratulationsTextView.text = "Неплохо!"
        }
        else {CongratulationsTextView.text = "К сожалению,"}
        ResultTextView.text = globalScore.toString()}
```

При выходе из экрана просмотра результатов тестирования запускается функция `Exit`, которая создает подузел в базе данных в разделе `Stat` и записывает в него получившиеся результаты. После выполнения этих действий открывается главный экран приложения. Реализация функции `Exit` представлена в листинге 7.

Листинг 7 Функция выхода из просмотра результатов тестирования

```
fun Exit(view: View?) {
        val globalScore = intent.getIntExtra("Score", 0)
        val timeInMilleseconds = intent.getLongExtra("timeInMilleseconds",
0)
        val userId = FirebaseAuth.getInstance().currentUser?.uid
        if (userId != null) {
            val database = FirebaseDatabase.getInstance()
            val usersRef = database.getReference("Users")
            val currentUserStatsRef = usersRef.child(userId).child("Stats")
            val attemptId = currentUserStatsRef.push().key
            val testAttemptData = HashMap<String, Any>()
            testAttemptData["Score"] = globalScore
```

```
            testAttemptData["Time"] = timeInMilleseconds
            currentUserStatsRef.child(attemptId ?: "").setValue(tes-
tAttemptData)}
        val intent = Intent(this, MainActivity::class.java)
        finish()
        startActivity(intent)}
```

Экран теста ЕГЭ по информатике и экран результатов представлены на рисунке 16.

Рисунок 16 – Экран тестирования: экран «Тест» (слева), экран «Результат теста» (справа)

Для перемещения по заданиям используются методы, отвечающие за нажатия кнопок, `buttonNext.onClickListener` и `buttonBack.onClick-Listener`. Они также выполняют проверку ответов. Подробная реализация этих методов представлена в листингах 8 и 9.

Листинг 8 – Фрагмент функции перемещения вперед по тесту

```
buttonNext.setOnClickListener { v ->
            if (number >= 0) {
                buttonBack.isEnabled = true
```

```
                buttonBack.setBackgroundResource(R.drawable.image_but-
ton_left)}
            val nice = v.context
            val userInput = answerEditText?.text.toString().trim { it <=''}
            if (answerEditText != null) {
                answerArray[index] = userInput
                answerEditText!!.setText("")
                if (userInput == rightAnswer) {
                    scoreArray[index] = scoreForTask
                } else {
                    if (scoreArray[index] != null) {
                        if (scoreArray[index]==1||scoreArray[index]==2){
                            scoreArray[index] = 0}}}} index++
```

Листинг 9 – Фрагмент функции перемещения назад по тесту

```
buttonBack.setOnClickListener { v ->
            if (number > 0) {
                if (number <= 26) {
                    buttonNext.setBackgroundResource(R.drawable.image_but-
ton_right)}if (number == 1){
                    buttonBack.setBackgroundResource(R.drawable.image_but-
ton_left_blocked)}
                val nice = v.context
                val userInput = answerEdit-
Text?.text.toString().trim{it<=''}
                if (answerEditText != null) {
                    answerArray[index] = userInput
                    answerEditText!!.setText("")
                    if (userInput == rightAnswer) {
                        scoreArray[index] = scoreForTask} else {
                        if (scoreArray[index] != null) {
                            if(scoreArray[index]==1||scoreArray[index]==2){
                                scoreArray[index] = 0}}}} index--
```

Функции `buttonNext.setOnClickListener` и `buttonBack.setOnClickListener` имеют схожий функционал. Они назначают обработчик нажатия на кнопку `buttonBack` и `buttonNext`. Функции работают так, что, если `number` меньше или равно 26, то извлекается контекст, в котором была нажата кнопка, и считывается текст из `EditText` с идентификатором `answerEditText`. Если `answerEditText` не является пустым, то в массив `answerArray` сохраняется введенный пользователем ответ, поле `EditText` очищается, и если введенный ответ совпадает с правильным ответом (`rightAnswer`), то в массив `scoreArray` сохраняется значение `scoreForTask`. В противном случае, если `scoreArray[index]` не равно `null` и равно 1 или 2, то `scoreArray[index]` устанавливается в 0. Затем индекс уменьшается, либо увеличивается на 1. Затем создается массив `taskKeys`, содержа-

щий названия заданий. Значение переменной number уменьшается либо увеличивается на 1. Если number меньше или равно 26, то происходит загрузка задания из Firebase с использованием ключа taskKeys[number] и номера "Number_1". Если number равно 26, то текст на кнопке buttonNext изменяется. Иначе происходит подсчет суммы оценок из массива scoreArray, значение сохраняется в переменную globalScore. Затем создается Intent для перехода на активность TestResult, и в Intent добавляется дополнительная информация Score с значением globalScore и запускается активность TestResult.

Метод onCreate(Bundle savedInstanceState), который выполняется при создании активности TestResult, устанавливает макет для активности, указывая на файл разметки activity_test_result.xml. Затем находит и инициализирует TextView элементы по их идентификаторам. После этого извлекает целочисленное значение globalScore из переданных данных, полученных через Intent. Зависимо от значения globalScore, функция устанавливает соответствующий текст для TextView элемента Score-TextView. Далее, в зависимости от значения globalScore, устанавливает различные тексты для TextView элемента CongratulationsTextView. Наконец, функция устанавливает значение globalScore в виде текста для TextView элемента ResultTextView.

Также в экзаменационном варианте реализовано завершение теста по окончании времени экзамена (235 минут). Если пользователь не успеет завершить экзамен за отведенное время, которое отображается на таймере, то тест автоматические завершается и сохраняет введенные пользователем данные. Реализация функции таймера описана в листинге 10.

Листинг 10 – Функция таймера

```
private fun startTimer(milliseconds: Long, context: Context) {
        startTimeMillis = System.currentTimeMillis()
        countDownTimer = object : CountDownTimer(milliseconds, 1000){
            override fun onTick(millisUntilFinished: Long) {
                val minutes = millisUntilFinished / 1000 / 60
                val seconds = (millisUntilFinished / 1000) % 60
```

```
                val timeLeftFormatted = String.format("%02d:%02d", minutes,
seconds)if (minutes == 15L) {
            timerTextView.setTextColor(Color.parseColor("#B34747"))}
                timerTextView.text = timeLeftFormatted}
                override fun onFinish() {
                Toast.makeText(context, "Время экзамена окончено",
Toast.LENGTH_SHORT).show()
              for (i in scoreArray.indices) {
                      if(scoreArray[i]!=null) {
                       scoreArray[i] = scoreArray[i]!! + scoreArray[i]!!}}
                globalScore = sumOfScore
                val intent = Intent(context, TestResult::class.java)
                intent.putExtra("Score", globalScore)
                context.startActivity(intent)}}.start()}
```

Режим тренировки

На рисунке 17 представлена страница создания теста из заданий для дальнейшего решения выбранных пользователем заданий.

Рисунок 17 – Экран создания теста из заданий

При открытии раздела для составления тестов из заданий запускается `ExActivity`. Вместе с активностью запускается метод класса `OnCreate`, ко-

торый загружает макет страницы и передает данные в следующую активность, которая в свою очередь загружает тест. Подробная реализация метода `OnCreate` в листинге 11.

Листинг 11 – Метод загрузки страницы и передачи данных

```
override fun onCreate(savedInstanceState: Bundle?) {
              super.onCreate(savedInstanceState)
              supportActionBar?.setBackgroundDrawable(ColorDrawable
(Color.parseColor("#FEFAE0")))
              setContentView(R.layout.activity_ex)
              val buttonExTest = findViewById<Button>(R.id.buttonExTest)
              buttonExTest.setOnClickListener
                    {
                  val editTextValues = getEditTextValues()
                  val hasValueGreaterThanOne = checkEditTextValues(edit-
TextValues)
                  if (hasValueGreaterThanOne) {
                      val intent = Intent(this, ExTestActivity::class.java)
                      intent.putExtra("editTextValues", editTextValues)
                      startActivity(intent)
                  }
                  else
                  {
                      Toast.makeText(this, "Вы выбрали некорректное
количество заданий (1-10)", Toast.LENGTH_SHORT).show()}
                  }
      }
```

Также в классе `ExActivity` имеется метод `getEditTextValues`, который необходим для считывания данных из введенных пользователем данных и преобразования их в словарь, который в дальнейшем и передается в следующую активность. Подробная реализация метода `getEditTextValues` представлена в листинге 12.

Листинг 12 – Функция считывания и преобразования данных

```
private fun getEditTextValues(): HashMap<String, Int> {
        val editTextValues = HashMap<String, Int>()
        for ((index, editTextId) in editTextIds.withIndex()) {
            val editText = findViewById<EditText>(editTextId)
            val value = editText.text.toString().toIntOrNull() ?: 0
            val taskKey = "Task_${index + 1}"
            editTextValues[taskKey] = value
        }
        return editTextValues
    }
```

После ввода пользователем данных для составления теста и по нажатии на кнопку `buttonExTest` запускается активность `ExTestActivity`, ко-

торая отображает перед пользователем задания, которые он выбрал, обрабатывает введенные ответы. На экране у пользователя есть кнопка `buttonHelp`, которая отображает подсказку для задания, на котором находится пользователь в конкретный момент времени. По завершении созданного теста, пользователь попадает в активность `ExTestReusltActivity`, где перед ним отображаются количество правильных ответов из общего их числа. На рисунке 18 показаны экраны, на которых отображается тест, подсказка для конкретного задания и результаты тестирования.

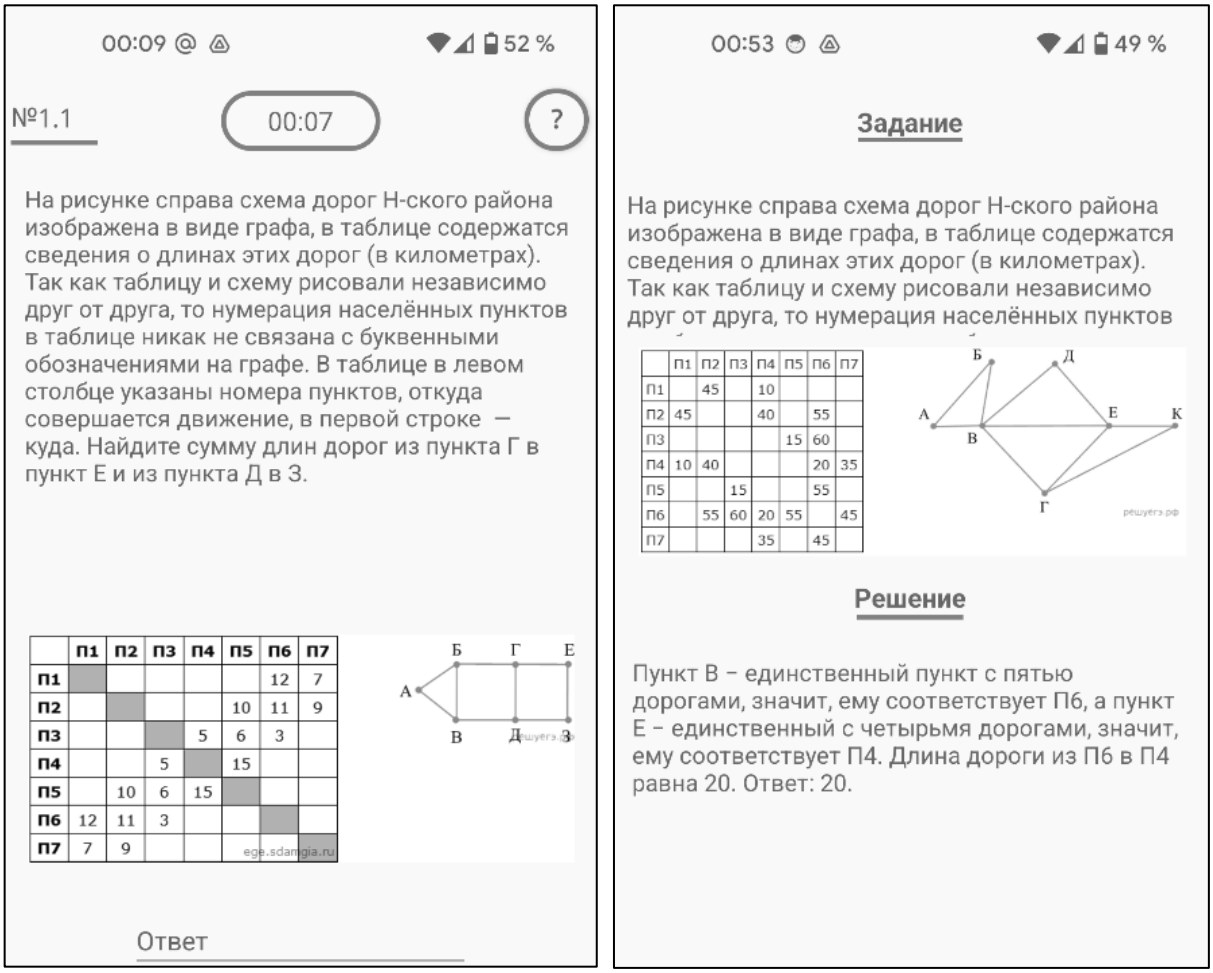

Рисунок 18 – Экран «Тест из заданий» (слева),
экран «Подсказка» (посередине)

Реализация функции для отображения заданий в тесте похожа на реализацию отображения заданий в экзаменационном варианте. Однако в отличие от экзаменационного варианта количество заданий в тесте задает сам

пользователь. Ко всему этому необходимо составить тест так, чтобы задания внутри своих тем не повторялись. Поэтому были реализованы вспомогательные функции для отображения заданий. Так в листинге 13 представлена функция для списка генерируемых заданий из полученного словаря.

Листинг 13 – Функция преобразование словаря в список

```
fun generateTaskLists(editTextValues: HashMap<String, Int>, getRandomKey:
() -> String?): Pair<List<String>, List<String>> {
        val taskNames = mutableListOf<String>()
        val taskNumbers = mutableListOf<String>()
        for ((task, value) in editTextValues) {
            repeat(value) {
                taskNames.add(task)
                val randomNumber = getRandomKey() ?: ""
                taskNumbers.add(randomNumber)
            }
}
return Pair(taskNames, taskNumbers)}
```

Чтобы задания внутри своих тем не повторялись, была написана функция генерации случайного ключа для заданий. Работа данной функции заключается в том, что она первоначально она проверяет наличие ключей в множестве `remainingKeys` и, если ключей нет, то заполняет множество доступными ключами. Затем генерирует случайный индекс в диапазоне от ноля до числа равному числу ключей в множестве `remainingKeys` и присваивает переменной `randomKey` значение из того же множества, которое имеет сгенерированный случайным образом индекс. Полная реализация этой функции представлена в листинге 14.

Листинг 14 – Функция генерации случайного ключа для заданий

```
fun getRandomKey(): String? {
        if (remainingKeys.isEmpty()) {
            remainingKeys.addAll(numberKeys)
        }
        val randomIndex = (0 until remainingKeys.size).random()
        val randomKey = remainingKeys[randomIndex]
        remainingKeys.removeAt(randomIndex)
        return randomKey
    }
```

Для возможности отображения подсказки с помощью информации из базы данных по нажатии на кнопку `buttonHelp` была реализована функция `loadHelpFromFirebase`. Функция получает фрагменты из Google Firebase,

которые необходимы для отображения подсказки, присваивает эти данные переменным и отправляет их значения в активность для отображения подсказок `HelpActivity`. Полная реализация функции `loadHelpFromFirebase` представлена в листинге 15.

Листинг 15 – Функция для загрузки подсказки из базы данных

```
private fun loadHelpFromFirebase(context: Context, taskKey: String, num-
berKey: String?) {
        val helpRef = FirebaseDatabase.getInstance("https://appbase-66912-
default-rtdb.europe-west1.firebasedatabase.app").refer-
ence.child(taskKey).child(numberKey!!)
            helpRef.addListenerForSingleValueEvent(object : ValueEventLis-
tener {
                override fun onDataChange(dataSnapshot: DataSnapshot) {
                if (dataSnapshot.exists()) { helpText = dataSnap-
shot.child("HelpText").getValue(String::class.java)
                    helpImg = dataSnapshot.child("HelpImg").get-
Value(String::class.java)
                    textForTask = dataSnapshot.child("Question").get-
Value(String::class.java)
                    imageUrl = dataSnapshot.child("Img").get-
Value(String::class.java)
                    val intent = Intent(context, HelpActivity::class.java)
                    intent.putExtra("helpText", helpText)
                    intent.putExtra("helpImg", helpImg)
                    intent.putExtra("textForTask", textForTask)
                    intent.putExtra("imageUrl", imageUrl)
                    startActivity(intent)}}
```

Для отображения подсказки была реализована функция `OnCreate` в классе `HelpActivity`. В ней переменным, которые отвечают за определенные `ImageBox` и `EditText`, присваиваются значения полученные в `loadHelpFromFirebse` класса `ExTestActivity`. Полная реализация функции `OnCreate` представлена в листинге 16.

Листинг 16 – Функция для отображения подсказки

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        supportActionBar?.setBackgroundDrawable(ColorDrawable(Color.par-
seColor("#FEFAE0")))
        setContentView(R.layout.activity_help)
        TaskText = findViewById(R.id.TextTask)
        TextHelp = findViewById(R.id.TextHelp)
        imageBox = findViewById(R.id.imageBox)
        imageBoxHelp = findViewById(R.id.imageBoxHelp)
        imageBox?.setVisibility(View.GONE)
        imageBoxHelp?.setVisibility(View.GONE)
        val helpText = intent.getStringExtra("helpText")
        val helpImg = intent.getStringExtra("helpImg")
        val imageUrl = intent.getStringExtra("imageUrl")
        val textForTask = intent.getStringExtra("textForTask")
        if (helpText != null) { TextHelp!!.text = helpText}
```

```
if (helpImg != null) { imageBoxHelp!!.visibility = View.VISIBLE
      Picasso.get().load(imageUrl).into(imageBoxHelp)}
if (textForTask != null) { TaskText!!.text = textForTask}
if (imageUrl != null) {
      imageBox!!.visibility = View.VISIBLE
      Picasso.get().load(imageUrl).into(imageBox)}}
```

Статистика

Чтобы отобразить статистику по вариантам экзамена, которые пользователь решал за все время использования приложения была реализована активность `StatActivity`. В ней имеются несколько функций которые считают среднее, максимальное и минимальное время и балл за попытки решения экзаменационного варианта и общее время, проведенное за решением вариантов, а также функции для отображения графиков, на которых по оси X находится число попыток, а на оси Y находится затраченное время или количество баллов, которые пользователь получил за попытки решения экзаменационных вариантов. На рисунке 19 представлен экран статистики.

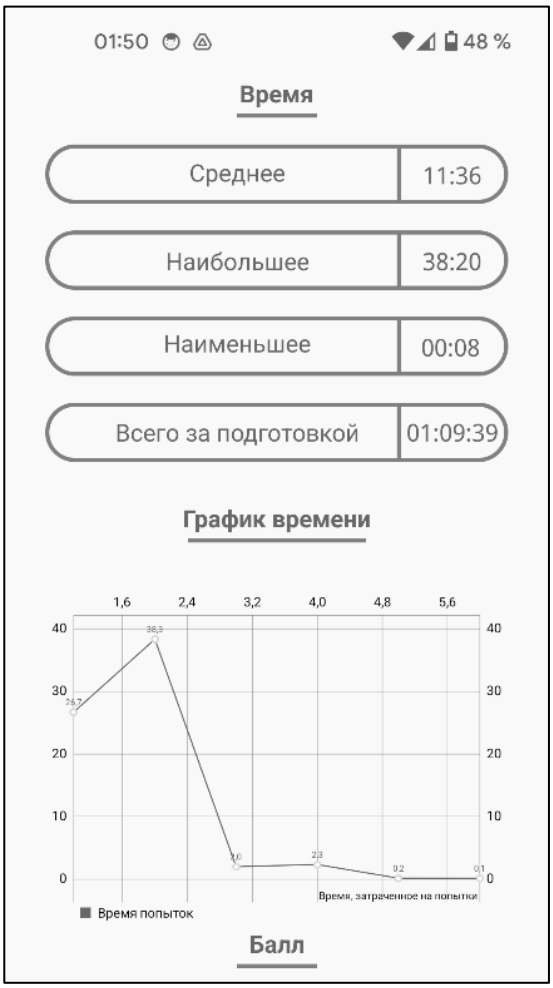

Рисунок 19 – Экран «Статистика»

Для получения статистики пользователя по баллам был использован метод addListenerForSingleValueEvent, который реализует интерфейс ValueEventListener. Данный метод слушает однократные изменения по ссылке в базе данных, которая заложена в userStatsRef, и выполняет чтение данных по попыткам решения экзаменационного теста в разделе Score, которые в свою очередь имеют уникальные идентификаторы в Google Firebase. Затем происходит вычисление и присвоение переменным среднего, наибольшего и наименьшего балла по всем попыткам решения экзаменационного теста. Далее с помощью функции showStatistics-Score, которая присваивает полученные значения о баллах определенным частям макета, в методе происходит демонстрация о статистики баллах на экран пользователю. Затем происходит составление датасета точек по количеству попыток и полученным за них баллы. Потом полученный датасет используется для отображения графика по баллам на экран пользователя. Частичная реализация отображения статистики на экране по баллам представлена в листинге 17.

Листинг 17 – Фрагмент функции отображения статистики по баллам

```
userStatsRef.addListenerForSingleValueEvent(object : ValueEventListener
          {
          val scores = mutableListOf<Long>()
          var attemptNumberScore = 1f
          override fun onDataChange(statsSnapshot: DataSnapshot)
            {
            statsSnapshot.children.forEach { attemptSnapshot ->
               val scoreStatsRef = attemptSnapshot.child("Score")
               val scoreSnapshot = scoreStatsRef.value as Long?
               if (scoreSnapshot != null) {entriesScore.add(Entry(attempt-
NumberScore,scoreSnapshot.toFloat())) attemptNumberScore++
              scores.add(scoreSnapshot)
             }
              val averageScore = scores.average().toInt()
              val maxScore = scores.maxOrNull()?.toInt() ?: 0
              val minScore = scores.minOrNull()?.toInt() ?: 0
              showStatisticsScore(averageScore,maxScore,minScore)
              val lineChartScore = findViewById<LineChart>(R.id.lineChart2)
              val dataSetScore=LineDataSet(entriesScore,"Баллы за попытки")
              val greenColor = Color.parseColor("#606C38")
              dataSetScore.color = greenColor
              dataSetScore.valueTextColor = greenColor
              val dataSetsScore: ArrayList<ILineDataSet> = ArrayList()
              dataSetsScore.add(dataSetScore)
              val lineDataScore = LineData(dataSetsScore)
              lineChartScore.data = lineDataScore
              lineChartScore.setTouchEnabled(true)
```

```
                lineChartScore.setPinchZoom(true)
                lineChartScore.description.text = "Баллы"
                lineChartScore.animateY(1000)
                }
}
```

Для получения статистики пользователя по времени был также использован метод `addListenerForSingleValueEvent`, только теперь чтение данных происходит в разделе `Time`. После чтения данных происходит их преобразование. Время, затраченное на экзаменационный тест, хранится в базе данных в миллисекундах. Код преобразует это время в более понятный формат: сначала в часы, затем в минуты, а затем в секунды. Это делает данные более понятными для пользователя и позволяет нам более точно анализировать результаты. После происходит вычисление и присвоение переменным среднего, наибольшего, наименьшего и общего времени по всем попыткам решения экзаменационного теста. Далее с помощью функции `showStatisticsTime`, которая присваивает полученные значения о времени определенным частям макета, в методе происходит демонстрация статистики о времени на экран пользователю. Далее происходит составление датасета точек по количеству попыток и затраченному на них времени. Затем полученный датасет используется для отображения графика по времени на экран пользователя. Частичная реализация отображения статистики по времени представлена в листинге 18.

Листинг 18 – Фрагмент функции отображения статистики по времени

```
userStatsRef.addListenerForSingleValueEvent(object : ValueEventListener {
    override fun onDataChange(statsSnapshot: DataSnapshot)
         {
         val times = mutableListOf<Long>()
         var attemptNumberTime = 1f
         statsSnapshot.children.forEach { attemptSnapshot ->
         val timeStatsRef = attemptSnapshot.child("Time")
         val timeSnapshot = timeStatsRef.value as Long?
         if (timeSnapshot != null)
         {
         entriesTime.add(Entry(attemptNumberTime, timeSnap-
shot.toFloat()/60000))
         attemptNumberTime++
         times.add(timeSnapshot)
                 }
         }
            val averageTime = times.average().toLong()
            val averageTimeFormatted = formatTime(averageTime)
```

```
            val maxTime = times.maxOrNull() ?: 0
            val minTime = times.minOrNull() ?: 0
            val maxTimeFormatted = formatTime(maxTime)
            val minTimeFormatted = formatTime(minTime)
            val totalTime = times.sum()
            val totalTimeFormatted = formatTime(totalTime)
            showStatisticsTime(averageTimeFormatted, maxTimeFormatted,
minTimeFormatted, totalTimeFormatted)
            val lineChartTime = findViewById<LineChart>(R.id.lineChart1)
            val dataSet = LineDataSet(entriesTime, "Время попыток")
            val greenColor = Color.parseColor("#606C38")
            dataSet.color = greenColor
            dataSet.valueTextColor = greenColor
            val dataSets: ArrayList<ILineDataSet> = ArrayList()
            dataSets.add(dataSet)
            val lineData = LineData(dataSets)
            lineChartTime.data = lineData
            lineChartTime.setTouchEnabled(true)
            lineChartTime.setPinchZoom(true)
            lineChartTime.description.text = "Время, затраченное на по-
пытки"
            lineChartTime.animateY(1000)
}
```

Выводы по четвертой главе

В данной главе был проведен обзор современных технологий, которые используются для разработки Android-приложений в 2024 году. Этот обзор включал в себя анализ различных интегрированных сред разработки (IDE), и в результате была выбрана IDE Android Studio. Также были описаны и реализованы ключевые компоненты облачной базы данных Google Firebase.

Было разработано мобильное приложение для платформы Android в соответствии с сформированными требованиям. Приложение обладает интуитивно понятным интерфейсом, позволяющим быстро находить нужную информацию, что очень важно для эффективного обучения. Кроме того, приложение обеспечивает высокий уровень безопасности и надежности хранения данных. Это было достигнуто благодаря использованию аутентификации пользователей, а это значит, что личные данные пользователей, которые будут использовать это мобильное приложение защищены, потому что только авторизованные пользователи имеют доступ к своим данным.

5. ТЕСТИРОВАНИЕ ANDROID-ПРИЛОЖЕНИЯ

5.1. Функциональное тестирование

Было проведено функциональное тестирование приложения, которое направлено на проверку реализуемости функциональных требований. Используя методологию функционального тестирования [20], проверим работоспособность мобильного приложения.

Для тестирования приложения было использовано мобильное устройство Google Pixel 5a, а также эмулятор устройства Samsung S21 Ultra. Набор тестов приведен в таблице 2.

Таблица 2 – Набор тестов для мобильного приложения

№	Функция	Шаги	Ожидаемый результат
1	Корректный запуск приложения	Запустить приложение	Приложение запустилось
2	Авторизация через Google	1. Запустить приложение. 2. Авторизоваться через Google.	Пользователь авторизован
3	Регистрация	1. Запустить приложение. 2. Нажать на кнопку «Зарегистрироваться». 3. Ввести данные. 4. Нажать на кнопку «Сохранить». 5. Нажать на кнопку «Войти».	Происходит авторизация пользователя
4	Корректное отображение экранов	1. Запустить приложение. 2. Авторизоваться. 3. Перейти на разные экраны приложения.	Экраны приложения отображаются корректно
5	Корректное отображение результатов теста	1. Запустить приложение. 2. Авторизоваться. 3. Нажать на кнопку «Экзамен».	На экране результатов отображается корректное количество баллов
6	Изменение ответа в тесте	1. Запустить приложение. 2. Нажать на кнопку «Экзамен». 3. Вписать правильный ответ в поле. 4. Перейти к следующему вопросу. 5. Вернуться к вопросу, на который был дан правильный ответ. 6. Внести новый неправильный ответ. 7. Завершить тест.	На экране результатов будет 0 баллов

Продолжение таблицы 2

№	Функция	Шаги	Ожидаемый результат
7	Корректное отображение картинок в тесте и разделе с теорией	1. Запустить приложение. 2. Авторизоваться. 3. Перейти на экран с ЕГЭ. 4. Завершить тест. 5. Перейти на экран с теорией. 6. Выбрать номер задания.	Картинки в тесте и в разделе с теоретическими материалами отображаются корректно
8	Свободное перемещение по вопросам	1. Запустить приложение. 2. Авторизоваться. 3. Перейти на экран с ЕГЭ. 4. Дойти до любого вопроса в тесте. 5. Вернуться к первому вопросу.	Доступно перемещение по любым вопросам теста
9	Регистрация	1. Запустить приложение. 2. Зарегистрироваться.	Происходит автоматическая авторизация пользователя
10	Выйти из аккаунта	1. Запустить приложение. 2. Авторизоваться. 3. Нажать на кнопку «Выйти из аккаунта».	Происходит переход на экран авторизации
11	Правильное отображение статистики пользователя	1. Запустить приложение. 2. Открыть раздел статистики.	Численная статистика, отображаемая на экране, принадлежит текущему пользователю, достоверна
12	Правильное отображения подсказки к заданию	1. Запустить приложение. 2. Открыть раздел с составлением теста по заданиям. 3. Задать любые задания. 4. Нажать на кнопку «Решать». 5. На одном из заданий нажать на кнопку подсказки.	Подсказка, которая отображается на экране относится к заданию, с которого был совершен переход
13	Корректное отображение таблицы перевода баллов в разделе «Об экзамене»	1. Запустить приложение. 2. Открыть нажать на кнопку «Об экзамене».	Таблица, которая отображается в разделе соответствует реальной таблице перевода баллов.
14	Отображение заданных заданий в созданном тесте	1. Запустить приложение. 2. Нажать на кнопку «Задания». 3. Выбрать несколько заданий для теста. 4. Нажать на кнопку «Решать».	В тест загружены те задания, которые были выбраны раннее. Задания отображаются корректно.
15	Корректное отображение теоретических материалов в выбранном разделе	1. Запустить приложение. 2. Нажать на кнопку «Теория». 3. Выбрать раздел с теоретическими материалами.	Теоретические материалы соответствуют выбранному разделу

Окончание таблицы 2

№	Функция	Шаги	Ожидаемый результат
16	Корректное отображение результатов созданного теста	1. Запустить приложение. 2. Нажать на кнопку «Задания». 3. Выбрать несколько заданий. 4. Нажать на кнопку «Решать». 5. Дать несколько правильных ответов. 6. Завершить тест.	По завершении теста в сообщении указано верное количество данных правильных ответов и верное количество общего количества заданий.
17	Завершение экзамена по окончании времени таймера	1. Запустить приложение. 2. Авторизоваться. 3. Нажать на кнопку «Экзамен». 4. Дождаться время окончания экзамена.	По истечении времени на таймере тест завершается, открывается экран с результатами. Результаты соответствуют реальным.
18	Корректное отображение графиков в статистике	1. Запустить приложение. 2. Открыть раздел статистики. 3. Пролистать экран вниз.	Отображаемые на экране статистики графики соответствуют реальной статистики. Не отличаются от численной статистики, которая представлена на этом же экране

В ходе функционального тестирования мобильного приложения было обнаружено, что полученные результаты полностью соответствуют ожиданиям. Это означает, что все тесты были успешно пройдены, и нет никаких проблем или ошибок, которые могли бы повлиять на функциональность или производительность приложения. Это важный момент, поскольку успешное прохождение всех тестов гарантирует, что мобильное приложение работает должным образом и готово к использованию.

5.2. Юзабилити-тестирование

Юзабилити-тестирование [20] – это процесс оценки удобства использования интерфейса продукта с участием конечных пользователей. Этот вид тестирования помогает определить, насколько продукт удовлетворяет ожиданиям пользователей, выявить проблемные аспекты интерфейса и понять восприятие продукта с точки зрения пользователей. Во время юзабилити-

тестирования пользователь выполняет типичные задачи с продуктом под наблюдением тестировщика.

Юзабилити-тестирование было проведено с участием друзей и знакомых людей. В общей сложности в тестировании участвовали четыре человека, которые знакомы со структурой экзамена.

Для проверки разработанного мобильного приложения участникам юзабилити-тестирования были предложены следующие задачи:

1) зарегистрироваться и войти в профиль;
2) создать и пройти тест;
3) во время прохождения созданного теста воспользоваться подсказкой в приложении;
4) пройти экзаменационный тест;
5) просмотреть свою статистику;
6) найти теоретические материалы по определенной теме.

Участники тестирования выполнили все задачи без каких-либо сложностей. В ходе данного тестирование были получены предложения по улучшениям, которые касаются создания теста. Также участники тестирования отметили удобный и интуитивно понятный интерфейс мобильного приложения. В результате тестирования было выявлено, что функционал мобильного приложения полностью соответствует разработанным функциональным требованиям.

Выводы по пятой главе

В данной главе было подготовлено 18 функциональных тестов для разработанного приложения. Все тесты были пройдены успешно. Также было проведено юзабилити-тестирование для конечных пользователей, значит разработанного мобильного приложения. Все тесты были пройдены успешно.

6. ВЕБ-ИНТЕРФЕЙС АДМИНИСТРАТОРА

6.1. Проектирование

Разрабатываемый веб-интерфейс администратора представляет собой сайт для добавления новых заданий и редактирования старых заданий ЕГЭ по информатике в базе данных Google Firebase. Для работы с сервисом пользователь должен авторизоваться. После авторизации пользователю будет доступен функционал для добавления и редактирования заданий ЕГЭ. Регистрация в данном веб-интерфейсе не предусмотрена, потому что пользователей, которые могут добавлять задания в базу данных, должен назначать администратор.

Веб-интерфейс построен на базе архитектуры MVT (Model-View-Template). На рисунке 20 представлена диаграмма архитектуры MVT.

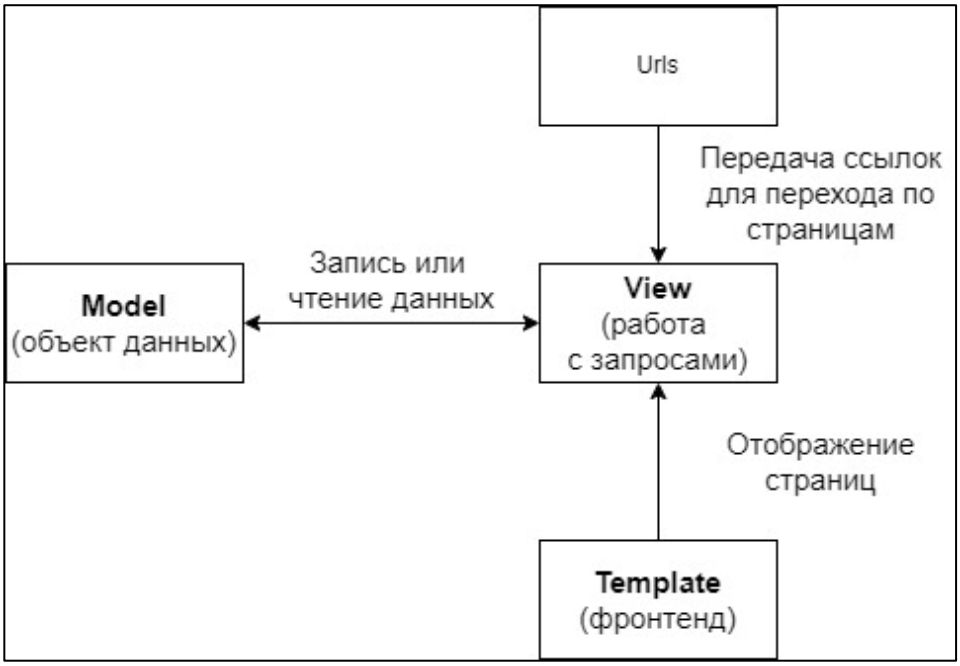

Рисунок 20 – Диаграмма архитектуры MVT

Model, View и Template (модель, шаблон и представление) – это набор из трех важных компонентов, которые в совокупности обеспечивают работу веб-интерфейса.

1. Модель помогает обрабатывать базу данных. Это уровень доступа к данным, который обрабатывает данные.

2. Шаблон – это уровень представления, который полностью обрабатывает часть пользовательского интерфейса.

3. Представление используется для выполнения бизнес-логики и взаимодействия с моделью для переноса данных и отображения шаблона.

На рисунке 21 представлена детальная диаграмма основных компонентов веб-интерфейса администратора, используя шаблон архитектуры MVT.

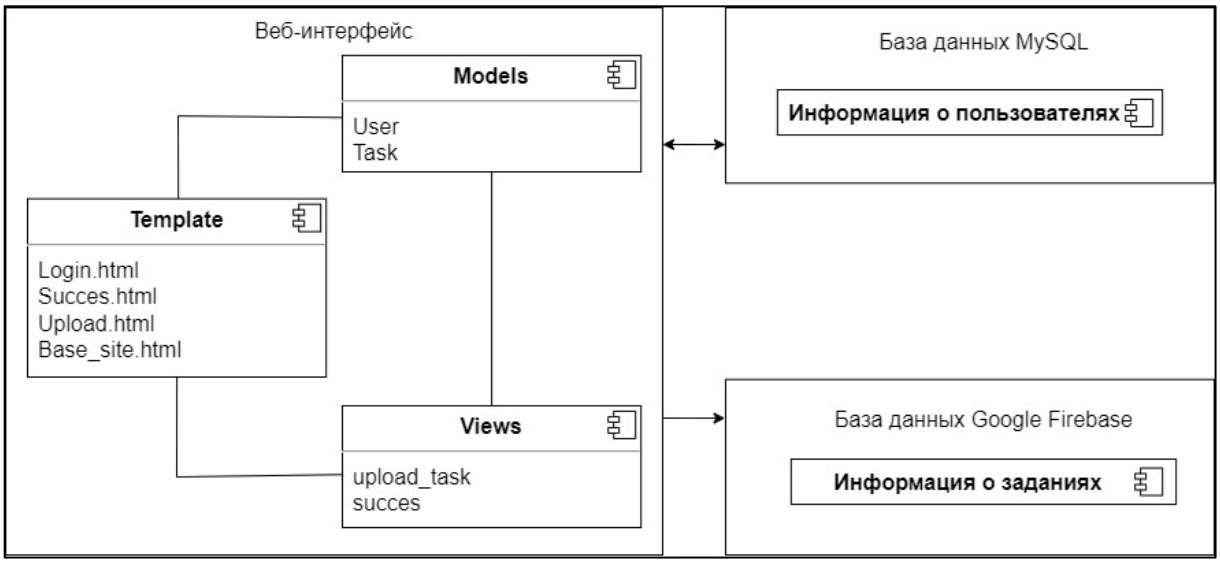

Рисунок 21 – Диаграмма основных компонентов веб-интерфейса

Анализ требований

В ходе анализа требований к веб-интерфейсу администратора были сформированы варианты использования системы.

1. Добавить задание – пользователь может добавить новое задание в Realtime Database.

2. Добавить пользователя – пользователь может добавить нового модератора, который сможет добавлять задания.

3. Авторизоваться – пользователь может пройти процедуру аутентификации.

6.2. Реализация

Технологии для реализации веб-интерфейса администратора

За основу веб-интерфейса для администратора был выбран фреймворк Django [21], который работает на языке Python. Django, созданный в 2005 году, до сих пор остается одним из наиболее широко используемых веб-фреймворков в мире. Он облегчает решение стандартных задач, таких как подключение к базе данных, обработка пользовательских данных, сохранение файлов, загруженных пользователем, и генерация шаблонов. Кроме того, Django предлагает настраиваемую панель администратора, что помогает легко реализовать аутентификацию и добавление новых сущностей.

В качестве среды для разработки было принято решение использовать PyCharm 2020.2.1 Community Edition [22]. Из плюсов IDE можно выделить:

1) тесная интеграция с Django;
2) простая и понятная организация проектов;
3) встроенный редактор для HTML и JavaScript файлов;
4) удобный интерфейс идентичный интерфейсу в Android Studio.

Реализация веб-интерфейса администратора

На момент реализации веб-интерфейс не загружен на хостинг, поэтому доступ на виртуальный сервер интерфейса осуществляется, только при помощи файла `manage.py`, который запускается через командную строку. Запуск виртуального сервера продемонстрирован на рисунке 22.

```
PS C:\Users\delsi\WebApp\webapp> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 20, 2024 - 14:34:26
Django version 3.2.25, using settings 'webapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 22 – Запуск виртуального сервера

При первом переходе в веб-интерфейс пользователь попадает на страницу авторизации, где пользователю необходимо пройти процедуру аутентификации для пользования основным функционалом веб-интерфейса. На рисунке 23 представлена форма для авторизации на сайте.

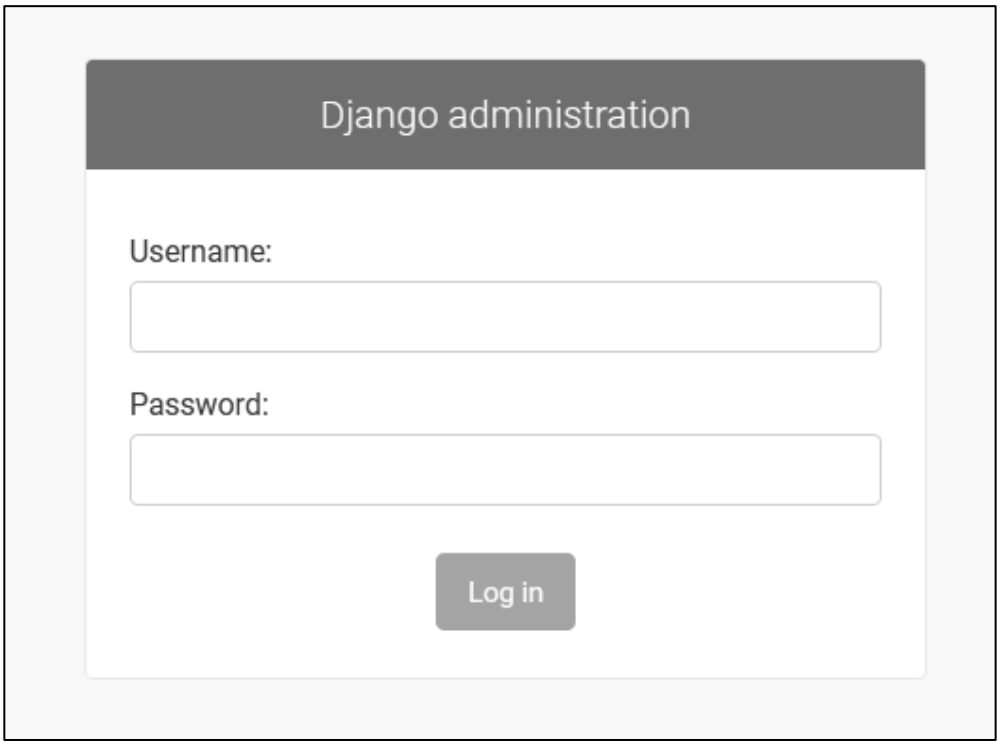

Рисунок 23 – Форма для авторизации пользователя

После успешной авторизации пользователь попадает в администраторскую панель. Форма авторизации и администраторская панель уже реализована во фреймворке Django вместе со всем необходимым функционалом. На рисунке 24 представлена администраторская панель.

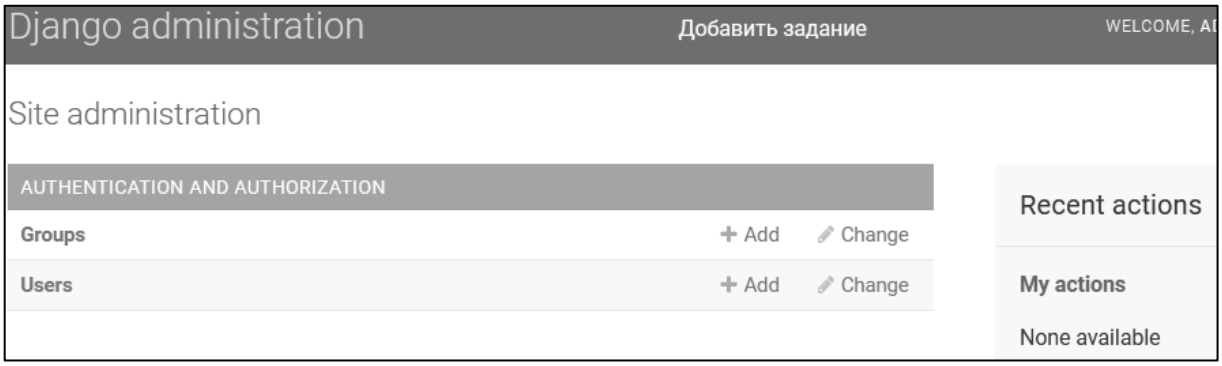

Рисунок 24 – Администраторская панель

Обработка заданий для загрузки их через веб-форму и сохранения данных в Realtime Database происходит при помощи функции `upload_task(request)`. Если метод запроса, который приходит в функцию `POST`, то создается экземпляр формы `TaskForm` с данными из запроса. Если форма валидна, то из экземпляра формы извлекаются данные, такие как вопрос, правильный ответ, оценка, текст подсказки, ссылки на изображения и файл. Затем формируются данные задания в виде словаря. Определяются папки для хранения задания в Firebase на основе номера задания и количества заданий. Данные задания отправляются в Firebase с использованием HTTP-запроса `PUT`. Если запрос успешен (статус 200), происходит перенаправление на страницу успеха. Полная реализация функции описана в листинге 19.

Листинг 19 – Функция обработки формы и отправки задания в базу данных

```
def upload_task(request):
    if request.method == 'POST':form = TaskForm(request.POST)
        if form.is_valid():
            task_data = {
                "Question": form.cleaned_data['Question'],
                "Correct_Answer": form.cleaned_data['Correct_Answer'],
                "Score": form.cleaned_data['Score'],
                "HelpText": form.cleaned_data['HelpText'],}
            image_link = form.cleaned_data['Img']
            task_data["Img"] = image_link
            help_image_link = form.cleaned_data['HelpImg']
            task_data["HelpImg"] = help_image_link
            file_link = form.cleaned_data['FileLink']
            task_data["FileLink"] = file_link
            task_folder = f"Task_{form.cleaned_data['task_number']}"
            task_subfolder = f"Number_{form.cleaned_data['task_count']}"
            response = requests.put(f"{data-
base_url}/{task_folder}/{task_subfolder}.json", json=task_data)
            if response.status_code == 200: return redirect('sucess')else:
form = TaskForm()
    return render(request, 'upload_task.html', {'form': form})
```

На рисунке 25 представлена форма, включающая в себя различные поля, необходимые для добавления нового задания в базу данных или обновления существующего задания. Поля формы включают в себя номер задания в ЕГЭ, порядковый номер задания в базе, текст задания, ответ, количество баллов, текст подсказки и ссылки на изображения или файлы, связанные с заданием. Важно отметить, что в форме предусмотрена валидация

данных, которая помогает гарантировать, что введенная информация соответствует необходимым требованиям.

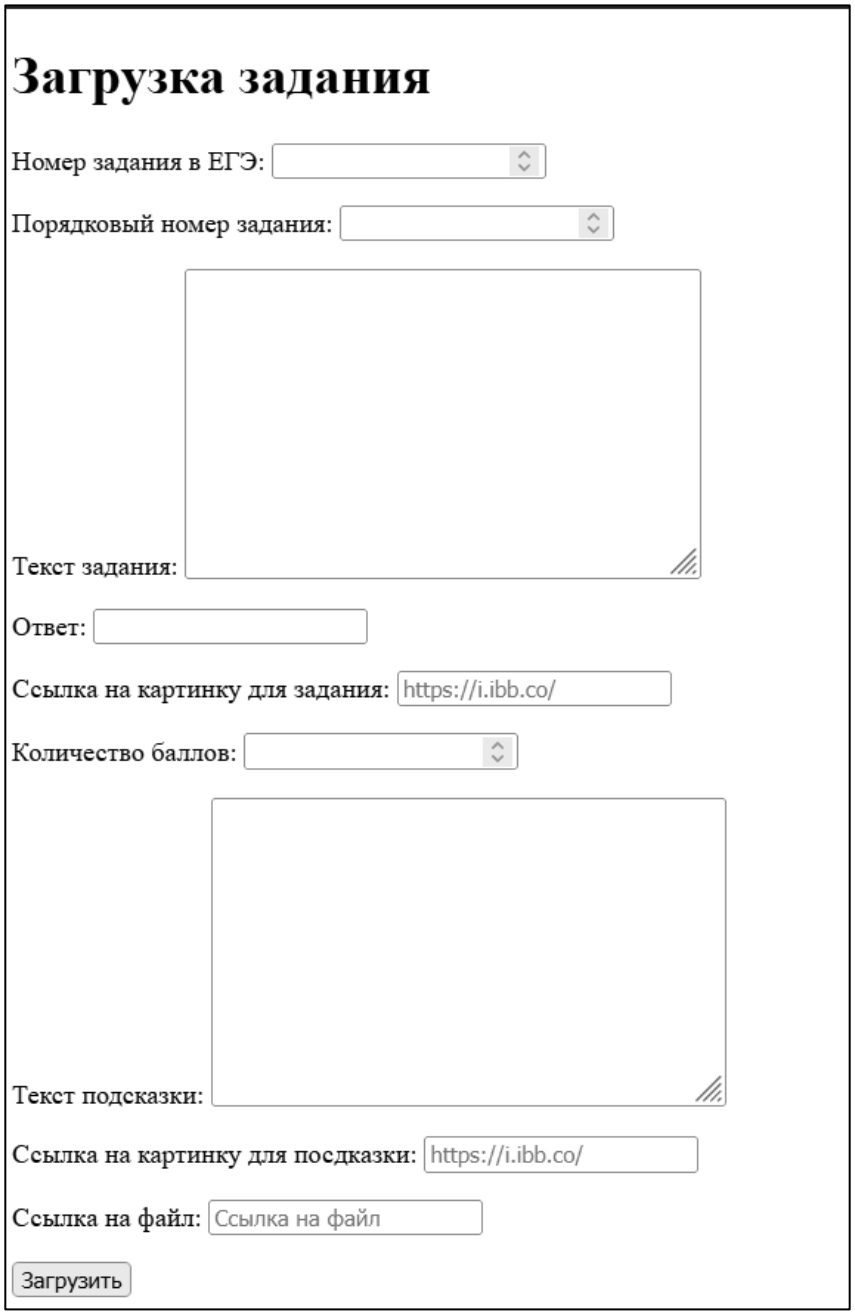

Рисунок 25 – Форма отправки задания

Функция `success(request)` необходима для отображения страницы, которая сигнализирует о том, что задание успешно загружено в базу данных. Реализация функции `success(request)` представлена в листинге 20.

Листинг 20 – Функция для отображения страницы-уведомления

```
def sucess(request):
    return render(request, 'sucess.html')
```

На рисунке 26 представлена страница-уведомление, информирующая пользователя о том, что загрузка задания ЕГЭ по информатике в базу данных Google Firebase прошла успешно. Эта страница появляется после того, как пользователь завершает процесс добавления задания, подтверждая, что все данные были корректно загружены и сохранены в облачное хранилище.

Рисунок 26 – Страница-уведомление об успешности загрузки задания

После нажатия на кнопку «Назад» пользователь вновь, возвращается на форму загрузки задания. Чтобы поля формы очищались и у пользователя не возникало ощущение, что задание не отправилось на странице upload_task.html с помощью JavaScript [23] были реализованы две функции, которые очищают поля и не позволяют браузеру на компьютере заполнять эти поля из буфера. Реализация этих функций представлена в листинге 21.

Листинг 21 – Функции отображения формы при переходе «Назад»

```
window.onload = function() {
        document.getElementById("TaskForm").reset();
    }
    window.onbeforeunload = function() {
        document.getElementById("TaskForm").reset();
    }
```

Класс `TaskForm` отвечает за форму для загрузки в базу данных задания и имеет поля для присваивания введенных пользователем данных. В свою очередь эти поля в дальнейшем использует функция `upload_task`, чтобы передать информацию о задании в базу данных. Реализация класса формы задания представлена в листинге 22.

Листинг 22 – Класс формы задания

```
class TaskForm(forms.Form):
 task_number = forms.IntegerField(label='Номер задания в ЕГЭ')
 task_count = forms.IntegerField(label='Порядковый номер задания')
 Question = forms.CharField(widget=forms.Textarea, label='Текст задания')
 Correct_Answer = forms.CharField(label='Ответ')
```

```
 Img = forms.CharField(widget=forms.TextInput(attrs={'placeholder':
'https://i.ibb.co/'}), label='Ссылка на картинку для задания', re-
quired=False)
 Score = forms.IntegerField(label='Количество баллов')
 HelpText = forms.CharField(widget=forms.Textarea, label='Текст подсказки')
 HelpImg= forms.CharField(widget=forms.TextInput(attrs={'placeholder':
'https://i.ibb.co/'}), label='Ссылка на картинку для посдказки', re-
quired=False)
 FileLink = forms.CharField(widget=forms.TextInput(attrs={'placeholder':
'Ссылка на файл'}), label='Ссылка на файл', required=False)
```

6.3. Тестирование

Было проведено функциональное тестирование веб-интерфейса администратора, которое направлено на проверку реализуемости функциональных требований, предъявляемых к системе. Для проведения тестирования веб-интерфейса использовались два устройства с различными характеристиками экранов. Первым устройством был ноутбук со стандартным разрешением экрана 1920x1080 пикселей и диагональю экрана 16 дюймов. Вторым устройством был домашний компьютер, оснащенный монитором с более высоким разрешением экрана 2560x1440 пикселей и диагональю экрана 27 дюймов. Набор тестов приведен в таблице 3.

Таблица 3 – Набор тестов для веб-интерфейса

№	Функция	Шаги	Ожидаемый результат
1	Корректный запуск сайта	Открыть веб-интерфейс по локальной ссылке, которая появляется в терминале.	Сайт открылся
2	Авторизация	1. Открыть веб-интерфейс по локальной ссылке. 2. Авторизоваться в веб-интерфейсе помощью логина и пароля супер-пользователя.	Пользователь авторизован
3	Добавление задания в базу данных	1. Открыть сайт. 2. Заполнить необходимые поля формы для добавления задания. 3. Нажать на кнопку «Отправить».	После открытия сайта запускается панель администратора. После отправки задания открывается страница, где есть текст, который сигнализирует отправке задания.
4	Корректное работа формы загрузки задания после перехода «Назад».	1. Открыть сайт. 2. Перейти на форму отправки задания. 3. Заполнить необходимые поля для отправки задания. 4. Нажать на кнопку «Назад».	Открывается форма, полностью очищенная от введенных ранее данных.

Окончание таблицы 3

№	Функция	Шаги	Ожидаемый результат
5	Валидация данных в форме	1. Открыть сайт. 2. Нажать на кнопку «Добавить задание». 3. Ввести в поля формы некорректные данные. 1. Нажать кнопку «Отправить».	Над неправильно заполненными полями появляется уведомление о том, что их нужно заполнить, либо исправить те данные, что уже имеются.
6	Добавление модератора	2. Открыть сайт. 3. Авторизоваться под данными супер-пользователя. 4. Нажать на кнопку добавления нового пользователя. 5. Ввести необходимые данные для регистрации модератора. 6. Нажать на кнопку «Добавить». 7. Выйти из аккаунта администратора. 8. Авторизоваться с данными модератора.	Открывается администраторская панель, но без возможности добавлять новых пользователей.
7	Корректное отображение всех страниц веб-интерфейса	1. Открыть веб-интерфейс. 2. Авторизоваться как супер-пользователь. 3. Перейти на страницу добавления задания. 4. Заполнить необходимые поля формы. 5. Выйти на страницу с панелью администратора. 6. Перейти на страницу добавления нового пользователя.	Текст на всех страницах читаемый, верстка страниц не нарушена.

В ходе функционального тестирования сайта не было выявлено ошибок. Все ожидания из набора тестов соответствуют действительности.

Выводы по шестой главе

В данной главе был спроектирован на основе шаблона архитектуры MVT и реализован при помощи фреймворка Django на языке Python веб-интерфейс администратора. Все функции веб-интерфейса успешно прошли тесты.

ЗАКЛЮЧЕНИЕ

В рамках данной работы были разработаны Android-приложения для подготовки к ЕГЭ по информатике и веб-интерфейс ввода заданий ЕГЭ для реализованного мобильного приложения. При этом были решены следующие задачи:

1) изучены методы создания приложений на мобильной операционной системе Android;

2) изучены методы создания веб-приложений с использованием фреймворка Django;

3) проведен анализ предметной области и сделан обзор существующих приложений по данной тематике;

4) спроектирована база данных для хранения информации приложения;

5) спроектировано и реализовано мобильное приложение;

6) протестирована работа реализованного приложения;

7) спроектирован, реализован и протестирован веб-интерфейс администратора.

Также был изучен язык программирования Kotlin для разработки Android-приложений.

Планируется дальнейшее развитие проекта, включающее в себя добавление следующих новых функций в приложение: онлайн режим между пользователями, возможность выбора учеников для репетиторов и преподавателей и назначение для них тестов. Разработанное приложение планируется разместить в магазине RuStore и HuaweiGallery. Также планируется улучшить дизайн и расширить функционал веб-интерфейса администратора возможностью редактировать теорию в мобильном приложении.

ЛИТЕРАТУРА

1. Вся статистика интернета и соцсетей на 2023 год – цифры и тренды в мире и в России. [Электронный ресурс] URL: https://www.web-canape.ru/business/vsya-statistika-interneta-i-socsetej-na-2022-god-cifry-i-trendy-v-mire-i-v-rossii/ (дата обращения: 10.02.2024 г.).

2. Шкарбан Ф.В., Авдиль С.Л., Эльвердинов Д.Э. Актуальность разработки мобильных приложений. // Информационно-компьютерные технологии в экономике, образовании и социальной сфере, 2017. – С. 176–184.

3. Колесников В.Н., Мельник Ю.И., Теплова Л.И. Мобильный телефон в учебной деятельности современного старшеклассника и студента. // Непрерывное образование: XXI век, 2018. – С. 5–6.

4. Google Play: ОГЭ, ЕГЭ информатика – программирование на Python. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.divschool.pythonprepare02 (дата обращения: 10.02.2024 г.).

5. Google Play: Информатика ЕГЭ. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.egeinf.ithandbook (дата обращения: 10.02.2024 г.).

6. Google Play: ЕГЭ по информатике 2023. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=amid.oge.egeinf (дата обращения: 12.02.2024 г.).

7. Вигерс К., Битти Д. Разработка требований к программному обеспечению // Санкт-Петербург: «Русская редакция», 2014. – 736 с.

8. Firebase. [Электронный ресурс] URL: https://firebase.google.com (дата обращения: 14.02.2024 г.).

9. Иванов Д. Моделирование на UML. / Д. Иванов, Ф. Новиков. // Санкт-Петербург: НИУ ИТМО, 2010. – 200 с.

10. Решу ЕГЭ. [Электронный ресурс] URL: https://inf-ege.sdamgia.ru/ (дата обращения 16.02.2024 г.).

11. Figma. [Электронный ресурс] URL: https://www.figma.com/ (дата обращения 01.04.2024 г.).

12. СУБД (Система управления базами данных) [Электронный ресурс] URL: https://itglobal.com/ru/company/glossary/subd-sistema-upravleniya-bazami-dannyh/ (дата обращения: 14.02.2024 г.).

13. База данных и СУБД: основные понятия и определения. [Электронный ресурс] URL: https://oracle-patches.com/db/3205-база-данных-и-субд-основные-понятия (дата обращения: 14.02.2024 г.).

14. SQLite. [Электронный ресурс] URL: https://www.sqlite.org/ (дата обращения: 14.02.2024 г.).

15. Realm. [Электронный ресурс] URL: https://realm.io (дата обращения: 16.02.2024 г.).

16. Смольянов А.Г. Краткий обзор мобильной СУБД Realm для Google Android / А.Г. Смольянов, В.В. Ивановичев // Международный научный журнал «Символ Науки, 2016. – 35 с.

17. Android Studio. [Электронный ресурс] URL: https://developer.android.com/studio/features.html (дата обращения: 12.03.2024 г.).

18. Eclipse. [Электронный ресурс] URL: https://eclipse.org (дата обращения: 12.03.2024 г.).

19. React Native. [Электронный ресурс] URL: https://reactnative.dev (дата обращения: 12.03.2024 г.).

20. Басок Б.М. Системы тестирования программного обеспечения // Москва: РТУ МИРЭА, 2021. – 47 с.

21. Django. [Электронный ресурс] URL: https://www.djangoproject.com/ (дата обращения: 22.05.2024 г.).

22. PyCharm. [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения 22.05.2024 г.).

23. Хантер И.Т. Многопоточный JavaScript / И.Т. Хантер, Б. Инглиш // ДМК Пресс, 2022. – 188 с.