|  |  |
| --- | --- |
|  | Министерство образования Калининградской области  Государственное бюджетное учреждение Калининградской области профессиональная образовательная организация  «Колледж информационных технологий и строительства» |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Разработка мобильного приложения «Магические рецепты»**

Выполнила:обучающаяся группы ИСп 20-1 Лаптева Ирина Константиновна

Специальность: 09.02.07 «Информационные системы и программирование»

(код, наименование)

Руководитель ВКР:

Большакова-Стрекалова А.В.

(Ф.И.О. преподавателя)

|  |  |
| --- | --- |
| Выпускная квалификационная работа допущена к защите  « » 20 г.  Заместитель директора по учебно – методической работе  \_\_\_\_\_\_\_\_\_\_ / Павленко Г.Я.  (подпись) (Ф.И.О) | Рецензент:      (Ф.И.О., место работы)  Председатель ГЭК: Наконечный А.Н.  (Ф.И.О.) |

Калининград

2024 г

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**для выполнения выпускной квалификационной работы**

Специальность 09.02.07 «Информационные системы и программирование» Группа ИСп 20-1

Ф.И.О. обучающаяся Лаптева Ирина Константиновна

Тема ВКР Разработка мобильного приложения «Магические рецепты»

Дата выдачи задания: «22» апреля 2024 г.

Работа должна быть сдана не позднее «15» июня 2024 г.

**Перечень вопросов, подлежащих разработке в ВКР**

1. Подготовка материала и оформление технического задания и последующего технического проекта по теме дипломной работы.

2. Подбор и анализ материалов для определения того нового, что будет разрабатываться в дипломной работе.

3. Анализ и проектирование структуры и дизайна мобильного приложения.

4. Программная реализация мобильного приложения.

5. Подготовка методического обеспечения.

6. Расчет себестоимости разработки.

Графическая часть (при наличии):

Архитектура приложения, схема базы данных, ERD-диаграмма, диаграммы UML.

Руководитель ВКР: Большакова-Стрекалова А.В.

(подпись) (Ф.И.О.)

Задание получила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лаптева И.К.

(подпись) (Ф.И.О.)

«22» апреля 2024 г.

Содержание

[Введение 4](#_Toc168347319)

[1 Анализ информации и постановка задачи 5](#_Toc168347320)

[1.1 Структура и особенности разработки мобильного приложения 5](#_Toc168347321)

[1.2 Анализ существующих аналогов 6](#_Toc168347322)

[1.3 Обзор и обоснование выбранных средств разработки 8](#_Toc168347323)

[1.4 Техническое задание на разработку 10](#_Toc168347324)

[2 Проектирование мобильного приложения 13](#_Toc168347325)

[2.1 Требования к мобильному приложения 13](#_Toc168347326)

[2.2 Варианты сценариев использования мобильного приложения 14](#_Toc168347327)

[2.3 Архитектура и компоненты мобильного приложения 16](#_Toc168347328)

[2.4 UX-дизайн мобильного приложения 18](#_Toc168347329)

[2.5 Создание ER-диаграммы базы данных 33](#_Toc168347330)

[3 Разработка мобильного приложения 34](#_Toc168347331)

[3.1 Реализация базы данных 34](#_Toc168347332)

[3.2 Создание приложения 36](#_Toc168347333)

[3.3 Методическое обеспечение 36](#_Toc168347334)

[3.3.1 Руководство для программиста 36](#_Toc168347335)

[3.3.2 Руководство пользователя 44](#_Toc168347336)

[3.4 Тестирование приложения 57](#_Toc168347337)

[4 Экономическая часть 60](#_Toc168347338)

[Заключение 72](#_Toc168347339)

[Список используемых источников 73](#_Toc168347340)

[Приложение. Листинг кода 74](#_Toc168347341)

Введение

Технологии развиваются и оказывают сильно влияние на жизнь людей, которое нельзя отрицать. Данная выпускная квалификационная работа посвящена разработке мобильного приложения «Магические рецепты» на платформе Android Studio с использованием языка программирования Java.

Целью данной работы является создание мобильного приложения, позволяющего любителям готовить искать и сохранять разнообразные рецепты блюд.

Основные задачи, необходимые для решения в рамках выпускной квалификационной работы:

* разработка базы данных с помощью Firebase, которая будет хранить информацию о пользователе (его личные данные и избранное) и рецепты блюд;
* подготовку дизайна приложения в тёмной и светлой теме, что позволит пользователям выбрать наиболее комфортный для них вариант интерфейса;
* создание формы авторизации, которая обеспечит доступ к функционалу приложения только зарегистрированным пользователям;
* реализацию функции добавления рецептов блюд в избранное только для авторизованных пользователей, что повысит удобство и персонализацию использования приложения.

В рамках данной работы будут рассмотрены основные этапы разработки приложения, начиная с анализа требований и проектирования базы данных, и заканчивая тестированием и внедрением готового продукта.

Ссылка на GitHub – https://github.com/Jittox/MagicRecipes

1 Анализ информации и постановка задачи

1.1 Структура и особенности разработки мобильного приложения

Разработка мобильного приложения «Магические рецепты» включает в себя несколько ключевых этапов, начиная от исследования и планирования и заканчивая запуском и поддержкой. Каждый этап имеет свои уникальные требования и задачи, которые необходимо решить для успешной реализации проекта. В данном разделе описывается структура и основные аспекты разработки мобильного приложения. [6]

Структура разработки мобильного приложения [7]:

* исследование и планирование: на этом этапе проводится анализ рынка кулинарных приложений, изучаются доступные инструменты и технологии, формулируются основные функции и цели приложения. Это помогает в создании детального технического задания, которое станет основой для последующих этапов разработки;
* проектирование интерфейса (UX дизайн): разработка дизайна пользовательского интерфейса, который должен быть интуитивно понятным и удобным. На этом этапе создаются модели экранов и разрабатываются сценарии взаимодействия пользователя с приложением, что позволяет обеспечить положительный пользовательский опыт;
* техническая разработка: на этом этапе реализуются функции приложения, используя выбранные технологии и языки программирования, такие как Android Studio и Java. Особое внимание уделяется интеграции с Firebase для хранения данных о пользователях и рецептах блюд. [2]

Особенности разработки мобильного приложения:

* приложение должно обеспечивать хранение и обработку персональных данных пользователей, включая их личные данные и избранные рецепты. Для этого необходима надежная система управления данными с использованием Firebase;
* дизайн приложения должен быть адаптирован под темную и светлую темы, чтобы пользователи могли выбрать наиболее комфортный для них вариант интерфейса. Это повышает удобство и привлекательность использования приложения;
* разработка формы авторизации является важным аспектом, так как добавление рецептов в избранное доступно только для авторизованных пользователей. Это позволяет защитить личные данные пользователей и обеспечить их конфиденциальность.

В результате выполнения всех этих этапов будет создано мобильное приложение «Магические рецепты», которое предоставит пользователям удобный доступ к разнообразным рецептам, возможность создания персонализированных подборок блюд и приятный опыт использования благодаря тщательно продуманному дизайну и функционалу.

1.2 Анализ существующих аналогов

Перед разработкой мобильного приложения «Магические рецепты» необходимо провести анализ существующих аналогов, чтобы понять текущие тенденции на рынке кулинарных приложений, выявить сильные и слабые стороны конкурентов и определить ключевые функции, которые будут востребованы пользователями. Такой анализ поможет разработать уникальное и конкурентоспособное приложение, удовлетворяющее потребности целевой аудитории.

Анализ конкурентов:

1. Cookpad

Cookpad — это популярное мобильное приложение для обмена рецептами еды, которое имеет ряд преимуществ, но также существуют и некоторые недостатки.

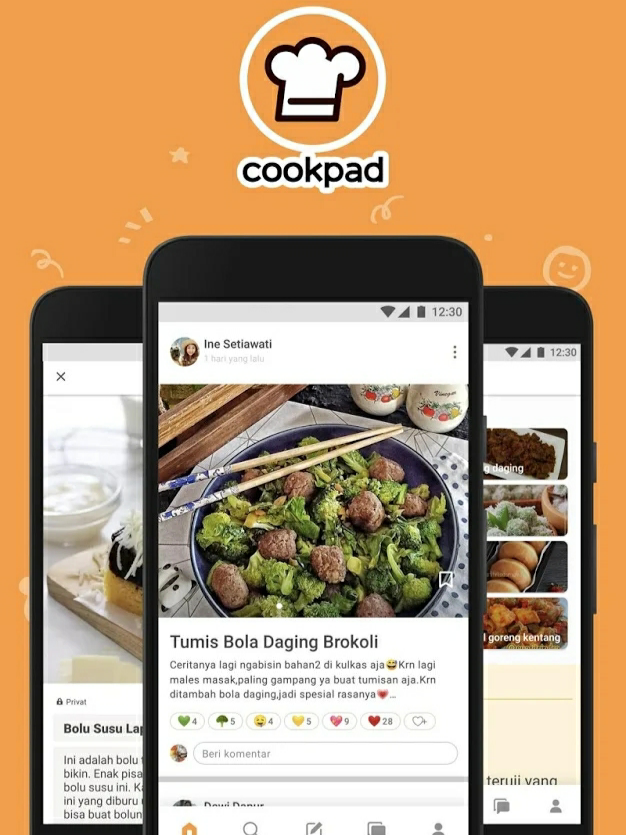


Рисунок 1 – Cookpad

Преимущества:

* широкая база рецептов;
* удобство использования.

Недостатки:

* реклама;
* ограниченная функциональность;
* неудобство с рекламными баннерами, которые мешают просмотру рецептов;
* недостаток контроля качества;
* с 24 февраля 2023 года больше не обслуживает российское сообщество.

1. Allrecipes

Allrecipes – популярный ресурс с рецептами еды, который имеет ряд преимуществ, но также существуют и некоторые недостатки.

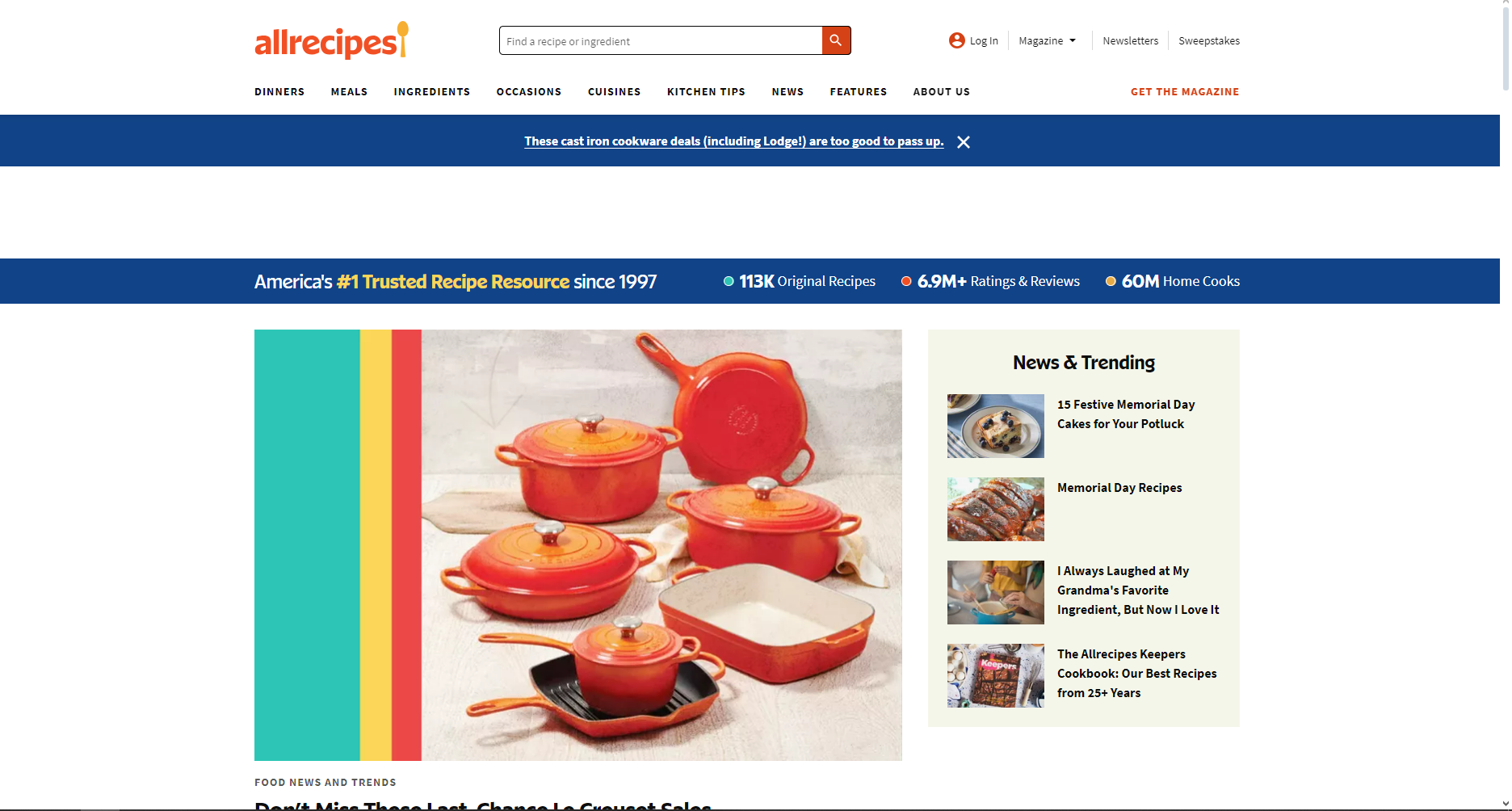


Рисунок 2 – главная страница Allrecipes

Преимущества:

* оценки и отзывы к рецептам;
* пользовательский контент;
* интерактивные функции (список покупок, пошаговое приготовление);
* бесплатное использование.

Недостатки:

* неудобство в навигации;
* неруссифицировано;
* медленная производительность;
* зависимость от пользовательского контента;
* неэффективная поддержка.

1.3 Обзор и обоснование выбранных средств разработки

Технологический анализ приложения, разрабатываемого с использованием Firebase, Android Studio и Java, позволяет оценить его технические возможности, производительность и масштабируемость. Рассмотрим каждый из этих инструментов:

Firebase:

* Firebase обеспечивает возможность работы с данными в режиме реального времени, что позволяет вашему приложению моментально реагировать на изменения данных и обновлять интерфейс пользователя [2];
* Firebase предлагает гибкую структуру данных и масштабируемость, что делает его подходящим выбором для приложений с разнообразными потребностями в хранении и обработке информации;
* Firebase предоставляет набор инструментов и API для интеграции с платформой Android, что упрощает разработку и поддержку приложения на этой платформе.

Android Studio:

* Android Studio предоставляет мощную и удобную среду разработки для создания приложений под платформу Android с использованием языка Java;
* Android Studio предлагает широкий спектр инструментов для разработки, от редактора кода и отладчика до визуальных редакторов пользовательского интерфейса и профилировщика производительности;
* Android Studio активно обновляется и поддерживает последние технологии и стандарты разработки Android-приложений, что позволяет использовать новейшие функции и возможности платформы.

Java:

* платформа с открытым исходным кодом: Java является одним из наиболее популярных и широко используемых языков программирования, с открытым исходным кодом и обширным сообществом разработчиков;
* Java обеспечивает поддержку многопоточности и позволяет создавать кроссплатформенные приложения, что полезно для разработки Android-приложений, работающих на разных устройствах и версиях операционной системы;
* Java имеет обширную экосистему библиотек и фреймворков, что упрощает разработку и добавление функциональности в приложение.

Figma:

* Figma предоставляет мощные возможности для проектирования пользовательских интерфейсов, позволяя создавать интерактивные прототипы и макеты;
* Figma поддерживает совместную работу в реальном времени, что облегчает взаимодействие между дизайнерами и разработчиками, позволяя вносить изменения и обсуждать их на ходу;
* Figma работает в браузере, что делает ее доступной на любой операционной системе и позволяет легко делиться проектами и ресурсами с командой.

Совокупное использование Firebase, Android Studio, Java и Figma обеспечивает приложению надежные инструменты для разработки, хранения данных в реальном времени и создания интерактивного пользовательского интерфейса.

1.4 Техническое задание на разработку

1. Введение

Необходимо разработать мобильное приложение для платформы Android с использованием интегрированной среды разработки Android Studio и языка программирования Java. Приложение должно предоставлять возможность регистрации, аутентификации пользователей и хранения информации о них и рецептах блюд в базе данных Firebase. Дизайн приложения должен быть подготовлен в тёмной и светлой темах с использованием Figma.

1. Функциональные требования

* регистрация нового пользователя с использованием электронной почты и пароля;
* авторизация зарегистрированных пользователей;
* хранение личных данных пользователей (имя, электронная почта и т.д.) и их избранных рецептов в базе данных Firebase;
* добавление рецептов блюд из готовых источников в избранное только для авторизованных пользователей.

1. Нефункциональные требования

* приложение должно быть разработано с использованием интегрированной среды разработки Android Studio и языка программирования Java;
* для хранения данных о пользователях и рецептах блюд должна использоваться база данных Firebase;
* дизайн приложения должен быть подготовлен в двух темах: светлой и тёмной, с использованием инструмента дизайна Figma;
* приложение должно быть доступно на устройствах с операционной системой Android версии не ниже 5.0 (API 21);
* приложение должно обеспечивать безопасную передачу данных и аутентификацию пользователей.

1. Дополнительные требования

* для интерфейса приложения следует использовать современные UX принципы и рекомендации платформы Android;
* все пользовательские действия, такие как регистрация, авторизация, добавление рецептов в избранное и т.д., должны быть интуитивно понятными и удобными для использования.

1. Дизайн

* дизайн интерфейса приложения должен быть разработан в тёмной и светлой темах с использованием инструмента дизайна Figma;
* должны быть предоставлены макеты экранов регистрации, авторизации, профиля пользователя, экрана со списком рецептов и экрана с деталями рецепта.

1. Управление проектом

* каждый этап разработки должен быть документирован;
* необходимо предоставить отчёт по выполненной работе и промежуточные результаты на каждом этапе разработки.

1. Тестирование

* перед выпуском приложения в продакшен необходимо провести тестирование функциональности, безопасности и производительности;
* приложение должно быть стабильным и работоспособным на различных устройствах и разрешениях экранов.

8. Развертывание и поддержка

* после успешного тестирования и релиза приложения в продакшен необходимо обеспечить его поддержку и обновления;
* приложение должно регулярно обновляться с добавлением новых функций и улучшением существующих.

2 Проектирование мобильного приложения

2.1 Требования к мобильному приложения

Требования к мобильному приложению «Магические рецепты» [1]

1. Авторизация и регистрация:

* пользователи должны иметь возможность создать учетную запись в приложении, используя логин и пароль;
* обеспечить защиту учетных данных пользователей и безопасное хранение паролей.

1. Просмотр и поиск рецептов:

* приложение должно предоставлять пользователю доступ к обширной базе данных рецептов;
* обеспечить возможность фильтрации рецептов по различным критериям.

1. Избранные рецепты:

* пользователи должны иметь возможность добавлять понравившиеся рецепты в избранное для последующего доступа;
* обеспечить возможность управления списком избранных рецептов: удаление, добавление.

1. Интерфейс и дизайн:

* разработать интуитивно понятный и привлекательный пользовательский интерфейс;
* обеспечить поддержку светлой и темной темы для удобства использования приложения в различных условиях освещения.

1. Интеграция с Firebase:

* использовать Firebase для хранения данных о пользователях и рецептах;
* обеспечить синхронизацию данных в режиме реального времени для мгновенного обновления контента.

1. Безопасность и защита данных:

* реализовать механизмы шифрования данных для обеспечения конфиденциальности личной информации пользователей;
* обеспечить защиту от взлома и несанкционированного доступа к данным приложения.

1. Тестирование и отладка:

* провести тщательное тестирование приложения на различных устройствах и операционных системах;
* обнаруженные ошибки и недочеты должны быть исправлены до выпуска приложения в продакшн.

1. Документация и поддержка:

* предоставить документацию к приложению, включая инструкции по использованию и контактные данные для обратной связи;
* обеспечить техническую поддержку пользователей и оперативное реагирование на обращения и запросы.

2.2 Варианты сценариев использования мобильного приложения

Сценарии, охватывающие основные взаимодействия пользователей с приложением «Магические рецепты», обеспечивая удобство использования и доступ ко всем необходимым функциям. [9]

Сценарий 1: Регистрация нового пользователя

1. Цель: пользователь хочет зарегистрироваться в приложении для получения доступа ко всем функциям.
2. Действия:
   * пользователь открывает приложение и нажимает кнопку «Регистрация»;
   * появляется форма регистрации, где пользователь вводит email, логин и пароль;
   * пользователь нажимает кнопку «Зарегистрироваться»;
   * система проверяет введенные данные и создает новую учетную запись;
   * пользователь получает уведомление об успешной регистрации и перенаправляется на главный экран.

Сценарий 2: Авторизация пользователя

1. Цель: пользователь хочет войти в приложение.
2. Действия:
   * пользователь открывает приложение и вводит логин и пароль в соответствующие поля на экране входа;
   * пользователь нажимает кнопку «Вход»;
   * система проверяет учетные данные и, при успешной авторизации, перенаправляет пользователя на главный экран.

Сценарий 4: Просмотр рецепта

1. Цель: пользователь хочет просмотреть детали рецепта.
2. Действия:
   * пользователь находит рецепт в списке результатов поиска или в своей избранной коллекции;
   * пользователь нажимает на рецепт;
   * система открывает экран с детальной информацией о рецепте: название, описание, ингредиенты, пошаговые инструкции, фотографии, рейтинг;
   * пользователь может просмотреть рецепт, поставить оценку или добавить комментарий.

Сценарий 5: добавление рецепта в избранное

1. Цель: пользователь хочет сохранить рецепт в избранное.
2. Действия:
   * пользователь находит интересующий его рецепт;
   * пользователь открывает детали рецепта;
   * пользователь нажимает кнопку «Добавить в избранное»;
   * система сохраняет рецепт в коллекцию избранных рецептов пользователя;
   * пользователь может увидеть рецепт в разделе «Избранное».

2.3 Архитектура и компоненты мобильного приложения

Для мобильного приложения «Магические рецепты» с использованием интегрированной среды разработки Android Studio, языка программирования Java и базы данных Firebase, можно рассмотреть следующую архитектурную концепцию:

1. Модель-Представление-Контроллер (Model-View-Controller, MVC):

Модель (Model): этот компонент будет отвечать за управление данными. Здесь будет взаимодействовать с базой данных Firebase для получения и обновления информации о пользователях и рецептах блюд. [4]

Представление (View): в этом компоненте будет реализован пользовательский интерфейс приложения. Каждый экран приложения, включая экраны регистрации, авторизации, списка рецептов и деталей рецепта, будет представлен в соответствующем классе View. [4]

Контроллер (Controller): этот компонент будет отвечать за логику приложения и управление потоком данных между моделью и представлением. Здесь будет обрабатывать действия пользователя, такие как регистрация, авторизация и добавление рецептов в избранное. [4]

2. Использование архитектурного паттерна MVVM (Model-View-ViewModel):

Модель (Model): остается неизменной, управление данными осуществляется через базу данных Firebase.

Представление (View): здесь отображается пользовательский интерфейс приложения.

ViewModel: этот компонент будет промежуточным звеном между моделью и представлением. ViewModel будет содержать логику бизнес-логики приложения и отвечать за предоставление данных, необходимых для отображения в пользовательском интерфейсе. Он также будет обрабатывать действия пользователя и обновлять данные в модели. [4]

3. Компоненты приложения:

Активности (Activities): каждый экран приложения будет представлен отдельной активностью. Например, LoginActivity, RegisterActivity, RecipesActivity и RecipeDetailActivity.

ViewModels: каждый экран может иметь свой собственный ViewModel, который будет управлять данными и бизнес-логикой для этого экрана. [4]

Репозитории (Repositories): репозитории будут обеспечивать доступ к данным из базы данных Firebase и предоставлять методы для выполнения операций чтения и записи. [4]

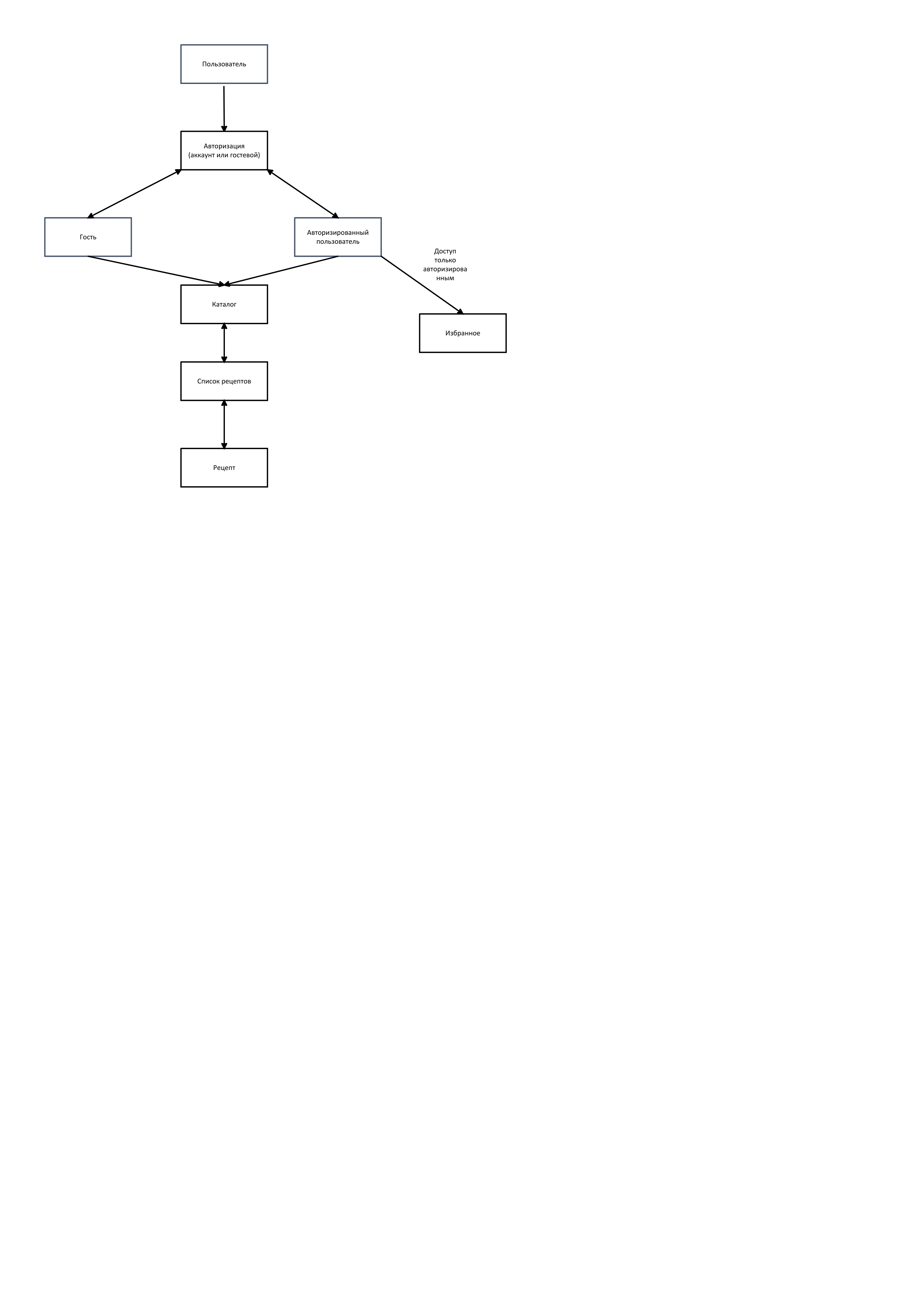


Рисунок 3 - UML-диаграмма

2.4 UX-дизайн мобильного приложения

Дизайн приложения разработан в двух цветовых гаммах для темной и светлой темы соответственно, а также была нарисована иконка для приложения.

Дизайн приложения соответствует основным принципам разработки дизайна приложения [8]:

* простота и удобство. Интерфейс должен быть простым и понятным для пользователя. Все функции должны быть легко доступными. Лишние элементы, чтобы не перегружать пользователя. Простота улучшает восприятие и взаимодействие с приложением.;
* последовательность и единообразие. Одинаковые цвета, шрифты и размеры для схожих элементов интерфейса. Все элементы должны следовать установленным шаблонам и паттернам, что помогает пользователям быстрее ориентироваться.;
* доступность. Светлая и тёмная темы, чтобы пользователи могли выбрать наиболее комфортный для них режим. Четкие и крупные шрифты, о достаточный контраст между текстом и фоном.;
* эстетика и привлекательность. Приятные и сочетаемые между собой цвета, формы и текстуры.;
* отзывчивость. Интерфейс должен корректно отображаться на различных устройствах с разными разрешениями экранов. Приложение должно быстро реагировать на действия пользователя, без задержек и зависаний.

Дизайн иконки соответствует основным принципам разработки дизайна иконки [8]:

* простота и узнаваемость. Иконка должна быть простой и легко узнаваемой даже при небольшом размере. Символизм, элементы, которые сразу ассоциируются с кулинарией и магией;
* согласованность с брендом. Иконка должна использовать те же цвета, что и основной дизайн приложения и соответствовать общему стилю приложения, создавая гармоничный образ;
* доступность. Иконка должна хорошо смотреться в разных размерах, от маленькой иконки на экране телефона до крупного изображения в магазине приложений;
* эстетичность. Иконка должна быть визуально приятной и привлекать внимание пользователей, иметь четкие линии и формы, чтобы оставалась понятной при любых размерах.



Рисунок 4 – Иконка приложения



Рисунок 5 – Загрузка (светлая тема)

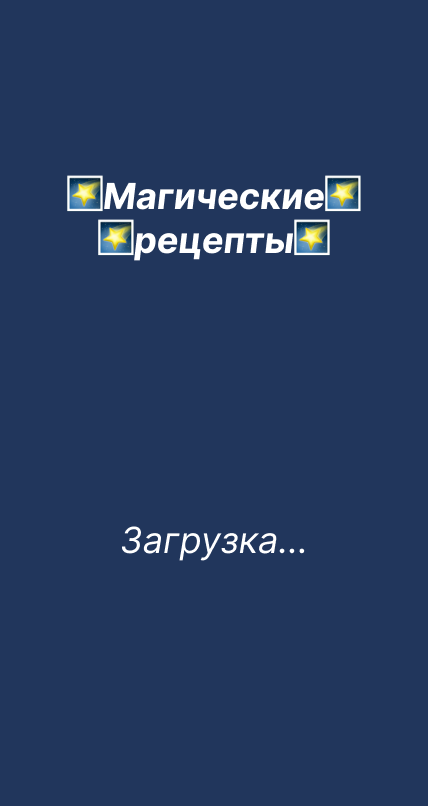


Рисунок 6 - Загрузка (темная тема)

1. Экран «Загрузка»

Назначение экрана: после окончания загрузки отправляет либо на экран авторизации, либо на главный экран, в зависимости от того был ли авторизован уже пользователь или нет.

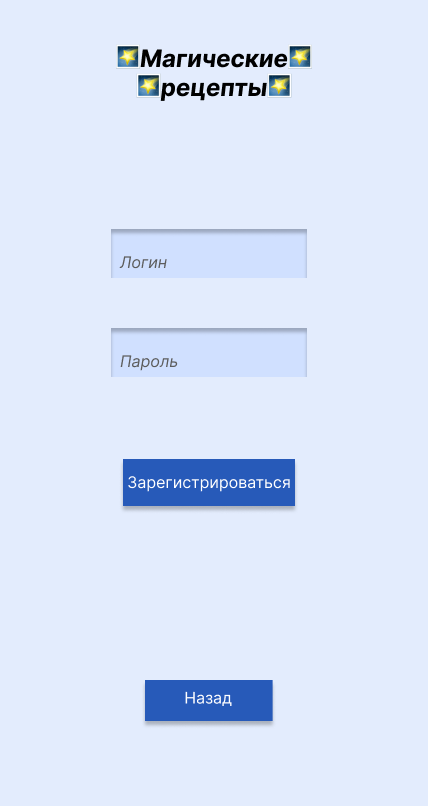


Рисунок 7 – Регистрация (светлая тема)

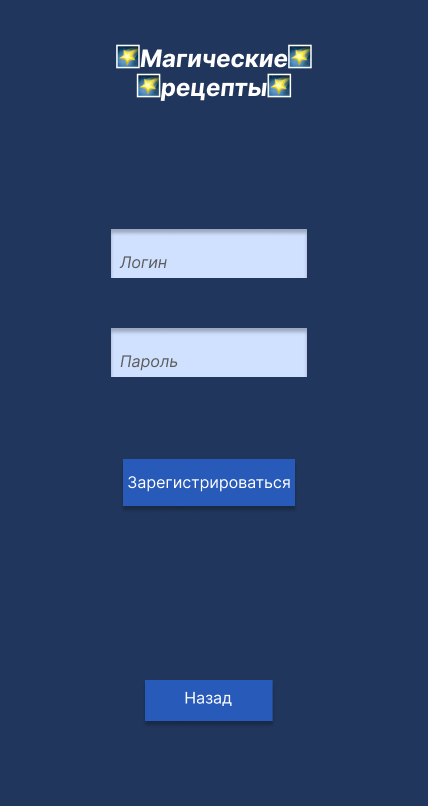


Рисунок 8 - Регистрация (темная тема)

1. Экран «Регистрация»

Назначение экрана: регистрация нового пользователя.

Компоненты экрана:

* TextField, предназначенный для ввода логина пользователя;
* TextField, предназначенный для ввода пароля пользователя.

Присутствует защита пароля от посторонних при вводе.

* кнопка «Зарегистрироваться» для регистрации аккаунта в приложении с веденными данными. Присутствует проверка наличия введенных логин и пароля
* кнопка «Назад» для перехода обратно на экран авторизации пользователя.

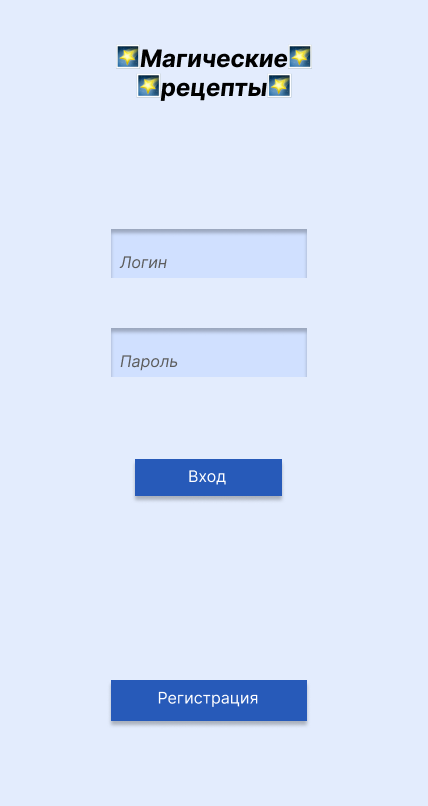


Рисунок 9 - Авторизация (светлая тема)

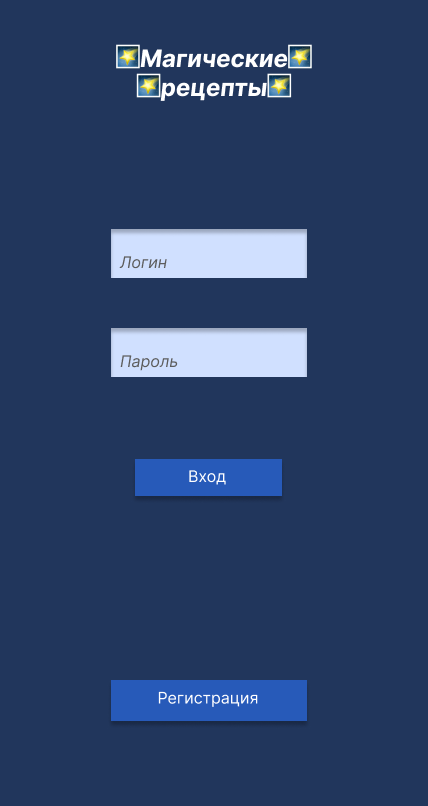


Рисунок 10 - Авторизация (темная тема)

1. Экран «Авторизация»

Назначение экрана: авторизация пользователя.

Компоненты экрана:

* TextField, предназначенный для ввода логина пользователя;
* TextField, предназначенный для ввода пароля пользователя. Присутствует защита пароля от посторонних при вводе;
* кнопка «Вход» для авторизации уже созданного аккаунта в приложении. Присутствует проверка корректности введенных данных, а также скрытие пароля при вводе. (присутствует гостевой вход без аккаунта);
* кнопка «Регистрация» для перехода на экран регистрации нового пользователя.

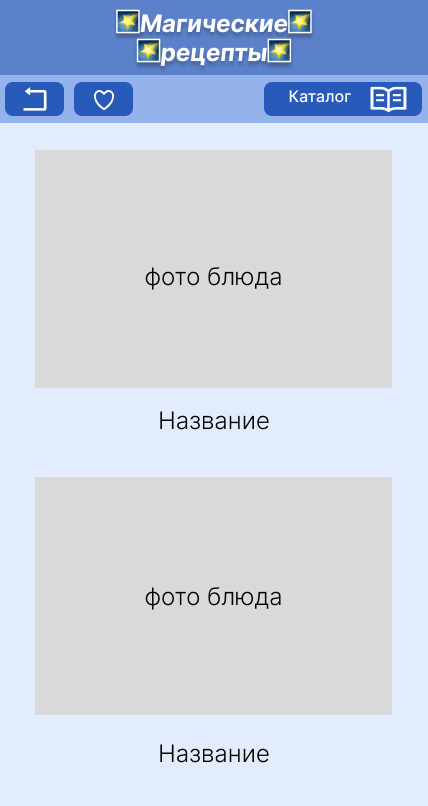


Рисунок 11 - Просмотр рецептов (светлая тема)

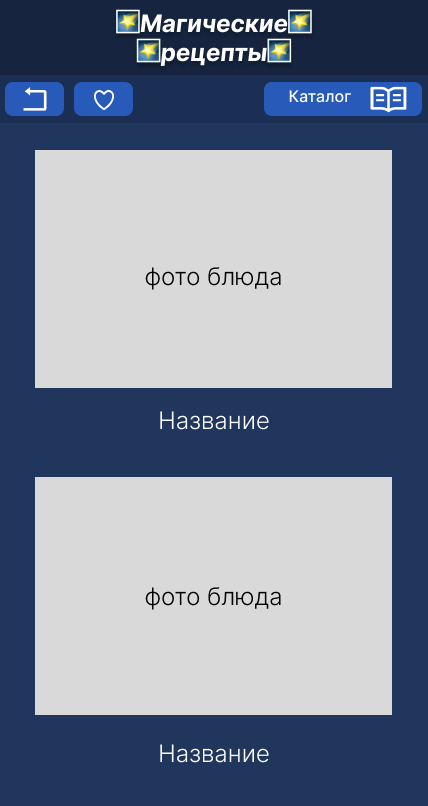


Рисунок 12 - Просмотр рецептов (темная тема)

1. Экран «Просмотр рецептов»

Назначение экрана: просмотр и выбор рецептов. Содержание экрана меняется в зависимости от того, что было выбрано: какой-то определенный каталог, который отображает выбранный тип рецептов; избранное, отображающее рецепты, которые пользователь добавил; главное меню, где отображаются все рецепты.

Компоненты экрана:

* меню в верхней части экрана, содержащая кнопки для переходов на экраны избранных рецептов, каталога, главной страницы, а также кнопки выхода из аккаунта;
* ScrollView, предназначенный для хранения названий и изображений рецептов;
* ImageView, отображающий изображение рецепта;
* TextView, отображающий название рецепта.

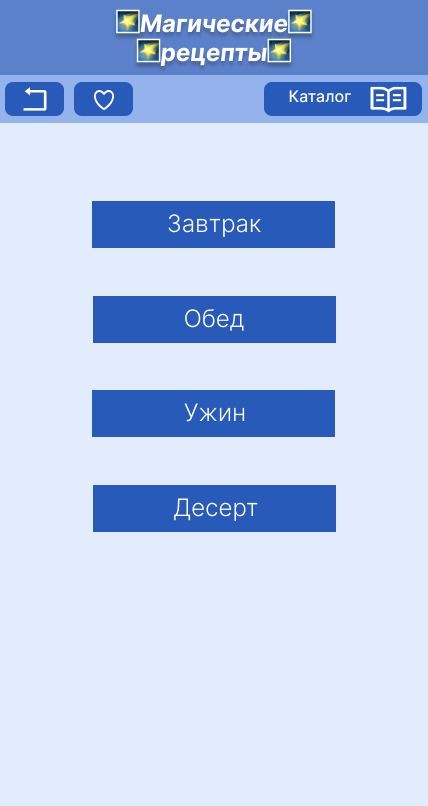


Рисунок 13 - Каталог (светлая тема)

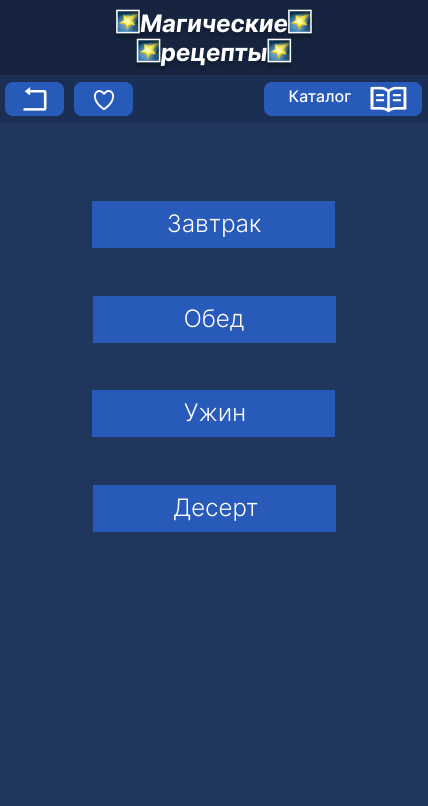


Рисунок 14 - Каталог (темная тема)

1. Экран «Каталог»

Назначение экрана: выбор каталога рецептов.

Компоненты экрана:

* меню в верхней части экрана, содержащая кнопки для переходов на экраны избранных рецептов, каталога, главной страницы, а также кнопки выхода из аккаунта;
* кнопка «Завтрак», предназначенная для перехода к экрану с рецептами для завтрака;
* кнопка «Обед», предназначенная для перехода к экрану с рецептами для обеда;
* кнопка «Ужин», предназначенная для перехода к экрану с рецептами для ужина;
* кнопка «Десерт», предназначенная для перехода к экрану с рецептами десертов.

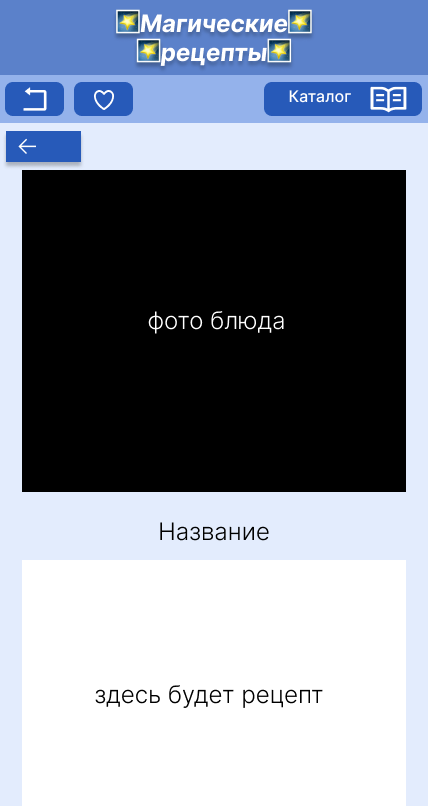


Рисунок 15 - Профиль рецепта (светлая тема)

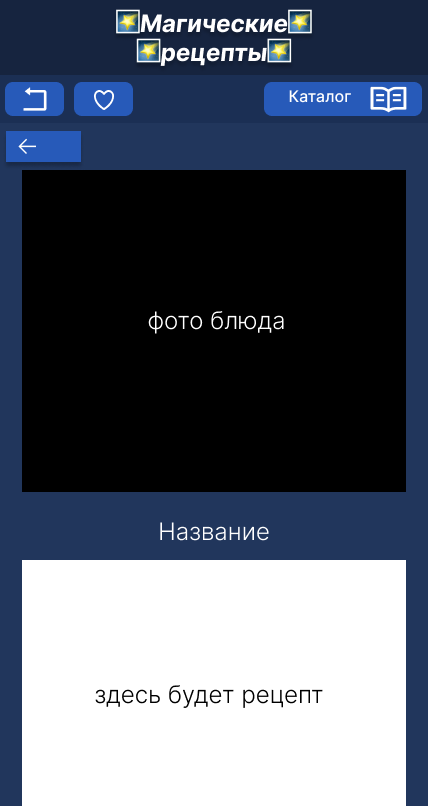


Рисунок 16 - Профиль рецепта (темная тема)

1. Экран «Профиль рецепта»

Назначение экрана: просмотр выбранного рецепта.

Компоненты экрана:

* ImageView, отображающий изображение выбранного рецепта;
* TextView, отображающий название выбранного рецепта;
* ScrollView, содержащий в себе поэтапное приготовление рецепта и требуемые ингридиенты.

## 2.5 Создание ER-диаграммы базы данных

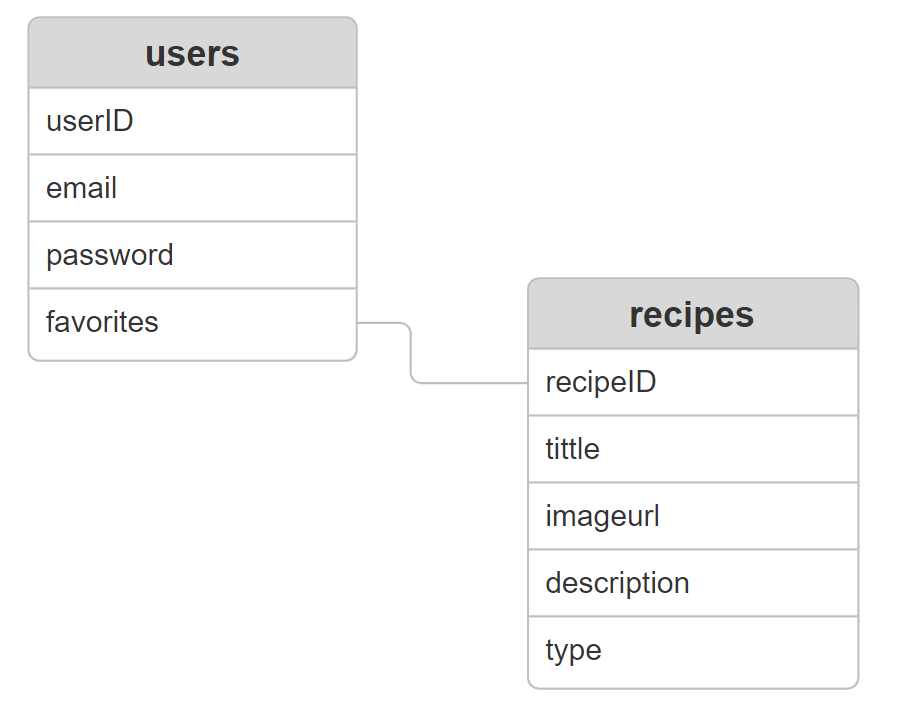


Рисунок 17 - ER-диаграмма

В данной диаграмме «users» и «recipes» представлены как две отдельные коллекции в базе данных Firestore. Связь между ними осуществляется через массив favorites, который содержит идентификаторы избранных рецептов в документе пользователя. Каждый элемент этого массива указывает на документ рецепта в коллекции «recipes» по его recipeId.

3 Разработка мобильного приложения

3.1 Реализация базы данных

Для реализации базы данных приложения использована Firebase Firestore. Firestore является облачной базой данных NoSQL, которая позволяет хранить и синхронизировать данные для клиентских и серверных приложений.

Первым этапом реализации бд (базы данных) будет создание проекта в Firebase. Для этого на официальном сайте Firebase необходимо создать новый проект и следовать инструкциям, которые выданы на сайте.

Вторым этапом будет добавление Android-приложения. Для этого нужно на сайте Firebase для уже созданного проекта необходимо скачать файл «google-services.json» и поместить в корень каталога приложения внутри Android Studio.

Третий этап — это добавление зависимостей для Firebase. Внутри build.gradle (app) добавляются зависимости кодом:

dependencies {

implementation 'com.google.firebase:firebase-firestore-ktx' // зависимость для Firestore

implementation 'com.google.firebase:firebase-database:21.0.0' // зависимость для Firebase

}

Четвертый этап — это непосредственная настройка Firestore в Firebase Console. Этот этап включает в себя создания базы данных во вкладке Firestore Database. После нажатия кнопки «Create database» нужно просто следовать инструкциям. При выборе режима базы данных для простоты настройки был выбран тестовый режим, но затем в целях безопасности изменен на обычный.

Пятый этап — это настройка коллекций и документов Firestore.

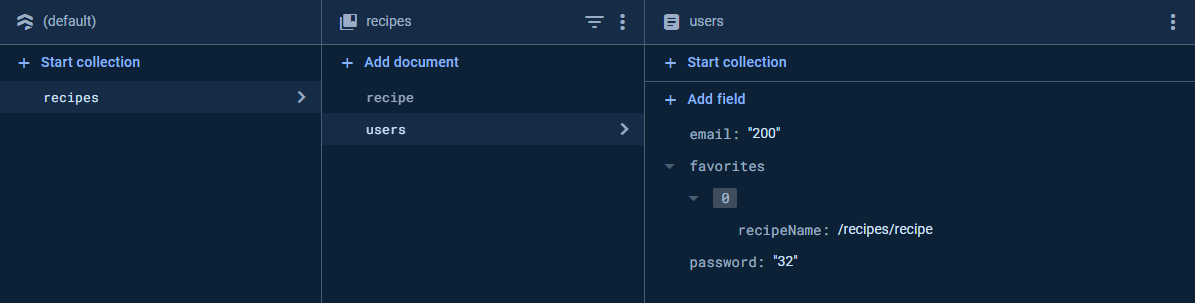


Рисунок 18 - Документ «user»

Имя коллекции: users. Внутри документа users созданы поля:

1. Поле email с типом string и значением 200.
2. Поле favorites с типом array, который связан с полем recipeName в документе recipe.
3. Поле password с типом number и значением 32.



Рисунок 19 - Документ «recipe»

Имя коллекции: recipes. Внутри документа recipe созданы поля:

1. Поле descriptionRecipe с типом string и значением 32000.
2. Поле imageURL с типом string и значением 1000.
3. Поле nameRecipe с типом string и значением 50.
4. Поле type с типом string и значением 15.

Таким образом при помощи Firebase Firestore были настроены коллекция и документ. Это позволило обеспечить гибкость и масштабируемость приложения.

3.2 Создание приложения

Программные модули в Android-приложении представляют собой отдельные компоненты кода, которые выполняют определенные функции. Разделение приложения на модули помогает организовать код, упростить его поддержку и повторное использование. В данном контексте Activity является одним из таких модулей, отвечающим за пользовательский интерфейс и взаимодействие пользователя с приложением. В этом разделе мы рассмотрим создание основных модулей для приложения, каждый из которых будет выполнять специфическую задачу.

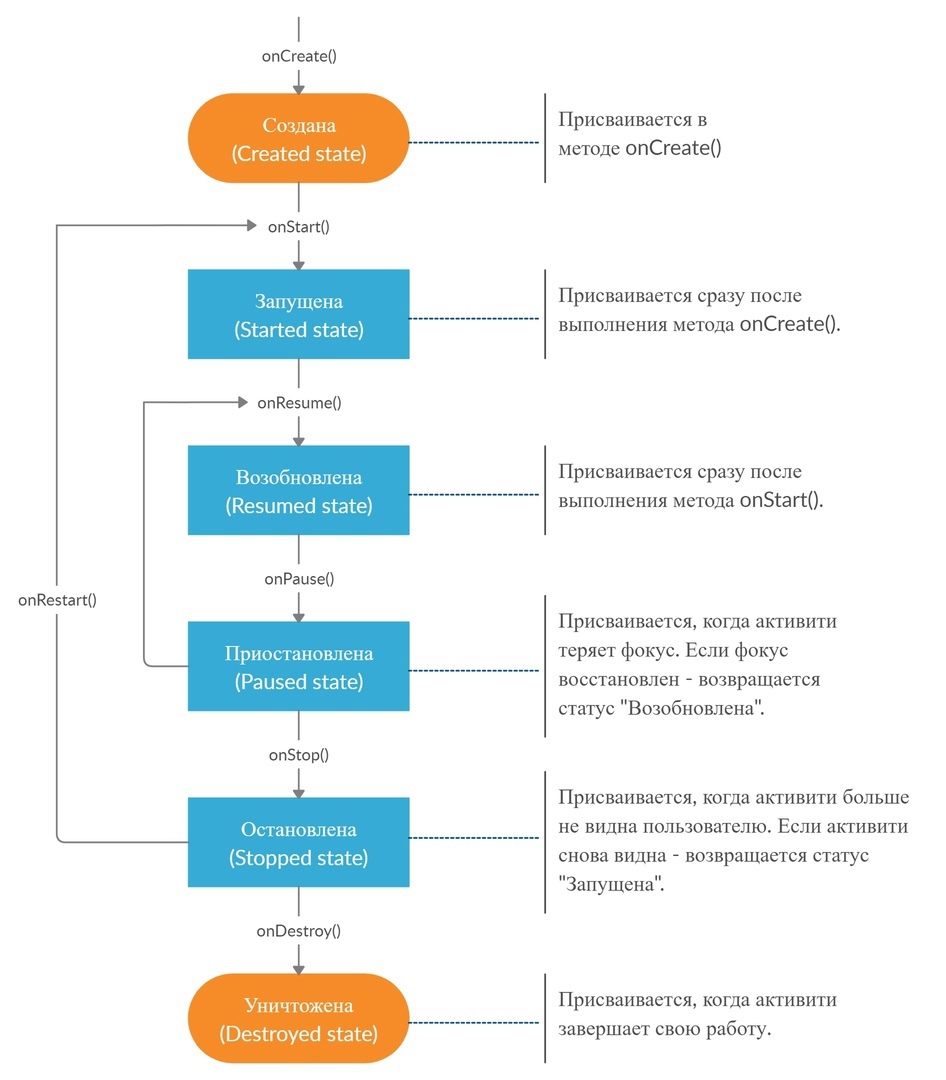


Рисунок 20 - Схема состояний activity

Класс LoadingActivity является начальной активностью приложения. Служит для плавного начала работы приложения, обеспечивая задержку для отображения экрана загрузки и затем переходя к основному экрану приложения.

Класс LoginActivity является активностью, которая служит для управления процессом аутентификации пользователей в приложении, обеспечивая проверку введенных данных, вход в систему и переходы между экранами на основе состояния пользователя и наличия подключения к интернету.

Позволяет зайти пользователю в качестве гостя или с помощью введенных данных с помощью FirebaseAuth. [2] В случае успешной аутентификации сохраняет состояние пользователя и происходит переход на главный экран.

Класс RegisterActivity является активностью, которая отвечает за процесс регистрации новых пользователей, обеспечивая проверку введенных данных, создание новой учетной записи и переход на экран входа после успешной регистрации.

Последующие активности имеют центральную точку управления различными функциями приложения, обеспечивая удобный доступ к разделам избранного, каталога, главного меню, а также управляя процессом выхода из аккаунта.

Класс MainActivity является активностью, которая является основным экраном, на котором отображаются случайные рецепты.

Класс CatalogActivity является активность, которая отображает различные категории рецептов и предоставляет возможность выбора конкретной категории для просмотра рецептов из нее. Имеет четыре кнопки, отвечающие за переходы на соответствующие экраны с рецептами завтраков, обедов, ужинов и десертов.

Классы DinnerActivity, DessertActivity, BreakfastActivity и LunchActivity являются активностями, которые отображают указанные ранее каталоги. Подобно MainActivity позволяют открывать конкретные рецепты при нажатии на изображение или название рецепта.

Класс FavoriteActivity является активностью, доступной только для пользователей с аккаунтом. Также является активностью, которая служит главным экраном при отсутствии интернета и при ранее авторизированном аккаунте. Позволяет просматривать ранее сохраненные рецепты в любое время, не зависит от подключения к интернету благодаря тому, что загружает рецепты из локального хранилища.

При первом посещении FavoriteActivity проверяется, первый ли раз пользователь зашел в раздел избранного, и в случае первого посещения отображается диалоговое окно с инструкциями о добавлении и удалении рецептов из избранного.

Класс ProfileRecipe является вспомогательной активностью, а потому не содержит навигационного меню по приложению. Служит для отображения подробной информации о рецепте. Получает данные из объекта Recipe, который содержит информацию о выбранном рецепте. Полученные данные используются для заполнения соответствующих компонентов на экране, таких как название, его описание и изображение. Содержит метод, вызывающий стандартное поведение кнопки «назад».

Класс Recipe представляет рецепт блюда в приложении. обеспечивает удобную работу с данными рецептов в приложении, позволяя удобно передавать их между различными компонентами и сохранять в базе данных Firestore. Основные аспекты этого класса:

* + - 1. Поля:
* recipeID: идентификатор рецепта;
* title: название рецепта
* description: описание рецепта;
* imageUrl: URL-адрес изображения блюда;
* type: тип блюда.
  + - 1. Конструкторы:
* пустой конструктор, необходимый для работы с Firestore;
* конструктор, принимающий параметры и инициализирующий поля объекта.
  + - 1. Методы:
* toJson(): преобразует объект Recipe в строку JSON;
* fromJson(): создает объект Recipe из строки JSON;
* fromDocumentSnapshot(): создает объект Recipe из документа Firestore.
  + - 1. Переопределенные методы:
* describeContents(): возвращает флаг, описывающий тип содержимого объекта;
* writeToParcel(): записывает содержимое объекта в Parcel для передачи между компонентами.

3.3 Методическое обеспечение

### 3.3.1 Руководство для программиста

Данное руководство предназначено для системного программиста, который занимается развертыванием, поддержкой и обслуживанием системы управления заявками на ремонт оборудования. В документе подробно описываются процедуры развертывания системы, работа с исходным кодом, а также методы решения возникающих проблем при использовании программного обеспечения. [11]

Руководство включает в себя детальные инструкции по конфигурации, установке, тестированию, диагностике и оптимизации системы. Особое внимание уделено описанию исходного кода, включая детализацию того, что содержится в каждом файле программного модуля. [11]

Руководство системного программиста включает следующие разделы [11]:

1. Общие сведения о программе.

2. Назначение и функции программы.

3. Настройка программы.

4. Описание дополнительных функциональных возможностей программы.

5. Сообщения системному программисту.

1. Общие сведения о приложении:

Название проекта: MagicRecipes

Версия: 1.0

* 1. Краткое описание: приложение предназначено для просмотра рецептов. Оно обеспечивает эффективный и удобный процесс просмотра рецептов для пользователей с возможностью сохранения понравившихся рецептов.
  2. Основные функции системы:
* просмотр рецептов;
* доступ к избранным рецептам без интернета;
* кроссплатформенность;
* безопасность;
* удобство использования;
* производительность.

1. Назначение и функции программы.

2.1. Технические и программные средства, необходимые для работы приложения:

1. Язык программирования: Java, Gradle 8.7, Android Gradle Plugin Version 8.4.1
2. Android SDK: Min SDK Version 31, Target SDK Version 34; SDK Tools: Android SDK Platform-Tools v. 35.0.1
3. База данных: Firebase Firestore используется для авторизации и хранения всех данных приложения.
4. Операционная система: Android
5. Системные требования: Android выше 13.0

2.2. Структура программы

2.2.1 Сведения о структуре программы, ее составных частях.

Кратко описание каждого этапа:

* + - авторизация: ввод логина и пароля или гостевой вход;
    - определение метода входа: гостевой вход или вход с аккаунтом;
    - меню для каждого метода:
      1. Гость: просмотр рецептов.
      2. Пользователь с аккаунтом: просмотр рецептов, доступ к избранному.
    - работа с рецептами: просмотр рецептов, добавление в избранное, удаление из избранного.

2.2.2 Связи между частями.

1. Entities (Сущности):
   * User: ID, email, favorites;
   * Recipe: recipeID, description, tittle, imageUrl, type.
2. Repositories:
   * UserRepository: Интерфейс и реализация для работы с данными пользователя;
   * RecipeRepository: Интерфейс и реализация для работы с данными рецептов
3. Use Cases (Прецеденты использования):
   * RegisterUserUseCase: Регистрация пользователя;
   * LoginUserUseCase: Авторизация пользователя;
   * GetRecipeUseCase: Получение информации о рецепте;
   * AddToFavoritesUseCase: Добавление рецепта в избранное;
   * MainManuUseCase: Навигация по экранам в приложении.
4. UI (Пользовательский интерфейс):
   * классы: LoadingScreen, Register, Login, Main, Favorite, Dinner, Breakfast, Dessert, Catalog, Lunch, ProfileRecipe.

Таблица 1 - Основные и критические переменные исходного кода

|  |  |  |  |
| --- | --- | --- | --- |
| Класс/Файл в проекте | Имя переменной | Тип переменной | Назначение |
| LoginActivity | mAuth | FirebaseAuth | Экземпляр FirebaseAuth для выполнечния операций аутентификации. |
| LoginActivity | emailEditText | EditText | Поле ввода электронной почты пользователя. |
| LoginActivity | passwordEditText | EditText | Поле ввода пароля пользователя. |
| LoginActivity | currentUser | FirebaseUser | Текущий аутентифицированный пользователь Firebase. |
| Recipe | title | String | Уникальный идентификатор рецепта. |
| Recipe | recipeID | String | Название рецепта. |
| Recipe | description | String | Описание рецепта. |
| Recipe | imageUrl | String | URL изображения рецепта. |
| Recipe | type | String | Тип рецепта (например, категория блюда). |
| Recipe | in | Parcel | Parcel для чтения данных при создании объекта из Parcelable. |
| Recipe | CREATOR | Parcelable.Creator<Recipe> | Создатель объектов Recipe из Parcel. |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Recipe | gson | Gson | Экземпляр Gson для сериализации и десериализации объектов. |
| Recipe | document | DocumentSnapshot | Снимок документа Firestore, из которого создается объект Recipe. |
| FavoriteActivity | builder | AlertDialog.Builder | Построитель диалогового окна для отображения инструкций или подтверждения действий. |
| FavoriteActivity | dialog | AlertDialog | Диалоговое окно для отображения инструкций или подтверждения действий. |
| FavoriteActivity | preferences | SharedPreferences | Объект для сохранения состояния первого захода в избранное. |
| FavoriteActivity | editor | SharedPreferences.Editor | Редактор для сохранения данных в SharedPreferences. |
| FavoriteActivity | sharedPreferences | SharedPreferences | Объект для сохранения данных пользователя, таких как избранные рецепты. |
| FavoriteActivity | allEntries | Map<String, ?> | Коллекция всех записей (избранных рецептов) в SharedPreferences. |
| FavoriteActivity | viewToRemove | View | Представление для удаления из макета после удаления рецепта из избранного. |

3 Настройка программы

3.1 Описание действий по настройке программы для конкретного применения и работы.

Для корректного запуска и настройки программного модуля требуется установить среду разработки от JetBrains, под названием Android Studio, доступная бесплатно с официального сайта разработчика среды разработки.

В случае использования стандартного, описанного выше программного обеспечения, в лице Android Studio, требуется:

− установить само программное обеспечение с использованием установочного пакета с официально сайта компании JetBarins. Версия программного обеспечения не влияет на работу с программным модулем, ни на его непосредственную работу;

− после установки и первичного запуска среды разработки, требуется внедрить в него проект/исходный код программы, используя либо контроль версии с GitHub (при наличии такой ссылки на проект) или же распакованный .zip архив с вложенной в него папкой проекта и исходным кодом внутри;

− выбрать версию SDK Version 34, дождаться ее скачивания и обработки, дождаться компиляции Gradle версии 8.7.

Работа с базой данных осуществляется в Firebase Console, на официальном сайте Firebase.

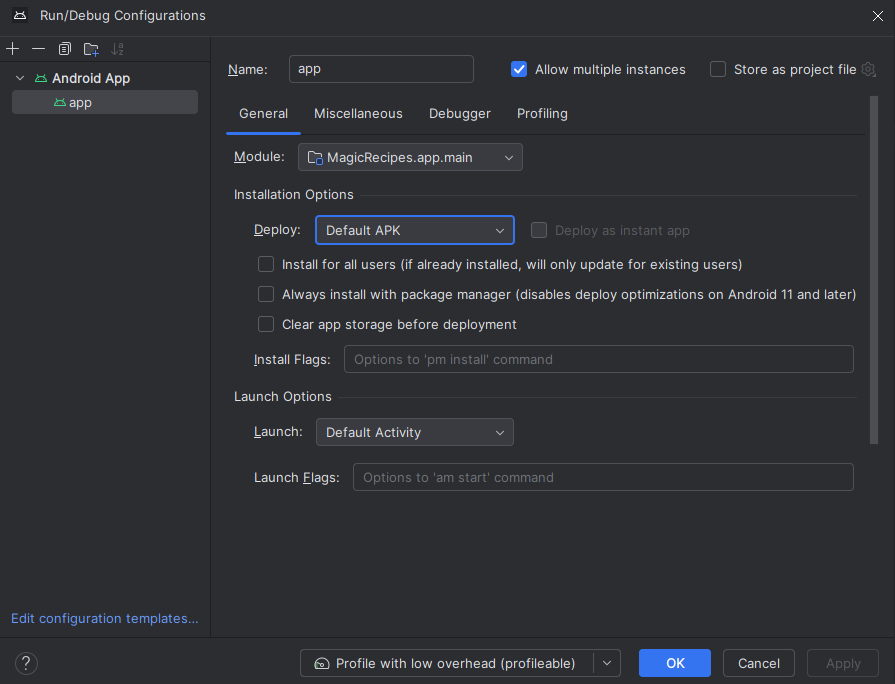


Рисунок 22 - пример настройки конфигурации для запуска модуля в среде разработки

3.2 Проверка программы

После корректной установки, настройки программного обеспечения и конфигурации проекта, следует запустить модуль, для проверки отображения интерфейса и подключения к БД, после корректного запуска, можно начать полноценную работу с приложением, его настройкой под заказчика и разработкой/добавлением нового функционала и при надобности – пользователей платформы.

3.3. Описание дополнительных функциональных возможностей приложения

В рамках расширения функционала приложения была добавлена новая возможность — режим гостя. Гостевой режим позволяет пользователям не регистрировать аккаунт, чтобы начать пользоваться приложением. Основные отличия гостя от пользователя с аккаунтом:

* невозможность переносить рецепты, сохраненные в избранном, с устройства на устройство;
* нет доступа к избранному.

4. Сообщения системному программисту

Создание резервной копии базы данных в Firestore через консоль Firebase включает несколько шагов:

1. Вход в консоль Firebase.

* перейдите на сайт Firebase Console;
* войдите в свою учетную запись Google, если вы еще не вошли.

1. Выбор проекта.

* выберите проект, в котором находится ваша база данных Firestore.

1. Переход в раздел Firestore.

* в меню слева выберите «Firestore Database».

1. Переход в раздел «Export/Import».

* в верхней части экрана выберите вкладку «Backups»;
* нажмите на кнопку «Export/Import».

1. Создание резервной копиию.

* в разделе «Export / Import» выберите опцию «Export»;
* укажите Google Cloud Storage (GCS) bucket, в который будет сохранена резервная копия. Если у вас еще нет настроенного GCS bucket, следуйте инструкциям для его создания;
* перейдите на Google Cloud Console;
* в меню слева выберите "Storage" и затем «Browser»;
* нажмите «Create bucket» и следуйте инструкциям для создания нового bucket;
* вернитесь в консоль Firebase и укажите путь к вашему GCS bucket;
* нажмите «Export» для создания резервной копии.

1. Мониторинг процесса экспорта.

* процесс экспорта может занять некоторое время в зависимости от размера вашей базы данных;
* мониторить статус экспорта можно в разделе «Export/Import» в консоли Firebase.

### 3.3.2 Руководство пользователя

Приложение «Магические рецепты» управляет процессами регистрации и авторизации пользователя, просмотром и сохранением рецептов. Пользователи могут добавить рецепт в избранное для быстрого доступа. Приложение включает сортировку рецептов по каталогам, что позволяет легко найти рецепт по нужной тематике.

Запуск приложения осуществляется при помощи клика на иконку приложения. Для использования приложения требуется доступ к интернету. Без интернета доступны только заранее добавленные в избранное рецепты.

При запуске приложения нужно произвести авторизацию: ввести логин и пароль от уже существующего аккаунта, либо войти как гость. Как логин используется электронная почта.

Гость: при входе через режим гостя не требуется создавать аккаунт. При использовании данного режима отсутствует возможность добавлять рецепты в избранное.

Авторизированный пользователь: требуется зарегистрировать аккаунт. Рецепты, сохраненные в избранном, переносятся с устройство на устройство.

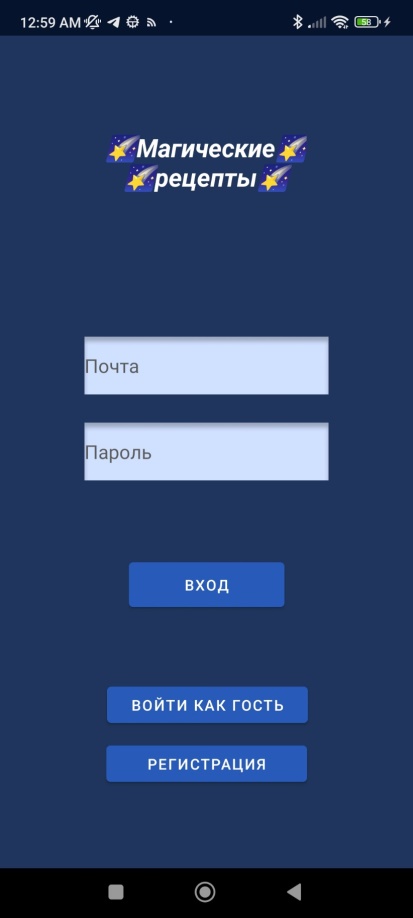


Рисунок 23 - Экран авторизации

Если аккаунт еще не зарегистрирован, то нажатие кнопки «Регистрация» откроет экран регистрации. В открывшимся окне необходимо указать данные, которые в будущем будут использоваться для входа в аккаунт.

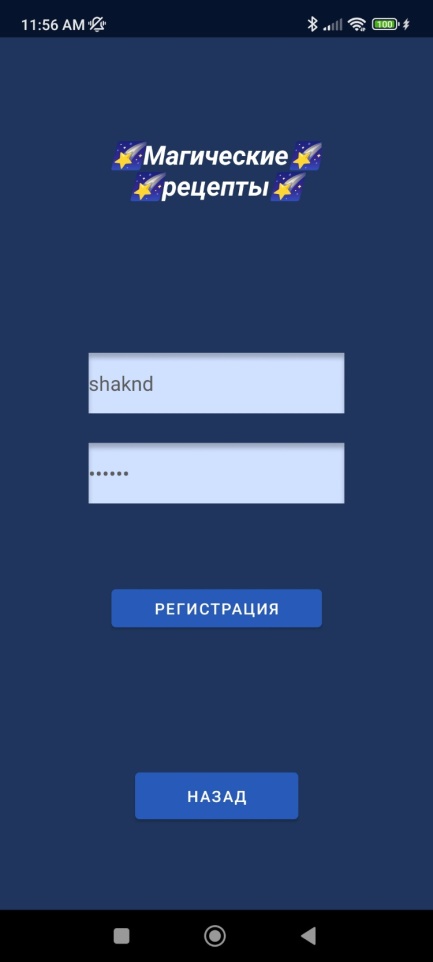


Рисунок 24 – Регистрация нового аккаунта

После успешной регистрации экран регистрации закрывается, отправляя обратно к экрану авторизации.

После авторизации или входа с гостевым аккаунтом, открывается главный экран, где отображаются случайные рецепты. В верхней части расположено главное меню, которое является навигацией по приложению.

Кнопка «Выйти из аккаунта» отправляет обратно на экран авторизации.



Рисунок 25 - Кнопка «Выйти из аккаунта»

Кнопка «Избранное» доступна только пользователям с авторизированным аккаунтом. Отправляет на экран с рецептами, которые были добавлены в избранное.



Рисунок 26 – Кнопка «Избранное»

Кнопка «Каталог» отправляет на экран с выбором каталогов рецептов.

**

Рисунок 27 - кнопка «Каталог»



Рисунок 28 – Экран выбора каталога

На данный момент существует 4 каталога:

* «Энергетический завтрак» — это каталог, содержащий рецепты, подходящие для утреннего приема пищи;
* «Мистический обед» — это каталог, содержащий рецепты, подходящие для обеденного приема пищи;
* «Потусторонний ужин» — это каталог, содержащий рецепты, подходящие для вечернего приема пищи;
* «Сладкое волшебство» — это каталог, содержащий рецепты десертов.

Для того, чтобы посмотреть понравившийся рецепт нужно нажать на картинку или название рецепта. Рецепт содержит изображение рецепта, название рецепта, требуемые ингредиенты, а также пошаговая инструкция к приготовлению. Кнопка «Назад» при просмотре рецепта возвращает к экрану, с которого был открыт рецепт.

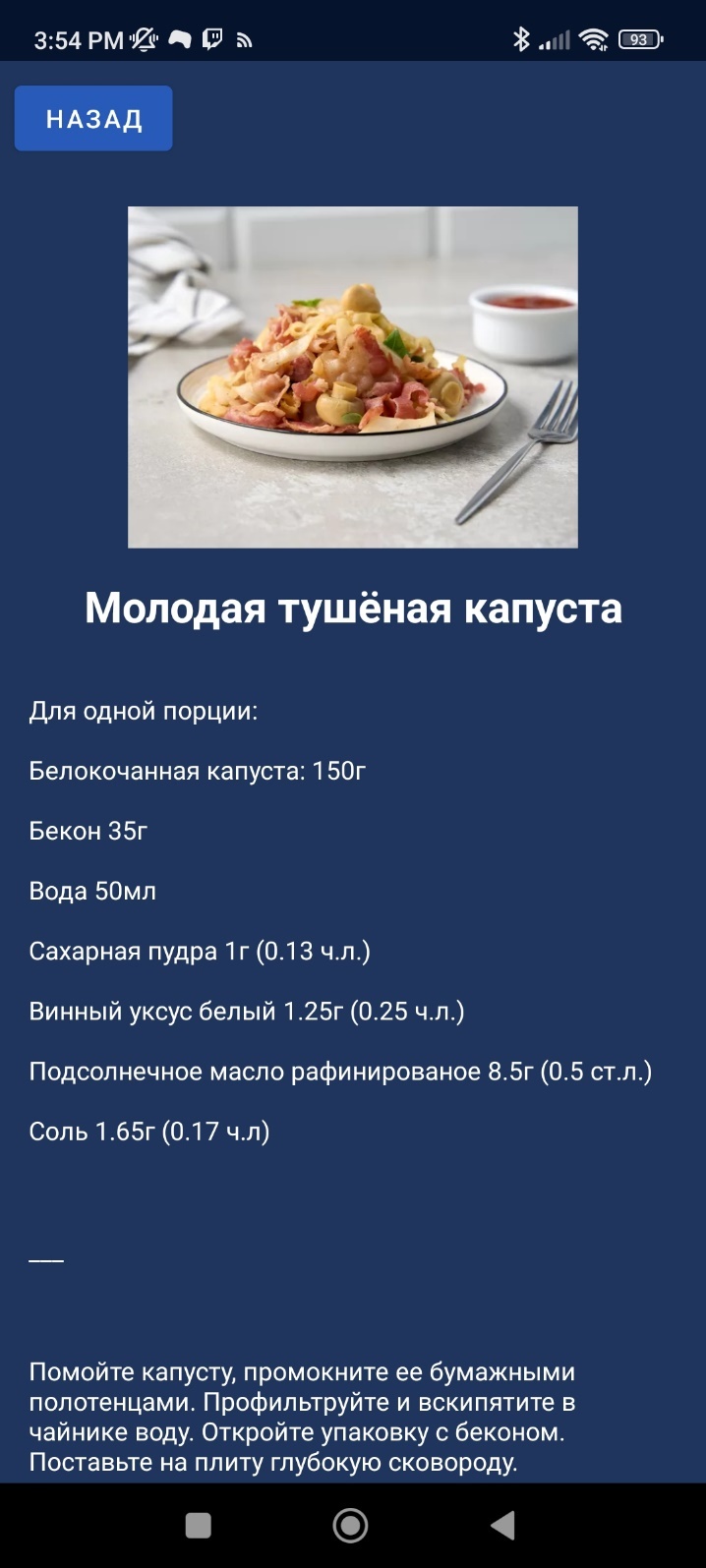


Рисунок 29 - Профиль рецепта

1. Гость

Гость имеет меньше взаимодействий с приложением, чем авторизированный пользователь. Вход гостя сопровождается соответствующим уведомлением.



Рисунок 30 - Уведомлением о входе, как Гость

Попытка зайти в избранное или добавить рецепт в избранное также сопровождается уведомлением о том, что требуется авторизировать аккаунт.



Рисунок 31 - Уведомление при попытке зайти в раздел «Избранное» в режиме Гостя



Рисунок 32 - Уведомление при попытке добавить рецепт в избранное в режиме гостя

1. Авторизированный пользователь

Авторизированный пользователь имеет полный доступ ко всем возможностям приложения. Вход с авторизированным аккаунтом сопровождается уведомлением об успешной авторизации.



Рисунок 33 - Уведомление об успешной авторизации аккаунта

При первом переходе в «Избранное», появляется краткая инструкция по добавлению рецепта в избранное.

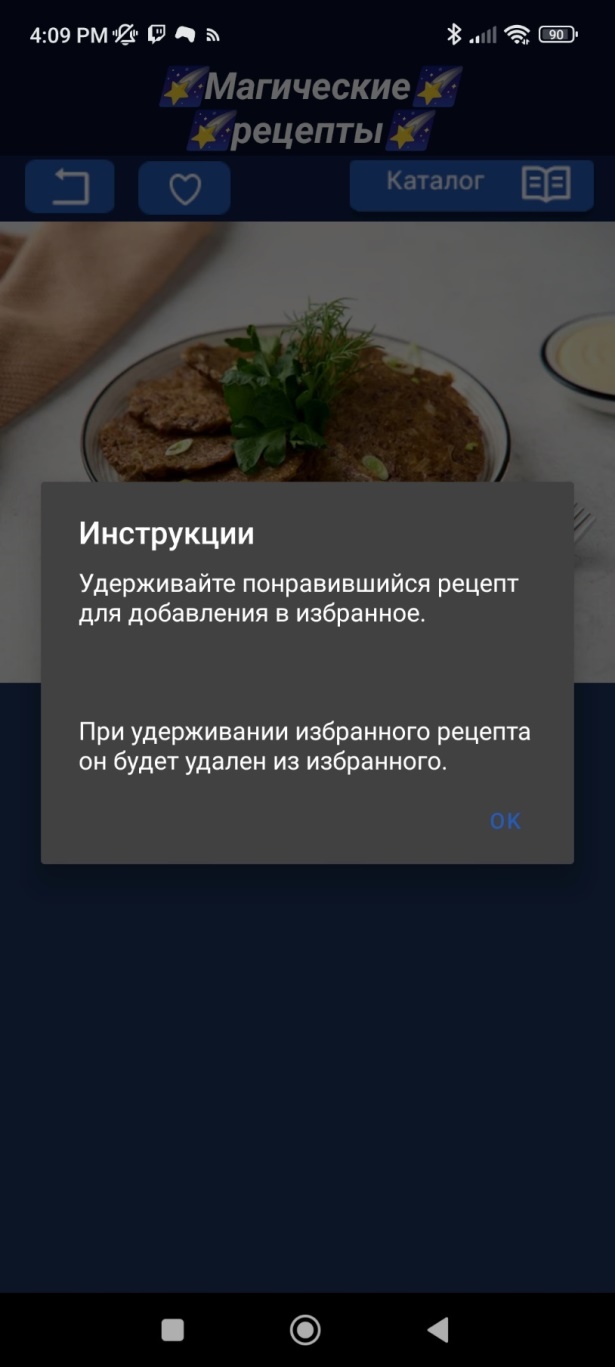


Рисунок 34 - Инструкция к добавлению и удалению рецептов в/из «Избранное»

Для того, чтобы добавить рецепт в избранное, необходимо долго удерживать изображение понравившегося рецепта в списке рецептов.

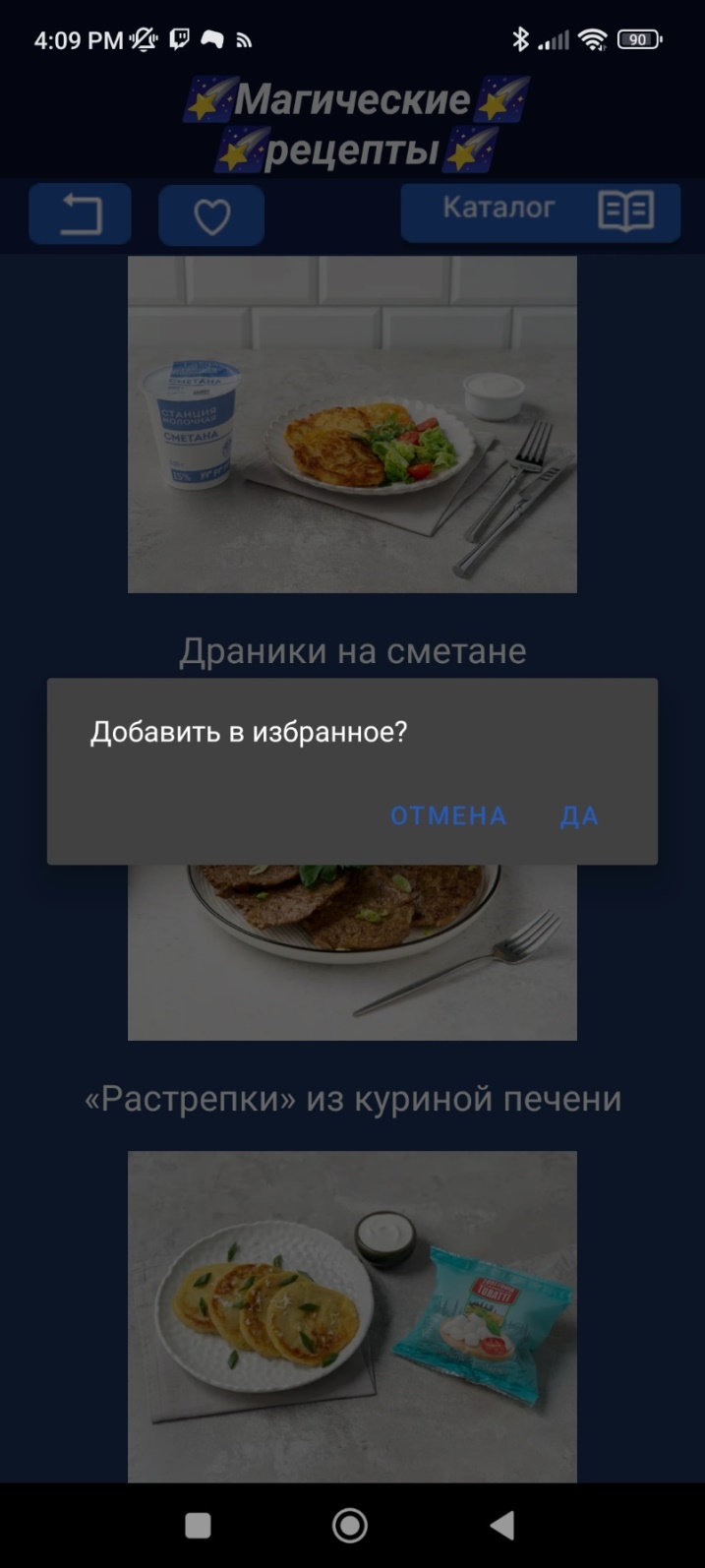


Рисунок 35 - Подтверждение добавления рецепта в избранные



Рисунок 36 - Отображение добавленного рецепта в избранном

Для того, чтобы удалить рецепт необходимо долго удерживать изображение рецепта в списке избранных рецептов.

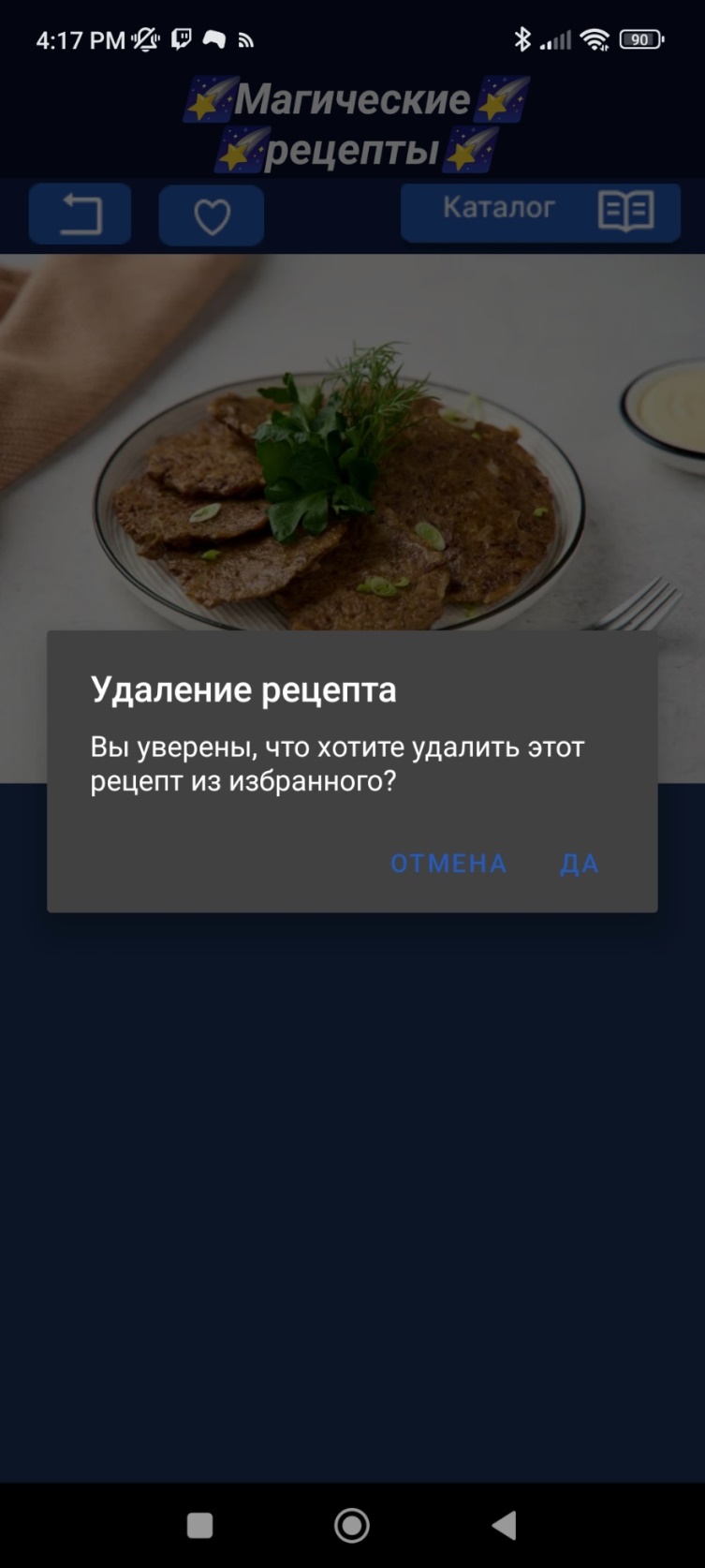


Рисунок 37 - Подтверждение удаления рецепта из избранного

3.4 Тестирование приложения

Для обеспечения качества и функциональности приложения «Магические рецепты» необходимо провести различные виды тестирования. Виды тестирования, которые будут проведены это функциональные, безопасности и юзабилити.

1. Функциональные тесты.

1.1 Тестирование авторизации:

Если вход в аккаунт не был выполнен, то после загрузки приложения открывается окно авторизации. После ввода верного логина и пароля должно появиться уведомление об успешной авторизации. В качестве логина используется электронная почта.

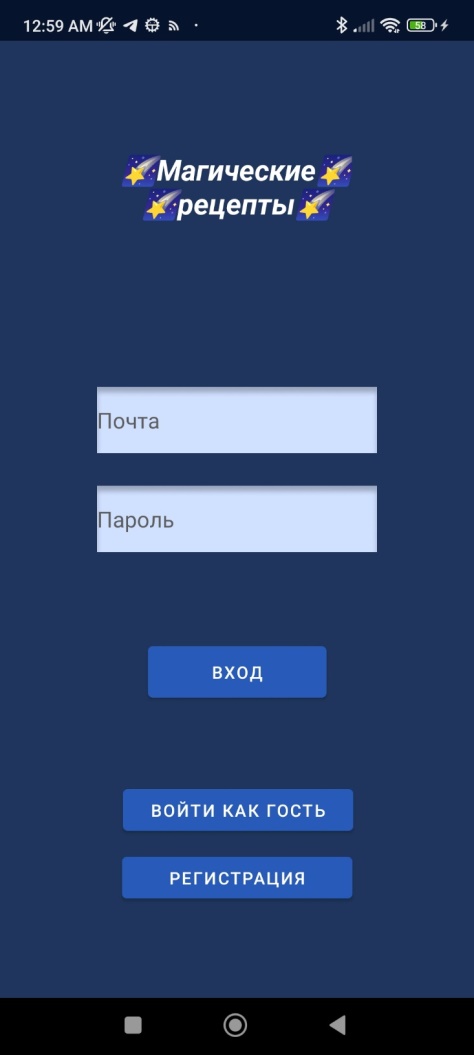


Рисунок 38 – Авторизация

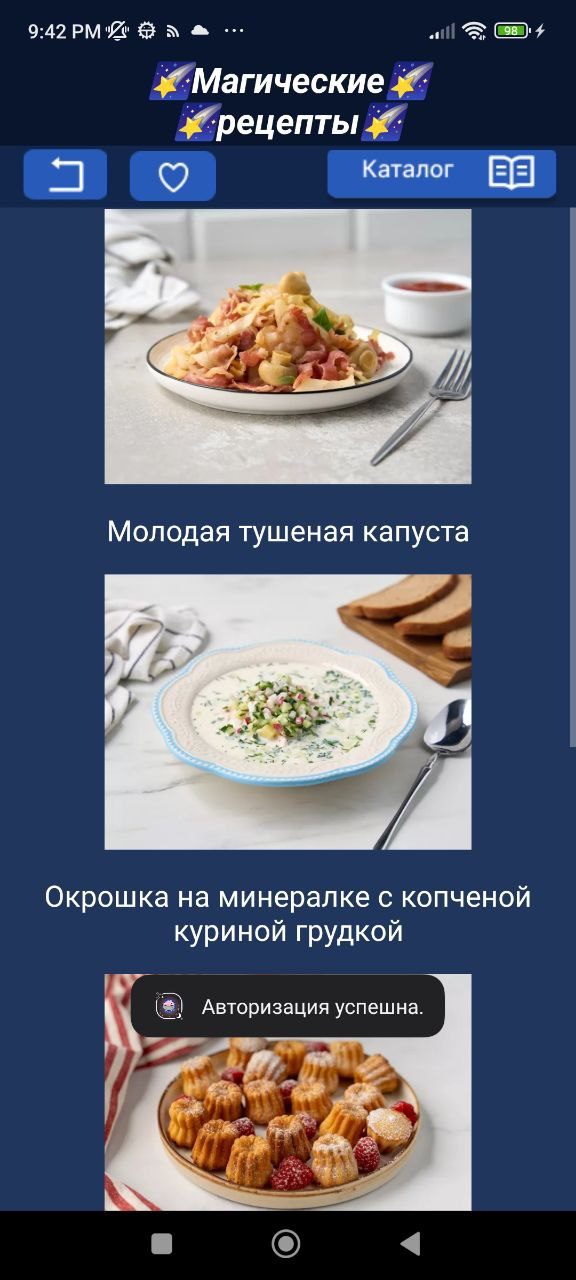


Рисунок 39 – Уведомление об успешной авторизации

Пользователь успешно прошел аутентификацию. Тестирование авторизации окончилось успехом.

1.2 Тестирование регистрации:

Для регистрации пользователь должен ввести электронную почту и пароль. При успешной регистрации пользователь будет перенаправлен обратно к экрану авторизации.

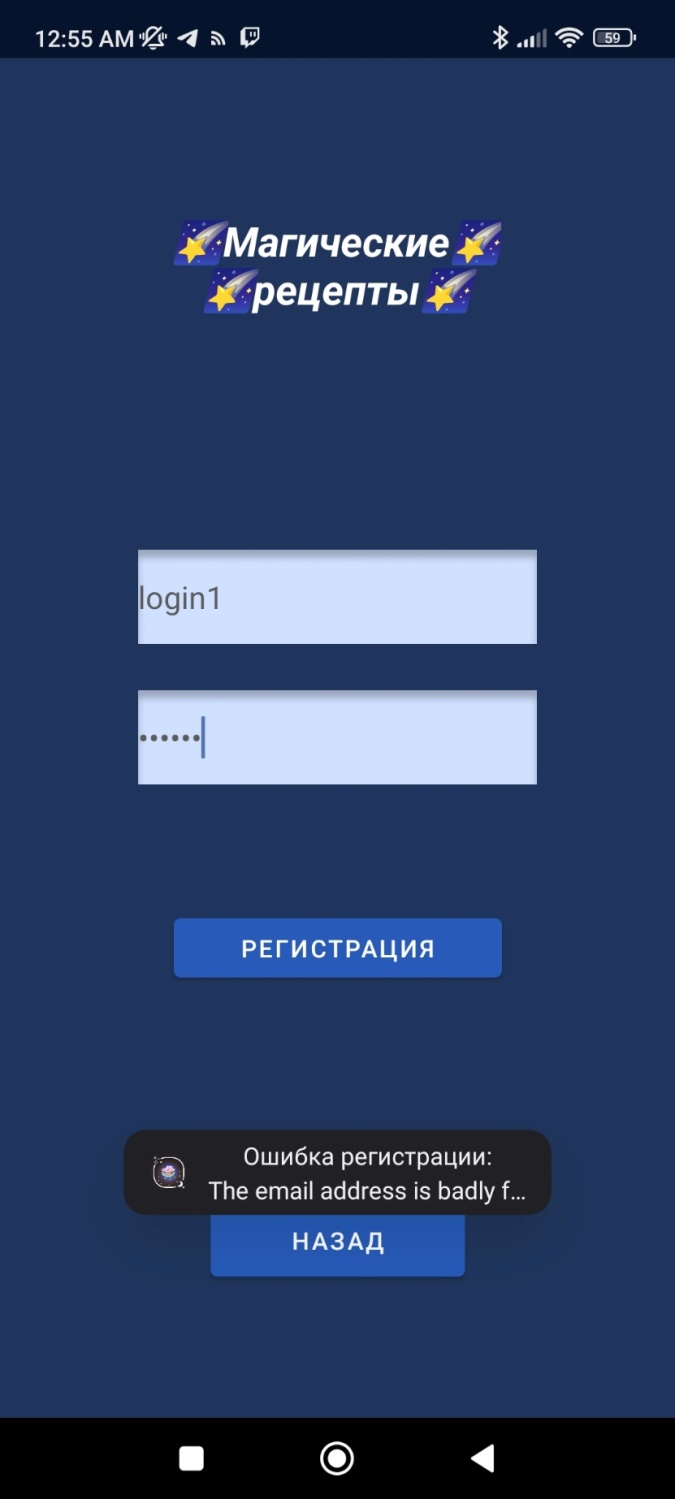


Рисунок 40 - Ошибка при регистрации

Пользователь не прошел регистрацию. Тестирование регистрации окончилось провалом.

Для того, чтобы улучшить опыт взаимодействия пользователя с приложением была добавлена обработка исключительной ситуации с выводом ошибки регистрации.

1.3 Тестирование выхода из учетной записи:

Для того, чтобы выйти из аккаунта пользователь должен нажать на кнопку «Выход из аккаунта». После этого появляется диалоговое окно, требующее подтверждения действия, чтобы пользователь не мог случайно выйти из аккаунта.

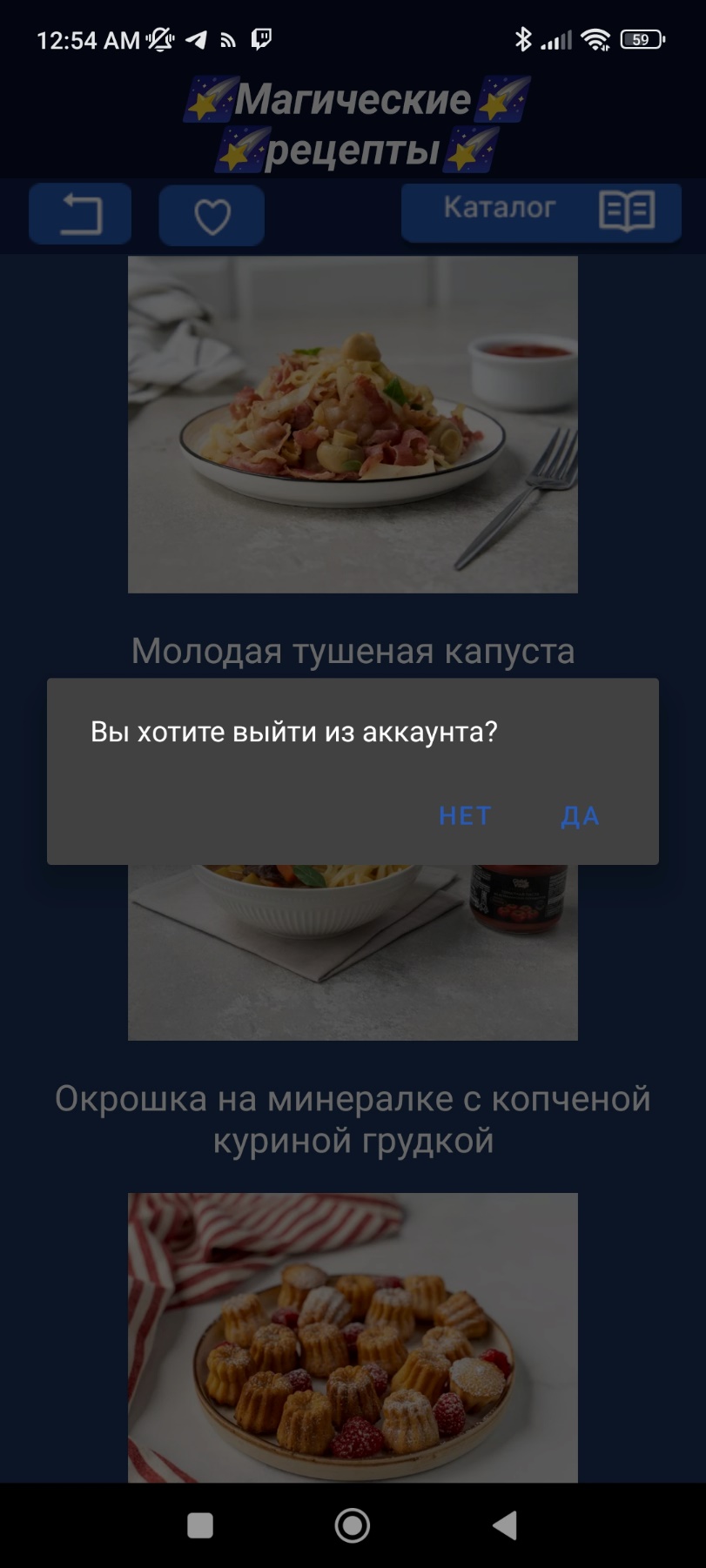


Рисунок 41 - Диалоговое окно для выхода из аккаунта

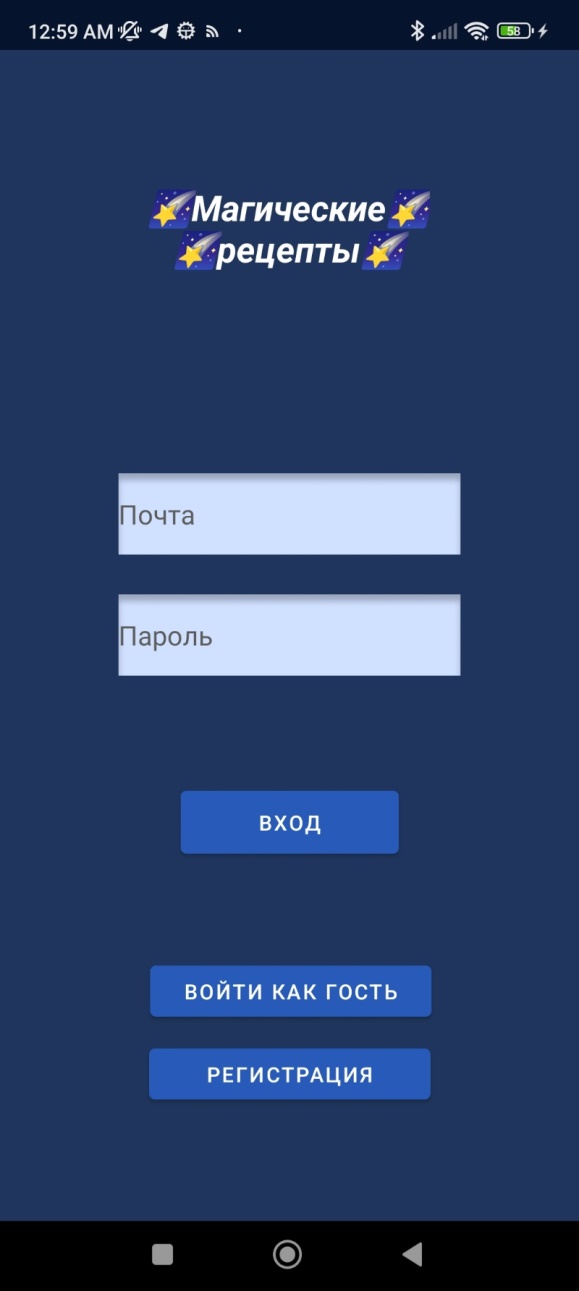


Рисунок 42 - Окно авторизации после подтверждения выхода из аккаунта

Пользователь вышел из аккаунта. Тестирование выхода из аккаунта окончилось успехом.

1. Тестирование безопасности.

2.1 Тестирование защиты данных пользователей при входе:

Ввод пароля при входе пользователя в аккаунт защищено шифрованием для того, чтобы злоумышленники не могли подсмотреть и украсть аккаунт. Отображается лишь последний введенный символ, а все предыдущие отображаются в виде точек.

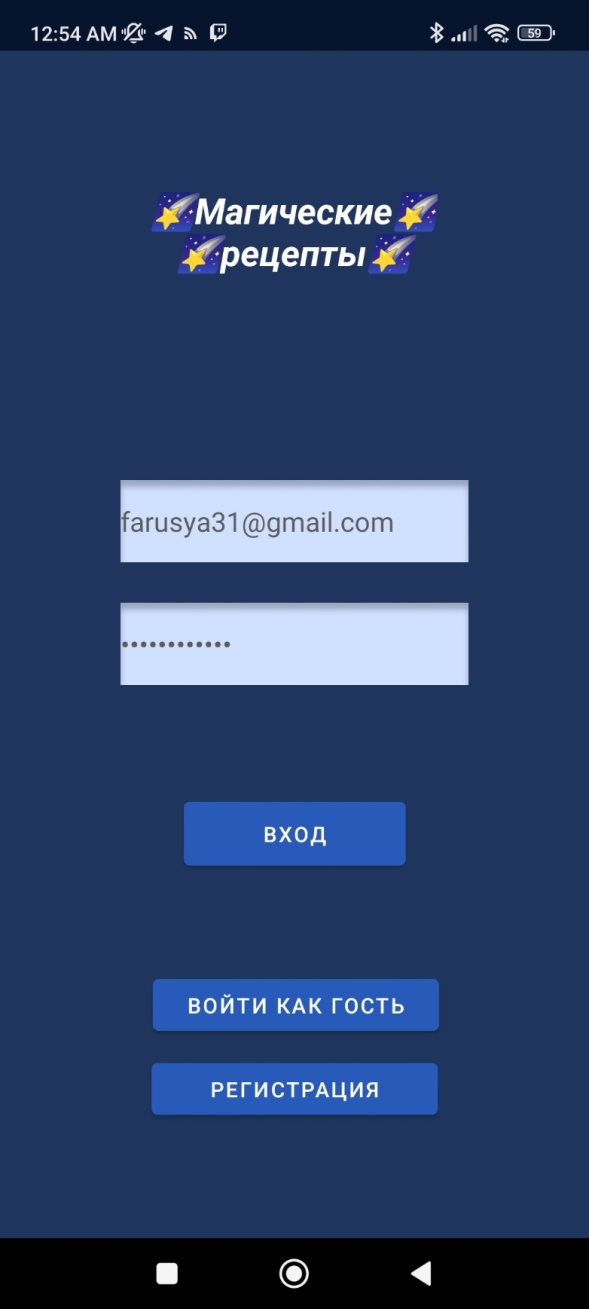


Рисунок 43 - Пароль при входе в аккаунт защищен шифрованием

Тестирование защиты данных пользователей при входе окончилось успехом.

2.2 Тестирование защиты данных пользователей при регистрации:

Ввод пароля при регистрации аккаунта нового пользователя защищено шифрованием для того, чтобы злоумышленники не могли подсмотреть и украсть аккаунт. Отображается лишь последний введенный символ, а все предыдущие отображаются в виде точек.

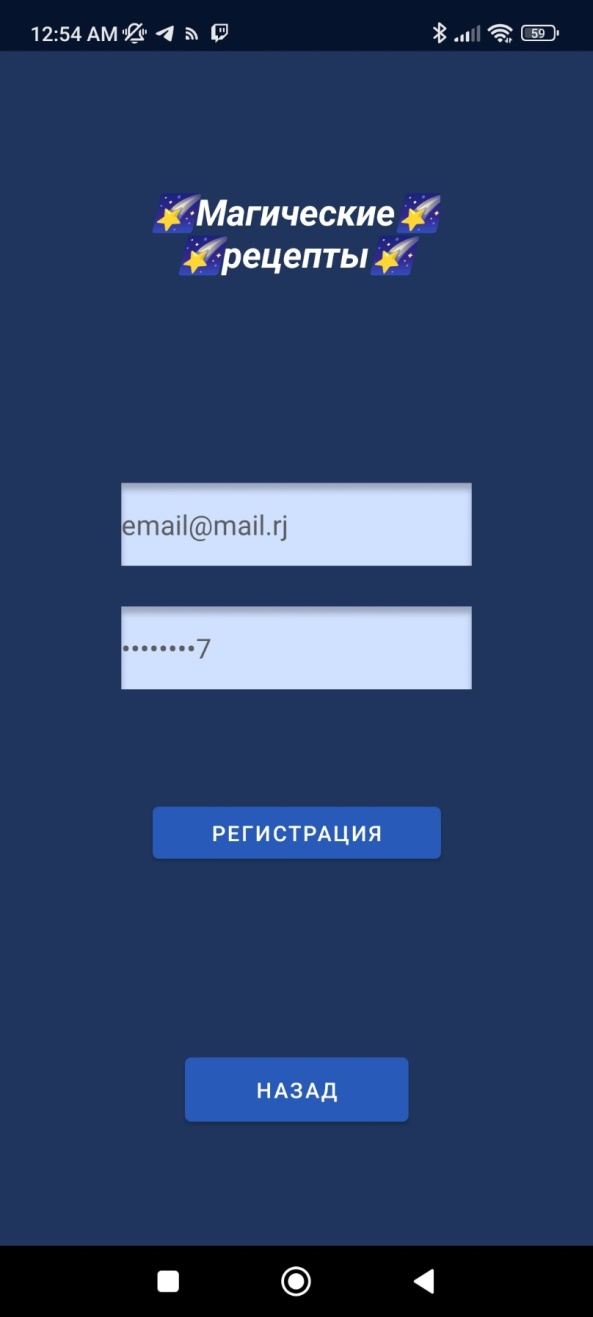


Рисунок 44 - Пароль при регистрации защищен шифрованием

Тестирование защиты данных пользователей при регистрации окончилось успехом.

1. Юзабилити тесты

Для оценки удобства и интуитивности использования мобильного приложения «Магические рецепты» был проведен юзабилити тест, включающий опрос 15 пользователей. В течение недели они активно использовали приложение, выполняя различные задачи.

Пользователи представляли собой разнообразную аудиторию с разным уровнем технической подготовки и интересами в кулинарии. Опрос был направлен на выявление их впечатлений, трудностей, с которыми они столкнулись. В ходе тестирования оценивались ключевые аспекты, включая интуитивность интерфейса, удобство навигации, восприятие дизайна и реакция на ошибки. Оценка проводилась по пятибалльной шкале.

3.1 Тестирование интуитивности интерфейса:

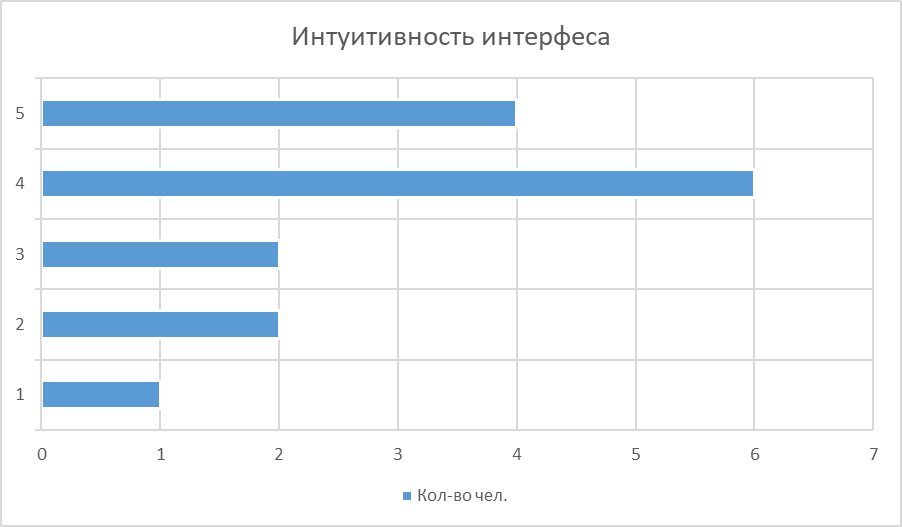


Рисунок 45 - Результаты опроса «Интуитивность интерфейса»

Пользователи оценили интуитивность интерфейса следующим образом:

1: 1 человек.

2: 2 человека.

3: 2 человека.

4: 6 человек.

5: 4 человека.

Большинство пользователей (10 из 15) оценили интуитивность интерфейса на уровне 4 и 5, что указывает на хорошую воспринимаемость и удобство использования интерфейса.

3.2 Тестирование удобства навигации:

Оценки удобства навигации распределились следующим образом:

1: 0 человек

2: 2 человека

3: 3 человека

4: 5 человек

5: 4 человека

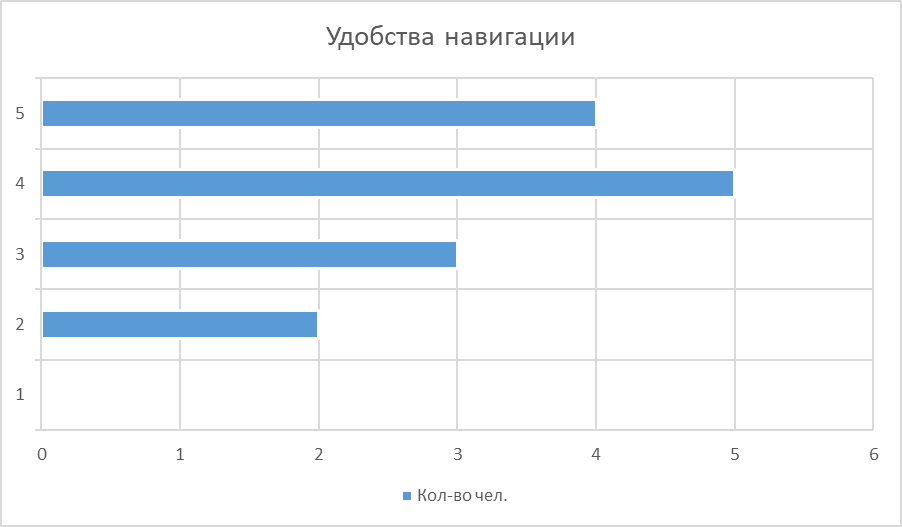


Рисунок 46 - Результаты опроса «Удобство навигации»

Небольшое количество пользователей (2 человека) нашли навигацию сложной, однако большая часть (9 человек) оценила её на 4 и 5, что говорит о хорошей навигационной структуре.

3.3 Тестирование восприятия дизайна:

Восприятие дизайна было оценено так:

1: 0 человек

2: 2 человека

3: 3 человека

4: 3 человека

5: 5 человек

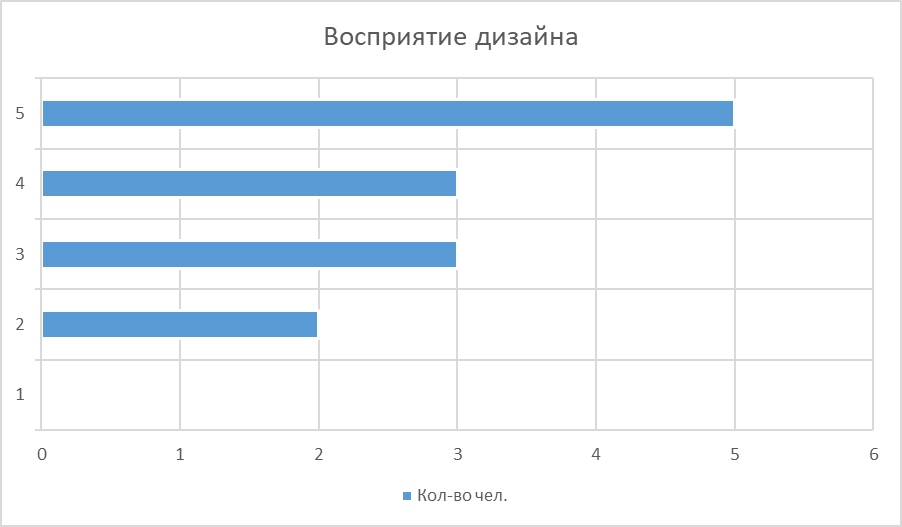


Рисунок 47 - Результаты опроса «Восприятие дизайна»

Большинство пользователей (8 из 15) оценили дизайн на 4 и 5, что свидетельствует о положительном восприятии визуального оформления приложения.

3.4 Тестирование функциональности форм:

Функциональность форм оценивалась следующим образом:

1: 1 человек

2: 3 человека

3: 0 человек

4: 2 человека

5: 9 человек

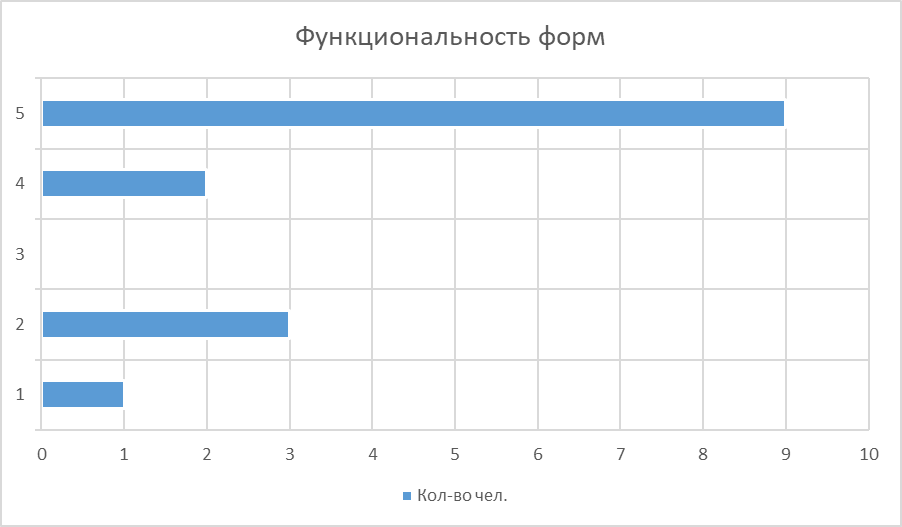


Рисунок 48 - Результаты опроса «Функциональность форм»

Большинство пользователей (11 из 15) высоко оценили функциональность форм, что указывает на их удобство и правильное функционирование.

3.5 Тестировании реакции на ошибки:

Оценки реакции на ошибки были следующими:

1: 2 человека

2: 4 человека

3: 1 человек

4: 5 человек

5: 9 человек

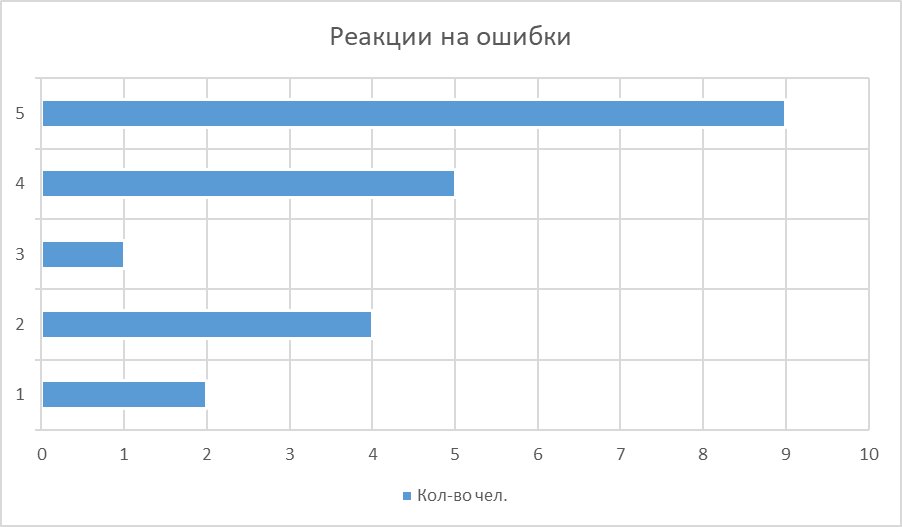


Рисунок 49 - Результат опроса «Реакции на ошибки»

Несмотря на несколько низких оценок, большинство пользователей (14 из 15) оценили реакцию на ошибки положительно, что говорит о корректной работе системы обработки ошибок.

3.6 Общие выводы юзабилити тестов:

Опросы показали, что пользователи в целом удовлетворены работой приложения «Магические рецепты». Большинство оценок интуитивности интерфейса, удобства навигации, восприятия дизайна, функциональности форм и реакции на ошибки находятся на высоком уровне. Это свидетельствует о том, что приложение предоставляет пользователям удобный и приятный опыт. Тем не менее, были выявлены некоторые области, требующие улучшений, такие как улучшение навигации и устранение некоторых ошибок. Эти результаты будут учтены для дальнейшей оптимизации и улучшения приложения.

4 Экономическая часть

1. Теоретическая часть

Расчет стоимости создания разработки зависит от объема работы и ресурсов, которые вложены в нее. В данной секции необходимо провести оценку затрат, связанных с разработкой. Затраты, издержки и себестоимость являются ключевыми финансовыми показателями, определяющими прибыль и эффективность деятельности предприятия. Сокращение и оптимизация расходов являются стратегическими приоритетами в улучшении финансовой устойчивости и конкурентоспособности предприятия.

Издержки на осуществление изготовления и реализации товаров представляют собой расходы компании, выраженные в денежной форме, и связанные с использованием сырья и материалов, компонентов продукции, топлива, энергии, рабочей силы, основных средств и прочих затрат не являющихся капитализированными. Они включаются в стоимость производимой продукции, уровень которой определяет прибыль, рентабельность производства и капитала, а также другие конечные результаты финансово-экономической деятельности предприятия.

В рыночной экономике все издержки рассматриваются как альтернативные, что подразумевает, что стоимость каждого использованного в производстве ресурса равна его ценности при оптимальном применении. Этот принцип управления фирмой позволяет ей приобретать ресурсы, выбирая их с учетом наиболее выгодного варианта использования и отвлекая их от возможного использования другими предпринимателями.

Себестоимость продукции – это оценка стоимости использованных в процессе производства природных и трудовых ресурсов. [15] Она отражает текущие затраты, связанные с производством и не являющиеся капитальными, необходимые для обеспечения процесса воспроизводства на предприятии. Себестоимость представляет собой экономический механизм возмещения затраченных производственных ресурсов.

Состав и структура затрат, включаемых в себестоимость продукции. [15]

Расходы, включаемые в общую себестоимость, могут быть разделены на следующие категории: издержки на материалы, оплата рабочей силы, взносы на социальные нужды, амортизация основных средств и прочие расходы. Формирование их структуры определяется различными факторами, включая тип производимой продукции и потребляемые сырьевые ресурсы, особенности технико-экономического процесса, его размещение и организацию, а также условия снабжения и реализации продукции.

Виды затрат, включаемые в себестоимость продукции, регулируется положением об их составе, утвержденным Правительством РФ.

Затраты на материальные ресурсы являются основной составляющей себестоимости продукции в различных областях экономики. Они включают в себя сырье, основные материалы, покупные полуфабрикаты, комплектующие, вспомогательные материалы, топливо, энергию и упаковочные материалы. Сырье и основные материалы представляют собой ресурсы, приобретаемые извне, которые составляют основу продукта или являются ключевым компонентом. Покупные полуфабрикаты и комплектующие поступают на предприятие для последующей обработки или сборки. Вспомогательные материалы добавляются к основным для изменения их характеристик или оформления, а также для производственного обслуживания. Топливо и различные виды энергии, приобретаемые извне, занимают особое место среди материальных затрат из-за их важности для народно-хозяйственной деятельности.

В состав каждой категории материальных расходов входят только расходы на ресурсы, которые приобретаются от сторонних поставщиков. Если компания выпускает собственные ресурсы (например, тепловую энергию, холод и другие), или выполняет определенные работы своими силами (например, разгрузочные операции, транспортировку и прочее), то затраты на них распределяются по соответствующим категориям: оплата труда, амортизация основных средств и другие.

Оценка стоимости материальных ресурсов, включаемых в себестоимость продукции, основывается на их закупочной цене (без учета НДС), а также всех дополнительных сборов и комиссий, уплачиваемых поставщикам, посредникам и торговым организациям, а также на расходах на услуги бирж, таможенные сборы и транспортировку. Исключаются расходы на возвратные отходы, то есть остатки материалов и сырья, которые потеряли свои потребительские свойства и не могут быть использованы возвращением к исходному назначению. Оценка возвратных отходов зависит от возможности их повторного использования и может быть полной или сниженной относительно исходной цены материальных ресурсов.

Расходы на оплату труда отражают участие затрат на рабочую силу в общей стоимости производства. В состав оплаты труда входят: заработная плата, которая рассчитывается на основе сдельных расценок, тарифных ставок и окладов в соответствии с принятыми на предприятии системами оплаты труда; стоимость продукции, выплачиваемой в натуральной форме; дополнительные надбавки; премии за результаты работы; стоимость услуг, предоставляемых бесплатно; единовременные выплаты за стаж работы.

Расходы на общественные нужды означают перераспределение национального дохода для финансирования общественных целей. Отчисления социального налога происходят от заработной платы трудящихся и являются основой для этого.

Начисление амортизации на полное восстановление включает в себестоимость производства определенные суммы, рассчитываемые на основе балансовой стоимости активов и устанавливаемых норм амортизации. Этот процесс включает в себя начисление износа как на собственные активы, так и на арендованные, а также применяется к стоимости помещений, которые предоставляются бесплатно предприятиям в общественном питании и медицинском обслуживании трудовых коллективов.

В общие затраты включены различные и многочисленные расходы. Их доля в общей стоимости за последние десять лет значительно увеличилась в различных отраслях экономики и превысила процент заработной платы. В сфере торговли они стали доминирующим элементом. Среди прочих затрат значительную часть составляют налоги, платежи и сборы, включая обязательное страхование и сборы за выбросы вредных веществ в окружающую среду, а также расходы на пожарную и охранную службу сторонних организаций, обучение и переобучение персонала, сертификацию продукции, гарантию по ремонту и обслуживанию и прочее. В прочие затраты входят вознаграждения работникам за изобретения и предложения по рационализации, командировочные, поощрения, аренда промышленных оборудованных помещений (если они используются на предприятии) и иные расходы.

Классификация затрат.

Затраты на производство и реализацию продукции классифицируются по ряду признаков:

* по функциям в процессе производства, разделяя на основные и накладные затраты. Основные расходы являются частью создания продукта и составляют его физическую основу: такие как сырье, материалы, заработная плата. Накладные затраты связаны с обслуживанием производственного процесса, включая содержание оборудования, цехового и административного персонала;
* по интеграции в общую себестоимость продукции на прямые и косвенные. Прямые затраты могут быть точно отнесены к себестоимости каждой единицы продукции, таких как сырье, основные материалы, энергия и заработная плата. Косвенные расходы сначала объединяются по определенным критериям и затем пропорционально распределяются между группами продукции на основе избранной базы: значений прямых затрат или зарплаты производственных работников;
* по связи с изменением объема выпуска продукции классифицируют как постоянные и переменные затраты. Величина постоянных расходов остается постоянной независимо от объема производства, в то время как переменные расходы колеблются в зависимости от динамики выпуска продукции;
* по периодичности использования в производстве различают каждодневные затраты, которые происходят регулярно, и единовременные затраты, происходящие реже одного раза в месяц;
* по методам планирования, учета и распределения затрат в соответствии с экономическим элементами - по структуре затрат и по местоположению их осуществления - по статьям калькуляции.

Группировка затрат по экономическим элементам. [15]

Распределение расходов по экономическим компонентам отражено в бюджете на производство и продажу продукции (работ, услуг). Здесь объединяются расходы по общности экономического содержания и по их естественному предназначению.

Общий объем ресурсов, затраченных в производственном процессе, называется "Итоговыми затратами на производство". С учетом этой суммы производится вычет не производственных расходов, что позволяет определить производственную себестоимость продукции.

Распределение затрат по экономическим категориям играет значительную роль для компании. Анализ затрат помогает установить общий объем используемых различных ресурсов на предприятии.

В бюджете учитываются расходы на ресурсы, приобретаемые у сторонних поставщиков. Предвиденные в бюджете расходы обеспечивают все нужды предприятия, связанные с изготовлением и сбытом продукции, разработкой незавершенного производства, поддержкой вспомогательных процессов и служб, включая обслуживание, вспомогательные и вспомогательные службы.

План производственно-финансовой деятельности предприятия основывается на разработанной смете, которая включает анализ потребности в материально-техническом обеспечении, рабочей силе, а также в оборотных средствах и других аспектах. Все виды бизнес-планов в первую очередь учитывают показатели сметы. Экономические взаимосвязи предприятия со своими поставщиками и финансово-кредитными учреждениями строятся с учетом сметы затрат на производство.

Однако, с помощью сметного разреза невозможно точно установить, где и как используются затраты (в производственных процессах, для обслуживания цеха и прочее), что мешает оценить эффективность использования ресурсов и выявить потенциальные возможности экономии. Особенно важно отметить, что на основе детальных данных в смете невозможно определить точную себестоимость единицы произведенной продукции для различных видов продукции или отдельных групп товаров. Для эффективного решения этих задач необходимо провести классификацию затрат по калькуляционным статьям.

Группировка затрат по статьям калькуляции. [14]

Классификация затрат по статьям калькуляции играет важную роль в определении стоимости производства, распределении затрат между группами товаров, определении объема затрат по различным видам деятельности, подразделениям и управленческим подразделениям, а также выявлении возможностей сокращения расходов. Принцип группировки затрат по калькуляционным статьям лежит в основе формирования плана счетов бухгалтерского учета в различных областях экономики. Отчетность преимущественно составляется и анализируется на основе данных статей калькуляции.

При классификации расходов по категориям в учете, они объединяются согласно их целям использования и месту происхождения, будь то в процессе изготовления продукции, обслуживании предприятия, управлении и т.д.

Типовая группировка затрат по статьям калькуляции имеет

следующий вид:

* материалы;
* основная заработная плата программиста;
* дополнительная заработная плата программиста;
* отчисления на социальные нужды от заработной платы программиста;
* амортизация компьютера;
* амортизация программного обеспечения;
* накладные расходы.

Расчет себестоимости единицы конкретного вида продукции или работ осуществляется по средствам калькулирования по установленным статьям затрат.

1. Расчет стоимости разработки приложения

Таблица 2 - Стоимость расходных материальных ресурсов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование материалов | Единица измерения | | Количество | Стоимость в рублях | | | | |
| Единица | | | Всего | |
| Диски CD -RW | шт. | | 1 | 88 | | | 88 | |
| Бумага А4 пачка | шт. | | 1 | 1 | | | 350 | |
| Ручка | шт. | 1 | | | 70 | 70 | |
| Итого | | | | | | 508 | |

Заработная плата — это основной вид вознаграждения, который работники получают за свой труд и который стимулирует их заинтересованность в результатах работы. [14] Существует основная и дополнительная заработная плата. Основная заработная плата включает в себя оплату за отработанное время, количество и качество выполненной работы, и несчастных случаев, не зависящих от работников.

При оплате времени сотрудника по часам или по рабочему дню, его заработок определяется с учетом тарифной ставки или оклада. При разработке программистом заработок рассчитывается в соответствии с требованиями повременной оплаты труда. Заработная плата определяется умножением отработанного времени на тарифную ставку. Повременщикам также может быть установлен оклад, который является фиксированной месячной ставкой.

Месячная тарифная ставка (оклад) Junior программиста составляет 52000 рублей.

Время, затраченное на создание разработки мобильного приложения «Магические рецепты» составляет 79 дней.

Произведён расчёт трудозатрат на разработку программного продукта, показано в таблице 3.

Таблица 3 - Трудоемкость создания программного продукта

|  |  |
| --- | --- |
| Вид выполняемой работы | Затраченное время (дни) |
| Фиксация требований, проектирование | 2 |
| Разработка технического задания | 3 |
| Создание приложения | 60 |
| Отладка и тестирование | 10 |
| Разработка методического обеспечения | 4 |
| Итого дней | 79 |

Расходы на заработную плату будут рассчитываться исходя из 23 нормативного рабочего дня за май месяц. Так как создание программного продукта занимало не полностью весь месяц, а 79 дней, необходимо рассчитать зарплату за отработанные дни.

Заработная плата за 1 рабочий день:

руб, (1)

где - средняя заработная плата за 1 рабочий день;

 - месячная тарифная ставка (31500 руб.);

 - нормативное рабочее время (23 день).

руб.

 руб, (2)

где  - основная заработная плата за отработанное время;

 – отработанное рабочее время.

руб.

К дополнительной заработной плате относятся выплаты за не проработанное время, предусмотренные законодательством о труде. [14] К ним относятся оплаты очередных, льготных часов подростков, выходные пособия при увольнении, пособия по временной нетрудоспособности, оплата отпусков работникам предприятия.

Дополнительная зарплата составляет 11 % от основной заработной платы:

руб, (3)

где  - дополнительная заработная плата,

 - основная заработная плата.

руб.

Отчисления на социальные нужды представляют собой форму перераспределения национального дохода на финансирования общественных потребностей и составляют 11% от основной и дополнительной заработной платы. Начисленные в организациях средства направляют в бюджет и используются на предусмотренные законом цели.

руб, (4)

где  - отчисления от оплаты труда.

руб.

В состав себестоимости программного продукта включаются амортизационные отчисления по использованному компьютеру, принтеру и лицензированным нематериальным активам. [15]

Амортизационные отчисления определяются по формуле:

руб, (5)

где — амортизационные отчисления;

 — стоимость технических средств, рубли;

— норма амортизации, На = 0,8% (ПК), На = 5,8% (ПО);

— время на разработку, Тз = 60 дней.

— количество рабочих дней в году.

Таблица 4 - Состав КТС на разработку приложения

|  |  |  |
| --- | --- | --- |
| Наименование | Параметры | Цена, руб |
| Процессор | AMD Ryzen 3 3200G BOX | 9300 |
| Материнская плата | ASRock B450M-HDV R4.0 | 7300 |
| Память | DEXP [DEXP8GD4UD26] 8 ГБ | 3200 |
| Винчестер | 1024 ГБ 2.5" SATA накопитель DEXP C100 | 6100 |
| Видеокарта | GIGABYTE GeForce RTX 4060 D6 | 34000 |
| Блок питания | AeroCool ECO 600W | 3000 |
| Монитор | 23.8" Монитор Cooler Master GM238-FFS | 15500 |
| Корпус | DEXP DC-101B черный | 1800 |
| Клавиатура | Проводная DEXP Revenge [мембранная, клавиш - 87, USB, белая] | 800 |
| Мышь | Logitech M90 [910-001970] черный | 800 |
| Итого | | 81800 |

Таблица 5 - Состав лицензированного программного обеспечения

|  |  |
| --- | --- |
| Наименование показателя | Цена, рубли |
| Операционная система Windows 11 | 24000 |
| Figma | 4100 |
| MS-Office 2021 | 1800 |
| Итого: | 29900 |

Сумма амортизации составляет:

1 Персональный компьютер:

руб.

2 Программное обеспечение

руб.

Итого:

 руб, (6)

где  - амортизационные отчисления;

 - амортизация компьютера;

 - амортизация программного обеспечения.

руб.

Накладные расходы — это расходы, связанные с обслуживанием процесса производства и управления: амортизация, оплата обслуживающего и административного персонала, платежи за кредит, страхование, ремонт. [15]

Накладные расходы составляют 20 % от основной заработной платы.

**руб,(7)

где  - накладные расходы.

руб.

Сумма всех расходов составляет производственную себестоимость.

**руб,(8)

где  - производственная себестоимость;

 - стоимость материалов.

руб.

Рентабельность программного продукта составляет 10% производственной себестоимости.

Рентабельность продукции - показатель эффективности производства, определяемый отношением прибыли к ее себестоимости выраженный в процентах.

Стоимость разработки равна:

руб, (9)

где  - стоимость разработки, рубли.

руб.

Стоимость разработки: 173704 руб.

Заключение

В ходе выполнения выпускной квалификационной работы по теме разработка мобильного приложения «Магические рецепты» были достигнуты следующие результаты:

1. Разработка технического задания и проекта:

Подготовлены материалы и оформлено техническое задание, включающее анализ требований, структуру, дизайн и архитектуру мобильного приложения. Это позволило определить ключевые аспекты и уникальные характеристики будущего приложения.

1. Анализ и выбор технологий:

Проведен детальный анализ существующих аналогов мобильных приложений, что позволило выбрать наиболее подходящие средства разработки, включая язык программирования Java и платформу Firebase для создания базы данных.

1. Проектирование мобильного приложения:

Спроектирована структура мобильного приложения, определены основные сценарии использования и разработан UX-дизайн. Также создана ER-диаграмма базы данных, обеспечивающая хранение данных о пользователях и рецептах.

1. Программная реализация:

Выполнена программная реализация мобильного приложения с использованием Android Studio. Реализована база данных в Firebase, включающая хранение личных данных пользователей и избранных рецептов.

1. Методическое обеспечение:

Подготовлены руководства для программиста и пользователя, обеспечивающие полное описание функциональности и инструкции по использованию приложения.

1. Тестирование:

Проведено тестирование мобильного приложения, что позволило выявить и устранить ошибки. Тестирование подтвердило корректность работы всех компонентов приложения, включая авторизацию и работу с базой данных.

1. Экономический расчет:

Рассчитана себестоимость разработки приложения, в состав которой вошли затраты на амортизацию, оплату труда и накладные расходы.

Список используемых источников

1. Выработка требований: https://studfile.net/preview/8716048/page:7/
2. Как управлять аккаунтами пользователей (Firebase): https://support.google.com/firebase/answer/6400802?hl=ru&ref\_topic=6386702
3. Методы получения кода с заданной функциональностью и степенью качества: http://www.psu.ru/sveden/education/eduOp/college/09.02.03/oo/educationAnnotation/rppm/11.pdf
4. Модели процесса создания программного обеспечения: https://studfile.net/preview/1848692/page:11/
5. Основные методологии процессов разработки программного обеспечения: https://www.purrweb.com/ru/blog/metodologii-dlya-razrabotki-po/
6. Основные принципы разработки программного обеспечения: https://dzen.ru/a/ZN3HUZkQtSkhG2PO
7. Программистам: модели и методологии разработки ПО: https://dzen.ru/a/XwWAfdcJ-g5C5ntx
8. Процесс разработки дизайна мобильного приложения с нуля: https://medium.com/product-design/процесс-разработки-дизайна-мобильного-приложения-с-нуля-909c0443ef92
9. Разработка тестовых наборов и тестовых сценариев: https://www.ibm.com/docs/ru/engineering-lifecycle-management-suite/lifecycle-management/7.0.1?topic=story-lesson-21-create-test-scripts
10. РД 50-34.698-90 Руководство пользователя (пример оформления): https://www.prj-exp.ru/patterns/pattern\_user\_guide.php
11. Руководство программиста: https://documenterra.ru/rukovodstvo-programmista/
12. Требования к ПО: https://habr.com/ru/articles/501468/
13. Шлыкова, Т. Н. Экономика: методические указания / Т. Н. Шлыкова. — Самара: СамГАУ, 2022. — 84 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/244496 (дата обращения: 05.06.2024).
14. Кузнецова, О. Н. Теория бухгалтерского учета: учебное пособие / О. Н. Кузнецова, И. Г. Шарапиева. — Иркутск: Иркутский ГАУ, 2022. — 210 с.— Текст: электронный// Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/300134 (дата обращения: 05.06.2024).

Приложение

Листинг кода

Recipe.java:

public class Recipe implements Parcelable {

private String recipeID;

private String title;

private String description;

private String imageUrl;

private String type;

public Recipe() {

// Пустой конструктор нужен для Firestore

}

public Recipe(String recipeID, String title, String description, String imageUrl, String type) {

this.recipeID = recipeID;

this.title = title;

this.description = description;

this.imageUrl = imageUrl;

this.type = type;

}

protected Recipe(Parcel in) {

recipeID = in.readString();

title = in.readString();

description = in.readString();

imageUrl = in.readString();

type = in.readString();

}

public static final Creator<Recipe> CREATOR = new Creator<Recipe>() {

@Override

public Recipe createFromParcel(Parcel in) {

return new Recipe(in);

}

@Override

public Recipe[] newArray(int size) {

return new Recipe[size];

}

};

public String getRecipeID() {

return recipeID;

}

public String getTitle() {

return title;

}

public String getDescription() {

return description;

}

public String getImageUrl() {

return imageUrl;

}

public String getType() {

return type;

}

@Override

public int describeContents() {

return 0;

}

@Override

public void writeToParcel(Parcel dest, int flags) {

dest.writeString(recipeID);

dest.writeString(title);

dest.writeString(description);

dest.writeString(imageUrl);

dest.writeString(type);

}

public String toJson() {

Gson gson = new Gson();

return gson.toJson(this);

}

// Статический метод для создания объекта из JSON

public static Recipe fromJson(String json) {

Gson gson = new Gson();

return gson.fromJson(json, Recipe.class);

}

// Статический метод для создания объекта из документа Firestore

public static Recipe fromDocumentSnapshot(DocumentSnapshot document) {

Recipe recipe = new Recipe();

recipe.recipeID = document.getId();

recipe.title = document.getString("title");

recipe.description = document.getString("description");

recipe.imageUrl = document.getString("imageUrl");

recipe.type = document.getString("type");

return recipe;

}

}

FavoriteActivity.java

public class FavoriteActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_favourite);

mAuth = FirebaseAuth.getInstance();

if (isFirstTime()) {

showInstructionsDialog();

}

// Загружаем избранные рецепты

loadFavoriteRecipes();

}

// Отображение инструкции для избранного

private void showInstructionsDialog() {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Инструкции");

builder.setMessage("Удерживайте понравившийся рецепт для добавления в избранное." +

"\n\n\n\nПри удерживании избранного рецепта он будет удален из избранного.");

builder.setPositiveButton("OK", null);

AlertDialog dialog = builder.create();

dialog.show();

}

// Проверка первого захода в избранное

private boolean isFirstTime() {

SharedPreferences preferences = getSharedPreferences("MyPrefs", MODE\_PRIVATE);

boolean isFirstTime = preferences.getBoolean("isFirstTime", true);

if (isFirstTime) {

SharedPreferences.Editor editor = preferences.edit();

editor.putBoolean("isFirstTime", false);

editor.apply();

}

return isFirstTime;

}

// Загружаем избранные рецепты

private void loadFavoriteRecipes() {

SharedPreferences sharedPreferences = getSharedPreferences("Favorites", Context.MODE\_PRIVATE);

Map<String, ?> allEntries = sharedPreferences.getAll();

for (Map.Entry<String, ?> entry : allEntries.entrySet()) {

String recipeName = entry.getKey();

Recipe recipe = getRecipeByName(recipeName);

if (recipe != null) {

addRecipeToScrollView(recipe);

}

}

}

private Recipe getRecipeByName(String recipeName) {

SharedPreferences sharedPreferences = getSharedPreferences("Favorites", Context.MODE\_PRIVATE);

String json = sharedPreferences.getString(recipeName, null);

if (json != null) {

return Recipe.fromJson(json);

}

return null;

}

// Загружаем рецепты в скроллвью

private void addRecipeToScrollView(final Recipe recipe) {

LinearLayout layout = findViewById(R.id.linearLayout);

// Создаем родительский макет для изображения и текста

LinearLayout itemLayout = new LinearLayout(this);

itemLayout.setLayoutParams(new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.MATCH\_PARENT,

LinearLayout.LayoutParams.WRAP\_CONTENT

));

itemLayout.setOrientation(LinearLayout.VERTICAL);

itemLayout.setGravity(Gravity.CENTER\_HORIZONTAL);

// Создаем новый ImageView и устанавливаем параметры

ImageView imageView = new ImageView(this);

LinearLayout.LayoutParams imageParams = new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.WRAP\_CONTENT, // Ширина

LinearLayout.LayoutParams.WRAP\_CONTENT // Высота

);

imageView.setLayoutParams(imageParams);

imageView.setImageResource(recipe.getImageResourceId());

imageView.setScaleType(ImageView.ScaleType.CENTER\_CROP);

imageView.setAdjustViewBounds(true);

// Создаем новый TextView и устанавливаем параметры

TextView textView = new TextView(this);

LinearLayout.LayoutParams textParams = new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.MATCH\_PARENT,

LinearLayout.LayoutParams.WRAP\_CONTENT

);

textParams.setMargins(0, 16, 0, 0); // Добавляем верхний отступ

textView.setLayoutParams(textParams);

textView.setText(recipe.getName());

textView.setGravity(Gravity.CENTER);

textView.setTextSize(TypedValue.COMPLEX\_UNIT\_SP, 20);

// Добавляем обработчик нажатия на изображение

imageView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Открываем рецепт

openRecipe(recipe);

}

});

// Добавляем обработчик долгого нажатия на изображение

imageView.setOnLongClickListener(new View.OnLongClickListener() {

@Override

public boolean onLongClick(View v) {

// Создаем диалоговое окно для подтверждения удаления

AlertDialog.Builder builder = new AlertDialog.Builder(FavoriteActivity.this);

builder.setTitle("Удаление рецепта");

builder.setMessage("Вы уверены, что хотите удалить этот рецепт из избранного?");

builder.setPositiveButton("Да", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// Удаляем рецепт из избранного

removeFromFavorites(recipe.getName());

// Удаляем изображение и текст из ScrollView

LinearLayout layout = findViewById(R.id.linearLayout);

layout.removeView(itemLayout);

}

});

builder.setNegativeButton("Отмена", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// Ничего не делаем, закрываем диалоговое окно

dialog.dismiss();

}

});

// Показываем диалоговое окно

builder.create().show();

return true; // Возвращаем true, чтобы обработать событие длительного нажатия

}

});

// Добавляем изображение и текст в родительский макет

itemLayout.addView(imageView);

itemLayout.addView(textView);

// Добавляем родительский макет в ScrollView

layout.addView(itemLayout);

}

// Открыть рецепт

private void openRecipe(Recipe recipe) {

// Открываем рецепт

Intent intent = new Intent(FavoriteActivity.this, ProfileRecipe.class);

intent.putExtra("recipe", recipe);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

}

// Удалить из избранного

private void removeFromFavorites(String recipeName) {

// Удаляем рецепт из SharedPreferences

SharedPreferences sharedPreferences = getSharedPreferences("Favorites", Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.remove(recipeName);

editor.apply();

// Удаляем изображение и текст из ScrollView

LinearLayout layout = findViewById(R.id.linearLayout);

View viewToRemove = layout.findViewWithTag(recipeName); // Используйте тег для поиска соответствующего представления

if (viewToRemove != null) {

layout.removeView(viewToRemove);

}

Toast.makeText(FavoriteActivity.this, "Рецепт удален.", Toast.LENGTH\_SHORT).show();

}

// Метод для отображения сообщения пользователю

private void showMessage(String message) {

Toast.makeText(this, message, Toast.LENGTH\_SHORT).show();

}

// Кнопка выхода из аккаунта

public void ExtAccBttn(View view) {

FirebaseUser currentUser = mAuth.getCurrentUser();

if (currentUser != null) {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("Вы хотите выйти из аккаунта?");

builder.setPositiveButton("Да", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i) {

// Действие при подтверждении выхода из аккаунта

logout(); // Метод для выхода из аккаунта

}

});

builder.setNegativeButton("Нет", null);

AlertDialog dialog = builder.create();

dialog.show();

} else {

// Пользователь не авторизован, показать сообщение или выполнить другие действия

showMessage("Вы не авторизованы.");

}

}

public void FavouriteButton(View view) {

FirebaseUser currentUser = mAuth.getCurrentUser();

if (currentUser != null) {

Intent intent = new Intent(FavoriteActivity.this, FavoriteActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

} else {

Intent intent = new Intent(FavoriteActivity.this, LoginActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

}

}

public void CatalogButton(View view) {

if (isInternetAvailable()) {

Intent intent = new Intent(FavoriteActivity.this, CatalogActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

} else {

showNoInternetMessage();

}

}

public void BackToMain(View view) {

if (isInternetAvailable()) {

Intent intent = new Intent(FavoriteActivity.this, MainActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

} else {

showNoInternetMessage();

}

}

// Метод для проверки подключения к интернету

private boolean isInternetAvailable() {

ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY\_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

return activeNetwork != null && activeNetwork.isConnectedOrConnecting();

}

// Метод для отображения сообщения об отсутствии интернета

private void showNoInternetMessage() {

Toast.makeText(this, "Отсутствует подключение к интернету", Toast.LENGTH\_SHORT).show();

}

// Метод для выхода из аккаунта

private void logout() {

// выход из Firebase

FirebaseAuth.getInstance().signOut();

// Переход на экран загрузки

Intent intent = new Intent(FavoriteActivity.this, LoginActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

}

}

RegisterActivity.java

public class RegisterActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

private EditText emailEditText, passwordEditText;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

mAuth = FirebaseAuth.getInstance();

emailEditText = findViewById(R.id.editTextEmail);

passwordEditText = findViewById(R.id.editTextPassword);

}

public void registerUser(View view) {

String email = emailEditText.getText().toString();

String password = passwordEditText.getText().toString();

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

// Регистрация успешна

Toast.makeText(RegisterActivity.this, "Регистрация успешна.",

Toast.LENGTH\_SHORT).show();

} else {

// Регистрация не удалась

Toast.makeText(RegisterActivity.this, "Ошибка регистрации: " +

task.getException().getMessage(), Toast.LENGTH\_SHORT).show();

}

}

});

}

public void goBack(View view) {

startActivity(new Intent(this, LoginActivity.class));

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

}

}

LoginActivity.java

package com.example.magicrecipes;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

private EditText emailEditText, passwordEditText;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

mAuth = FirebaseAuth.getInstance();

emailEditText = findViewById(R.id.editTextEmail);

passwordEditText = findViewById(R.id.editTextPassword);

}

public void loginAsGuest(View view) {

showMessage("Гостевой вход выполнен.");

navigateToMainActivity();

}

public void loginUser(View view) {

// Получаем введенные пользователем данные

String email = getEmail();

String password = getPassword();

// Проверяем, что email и пароль не пустые

if (email == null || email.isEmpty() || password == null || password.isEmpty()) {

// Если email или пароль пусты, выводим сообщение об ошибке

showMessage("Пожалуйста, введите адрес электронной почты и пароль");

return;

}

// Выполняем попытку входа в аккаунт Firebase

mAuth.signInWithEmailAndPassword(email, password)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

// Сохраняем состояние пользователя после успешной авторизации

saveUserState();

// Если авторизация успешна, переходим на главный экран

showMessage("Авторизация успешна.");

navigateToMainActivity();

} else {

// Если авторизация не удалась, выводим сообщение об ошибке

showMessage("Ошибка авторизации: " + task.getException().getMessage());

}

}

});

}

// Метод для получения введенного пользователем email

private String getEmail() {

return emailEditText.getText().toString().trim();

}

// Метод для получения введенного пользователем пароля

private String getPassword() {

return passwordEditText.getText().toString().trim();

}

// Метод для отображения сообщения пользователю

private void showMessage(String message) {

Toast.makeText(this, message, Toast.LENGTH\_SHORT).show();

}

// Метод для перехода на главный экран

private void navigateToMainActivity() {

Intent intent = new Intent(LoginActivity.this, MainActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

}

public void goToRegister(View view) {

startActivity(new Intent(this, RegisterActivity.class));

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

}

@Override

protected void onStart() {

super.onStart();

// Проверяем состояние пользователя при запуске приложения

FirebaseUser currentUser = mAuth.getCurrentUser();

if (currentUser != null) {

// Пользователь уже авторизован, проверяем подключение к интернету

if (isInternetAvailable()) {

navigateToMainActivity();

} else {

navigateToFavoriteActivity();

}

}

}

// Метод для перехода на экран избранного

private void navigateToFavoriteActivity() {

Intent intent = new Intent(LoginActivity.this, FavoriteActivity.class);

startActivity(intent);

overridePendingTransition(R.anim.fade\_in, R.anim.fade\_out);

finish();

}

// Метод для проверки подключения к интернету

private boolean isInternetAvailable() {

ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY\_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

return activeNetwork != null && activeNetwork.isConnectedOrConnecting();

}

private void saveUserState() {

// Здесь вы сохраняете состояние пользователя, например, его идентификатор или другие данные,

// которые позволят вам идентифицировать пользователя в будущем.

SharedPreferences.Editor editor = getSharedPreferences("UserData", MODE\_PRIVATE).edit();

editor.putString("userId", mAuth.getCurrentUser().getUid());

editor.apply();

}

}