МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 «Математическое обеспечение и администрирование информационных систем»
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения кулинарного помощника»

Обучающийся А.А. Разуваев
(Инициалы Фамилия) (личная подпись)

Руководитель к.п.н., доцент, О.В. Оськина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема бакалаврской работы: «Разработка мобильного приложения кулинарного помощника».

Выпускная квалификационная работа состоит из введения, трёх глав, заключения и списка литературы, включающего 25 источников. Общий объем работы составляет 72 страницы, содержит 33 рисунка, 4 таблицы и 3 диаграммы.

Целью данной работы является создание удобного мобильного приложения, которое поможет пользователям находить рецепты, сохранять и добавлять рецепты для удобного использования. Приложение разработано для платформы Android с возможностью работы даже без доступа к интернету, что делает его особенно полезным для пользователей, предпочитающих готовить в любых условиях.

В первой главе проводится анализ предметной области, включая изучение деятельности компании, на базе которой выполнялась выпускная квалификационная работа. На основе этого анализа формулируются функциональные и технические требования к приложению, определяющие его архитектуру и ключевые возможности. Дополнительно в рамках исследования была разработана диаграмма использования, визуализирующая взаимодействие пользователя с системой. Для её создания применялось CASE-средство, что позволило стандартизировать процесс проектирования и наглядно отобразить основные сценарии работы приложения.

Во второй главе представлено подробное описание этапов проектирования мобильного приложения. Рассмотрены основные принципы построения внутренней логики, структура взаимодействия компонентов, а также подходы к разработке архитектуры системы. В качестве основной среды разработки была выбрана Android Studio инструмент для создания Android-приложений. В качестве языка программирования использована

Java, что обусловлено её стабильностью, широким распространением и хорошей интеграцией с Android SDK.

Третья глава посвящена практической реализации приложения. В ней последовательно описан процесс создания пользовательского интерфейса, разработки компонентов базы данных для хранения рецептов, а также внедрения основных функций, обеспечивающих интерактивность и удобство взаимодействия с системой. Отдельное внимание уделяется вопросам тестирования как автоматического, так и ручного с целью проверки корректности работы всех модулей, выявления возможных ошибок и оценки удобства использования приложения с точки зрения конечного пользователя.

Заключение содержит выводы о проделанной работе, достигнутых результатах и возможных направлениях дальнейшего развития приложения. Основным результатом работы стало готовое мобильное приложение, позволяющее пользователям искать и добавлять рецепты, а также удобно их сохранять.

Abstract

The title of the graduation work is Development of a Mobile Culinary Assistant Application.

The graduation work consists of an introduction, three parts, 33 figures, 4 tables, 3 diagrams, a conclusion, and a list of 25 references including foreign sources.

The aim of this graduation work is to address the growing demand for intuitive digital tools in everyday cooking by developing a functional and user-friendly culinary assistant that supports key features such as recipe discovery, nutritional analysis, and personal menu creation.

The object of the graduation work is the concept of a mobile culinary assistant with offline capabilities.

The subject of the graduation work is the development process of a mobile application focused on recipe management, food composition analysis, and user interaction design.

The key issue of the graduation work is the implementation of a mobile solution that combines usability, functionality, and accessibility for daily cooking-related tasks.

The graduation work may be divided into several logically connected parts which are literature review, design and implementation, and testing.

The first part presents the analysis of the subject area, including a review of existing culinary applications and platforms.

The second part describes the design stages, including technology selection, system architecture development, internal logic structure, and component interaction. Android Studio was chosen as the main development environment, and Java was selected as the programming language due to its stability, widespread usage, and compatibility with the Android SDK.

The third part is devoted to the practical implementation of the application. It covers the development of the user interface, creation of database components

for storing recipes and user data, and integration of core functionalities such as search, filtering, and nutritional calculations.

In conclusion, we'd like to stress that the developed application serves as a useful tool for anyone who wants to diversify their diet, experiment with new dishes, and monitor the composition of food products.

Nevertheless, more experimental data and user feedback are required to further enhance the application's functionality and adaptability.

The work is of interest to a wide circle of readers interested in educational and utility mobile application development.

Содержание

Введение 7
Глава 1 Анализ предметной области 9
1.1 Характеристика предприятия – места практики 9
1.2 Экосистема программной разработки 10
1.3 Выбор CASE-средств для описания процессов приложения 14
1.4 Требования к разрабатываемому мобильному приложению 16
Глава 2. Проектирование мобильного приложения 20
2.1 Выбор платформы для приложения 20
2.2. Выбор языка программирования 25
2.3. Выбор и обоснование методов решения задачи 26
2.4. Структура и компоненты системы 29
2.4.1. Обеспечение офлайн-доступа и локальное хранение данных 29
2.4.2. Эффективное представление данных с помощью RecyclerView. 30
2.4.3. Управление данными при помощи ViewModel и LiveData. 31
2.4.4. Вспомогательные компоненты интерфейса. 32
2.4.5. Диаграмма классов 32
Глава 3. Разработка алгоритма, программирование и откладка 35
3.1 Описания средств для решения задач 35
3.2 Выполнение задания 37
3.3 Тестирование приложения 53
Заключение 67
Список используемой литературы и используемых источников 69

Введение

В условиях стремительного развития цифровых технологий мобильные приложения стали неотъемлемой частью повседневной жизни, существенно упрощая решение бытовых задач. Особое значение приобретают специализированные приложения, такие как кулинарные помощники, которые помогают пользователям эффективно организовывать процесс приготовления пищи. Разработка таких приложений требует тщательного подхода к проектированию интерфейса и функциональности, чтобы обеспечить удобство использования и востребованность среди целевой аудитории.

Цель работы заключается в разработке мобильного приложения "Кулинарный помощник" для платформы Android, которое предоставляет пользователям:

- доступ к базе рецептов с возможностью поиска по ингредиентам;
- функционал для создания и сохранения собственных рецептов;
- удобную систему навигации и визуализации кулинарных рецептов.

Объект исследования это процесс разработки мобильных приложений для кулинарных целей, включая проектирование пользовательского интерфейса, организацию хранения данных и реализацию ключевых функций.

Предмет исследования разработанное Android приложение, использующее:

- язык программирования Java;
- библиотеку Room для локального хранения данных;
- компоненты Android Jetpack (ViewModel, LiveData);
- Material Design для создания интуитивного интерфейса.

Задачи исследования:

– анализ существующих решений в области кулинарных приложений;

– выбор технологического стека для разработки;

– проектирование архитектуры приложения;

– реализация основных функций в виде хранения и отображения рецептов, поиск по ингридиентам и создание пользовательских рецептов;

– тестирование работоспособности приложения.

Научная новизна работы заключается в:

– оптимизации процесса взаимодействия пользователя с кулинарными рецептами;

– разработке адаптивного интерфейса для различных категорий пользователей;

– реализации эффективного механизма поиска рецептов по имеющимся ингредиентам.

Практическая значимость:

– приложение предоставляет удобный инструмент для планирования домашнего приготовления пищи;

– разработанная архитектура может служить основой для создания аналогичных приложений.

Для тестирования работоспособности приложения использовались:

– эмуляторы Android Studio;

– реальные устройства с различными версиями Android;

– инструменты профилирования для оценки производительности.

Особое внимание уделено:

– удобству пользовательского интерфейса;

– стабильности работы приложения;

– эффективности работы с локальной базой данных;

Разработанное приложение демонстрирует стабильную работу и соответствует требованиям к мобильным приложениям в своей категории.

Глава 1 Анализ предметной области

1.1 Характеристика предприятия – места практики

Производственная практика проходила в компании «ВорлдИнтерТех РУС», которая специализируется на разработке программных решений и внедрении цифровых продуктов для нужд бизнеса. Данная организация функционирует в области информационных технологий, предлагая услуги, связанные с автоматизацией процессов, построением информационных систем, а также созданием мобильных и веб-приложений под заказ.

Главной задачей компании является не просто выпуск готового программного продукта, а создание технологически гибкой платформы, способной адаптироваться под конкретные задачи клиента. Основное внимание уделяется удобству конечного пользователя, скорости работы решений и их интеграции с уже существующей цифровой инфраструктурой компаний. Благодаря этому подходу организация на протяжении нескольких лет сохраняет положительную динамику развития в своей нише.

Во время прохождения практики особое внимание уделялось анализу внутренней структуры компании и ознакомлению с рабочими процессами в отделе разработки, что дало возможность лучше понять реальный цикл создания программного обеспечения и сложности, возникающие на каждом его этапе. Внутренняя структура предприятия включает в себя несколько отделов, каждый из которых выполняет строго определённые функции. Наиболее значимым в контексте данной работы стал отдел программной разработки, в рамках которого осуществляется проектирование, написание и тестирование программных продуктов. Разработка ведётся как для нужд внешних клиентов, так и для внутреннего применения в корпоративной инфраструктуре. Этот отдел играет важную роль в данной работе, потому что в рамках исследования разрабатывается прототип мобильного кулинарного

помощника. Приложение создается с учетом современных требований к удобству, быстродействию и функциональности, что соответствует основным принципам работы компании.

В «ВорлдИнтерТех РУС» используют передовые технологии и инструменты, позволяющие разрабатывать мобильные приложения с удобным интерфейсом и высокой производительностью. Важное внимание уделяется не только функциональности, но и простоте использования, чтобы конечный пользователь мог легко разобраться в продукте без лишних сложностей.

Разработка мобильного кулинарного помощника в рамках данной работы полностью соответствует стратегии компании – создавать инновационные и полезные цифровые решения. Разрабатываемое приложение, представляет собой инструмент для хранения и поиска кулинарных рецептов, полностью вписывается в профиль деятельности предприятия. Более того, технические и методические рекомендации, полученные в процессе дипломной работы, оказали значительное влияние на подход к архитектуре приложения и выбор технологий для его реализации.

1.2 Экосистема программной разработки

В компании «ВорлдИнтерТех РУС» чётко выстроена организационная структура, которая позволяет эффективно управлять проектами и распределять ответственность между отделами. На Рисунке 1 представлена схема взаимодействия между структурными единицами компании.

Рисунок 1 – Схема взаимодействия между отделами

В компании функционируют несколько отделов, каждый из которых отвечает за определенные задачи. Отдел программной разработки занимается созданием программного обеспечения, которое используется как внутри компании для повышения эффективности работы сотрудников, так и предоставляется внешним клиентам для улучшения качества их услуг.

Отдел программной разработки это ключевое подразделение, где создаются как мобильные, так и веб-решения. Именно здесь формируются технические задания, пишется код, проводится тестирование и осуществляется внедрение. Помимо этого, сотрудники отдела занимаются поддержкой и развитием уже готовых решений.

В отделе программной разработки работают следующие сотрудники:

– руководитель отдела;
– ведущий разработчик (senior);
– разработчик среднего уровня (middle);
– начинающий разработчик (junior).

Рассмотрим их функции подробнее на рисунке 2.

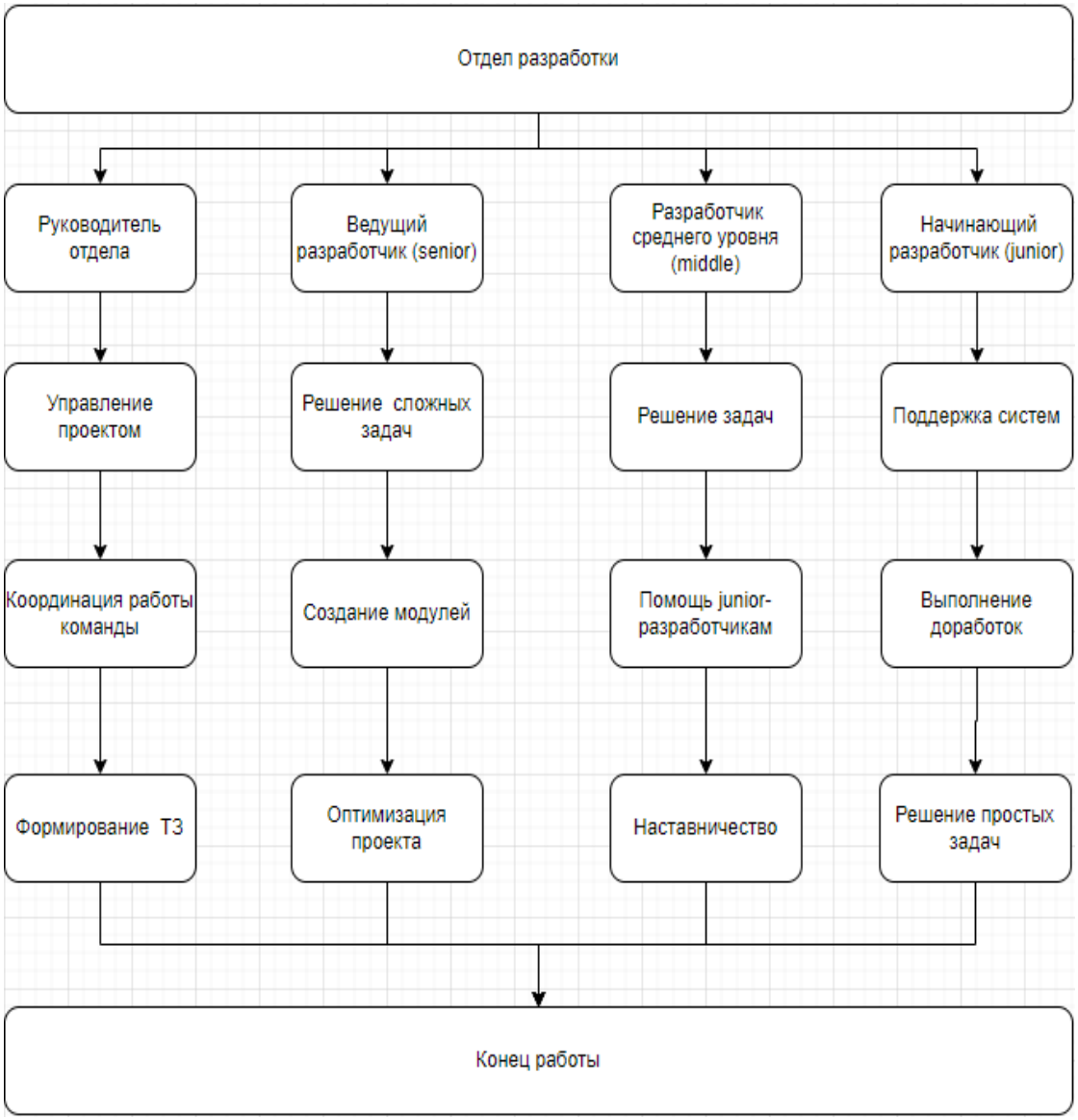


Рисунок – 2 Функции, выполняемые сотрудниками отдела разработок

Рассмотрим роли и задачи каждого из участников команды:

Начинающий разработчик (junior):

Специалист с минимальным опытом, чьи обязанности ограничены узкими задачами из-за недостатка знаний и опыта программирования. В основные обязанности входит поддержка существующих систем и внесение небольших улучшений. Может брать на себя более сложные задачи для развития навыков.

Разработчик среднего уровня (middle):

Специалист с опытом работы от 2 до 4 лет. Знает основной язык программирования из стека компании, хорошо ориентируется в структуре проекта, уверенно пользуется фреймворками, самостоятельно пишет код и решает задачи почти без ошибок. Мидл обладает насмотренностью, наработками, умеет решать задачи самостоятельно, но не всегда понимает общую картину и интеграцию в архитектуру проекта.

Ведущий разработчик (senior):

Более опытный специалист с опытом работы от 4 до 7 лет, которому поручаются наиболее сложные и нестандартные задачи. Он занимается проектированием архитектуры, созданием новых модулей, а также ревью кода коллег. Senior часто участвует в принятии ключевых технических решений и консультирует менее опытных участников команды

Руководитель отдела:

Отвечает за управление проектами, распределяет задачи и следит за соблюдением сроков. Его зона ответственности включает формулировку технических требований, контроль за качеством разработки, а также взаимодействие с другими отделами и клиентами. Помимо этого, Руководитель отдела участвует в оценке рисков, формировании отчётности и стратегическом планировании.

Каждая из перечисленных ролей имеет чётко определённую зону ответственности, и слаженность взаимодействия между ними является основой успешной работы над проектами. Благодаря такой иерархии и распределению задач, проекты компании выполняются качественно и в срок.

Разрабатываемое приложение разрабатывается с ориентацией на реальную производственную практику. Прототип был создан с учётом подходов, применяемых в отделе разработки, а также с использованием актуальных инструментов и технологий, используемых в компании.

1.3 Выбор CASE-средств для описания процессов приложения

Перед началом разработки мобильного приложения важно четко определить его процессы и структуру. Для этого используются различные инструменты моделирования, которые позволяют визуализировать процессы и взаимодействия внутри системы

Для описания бизнес-процессов существует несколько методов:

– текстовый метод, подробно описывает каждый этап с помощью слов;

– табличный метод, представляет информацию в виде структурированной таблицы или матрицы;

– графический метод, использует схемы и диаграммы для отображения процессов.

Решено было выбрать графический метод из за того что он позволяет наглядно предоставить информацию и облегчить восприятие информации.

«CASE-средства (Computer-Aided Software Engineering) это инструмент, который позволяет автоматизировать процесс разработки информационной системы и программного обеспечения. Разработка и создание информационных систем управления предприятием связаны с выделением бизнес-процессов, их анализом, определением взаимосвязи элементов процессов, оптимизации их инфраструктуры и т.д. Основной целью применения CASE средств является сокращение времени и затрат на разработку информационных систем, и повышение их качества. » [1]

Рассмотрим несколько CASE-средств такие как Lucidchart, Cacoo и Draw.io.

Все они имеют поддержку «IDEF0 (Integration Definition for Function Modeling) – это семейство методологий и нотаций, разработанных для моделирования различных аспектов деятельности организаций и систем. Изначально создано в рамках программы ICAM (Integrated Computer Aided Manufacturing) для ВВС США, целью которой было улучшение

производственных процессов. Сегодня методологии IDEF0 широко применяются в различных отраслях и бизнес-задачах: например, системном и бизнес-анализе, документировании процессов, управлении качеством на производстве, обучении персонала» [2], однако у Lucidchart нет строгой автоматической проверки синтаксиса IDEF0, а у Cacoo Нет готовых шаблонов IDEF0, и меньше возможностей изменять решение чем в аналогах:

– Lucidchart онлайн-сервис для создания диаграмм с возможностью совместной работы в реальном времени, но имеет ограничение на количество бесплатных диаграмм;

– Cacoo онлайн-сервис для создания диаграмм, позволяющий командам работать над схемами одновременно. Есть большое количество шаблонов и форм для различных типов диаграмм. Поддерживает шаблоны и интеграции с другими сервисами;

– Draw.io (diagrams.net) Бесплатное веб приложение которое позволяет экспортировать диаграммы в форматы JPG, PNG, SVG, можно сохранять схемы в Google Drive или на компьютер. Подходит и для маленьких учебных проектов, и для более серьёзных задач.

Для проведения сравнительного анализа будут использованы такие критерии, как доступность, простота освоения, возможность работы в браузере, возможность экспорта в разные форматы, разнообразие поддерживаемых методологий и ограничения на бесплатные схемы. Результаты анализа представлены в таблице 1.

Таблица 1 – Сравнение Case-средств

Критерий	Lucidchart	Cacoo	Draw.io
Бесплатность	-	-	+
Простота освоения	-	+	+
Работа в браузере	+	+	+
Возможность экспорта pdf, png	+	-	+
Ограничения на количество схем	+	+	-

Как видно из таблицы, Draw.io выигрывает по большинству параметров, особенно для небольших проектов, вроде курсовых или дипломов. Его не нужно устанавливать, работает в браузере, он не требует подписки, и в нем легко разобраться.

В рамках данной работы для построения диаграмм, описывающих взаимодействие пользователя с приложением, был выбран инструмент Draw.io. Онлайн-сервис позволяет наглядно представить процессы от поиска рецептов до добавления собственных блюд и составления меню, что способствует более эффективному проектированию структуры приложения.

1.4 Требования к разрабатываемому мобильному приложению

Перед началом разработки мобильного приложения нужно чтобы оно соответствовал определенным требованиям. Это позволит представить, какие задачи должно выполнять приложение и каким критериям должно соответствовать.

Все требования можно разделить на две основные категории, это «Функциональные требования (Functional Requirements) описывают, какие возможности должна предоставлять программа и как она обрабатывает ввод пользователя. Они задают правила работы системы: какие действия

доступны, как формируются результаты и какие данные передаются между интерфейсом, сервером и внешними сервисами.»[3] и «Нефункциональные требования это требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы.»[4]

При анализе существующих решений были сформулированы следующие функциональные требования для приложения:

– регистрация и авторизация пользователей для сохранения персональных данных и предпочтений;

– возможность выхода из учетной записи и удаления данных;

– каталог рецептов с функцией поиска;

– добавление и редактирование собственных рецептов;

– поиск рецептов по ингредиентам;

– добавление рецептов в избранное для быстрого доступа.

Помимо функциональных возможностей, приложение требует определения нефункциональных требований, которые напрямую влияют на удобство использования, стабильность работы и потенциальную масштабируемость проекта:

– приложение должно работать стабильно, не зависать и не выдавать ошибки при активном использовании;

– время отклика интерфейса должно быть минимальным, чтобы пользователь мог быстро находить нужную информацию;

– дизайн и интерфейс должны быть интуитивно понятными, чтобы даже новый пользователь легко разобрался в управлении;

– архитектура приложения должна быть гибкой, чтобы в будущем можно было легко добавлять новые функции без полной переделки системы;

– приложение должно корректно работать в офлайн-режиме, позволяя пользователям просматривать сохраненные рецепты без доступа к интернету;

– рабочая ориентация экрана – портретная, так как в этом режиме удобнее просматривать рецепты во время готовки.

Чтобы лучше понять, как пользователь будет взаимодействовать с приложением, можно представить это в виде диаграммы вариантов использования. Она представлена на рисунке 3 и показывает, какие действия доступны в приложении и какие основные сценарии работы с ним.

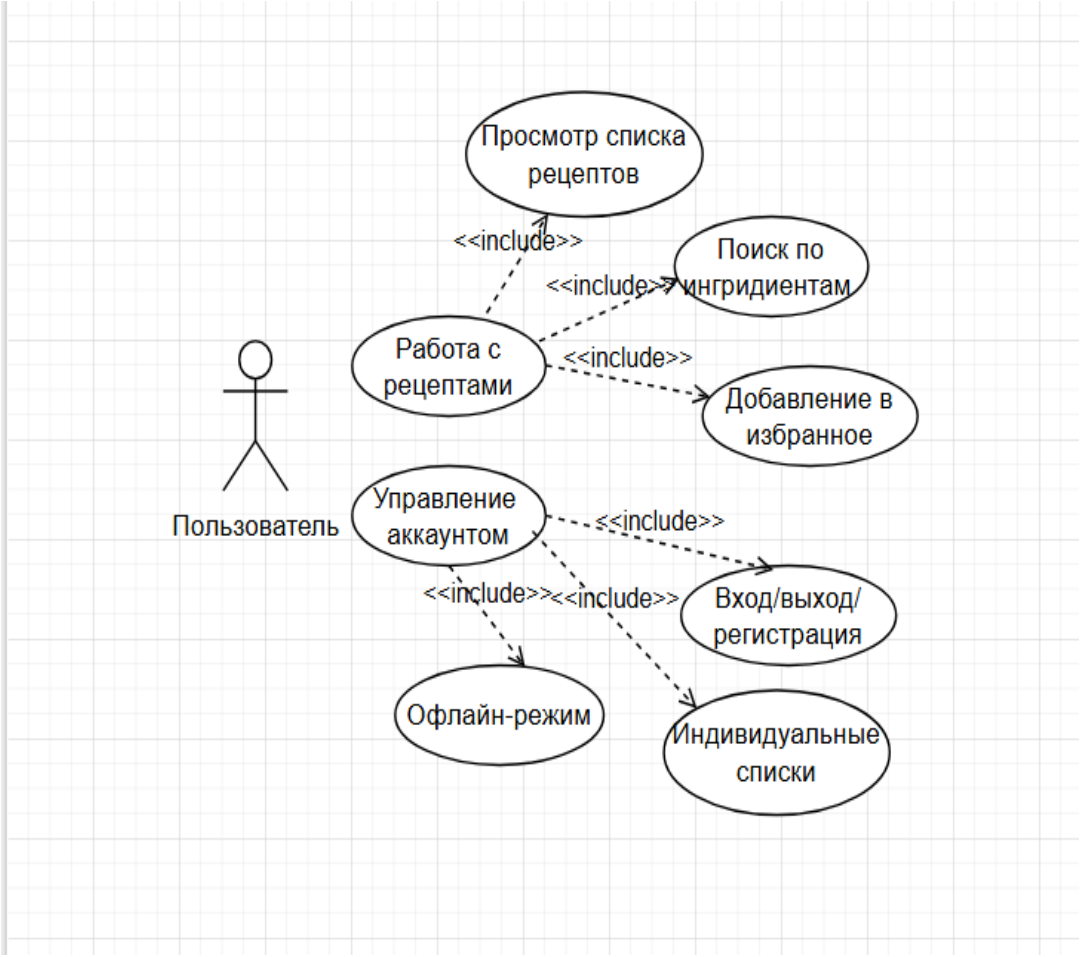


Рисунок – 3 Диаграмма действий и сценарии приложения.

Выводы по главе 1

В первой главе дипломной работы был проведён всесторонний анализ предметной области, связанной с разработкой мобильного кулинарного помощника и деятельности компании «ВорлдИнтерТех РУС». Компания специализируется на создании программных решений и внедрении цифровых продуктов для бизнеса, что делает её подходы и технологии подходящими для этого проекта.

Во время выполнения дипломной работы было изучено организационное устройство предприятия, с акцентом на отдел программной разработки, где осуществляется проектирование, написание и тестирование программных продуктов. Рассмотрены роли и обязанности сотрудников начиная от junior-разработчиков и заканчивая руководителем отдела, что позволило понять распределение задач и ответственности в команде.

При выборе Case-средств были исследованы различные CASE-инструменты, такие как Lucidchart, Cacoo и Draw.io. На основе анализа по критериям доступности, простоты освоения, набору функций и возможностей экспорта был выбран инструмент Draw.io, потому что он подходит по всем критериям.

Определены функциональные и нефункциональные требования к мобильному приложению, такие как стабильность работы, минимальное время отклика интерфейса, интуитивно понятный дизайн, и возможность офлайн-доступа к сохранённым рецептам. Также была разработана диаграмма вариантов использования, показывающая основные сценарии взаимодействия пользователя с приложением.

Глава 2. Проектирование мобильного приложения

2.1 Выбор платформы для приложения

Сегодня у большинства людей есть смартфон, который давно стал не просто средством связи, а настоящим помощником в повседневной жизни. Мы используем мобильные приложения буквально для всего от оплаты коммуналки до планирования питания. Поэтому идея создать удобное мобильное приложение-кулинарного помощника кажется вполне логичной и актуальной.

Прежде чем начинать разработку мобильного приложения, важно понять, для какой операционной системы оно будет создаваться. Сейчас на рынке представлено несколько мобильных ОС, но реально широко используются только две Android и iOS. Есть еще Windows Phone, но о ней чуть позже.

Рассмотрим эти платформы с точки зрения пяти ключевых критериев: функциональность, удобство, безопасность, стабильность и популярность. Это поможет объективно оценить, какая система подойдет лучше всего для моего проекта.

Функциональность:

Android это очень гибкая система, в ней много возможностей для разработчиков, можно использовать разные библиотеки, подключать сторонние сервисы.

iOS Apple жёстко ограничивает доступ к ряду системных функций. Из-за этого часть полезных функций просто невозможно реализовать без обходных путей.

Windows Phone когда-то был неплохим конкурентом, но сегодня он практически исчез с рынка, и даже базовые функции там реализованы не так удобно, как у Android и iOS.

Удобство использования:

Android хорошо знаком большинству пользователей всё логично, понятно, и каждый может подстроить интерфейс под себя.

iOS тоже очень прост в использовании. Интерфейс минималистичный, всё продумано и удобно, особенно для тех, кто давно пользуется техникой Apple.

Windows Phone имеет простой интерфейс, но непривычный. Плюс, сам подход к организации меню может вызвать затруднения у тех, кто с ним раньше не сталкивался.

Безопасность:

У Android с безопасностью стало намного лучше за последние годы: есть встроенные механизмы защиты, сканирование приложений в Google Play и т. д.

iOS традиционно считается более безопасной за счёт своей закрытой экосистемы и жёсткой модерации всех приложений.

У Windows Phone с безопасностью в целом всё было неплохо, но так как система устарела и почти не поддерживается, это делает её более уязвимой.

Стабильность:

Android развивается и обновляется постоянно. Конечно, стабильность может зависеть от конкретного устройства и производителя, но в целом система работает стабильно.

iOS это стабильность в чистом виде. Благодаря тому, что железо и система от одного производителя, всё работает максимально слаженно.

Windows Phone больше не развивается, обновления прекращены, а значит, стабильность там теперь это вопрос удачи.

Популярность и доступность:

Более 70% всех мобильных устройств в мире работают именно на этой системе. Приложение, сделанное под Android, потенциально доступно миллионам пользователей.

iOS тоже очень популярна, особенно в США и Европе. Но устройства Apple стоят дороже, и аудитория получается немного уже.

Windows Phone имеет мизерную долю рынка (около 0,02%), и рассчитывать на какую-то широкую аудиторию точно не стоит.

Все результаты данного анализа представлены в таблице 2, где указаны как достоинства, так и недостатки каждой операционной системы.

Таблица 2 – Сравнение операционных систем

Критерий	Android	iOS	Windows Phone
Функциональность	+	+	-
Удобство использования	+	+	-
Безопасность	+	+	-
Стабильность	+	+	+
Доступность	+	-	-

Выбор очевиден Android более доступен, открыт для разработчиков и охватывает гораздо больше устройств от разных производителей. Это значит, что моё приложение потенциально смогут использовать миллионы людей.

После определения платформы логично перейти к выбору инструмента, с которым будет вестись разработка. Существует несколько сред, с помощью которых можно создавать Android-приложения, и каждая из них имеет свои плюсы и минусы. Надо выбрать инструмент, который будет и прост в установке, и достаточно функционален, чтобы не ограничивать возможности проекта.

Visual Studio Code это скорее не полноценная IDE, а легковесный редактор с огромным количеством расширений. Если поставить нужные плагины и настроить SDK, можно писать и под Android. Но из коробки поддержка слабая, особенно если нужно тестировать приложение сразу.

Эмулятора нет, сборку приходится организовывать вручную. Подходит для опытных разработчиков, которые любят всё настраивать под себя.

Codename One — интересный вариант для кроссплатформенной разработки. Позволяет писать на Java и собирать приложение под Android, iOS и другие платформы. Работает через браузер или как плагин в других IDE. Удобно, но довольно ограниченно по возможностям — не всегда поддерживает последние версии Android SDK и может вести себя нестабильно при сборке.

AIDE (Android IDE) — приложение, которое позволяет писать Android-приложения прямо на смартфоне или планшете. Это очень удобно, если у тебя нет мощного ПК или хочется быстро протестировать идею «на ходу». Интерфейс простой, сборка APK доступна сразу, поддерживается Java и Kotlin. Однако при работе с большими проектами может быть неудобно — всё-таки экран телефона не заменит полноценный монитор.

Android Studio — IDE была изначально создана под Android, и это чувствуется сразу. В ней уже есть всё, что нужно: эмулятор, шаблоны, удобный редактор кода, подсказки, отладка и так далее. Она построена на базе IntelliJ IDEA, но адаптирована именно под мобильную разработку. Никаких плагинов вручную ставить не нужно — всё работает сразу после установки.

Чтобы было проще определиться, я сравнил эти среды по шести критериям в таблице 3.

Таблица 3 – Сравнение инструментов для разработки

Критерий	Visual Studio Code	Codename One	AIDE	Android Studio
Поддержка Android SDK	+/-	-	-	+
Встроенный эмулятор	-	-	-	+
Работа на мобильном устройстве	-	-	+	+
Поддержка Kotlin	+	-	+	+
Возможность собрать APK	+	+/-	+	+

По результатам проведённого анализа было принято решение использовать среду разработки Android Studio. Данная IDE разработана специально для создания мобильных приложений на платформе Android и включает в себя весь необходимый набор инструментов для реализации проекта. Android Studio предоставляет встроенный эмулятор мобильного устройства, современный редактор кода с интеллектуальными подсказками, поддержку языков программирования Java и Kotlin, а также широкий выбор шаблонов, упрощающих начальный этап разработки. В отличие от ряда альтернативных решений, Android Studio не требует дополнительной установки сторонних модулей и плагинов, что существенно снижает порог входа и упрощает процесс первоначальной настройки среды.

2.2. Выбор языка программирования

Android Studio поддерживает два основных языка программирования Kotlin и Java. Оба варианта активно используются в разработке под Android, и каждый из них имеет как свои преимущества, так и определённые особенности, которые следует учитывать при выборе.

Kotlin это сравнительно новый язык, разработанный компанией JetBrains и официально представленный в 2016 году. Он был создан с для упрощения и ускорения процесса разработки мобильных приложений. Основными достоинствами Kotlin можно назвать лаконичный синтаксис, удобную работу с API, встроенную защиту от null-ссылок и хорошую совместимость с кодом на Java, что особенно важно при переходе от одного языка к другому. Однако, язык пока ещё сравнительно новый, и в процессе решения задач может возникнуть нехватка примеров или подробной документации, особенно при решении нестандартных задач.

«Java это современный, объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems), и впервые представленный в 1995 году. Java сочетает в себе высокую производительность, надежность и простоту разработки благодаря своей универсальности и независимости от платформ.» Огромным преимуществом является масштаб экосистемы, потому что язык развивался очень долго, сформировался огромный список библиотек и фреймворков которые очень помогают при работе с приложениями такие как Gson для работы с JSON, разнообразные библиотеки для работы с базами данных, это ускоряет процесс разработки, потому что не нужно создавать базовые механизмы с нуля, также устойчивость и стабильность Java-платформы, обеспечиваемая долгосрочной поддержкой (LTS) версий и обратной совместимостью, минимизирует риски, связанные с долгосрочной поддержкой и развитием приложения в будущем. Кроме того большое количество документации и информации в интернете как по самому языку, так и по большинству

популярных библиотек, обилие обучающих материалов, форумов (Stack Overflow и др.), готовых примеров кода и решений распространенных проблем, позволяет в случае затруднения обратиться к ним.

2.3. Выбор и обоснование методов решения задачи

Первым шагом проектирования пользовательского интерфейса и пользовательского опыта мобильного приложения начался с моделирования его логической структуры и потоков взаимодействия. Для систематизации структуры приложения и логики взаимодействия пользователя для начала стало создание диаграммы состояний. Эта модель, разработанная с использованием стандартной нотации UML в редакторе Draw.io, наглядно отображает все основные экраны приложения (состояния), возможные действия пользователя (события) и правила перехода между ними, он также служит для наглядного представления совокупности возможных экранов приложения, и правил перехода между этими состояниями. Использование формальной нотации позволило не только систематизировать архитектуру экранов и их взаимосвязи, но и выявить потенциальные проблемы навигации на ранней стадии проектирования, минимизировав риски последующих изменений в кодовой базе.

Разработанная диаграмма состояний на рисунке 4 охватывает весь жизненный цикл взаимодействия пользователя с приложением, начиная с начальной точки входа.

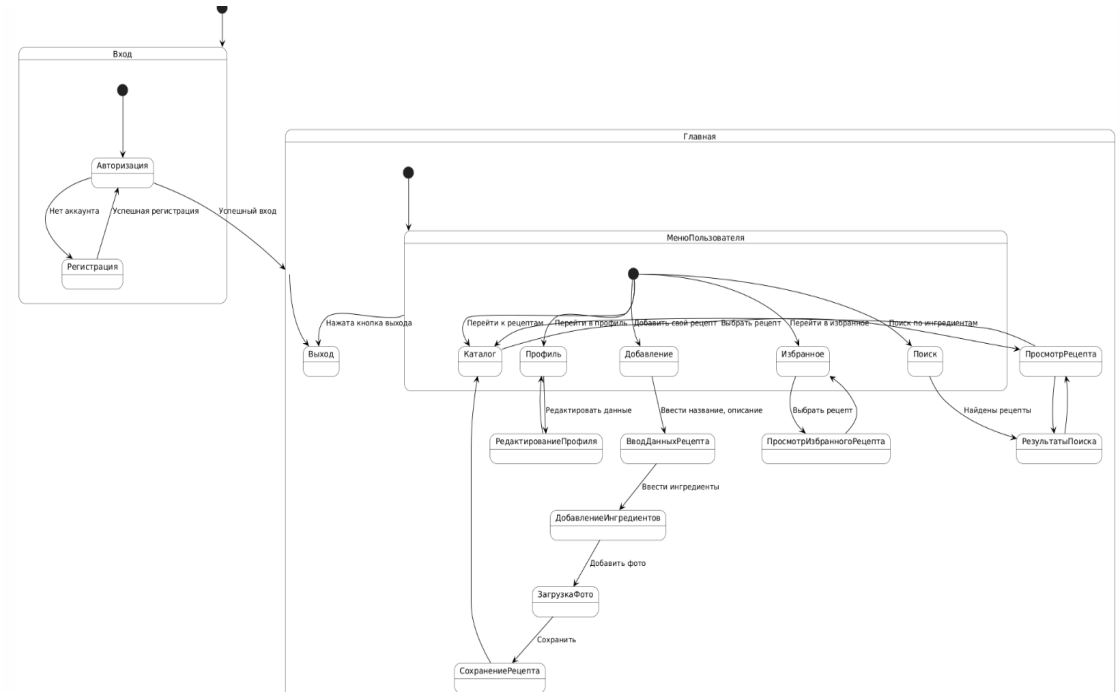


Рисунок – 4 Диаграмма состояний

Модель включает следующие ключевые состояния (экраны):

– авторизация это исходное состояние системы. Пользователю предоставляется возможность входа в существующую учетную запись. Также учтена альтернативная ветвь перехода для новых пользователей;

– регистрация это состояние для создания новой учетной записи, включающее ввод необходимых данных пользователя;

– главный экран это центральное состояние, доступное после успешной аутентификации. Представляет каталог рецептов, организованный по категориям;

– поиск рецептов это специализированное состояние для задания критериев поиска таких как доступных ингредиентов, названия блюда и отображения результатов, соответствующих введенным параметрам;

– просмотр рецепта это состояние детализированного представления выбранного рецепта оно содержит структурированные данные: список

ингредиентов с количественными показателями, пошаговую инструкцию приготовления а также элементы взаимодействия;

– добавление рецепта это состояние для создания пользователем собственного кулинарного рецепта,требует ввода метаданных таких как название, описание, категория, списка ингредиентов с указанием количества и единиц измерения и пошагового описания процесса приготовления а также предусмотрена загрузка изображения блюда;

– профиль пользователя это состояние для отображения и управления персональной информацией пользователя, а также доступом к сохраненным данным;

– избранное это состояние отображения рецептов, помеченных пользователем как предпочтительные;

– ввод ингредиентов это модальное окно или экран для удобного выбора или ввода ингредиентов при создании рецепта или использовании функции поиска;

– выбор категории это вспомогательный интерфейс для указания категории создаваемого рецепта;

– добавление фото это интерфейс для захвата изображения с камеры устройства или выбора из галереи.

Важным решением стало внедрение нижней панели навигации, эта общепринятая в мобильном дизайне паттерна «Material Design дизайн-система для создания интерфейсов программного обеспечения и приложений, разработанная компанией Google.»[5] обеспечивает:

− непрерывный доступ к ключевым разделам приложения таким как главный экран, поиск, профиль из любого состояния, кроме модальных окон и экранов авторизации/регистрации;

− прямую навигацию между основными разделами без необходимости последовательного возврата по иерархии;

− повышение удобности использования из за расположения основных элементов.

Построение диаграммы состояний обеспечило четкое понимание архитектуры взаимодействия, минимизировало риски логических ошибок на ранней стадии для создания интуитивно понятного и функционального пользовательского интерфейса приложения.

2.4. Структура и компоненты системы

Проектирование архитектуры мобильного приложения осуществлялось с приоритетом на надежность, производительность приложения и качество пользовательского опыта поэтому выбор инструментов и паттернов проектирования был обусловлен специфическими требованиями проекта, а именно, необходимость полнофункциональной офлайн-работы, эффективное управление коллекциями данных таких как рецепты, ингредиенты, избранное и устойчивость к изменениям конфигурации устройства например, поворот экрана.

2.4.1. Обеспечение офлайн-доступа и локальное хранение данных

Ключевым требованием от компании являлась возможность использования базового функционала (просмотр рецептов, доступ к избранному, создание новых рецептов) без зависимости от сетевого соединения. Эта задача была решена путем внедрения «Room Persistence Library – современной абстракции над SQLite, интегрированной в Android Jetpack». Room предоставляет мощный, безопасный и удобный в использовании API для работы с локальной базой данных. В контексте кулинарного приложения это означает, что:

− база рецептов, текст, метаданные хранятся локально на устройстве пользователя;

− операции чтения/записи (добавление нового рецепта пользователем, обновление статуса "Избранное", загрузка каталога) выполняются асинхронно и высокопроизводительно, минимизируя задержки интерфейса;

− сетевая нагрузка снижается до необходимого минимума для первоначальной синхронизации каталога или получения редких обновлений, что критично для пользователей с ограниченным трафиком или нестабильным интернетом;

− обеспечивается мгновенная реакция интерфейса на действия пользователя такие как добавление в избранное, поиск, потому что данные доступны локально.

2.4.2. Эффективное представление данных с помощью RecyclerView.

Для отображения обширного каталога рецептов, который потенциально может содержать сотни позиций, стандартный компонент ListView неприменим из-за проблем с производительностью и потреблением памяти. Вместо него был выбран RecyclerView – гибкий и хорошо оптимизированный компонент для представления больших динамических коллекций. Его преимущества для приложения:

− механизм повторного использования представлений позволяет эффективно управлять памятью, создавая только то количество элементов интерфейса (карточек рецептов), которое видно на экране, что критично для длинных списков;

− гибкая компоновка (LayoutManager): Позволяет легко реализовать различные типы отображения (линейный список, сетка) для каталога рецептов;

− четкое разделение ответственности между компонентами (Adapter для привязки данных, ViewHolder для хранения ссылок на элементы

View, LayoutManager для расположения) упрощает разработку и тестирование.

Каждый элемент списка (RecipeItemView) представляет рецепт в виде структурированной карточки, содержащей ключевую информацию: название блюда, основные ингредиенты в виде сокращенного списка, изображение в виде превью также время приготовления в минутах, калорийность и индикатор статуса "Избранное".

2.4.3. Управление данными при помощи ViewModel и LiveData.

Большой проблемой в Android является потеря данных и состояния интерфейса при смене конфигурации например, повороте экрана или временном уничтожении фонового Activity системой. Чтобы решить эту проблему были использованы компоненты ViewModel и LiveData из библиотеки Android Jetpack:

− ViewModel служит контейнером для данных, связанных с UI, но независимым от жизненного цикла Activity/Fragment. Данные например, загруженный список рецептов, результаты поиска, состояние формы добавления рецепта сохраняются при повороте экрана и также обеспечивает централизацию доступа к данным для одного или нескольких фрагментов и отделение логики подготовки данных от кода отображения;

− LiveData это держатель данных, который уведомляет об изменениях данных потому что автоматически учитывает состояние жизненного цикла и предотвращая утечки памяти, а также при изменении данных в базе данных Room, автоматически передает обновленные данные тоесть при добавлении рецепта в избранное через один экран, список избранного на другом экране обновится автоматически.

2.4.4. Вспомогательные компоненты интерфейса.

Для понятного и функционального интерфейса также использовались стандартные компоненты AndroidSdk такие как:

- EditText используется для ввода текста для названия рецепта, описания, поисковых запросов, ингредиентов при создании;
- Button используется для инициации действий, сохранить рецепт, добавить ингредиент, начать поиск, добавить в избранное;
- ImageView используется для корректного отображения фотографий блюд.

2.4.5. Диаграмма классов

Диаграмма на Рисунке 5 представляет основные взаимодействия ключевых компонентов приложения (ViewModel, Repository, DAO, Entity, Database).

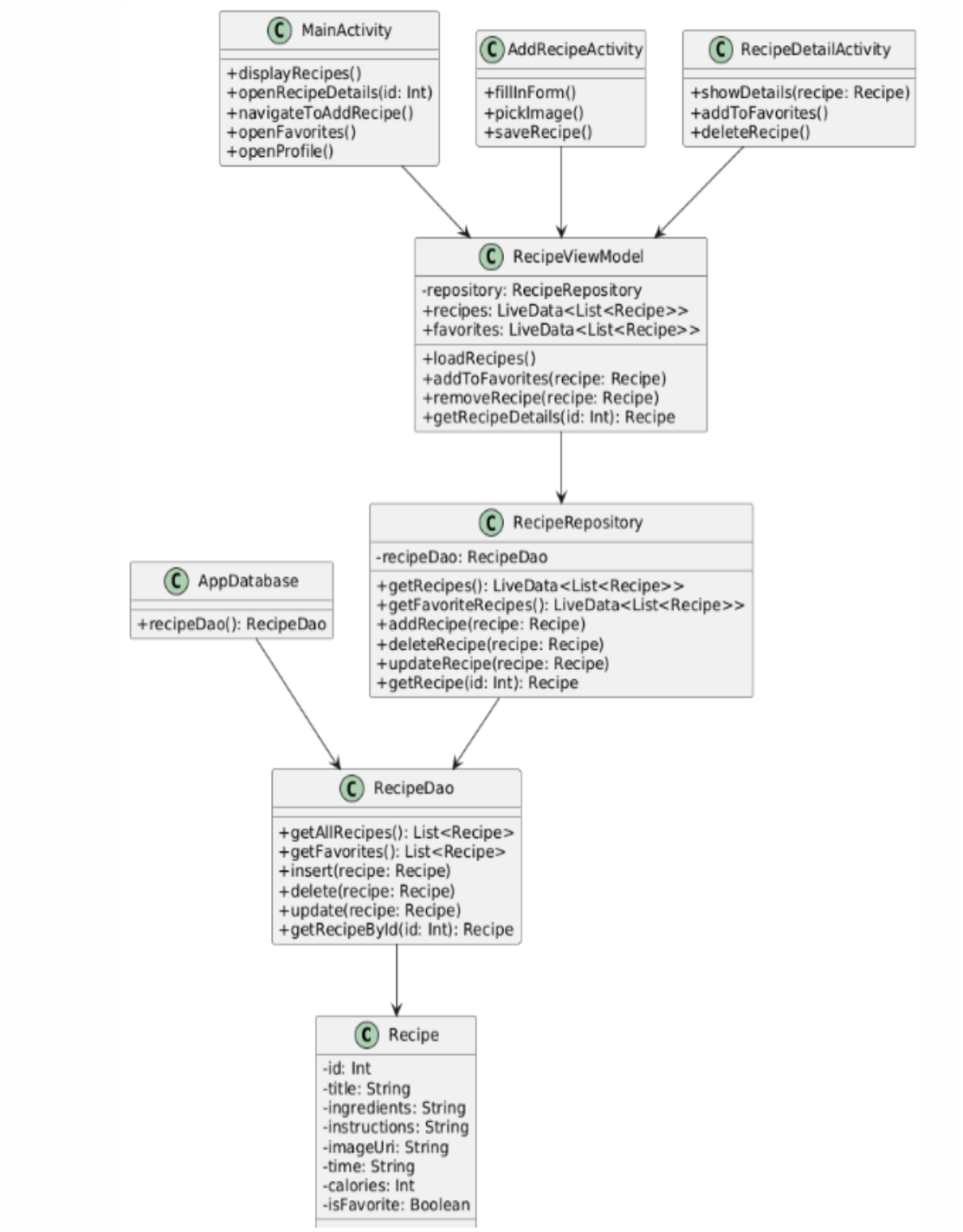


Рисунок – 5 Диаграмма классов

Диаграмма показывает основные взаимодействия классов:

− поток данных в виде запросов от ViewModel к Repository, которые транслируются в запросы к Dao, выполняемые RoomDatabase над сущностями (Recipe). Ответы возвращаются по цепочке, виде LiveData;

− ключевые публичные методы каждого компонента у Dao, свойства класса Recipe;

− зависимости между классами интерфейсы (Dao), реализация RoomDatabase и Repository.

Выводы по главе 2

Во второй главе была проведена работа по проектированию архитектуры мобильного приложения включая обоснование выбора платформы, инструментов разработки, языка программирования, методов проектирования интерфейса и структуры системы.

На основе анализа мобильных операционных систем (Android, iOS, Windows Phone) по критериям функциональности, удобства, безопасности, стабильности и популярности было принято решение разрабатывать приложение для Android. Эта платформа обладает наибольшей распространенностью открытостью для разработчиков и широкими возможностями кастомизации, что делает её оптимальным выбором для охвата максимальной аудитории.

Среди рассмотренных инструментов таких как Visual Studio Code, Codename One, AIDE, Android Studio предпочтение отдано Android Studio официальной IDE для Android-разработки, из за того что в нем имеется встроенная поддержка Android SDK, эмулятор, интеграция с языком java и Kotlin, готовые шаблоны проектов и удобные инструменты отладки.

В качестве основного языка выбор пал на Java потому что у него есть обширная библиотека для работ с данными, широкая поддержка сообщества и обратная совместимость.

Для стабильной и надежной работы приложения были выбраны основные компоненты, такие как Room для локального хранения рецептов, RecyclerView для отображения списка рецептов, ViewModel и LiveData для управления данными и стандартные компоненты AndroidSDK для взаимодействия с приложением.

Глава 3 Разработка алгоритма, программирование и откладка

3.1 Описания средств для решения задач

Разработка кулинарного помощника велась на языке Java в среде Android Studio, что позволило в полной мере использовать экосистему Android-разработки, включая Gradle гибкую систему сборки, упрощающая подключение библиотек, поддержку стандартов безопасности и android:exported для Android 12+.

Благодаря Room реализовано добавление, удаление, обновление и поиск рецептов, при этом данные сохраняются даже без подключения к интернету.

Для модели данных была создана сущность Recipe с полями title, description, imageUrl, ingredients, steps, stepImages и userId, дополнительно добавлен класс конвертации TypeConverter для сохранения списков изображений шагов.

В приложении реализована аутентификация пользователей через базу данных с таблицей User, содержащей email, пароль и никнейм. Никнейм отображается в профиле и используется для привязки рецептов. Архитектура приложения построена по шаблону MVVM.

Класс RecipeViewModel управляет данными рецептов и обеспечивает взаимодействие между репозиторием и пользовательским интерфейсом. Данные передаются через LiveData, что позволяет обновлять интерфейс автоматически при изменении информации в базе.

Взаимодействие с Room происходит через интерфейсы UserDao и RecipeDao, а также слой RecipeRepository, инкапсулирующий всю работу с базой. Для отображения рецептов используется RecyclerView с кастомным адаптером RecipeAdapter и элементами ViewHolder, что обеспечивает высокую производительность и плавную прокрутку даже при большом количестве данных.

Каждый рецепт отображается в виде карточки с изображением, названием и описанием, загружаемыми с помощью библиотеки Glide. При выборе рецепта осуществляется переход в RecipeDetailActivity, где отображаются полные шаги приготовления и изображения. Пользователь может создать рецепт через AddRecipeActivity, где реализована поддержка выбора изображения из галереи с использованием Intent.ACTION_PICK и хранения URI картинки.

В приложении реализована полноценная регистрация и вход пользователей, сохранение текущего пользователя в SharedPreferences, отображение профиля с никнеймом и кнопкой выхода. В приложении добавлена возможность сохранять рецепты в избранное. Для этого создана таблица FavoriteRecipe, связанная с пользователем и рецептами. Пользователь может добавлять или удалять рецепты из избранного, и они отображаются в соответствующем разделе. Также реализован функционал поиска рецептов по названию и ингредиентам. В RecipeDao добавлены SQL-запросы с оператором LIKE для фильтрации данных, а в SearchFragment реализован обработчик поиска и отображения результата.

Интерфейс приложения реализован с использованием Material Design. На всех экранах присутствует нижняя панель навигации BottomNavigationView, обеспечивающая доступ к четырем основным разделам: Главная, Поиск, Избранное и Профиль. Каждая вкладка реализована как Fragment, что позволяет избежать лишней перезагрузки данных и повысить отзывчивость интерфейса. Все элементы интерфейса адаптированы под современные устройства и обеспечивают удобное взаимодействие. Таким образом, соблюдена архитектура MVVM, обеспечена поддержка офлайн-доступа, реализовано удобное взаимодействие с базой данных через Room, обеспечена гибкая работа с изображениями и мультимедиа, а также реализована логика работы с пользователями, избранным и поиском. Таким образом приложение можно легко адаптировать под новые изменения.

3.2 Выполнение задания

Первым этапом для создания приложения было создание проекта на рисунке 6 в среде разработки Android Studio с использованием шаблона "Empty Activity".

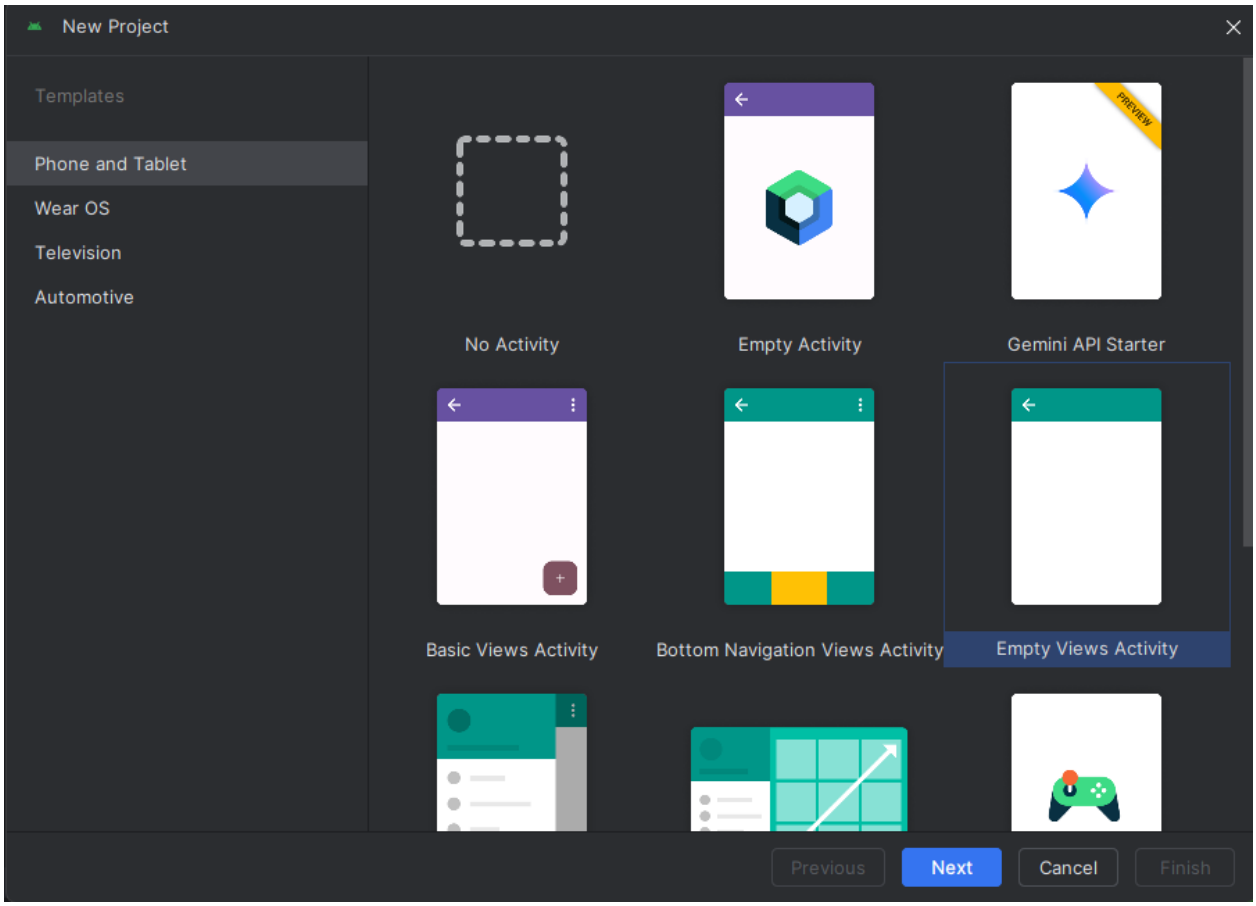


Рисунок – 6 Создание проекта

Были выбраны следующие параметры на рисунке 7:

– Название проекта: myapplication;

– Язык программирования: Java;

– Минимальная версия SDK: API 21 (Android 5.0).

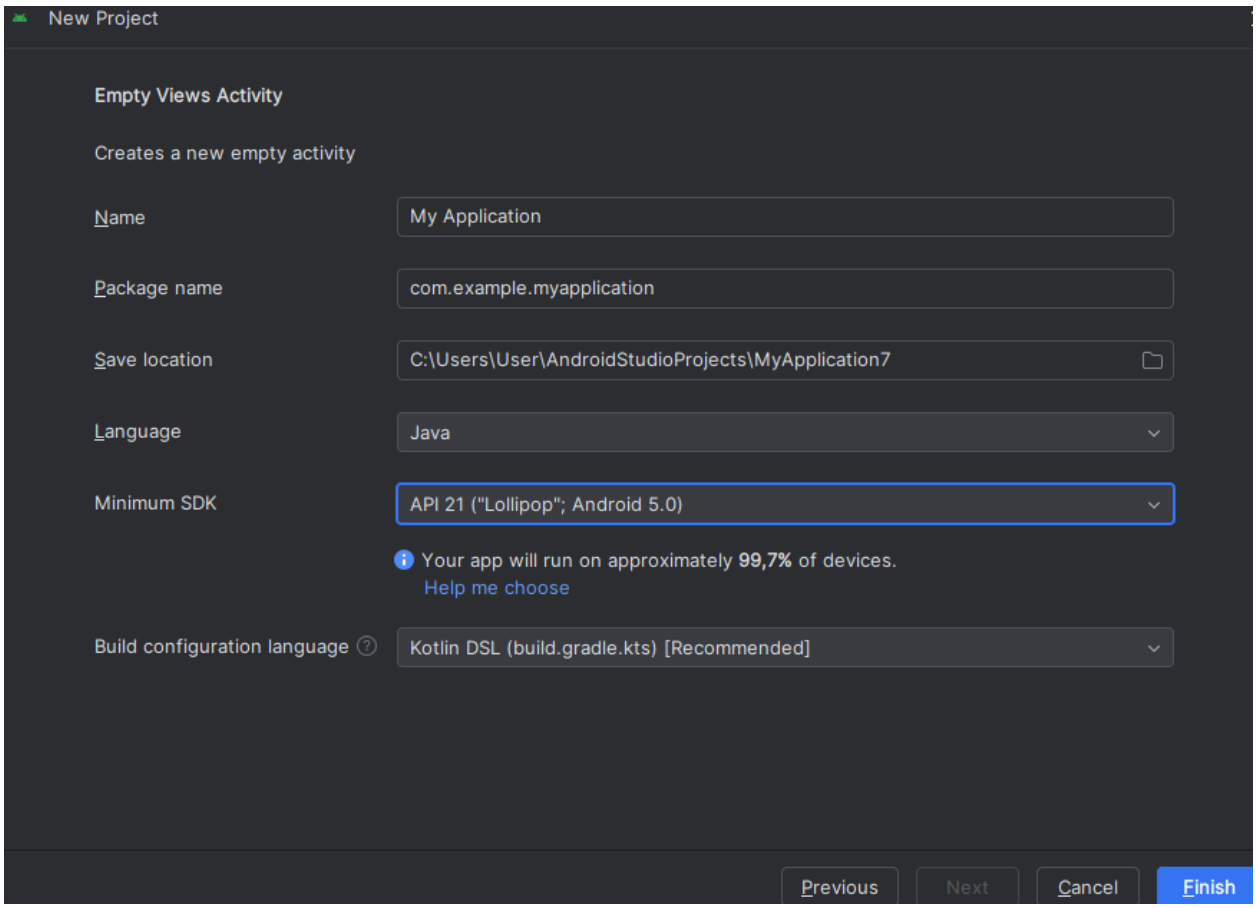


Рисунок – 7 Данные проекта

Класс Recipe на Рисунке 8 является основой для работы с рецептами в приложении от отображения в списках до сохранения в локальную БД.

```java
package com.example.cookhelper;

import androidx.room.Entity;
import androidx.room.Ignore;
import androidx.room.PrimaryKey;

import java.io.Serializable;
import java.util.List;

@Entity
public class Recipe implements Serializable {
    @PrimaryKey(autoGenerate = true)
    public int id;

    public int userId;
    public String title;
    public String description;
    public String imageUrl;

    public String ingredients;
    public String steps;

    @Ignore
    public List<String> stepImages;

    public Recipe(int userId, String title, String description, String imageUrl) {
        this.userId = userId;
        this.title = title;
        this.description = description;
        this.imageUrl = imageUrl;
    }

    public Recipe() {}

    public String getTitle() { return title; }

    public String getDescription() { return description; }

    public String getImageUrl() { return imageUrl; }

    public int getId() { return id; }

    public int getUserId() { return userId; }

    public String getIngredients() { return ingredients; }

    public String getSteps() { return steps; }

    public List<String> getStepImages() { return stepImages; }

    public void setIngredients(String ingredients) { this.ingredients = ingredients; }

    public void setSteps(String steps) { this.steps = steps; }

    public void setStepImages(List<String> stepImages) { this.stepImages = stepImages; }
}
```

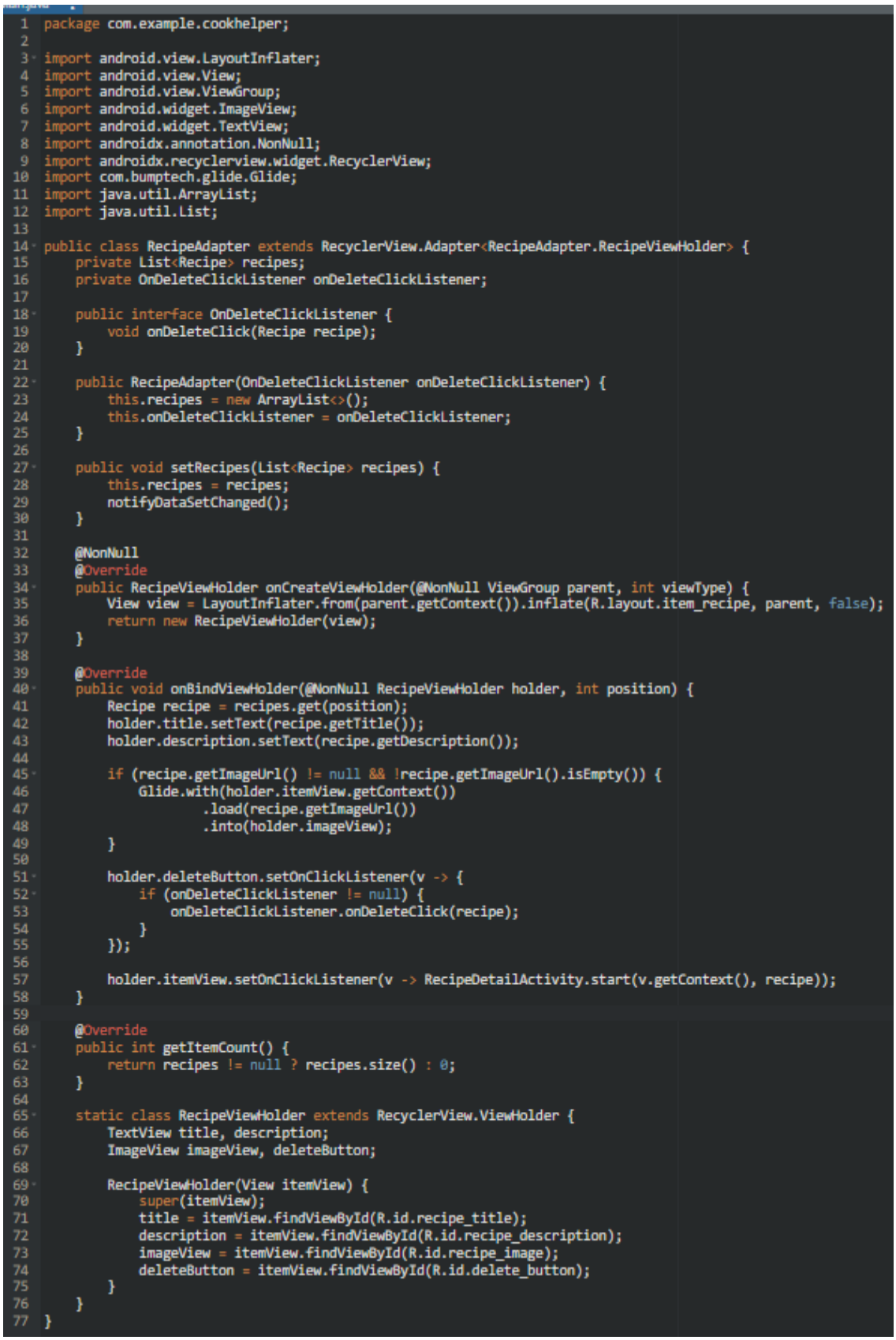
Рисунок – 8 Класс recipe

Класс AppDatabase Рисунке 9 для управления базой данных на устройстве с помощью Room.

```java
package com.example.cookhelper.data;

import android.content.Context;

import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import androidx.room.TypeConverters;

import com.example.cookhelper.Recipe;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

@Database(entities = {User.class, Recipe.class, FavoriteRecipe.class}, version = 1)  22 usages  1 inher
@TypeConverters(Converters.class)
public abstract class AppDatabase extends RoomDatabase {
    public abstract UserDao userDao();  4 usages  1 implementation
    public abstract RecipeDao recipeDao();  1 usage  1 implementation
    public abstract FavoriteRecipeDao favoriteRecipeDao(); // ← добавлено  no usages

    private static volatile AppDatabase INSTANCE;  4 usages
    private static final int NUMBER_OF_THREADS = 4;  1 usage
    public static final ExecutorService databaseWriteExecutor =  3 usages
            Executors.newFixedThreadPool(NUMBER_OF_THREADS);

    public static AppDatabase getDatabase(final Context context) {  5 usages
        if (INSTANCE == null) {
            synchronized (AppDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                                    AppDatabase.class,  name: "cook_helper_db")
                            .build();
                }
            }
        }
        return INSTANCE;
    }
}
```

Рисунок – 9 AppDatabase для управления базой данных.

RecipeAdapter на рисунке 10 отображает список рецептов с заголовком, описанием и изображением.



```java
package com.example.cookhelper;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import com.bumptech.glide.Glide;
import java.util.ArrayList;
import java.util.List;

public class RecipeAdapter extends RecyclerView.Adapter<RecipeAdapter.RecipeViewHolder> {
    private List<Recipe> recipes;
    private OnDeleteClickListener onDeleteClickListener;

    public interface OnDeleteClickListener {
        void onDeleteClick(Recipe recipe);
    }

    public RecipeAdapter(OnDeleteClickListener onDeleteClickListener) {
        this.recipes = new ArrayList<>();
        this.onDeleteClickListener = onDeleteClickListener;
    }

    public void setRecipes(List<Recipe> recipes) {
        this.recipes = recipes;
        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public RecipeViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_recipe, parent, false);
        return new RecipeViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull RecipeViewHolder holder, int position) {
        Recipe recipe = recipes.get(position);
        holder.title.setText(recipe.getTitle());
        holder.description.setText(recipe.getDescription());

        if (recipe.getImageUrl() != null && !recipe.getImageUrl().isEmpty()) {
            Glide.with(holder.itemView.getContext())
                    .load(recipe.getImageUrl())
                    .into(holder.imageView);
        }

        holder.deleteButton.setOnClickListener(v -> {
            if (onDeleteClickListener != null) {
                onDeleteClickListener.onDeleteClick(recipe);
            }
        });

        holder.itemView.setOnClickListener(v -> RecipeDetailActivity.start(v.getContext(), recipe));
    }

    @Override
    public int getItemCount() {
        return recipes != null ? recipes.size() : 0;
    }

    static class RecipeViewHolder extends RecyclerView.ViewHolder {
        TextView title, description;
        ImageView imageView, deleteButton;

        RecipeViewHolder(View itemView) {
            super(itemView);
            title = itemView.findViewById(R.id.recipe_title);
            description = itemView.findViewById(R.id.recipe_description);
            imageView = itemView.findViewById(R.id.recipe_image);
            deleteButton = itemView.findViewById(R.id.delete_button);
        }
    }
}
```

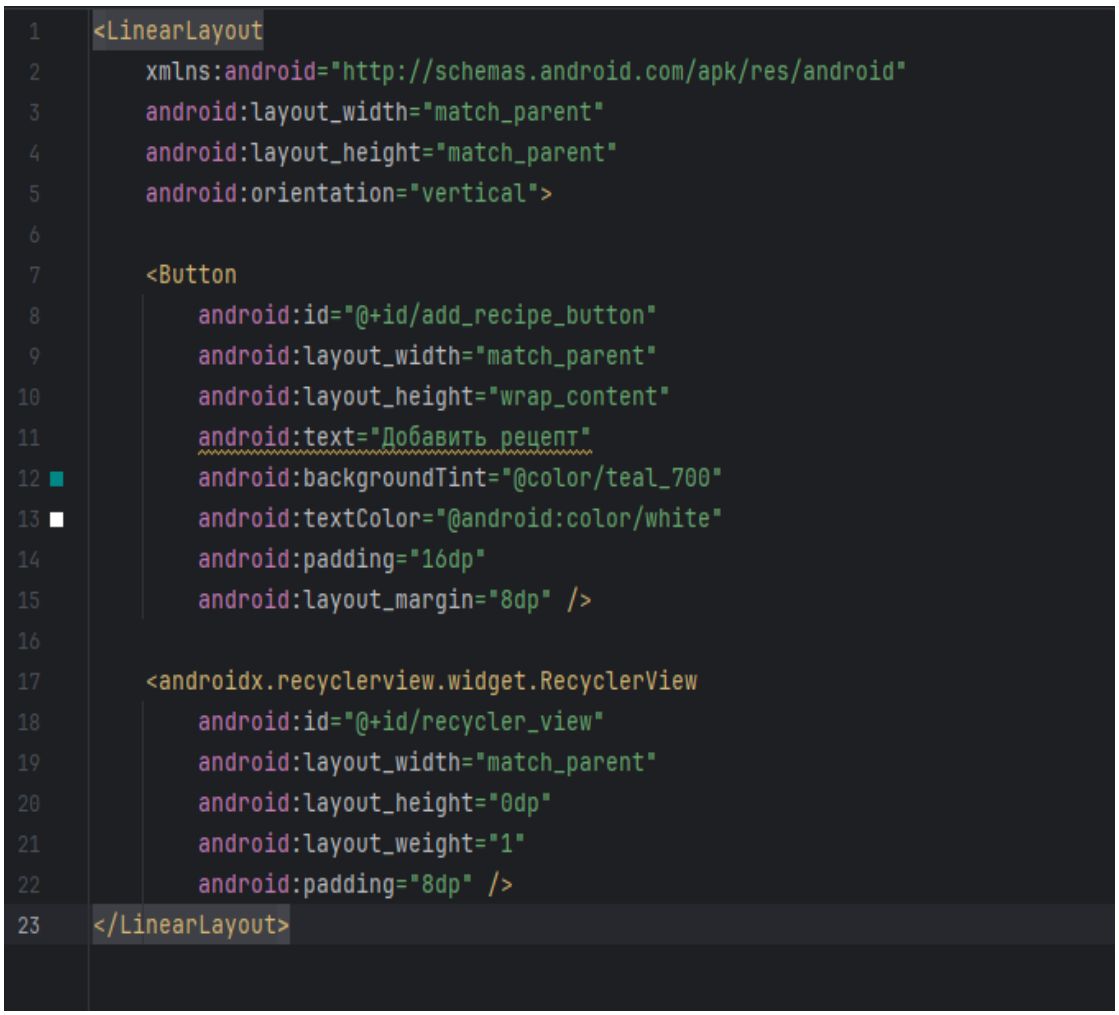
Рисунок – 10 RecipeAdapter

RecipeDetailActivity на рисунке 11 с подробной информацией о выбранном рецепте, включая шаги и изображения приготовления.

```java
import android.content.Intent;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import com.bumptech.glide.Glide;
import java.util.List;

public class RecipeDetailActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recipe_detail);

        Recipe recipe = (Recipe) getIntent().getSerializableExtra("recipe");

        if (recipe != null) {
            TextView titleTextView = findViewById(R.id.detail_title);
            TextView ingredientsTextView = findViewById(R.id.detail_ingredients);
            LinearLayout stepsContainer = findViewById(R.id.steps_container);

            titleTextView.setText(recipe.getTitle());
            ingredientsTextView.setText(recipe.getIngredients());

            String[] steps = recipe.getSteps() != null ? recipe.getSteps().split("\n") : new String[0];
            List<String> stepImages = recipe.getStepImages();

            for (int i = 0; i < steps.length; i++) {
                TextView stepTextView = new TextView(this);
                stepTextView.setText(steps[i]);
                stepTextView.setTextSize(16);
                stepTextView.setPadding(0, 8, 0, 8);
                stepsContainer.addView(stepTextView);

                if (stepImages != null && i < stepImages.size()) {
                    ImageView stepImageView = new ImageView(this);
                    stepImageView.setLayoutParams(new LinearLayout.LayoutParams(
                            LinearLayout.LayoutParams.MATCH_PARENT,
                            400
                    ));
                    stepImageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
                    stepsContainer.addView(stepImageView);

                    Glide.with(this)
                            .load(stepImages.get(i))
                            .into(stepImageView);
                }
            }
        }
    }

    public static void start(android.content.Context context, Recipe recipe) {
        Intent intent = new Intent(context, RecipeDetailActivity.class);
        intent.putExtra("recipe", recipe);
        context.startActivity(intent);
    }
```

Рисунок – 11 RecipeDetailActivity

Макет activity_main.xml на рисунке 12 для отображения списка рецептов.



```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/add_recipe_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Добавить рецепт"
        android:backgroundTint="@color/teal_700"
        android:textColor="@android:color/white"
        android:padding="16dp"
        android:layout_margin="8dp" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:padding="8dp" />
</LinearLayout>
```

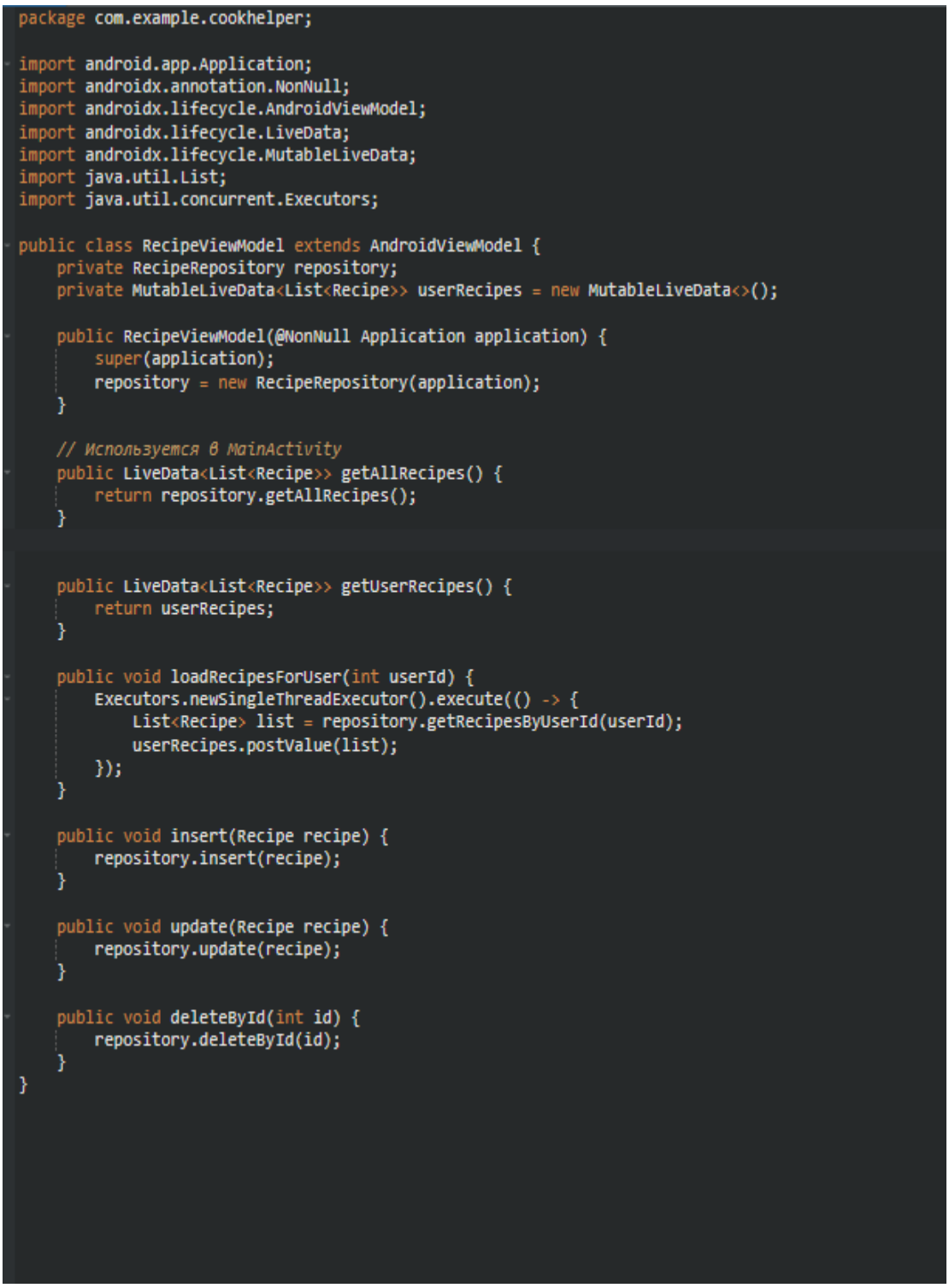
Рисунок – 12 activity_main.xml

Библиотеки и инстурменты для работы приложения Gradle на рисунке 13.

```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
    id("kotlin-kapt")
}

android {
    namespace = "com.example.cookhelper"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.cookhelper"
        minSdk = 24
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(getDefaultProguardFile("proguard-android-optimize.txt"), "proguard-rules.pro")
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
    }

    kotlinOptions {
        jvmTarget = "17"
    }

    buildFeatures {
        viewBinding = true
    }
}

dependencies {
    val roomVersion = "2.6.1"

    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation("com.google.code.gson:gson:2.10.1")
    // Room
    implementation("androidx.room:room-runtime:$roomVersion")
    kapt("androidx.room:room-compiler:$roomVersion")
    implementation("androidx.room:room-ktx:$roomVersion")

    // Glide
    implementation("com.github.bumptech.glide:glide:4.16.0")
    kapt("com.github.bumptech.glide:compiler:4.16.0")

    // Testing
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

Рисунок – 13 build.gradle.kts

RecipeViewModel на рисунке 14 хранит рецепты и обеспечивает обновления в пользовательский интерфейс.



```
package com.example.cookhelper;

import android.app.Application;
import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import java.util.List;
import java.util.concurrent.Executors;

public class RecipeViewModel extends AndroidViewModel {
    private RecipeRepository repository;
    private MutableLiveData<List<Recipe>> userRecipes = new MutableLiveData<>();

    public RecipeViewModel(@NonNull Application application) {
        super(application);
        repository = new RecipeRepository(application);
    }

    // Используется в MainActivity
    public LiveData<List<Recipe>> getAllRecipes() {
        return repository.getAllRecipes();
    }


    public LiveData<List<Recipe>> getUserRecipes() {
        return userRecipes;
    }

    public void loadRecipesForUser(int userId) {
        Executors.newSingleThreadExecutor().execute(() -> {
            List<Recipe> list = repository.getRecipesByUserId(userId);
            userRecipes.postValue(list);
        });
    }

    public void insert(Recipe recipe) {
        repository.insert(recipe);
    }

    public void update(Recipe recipe) {
        repository.update(recipe);
    }

    public void deleteById(int id) {
        repository.deleteById(id);
    }
}
```

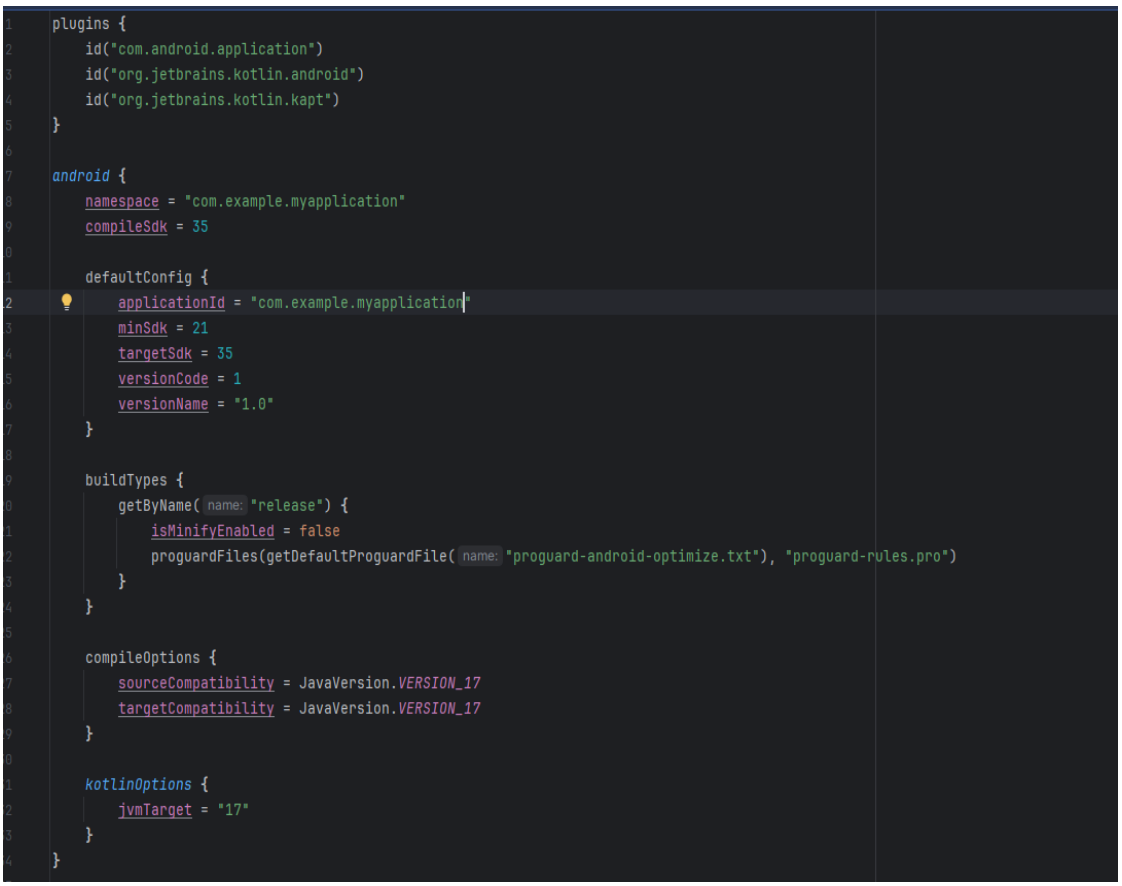
Рисунок – 14 RecipeViewModel

AddRecipeActivity на рисунке 15 экран создания нового рецепта с возможностью выбора изображения из галереи и сохранением всех данных .

```java
package com.example.cookhelper;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class AddRecipeActivity extends AppCompatActivity {
    private EditText titleEditText, descriptionEditText, imageUrlEditText;
    private RecipeRepository repository;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_recipe);

        titleEditText = findViewById(R.id.titleEditText);
        descriptionEditText = findViewById(R.id.descriptionEditText);
        imageUrlEditText = findViewById(R.id.imageUrlEditText);
        Button saveButton = findViewById(R.id.saveButton);

        repository = new RecipeRepository(getApplication());

        saveButton.setOnClickListener(v -> {
            String title = titleEditText.getText().toString().trim();
            String description = descriptionEditText.getText().toString().trim();
            String imageUrl = imageUrlEditText.getText().toString().trim();

            if (title.isEmpty() || description.isEmpty()) {
                Toast.makeText(this, "Заполните все поля", Toast.LENGTH_SHORT).show();
                return;
            }

            SharedPreferences prefs = getSharedPreferences("user", MODE_PRIVATE);
            int userId = prefs.getInt("currentUserId", -1);
            if (userId == -1) {
                Toast.makeText(this, "Ошибка пользователя", Toast.LENGTH_SHORT).show();
                return;
            }

            Recipe recipe = new Recipe(userId, title, description, imageUrl);
            repository.insert(recipe);
            finish();
        });
    }
}
```

Рисунок – 15 AddRecipeActivity

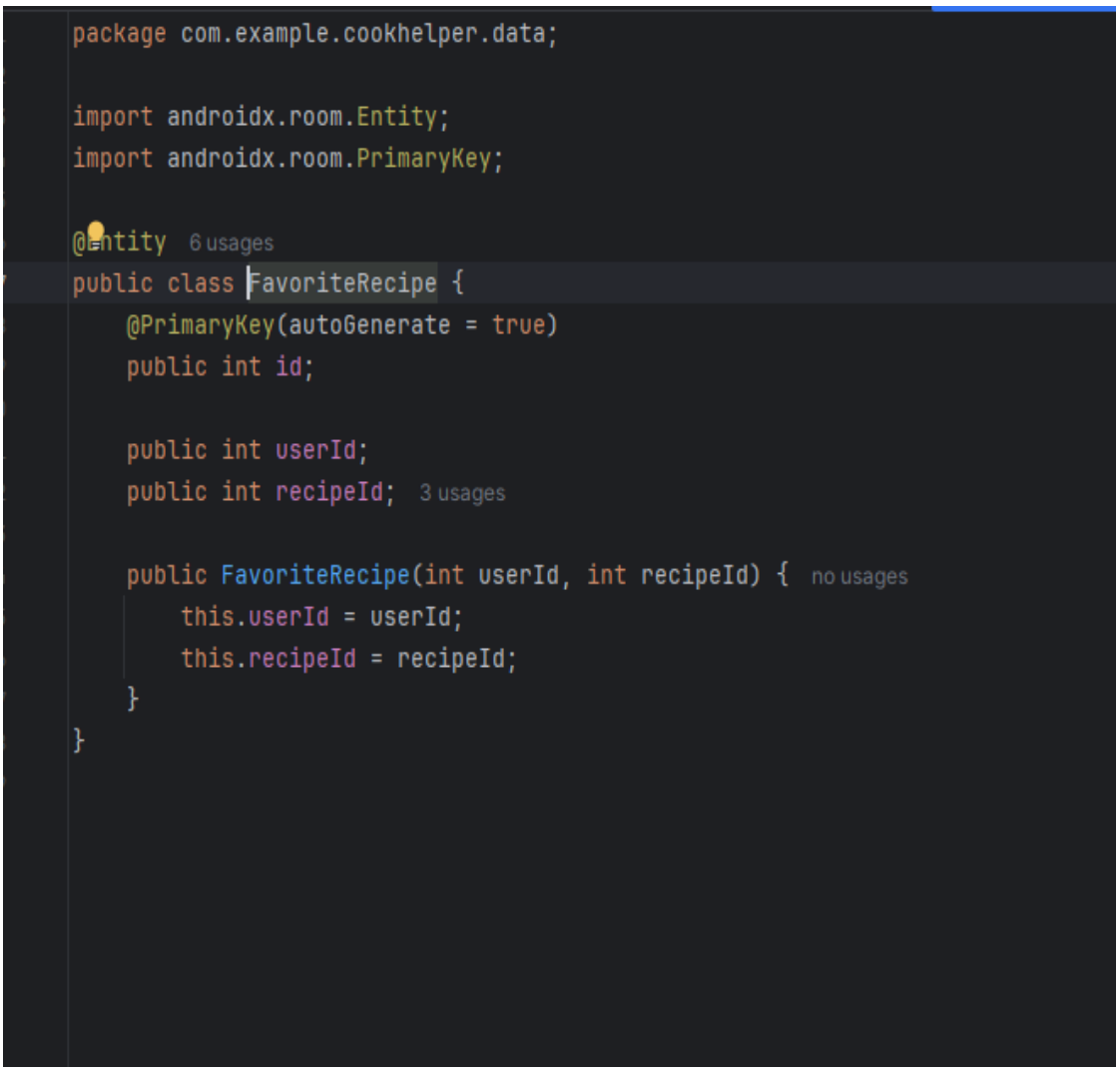
Исправление ошибок связанных с gradle app на рисунке 16.



```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
    id("org.jetbrains.kotlin.kapt")
}

android {
    namespace = "com.example.myapplication"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.myapplication"
        minSdk = 21
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"
    }

    buildTypes {
        getByName( name: "release") {
            isMinifyEnabled = false
            proguardFiles(getDefaultProguardFile( name: "proguard-android-optimize.txt"), "proguard-rules.pro")
        }
    }

    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_17
        targetCompatibility = JavaVersion.VERSION_17
    }

    kotlinOptions {
        jvmTarget = "17"
    }
}
```

Рисунок – 16 build.gradle.kts(app)

FavoriteRecipe на рисунке 17 для таблицы избранных рецептов, связывает ID пользователя и ID рецепта.



```java
package com.example.cookhelper.data;

import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity 6 usages
public class FavoriteRecipe {
    @PrimaryKey(autoGenerate = true)
    public int id;

    public int userId;
    public int recipeId; 3 usages

    public FavoriteRecipe(int userId, int recipeId) { no usages
        this.userId = userId;
        this.recipeId = recipeId;
    }
}
```

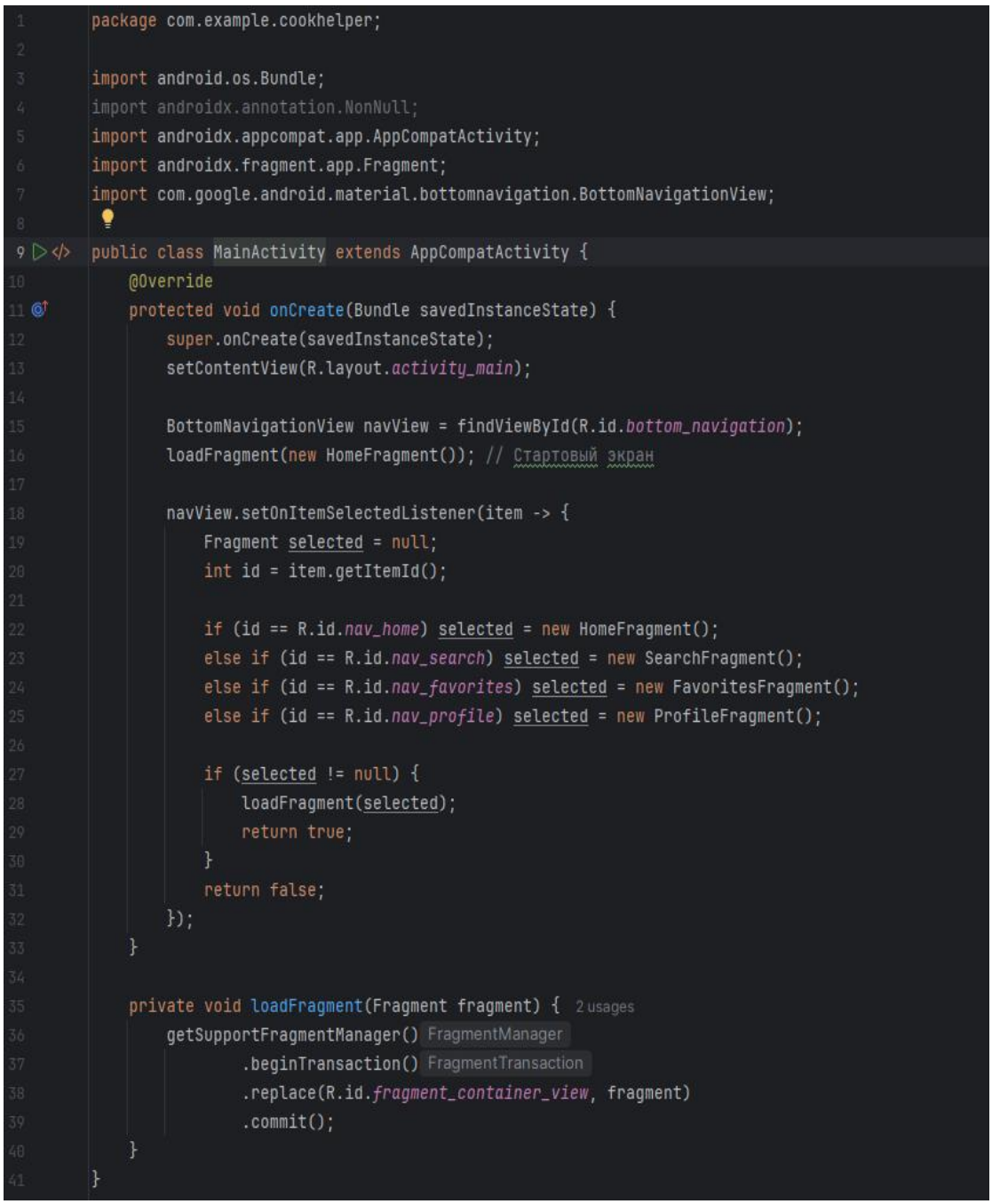
Рисунок – 17 FavoriteRecipe

Converters на рисунке 18 вспомогательный класс TypeConverter, преобразующий список изображений шагов в строку и обратно для хранения в базе.

```
package com.example.cookhelper.data;

import androidx.room.TypeConverter;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Converters {  1 usage
    @TypeConverter  no usages
    public static String fromList(List<String> list) {
        return list != null ? String.join( delimiter: ",", list) : "";
    }

    @TypeConverter  no usages
    public static List<String> toList(String data) {
        return data != null && !data.isEmpty() ? Arrays.asList(data.split( regex: ",")) : Collections.emptyList();
    }
}
```

Рисунок – 18 Converters

MainActivity на рисунке 19 основа с навигацией между главными разделами



```java
package com.example.cookhelper;

import android.os.Bundle;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import com.google.android.material.bottomnavigation.BottomNavigationView;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        BottomNavigationView navView = findViewById(R.id.bottom_navigation);
        loadFragment(new HomeFragment()); // Стартовый экран

        navView.setOnItemSelectedListener(item -> {
            Fragment selected = null;
            int id = item.getItemId();

            if (id == R.id.nav_home) selected = new HomeFragment();
            else if (id == R.id.nav_search) selected = new SearchFragment();
            else if (id == R.id.nav_favorites) selected = new FavoritesFragment();
            else if (id == R.id.nav_profile) selected = new ProfileFragment();

            if (selected != null) {
                loadFragment(selected);
                return true;
            }
            return false;
        });
    }

    private void loadFragment(Fragment fragment) { 2 usages
        getSupportFragmentManager() FragmentManager
                .beginTransaction() FragmentTransaction
                .replace(R.id.fragment_container_view, fragment)
                .commit();
    }
}
```

Рисунок – 19 MainActivity

User модель пользователя с полями email, password и nickname, на рисунке 20, используемая для аутентификации и отображения данных о пользователе.

```java
package com.example.cookhelper.data;

import androidx.room.Entity;
import androidx.room.Ignore;
import androidx.room.PrimaryKey;

@Entity 31 usages
public class User {
    @PrimaryKey(autoGenerate = true)
    public int id;

    public String email; 11 usages
    public String password; 10 usages
    public String nickname; // 👈 новое поле 2 usages

    @Ignore no usages
    public User(String email, String password, String nickname) {
        this.email = email;
        this.password = password;
        this.nickname = nickname;
    }

    @Ignore 1 usage
    public User(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public User() {} 3 usages
}
```

Рисунок – 20 User

RegisterActivity на рисунке 21 регистрация нового пользователя с проверкой на уникальность.

```java
package com.example.cookhelper;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cookhelper.data.AppDatabase;
import com.example.cookhelper.data.User;
import com.example.cookhelper.data.UserDao;

public class RegisterActivity extends AppCompatActivity {
    private EditText emailEditText, passwordEditText;
    private UserDao userDao;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        emailEditText = findViewById(R.id.emailEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        Button registerButton = findViewById(R.id.registerButton);

        AppDatabase db = AppDatabase.getDatabase(this);
        userDao = db.userDao();

        registerButton.setOnClickListener(v -> {
            String email = emailEditText.getText().toString();
            String password = passwordEditText.getText().toString();

            new Thread(() -> {
                User existingUser = userDao.getUserByEmail(email);
                if (existingUser != null) {
                    runOnUiThread(() -> Toast.makeText(this, "Пользователь уже существует", Toast.LENGTH_SHORT).show());
                    return;
                }

                User newUser = new User();
                newUser.email = email;
                newUser.password = password;
                long id = userDao.insert(newUser);

                runOnUiThread(() -> {
                    SharedPreferences.Editor editor = getSharedPreferences("user", MODE_PRIVATE).edit();
                    editor.putInt("currentUserId", (int) id);
                    editor.apply();

                    Toast.makeText(this, "Успешная регистрация", Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(this, MainActivity.class));
                    finish();
                });
            }).start();
        });
    }
}
```

Рисунок – 21 RegisterActivity

LoginActivity на рисунке 22 вход в приложение, проверяет данные пользователя и сохраняет текущий ID пользователя.

```java
package com.example.cookhelper;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.example.cookhelper.data.AppDatabase;
import com.example.cookhelper.data.User;
import com.example.cookhelper.data.UserDao;

public class LoginActivity extends AppCompatActivity {
    private EditText emailEditText, passwordEditText;
    private UserDao userDao;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        emailEditText = findViewById(R.id.emailEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        Button loginButton = findViewById(R.id.loginButton);
        Button registerButton = findViewById(R.id.registerButton);

        AppDatabase db = AppDatabase.getDatabase(this);
        userDao = db.userDao();

        loginButton.setOnClickListener(v -> {
            String email = emailEditText.getText().toString();
            String password = passwordEditText.getText().toString();

            new Thread(() -> {
                User user = userDao.getUserByEmailAndPassword(email, password);
                runOnUiThread(() -> {
                    if (user != null) {
                        SharedPreferences.Editor editor = getSharedPreferences("user", MODE_PRIVATE).edit();
                        editor.putInt("currentUserId", user.id);
                        editor.apply();

                        Toast.makeText(this, "Успешный вход", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent(this, MainActivity.class));
                        finish();
                    } else {
                        Toast.makeText(this, "Неверные данные", Toast.LENGTH_SHORT).show();
                    }
                });
            }).start();
        });

        registerButton.setOnClickListener(v -> startActivity(new Intent(this, RegisterActivity.class)));
    }
}
```

Рисунок – 22 LoginActivity

3.3 Тестирование приложения

Для подтверждения работоспособности основных функций разработанного мобильного приложения "Кулинарный Помощник" было проведено функциональное тестирование. Основная цель – убедиться, что приложение ведет себя так, как было запланировано в техническом задании,

и все его ключевые возможности работают корректно. Тестирование выполнялось вручную на эмуляторе Android и на реальном смартфоне (модель Samsung S6 Lite, Android 13). Фокус тестирования был направлен на проверку следующих основных блоков приложения, критически важных для пользователя:

- Работа с рецептами, проверялось, насколько удобно и правильно пользователь может создать рецепт и также внимание уделялось заполнению всех обязательных полей, загрузке фотографии блюда из галереи устройства и сохранению рецепта;
- Работа с избранным, проверялось добавление любого рецепта в «Избранное», корректное отображение списка избранных рецептов в соответствующем разделе и сохранение списка "Избранного" даже после закрытия и повторного открытия приложения;
- Авторизация и регистрация пользователей, проверялся процесс создания нового аккаунта: ввод email и пароля, успешное создание аккаунта и автоматический вход после него, также вход по существующему логину и паролю, в том числе с неверными данными и сохранение сессии;
- Навигация по приложению, проверялась работа нижнего меню навигации, также скорость и плавность переходов и стабильность приложения при частых сменах экранов.

Результаты функционального тестирования приведены в таблице 4.

Таблица 4- Результаты тестирования

Действие	Ожидаемый результат	Действительный результат
Добавление рецепта	Рецепт сохраняется, все данные видны правильно, картинка загружается и отображается.	Соответствует
Избранное	Рецепт добавляется в «Избранное», и удаляется из «Избранное», появляется в списке	Соответствует
Регистрация нового пользователя	Можно создать аккаунт с email и паролем, после регистрации пользователь сразу попадает в приложение.	Соответствует
Навигация	Переходы между экранами работают быстро и без ошибок.	Соответствует

По итогам проведенного функционального тестирования можно сделать вывод, что основное мобильное приложение "Кулинарный Помощник" реализовано корректно.

На рисунке 23 - 32 видно, что приложение работает и реализовывает описанные функции.

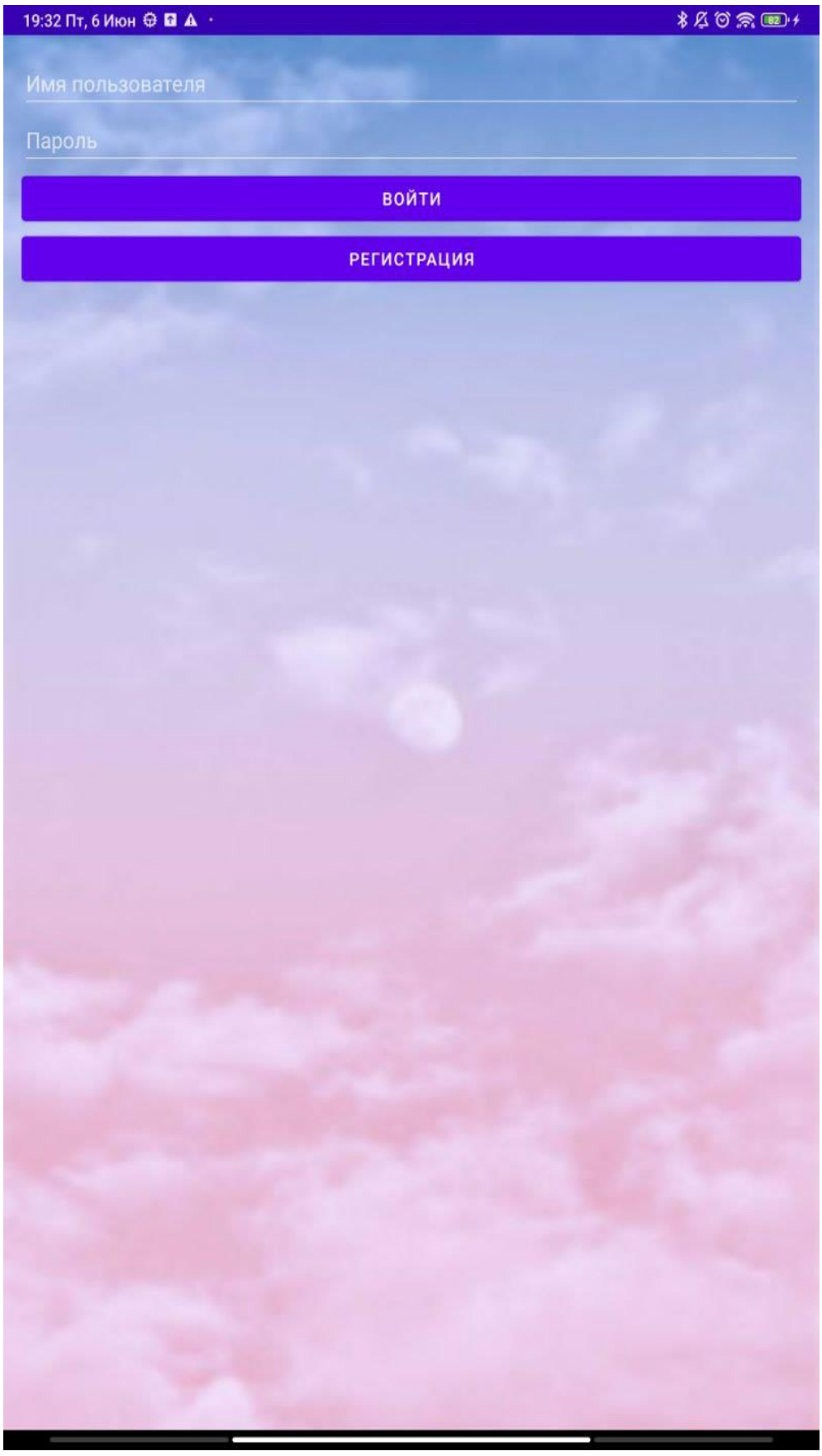


Рисунок – 23 Вход в приложение

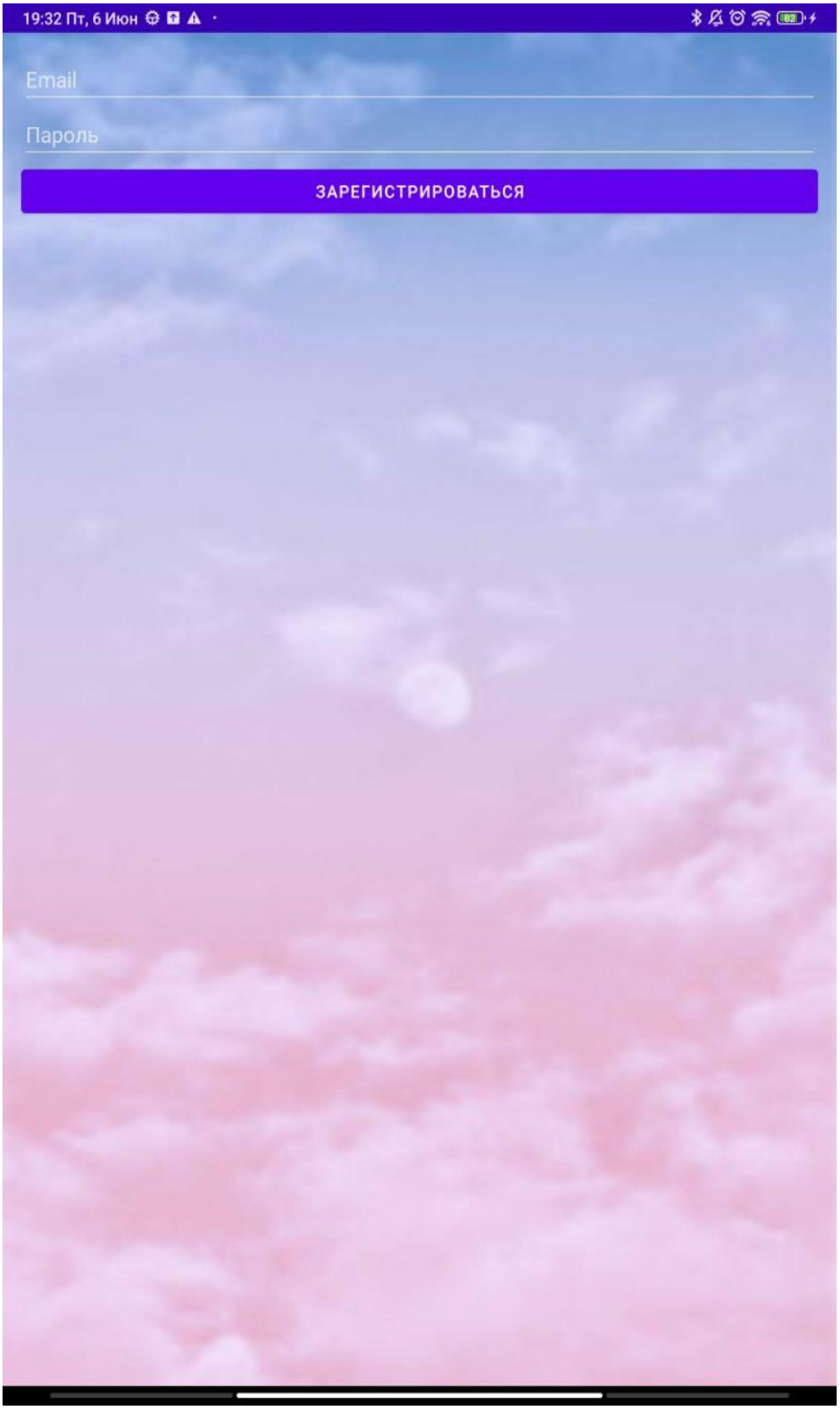


Рисунок – 24 Регистрация в приложении

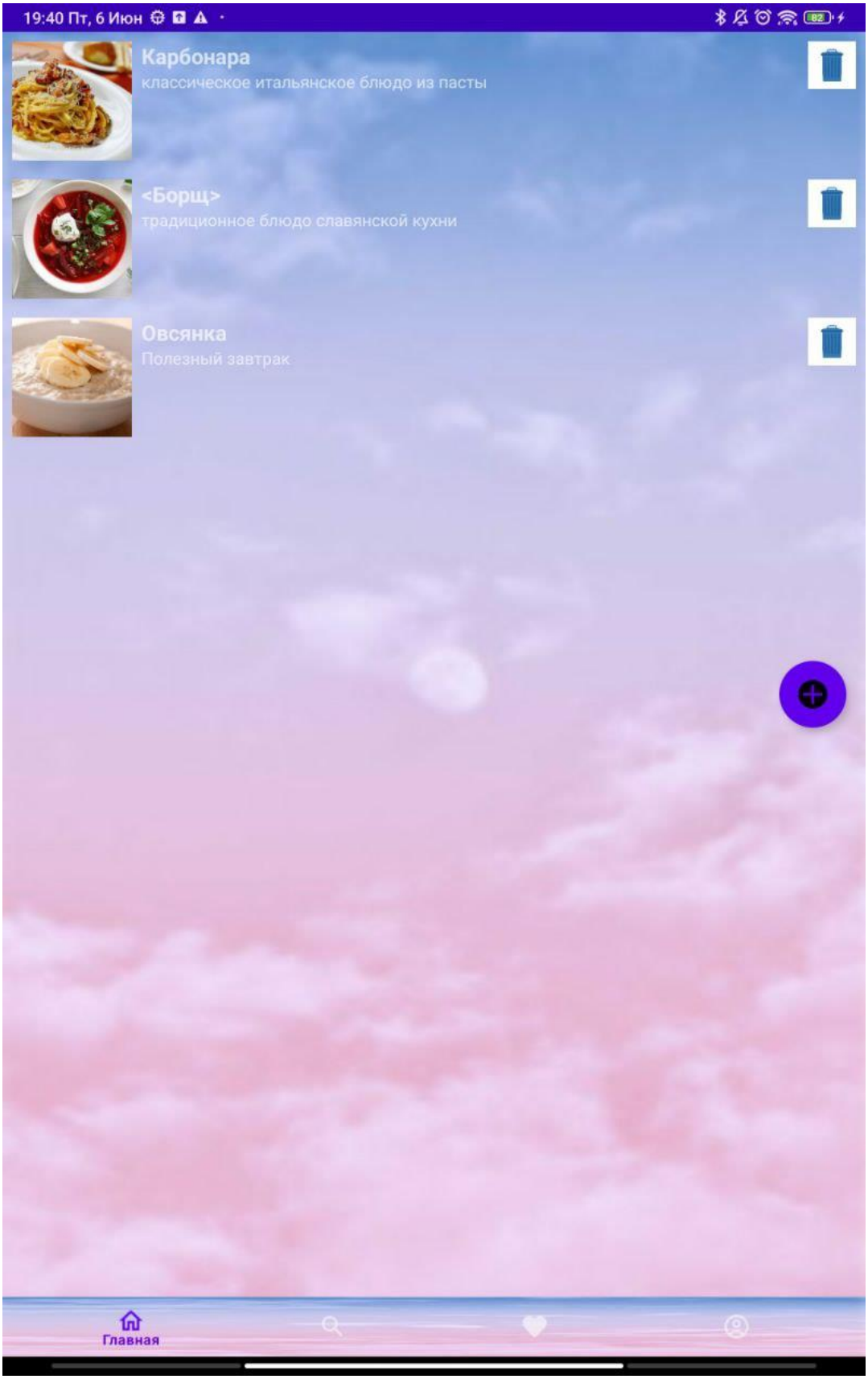


Рисунок – 25 Внешний вид приложения

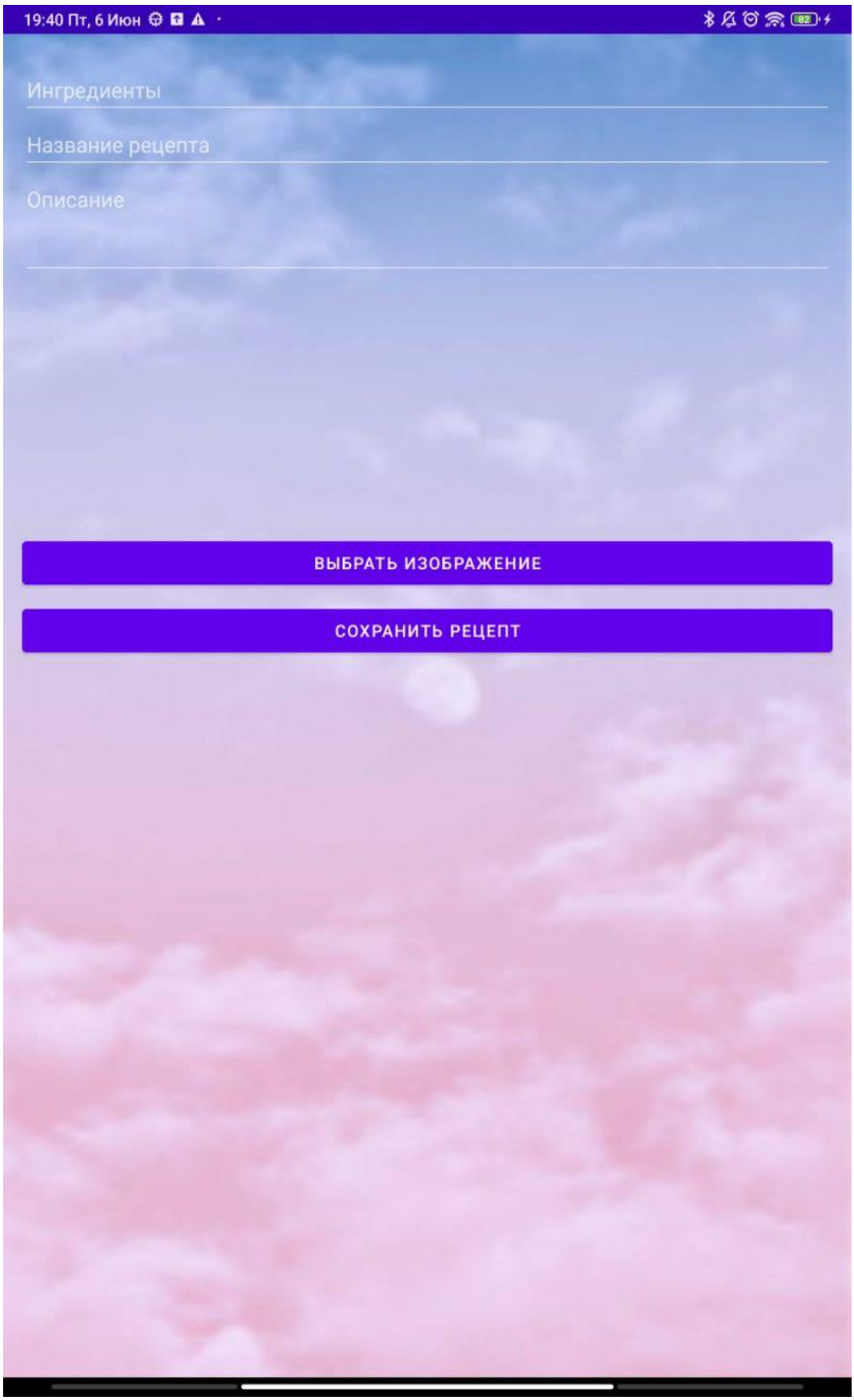


Рисунок – 26 Добавление рецептов

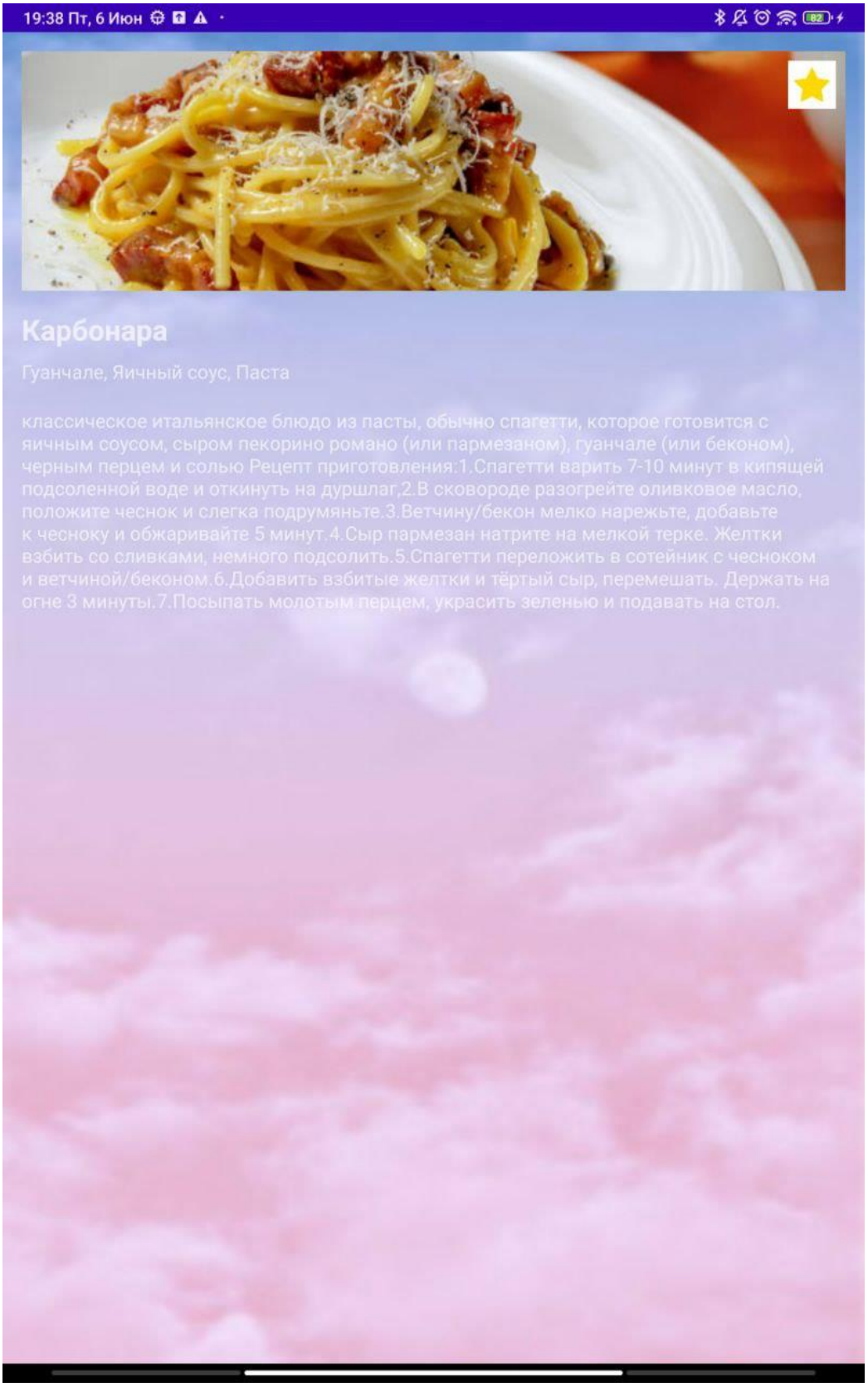


Рисунок – 27 Рецепт внутри приложения

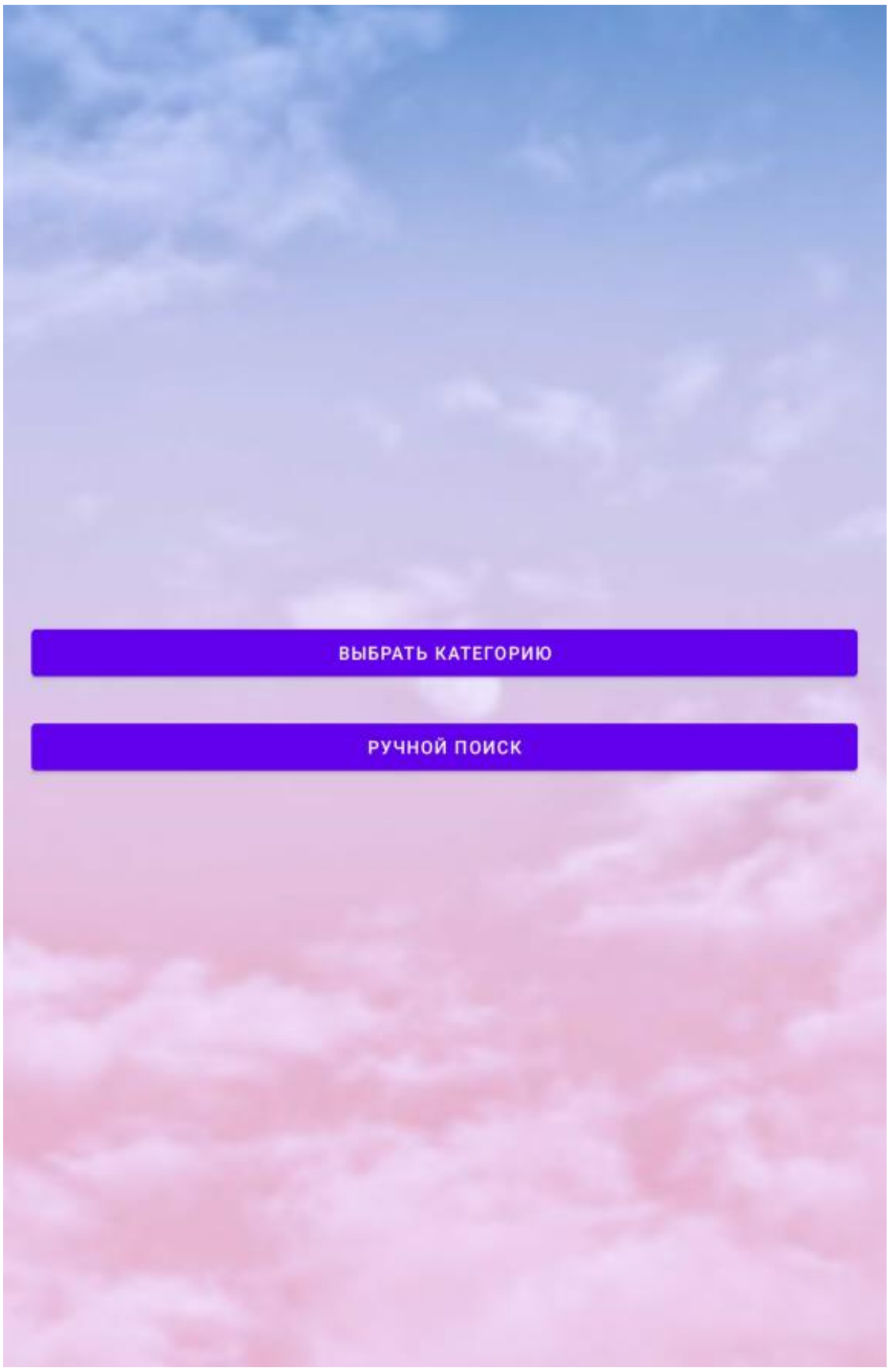


Рисунок – 28 Выбор поиска

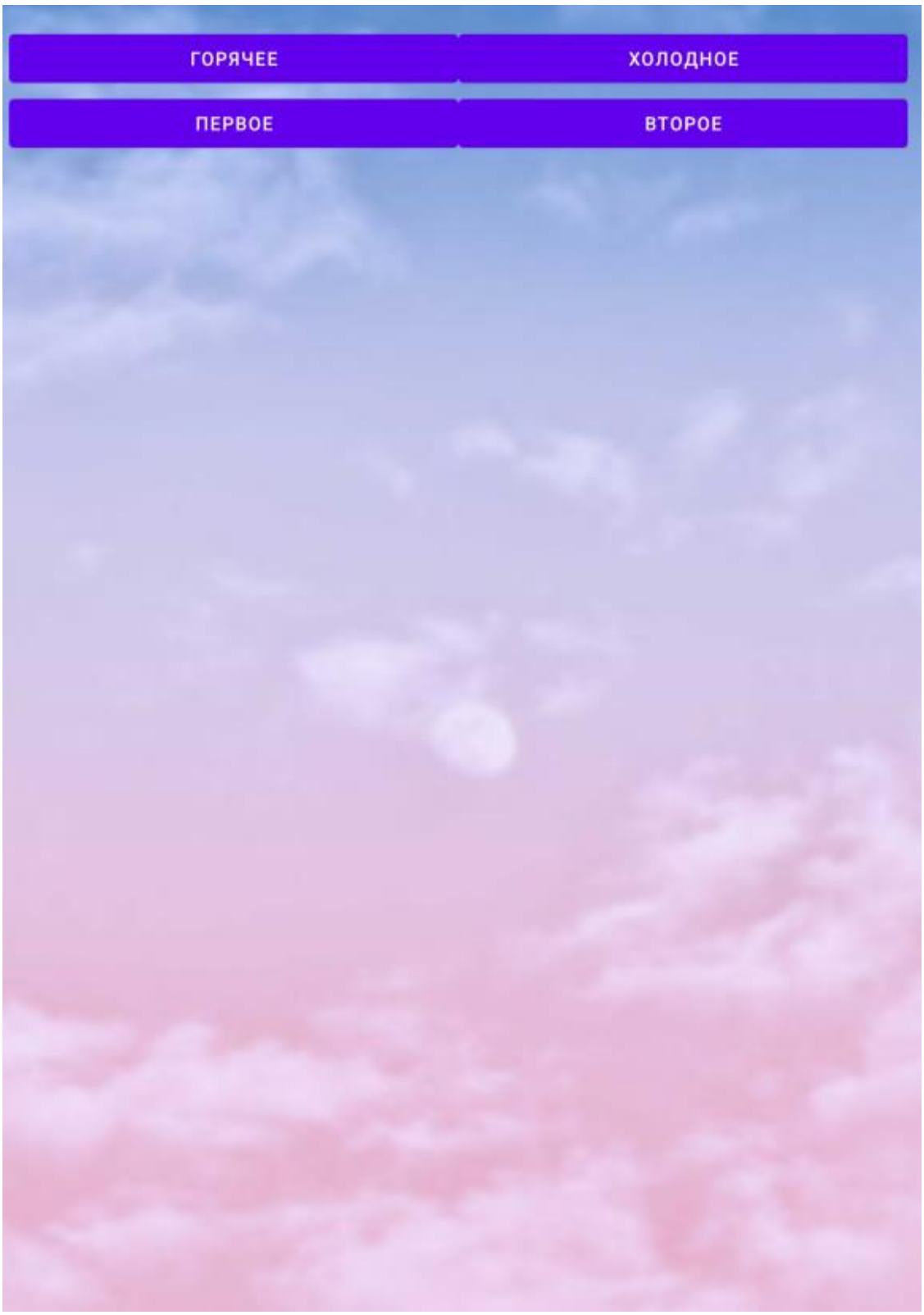


Рисунок – 29 Поиск по критериям

Рисунок – 30 Ручной поиск

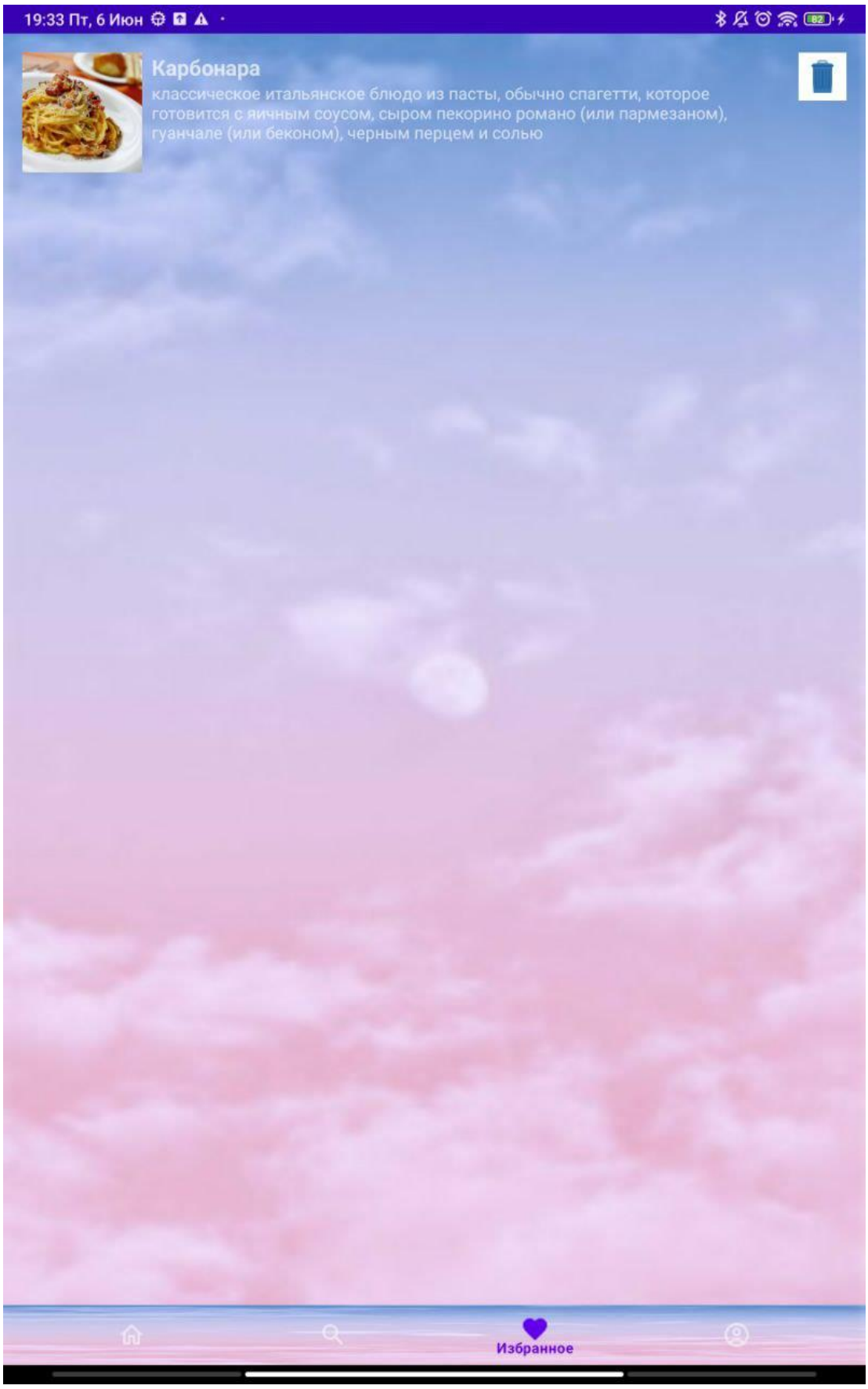


Рисунок – 31 Избранное

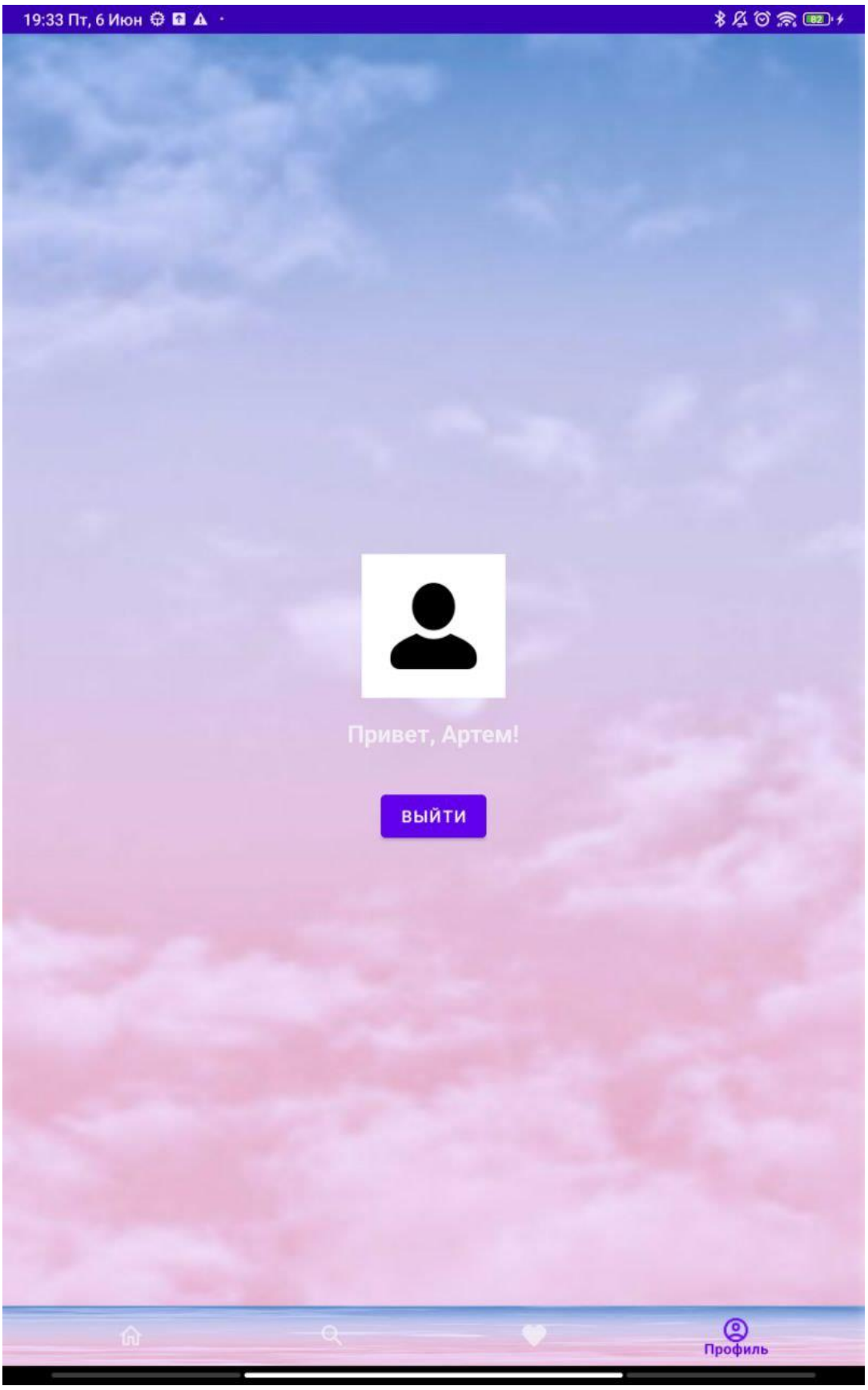


Рисунок – 32 Профиль

После завершения разработки всех основных функций мобильного приложения "Кулинарный Помощник" и успешного прохождения функционального тестирования, была выполнена финальная сборка проекта. Целью этого этапа было создание готового к установке APK-файла. Процесс компиляции исходного кода представлен на Рисунке 33.

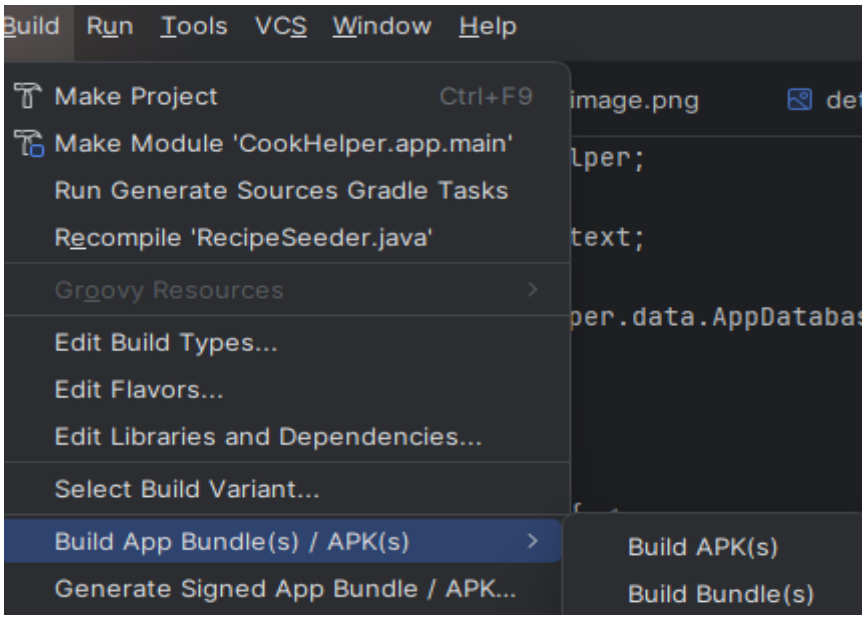


Рисунок – 33 Компиляция APK

Выводы по третьей главе

В результате работы, проведенной в третьей главе данной исследовательской работы, было создано рабочее мобильное приложение, разработанное с использованием Android Studio и языка Java. В процессе его создания реализованы ключевые функции: экраны авторизации и регистрации, добавление рецептов (включая загрузку изображений из галереи), управление списком "Избранное", поиск рецептов и навигация между основными разделами. Завершением разработки стала успешная сборка APK-файла, который был установлен и проверен на реальных устройствах (Samsung S6 lite), что подтвердило готовность приложения к использованию.

Заключение

В рамках данной выпускной квалификационной работы было разработано мобильное приложение «Кулинарный помощник», основная задача которого обеспечить пользователям простой и удобный способ поиска, хранения и создания рецептов. Проект ориентирован на то, чтобы предложить решение, способное работать офлайн, не перегружать устройство и в то же время предоставлять весь необходимый функционал для любителей домашней кулинарии.

На первом этапе работы был проведён анализ предметной области, изучены предпочтения пользователей в сфере кулинарных приложений, а также рассмотрены технологии, подходящие для реализации такого продукта. В ходе подготовки были выбраны инструменты проектирования (включая UML-диаграммы), что позволило чётко обозначить структуру будущего приложения и определить его ключевые компоненты.

В качестве основной среды разработки использовалась Android Studio, язык программирования Java. Для хранения рецептов без необходимости постоянного подключения к сети применена база данных Room, которая обеспечивает быструю работу и надёжное сохранение информации. Пользовательский интерфейс был создан с учётом современных требований к дизайну мобильных приложений с использованием компонентов RecyclerView, ViewModel, а также встроенных элементов Android SDK для диалогов, навигации и работы с данными.

Особое внимание при реализации было уделено удобству взаимодействия с рецептурной базой, пользователь может не только просматривать и искать блюда по ингредиентам, но и сохранять избранное, добавлять собственные рецепты и управлять ими. Логика приложения построена по архитектурному принципу разделения ответственности, что упростило как реализацию, так и последующую модификацию кода.

В процессе тестирования были проверены основные функции: навигация, сохранение данных, корректность отображения рецептов, устойчивость при смене ориентации экрана и перезапуске приложения. По результатам проверок все модули работали стабильно, а интерфейс оказался понятным даже для неподготовленного пользователя.

Таким образом, разработанное приложение в полной мере соответствует поставленным задачам и может быть использовано как готовый продукт. В дальнейшем возможно его расширение: добавление поддержки облачной синхронизации, улучшенный поиск по категориям, встроенный календарь для планирования питания, а также интеграция с внешними источниками данных например, с API магазинов или системами подсчёта калорий.

Тем не менее, уже в текущей версии «Кулинарный помощник» способен решать повседневные задачи пользователей, облегчая процесс хранения и подбора рецептов и делая кулинарию ещё доступнее.

Список используемой литературы и используемых источников

1. Android Studio User Guide. – [Электронный ресурс].

2. Brown, L. Non-Functional Requirements in Mobile Applications / L. Brown // Journal of Systems and Software. – 2022. – Vol. 185. – P. 111-125.

3. Брукс, Ф. Мифический человеко-месяц / Ф. Брукс. – М.: Символ-Плюс, 2021. – 304 с.

4. Власов С.Л. Информационные технологии: основы и практика. – М.: Форум, 2020. – 432 с.

5. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2022. – 366 с.

6. ГОСТ Р 57724-2017. Требования к функциональным характеристикам программного обеспечения.

7. ГОСТ Р 57725-2017. Требования к нефункциональным характеристикам ПО.

8. ГОСТ Р ИСО/МЭК 12207-2020. Процессы жизненного цикла программного обеспечения.

9. Google. Material Design Guidelines / Google, 2023. – [Электронный ресурс].

10. IDEF0. Functional Modeling Method / FIPS PUB 183. – 1993.

11. Иванов, А.А. CASE-средства в разработке программного обеспечения / А.А. Иванов // Программная инженерия. – 2022. – № 4. – С. 45-52.

12. Иванов, А.А. CASE-средства в разработке программного обеспечения / А.А. Иванов // Программная инженерия. – 2022. – № 4. – С.52.

13. Куликов С. С., Данилова Г. В., Смолякова О. Г., Меженная М. М. Тестирование программного обеспечения - 2019 – 277с.

14. Леффингуэлл, Д. Agile-требования: гибкое управление проектами / Д. Леффингуэлл. – М.: Вильямс, 2022. – 528 с.

15. Макконнелл, С. Совершенный код / С. Макконнелл. – М.: Русская редакция, 2021. – 896 с.

16. Martin, R.C. Clean Code: A Handbook of Agile Software Craftsmanship / R.C. Martin. – Pearson, 2022. – 464 p.

17. Официальная документация Android Developers. – [Электронный ресурс].

18. Оськин В.В. Сетевые технологии и администрирование: учебное пособие. – М.: Академия, 2021. – 256 с.

19. Петров, В.С. Методологии проектирования баз данных / В.С. Петров // Информационные технологии. – 2023. – № 5. – С. 34-41.

20. Роберт Мартин. Чистая архитектура / Р. Мартин. – М.: Питер, 2021. – 352 с.

21. Room Persistence Library Guide. – [Электронный ресурс].

22. Smith, J. Functional Requirements in Agile Development / J. Smith // IEEE Software. – 2021. – Vol. 38, No. 3. – P. 78-85.

23. Фаулер, М. Рефакторинг. Улучшение существующего кода / М. Фаулер. – СПб.: Питер, 2023. – 448 с.

24. Цуканова О.А. Методология и инструментарий моделирования бизнес-процессов: Учебное пособие. – Университет ИТМО, 2015. – 101 с.

25. Чесноков Д.А. Тестирование и отладка программного обеспечения. – СПб.: Питер, 2022. – 368 с.