

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

<u>Факультет</u>	<u>Кафедра</u>
<u>вычислительной техники</u>	<u>МОиПЭВМ</u>

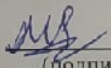
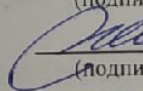
Направление
подготовки
Профиль

09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное электронное приложение
«кулинарная книга»

Студент	 (подпись, дата)	<u>Меркурьев Дмитрий Сергеевич</u> (ФИО полностью)
Руководитель	 (подпись, дата)	<u>Шашков Б. Д.</u> (фамилия, инициалы)
Нормоконтролёр	 (подпись, дата) 22.06.21	<u>Такташкин Д.В.</u> (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____
(подпись) Макарычев П.П.
(фамилия, инициалы)

Работа защищена с отметкой 3 (протокол заседания ГЭК от 29.06.21 № 15)

Секретарь ГЭК _____
(подпись) Попова Н.А.
(фамилия, инициалы)

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ П.П. Макарычев
«__» _____ 2021 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Меркурьев Дмитрий Сергеевич гр. 17ВИ1 факультета ФВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Мобильное электронное приложение «Кулинарная книга»

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., профессор кафедры «МО и ПЭВМ» Шашков Б.Д.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: Электронное мобильное кулинарное приложение предназначено для упрощения процесса готовки продуктов посредством рецептов.

2) Среда выполнения: Операционная система Android выше версии 5.6.

Средства разработки: языки программирования Java, SQLite, JUnit.

3) Основные функции электронного мобильного приложения «кулинарная книга»:

а) Авторизация и аутентификация пользователя;

б) Создание рецепта с заполнением соответствующих полей (фото, описание);

в) Просмотр рецепта

г) Удаление рецепта

5. Перечень подлежащих разработке вопросов

Анализ предметной области

Анализ требований на разработку

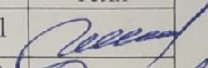
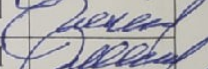
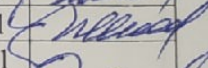
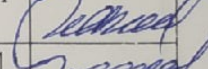
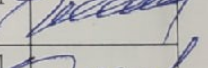
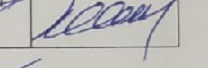
Проектирование приложения

Реализация приложения

Тестирование и контроль качества кода приложения

Планирование разработки и оценка бюджета приложения
Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	5%	20.04.2021 - 23.04.2021	
2.	Анализ требований на разработку	5%	24.04.2021 - 29.04.2021	
3.	Проектирование приложения	25%	30.04.2021 - 11.05.2021	
4.	Реализация приложения	35%	12.05.2021 - 27.05.2021	
5.	Тестирование и контроль качества кода приложения	10%	28.05.2021 - 01.06.2021	
6.	Планирование разработки и оценка бюджета приложения	10%	02.06.2021 - 04.06.2021	
7.	Оформление пояснительной записки	10%	05.06.2021 - 09.06.2021	

Дата выдачи задания "___" _____ 20___

Руководитель бакалаврской работы
(подпись, дата)

Задание к исполнению принял студент
(подпись, дата)

Работу к защите допустить
Декан факультета
(подпись, дата)

 Л.Р. Рогов
29.06.2021

Реферат

Пояснительная записка содержит 124 листа, 21 рисунок, 6 таблиц, 12 использованных источников и 1 приложение.

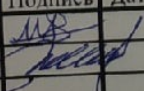
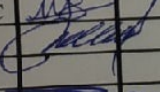
ANDROID STUDIO, JAVA, ИНТЕРФЕЙС, КУЛИНАРИЯ,
КУЛИНАРНОЕ ПРИЛОЖЕНИЕ.

Объект исследования: мобильное электронное приложение
«Кулинарная книга»

Цель работы: разработать мобильное электронное кулинарное
приложение.

Средства разработки: Android Studio, Java.

Результаты разработки: спроектировано и разработано мобильное
электронное приложение «Кулинарная книга».

					ПГУ 09.03.02–08ВР171.09 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Мобильное электронное приложение «Кулинарная книга».	Лит.	Лист	Листов
Разраб.		Меркурьев Д.С.					4	125
Провер.		Шашков Б.Д.				Группа 17ВИ1		
Н. Контр.		Такташкин Д.В.		21.06.17				
Утверд.		Макарычев П.П.						

Содержание

Введение.....	7
1 Анализ предметной области.....	8
1.1 Анализ предметной области.....	8
1.2 Обзор современных решений.....	9
1.3 Анализ рынка конкурирующего программного обеспечения.....	11
1.4 Анализ функциональных требований.....	13
1.5 Анализ требований к архитектуре ПО.....	14
1.6 Выбор средств разработки.....	14
1.7 Результаты анализа предметной области.....	15
2 Проектирование системы.....	17
2.1 Архитектура системы.....	17
2.2 Логическая модель.....	17
2.3 Физическая модель.....	18
2.4 Проектирование программного интерфейса.....	19
2.5 Проектирование программных средств.....	29
2.6 Планирование и расчет бюджета разрабатываемой системы.....	32
3 Реализация системы.....	35
3.1 Диаграмма компонентов приложения.....	35
3.2 Реализация программных средств.....	36
4 Тестирование системы и анализ кода.....	39
4.1 Функциональное тестирование методом черного ящика.....	39
4.2 Структурное тестирование.....	41
Заключение.....	46

Список использованных источников.....	47
Приложение А. Листинг программы на языке java.....	49

Введение

Сегодня сложно представить, что на мобильном устройстве любого человека не стоит ни одного приложения. Они появились практически одновременно со смартфонами и невероятно быстро развиваются. Все больше растет рынок мобильных приложений, а крупные компании все чаще понимают необходимость их разработки.

Не так давно мобильные приложения были исключительно играми, но сегодня приложения могут быть полезны в самых разных сферах. Использоваться для просмотра кино и видео, использоваться для общения между людьми, помогать ориентироваться в незнакомом месте, быть маркетинговым инструментом с помощью которого можно повысить узнаваемость бренда или быть удобным инструментом для помощи в обучении. На сегодняшний день мобильные приложения часто используются и в быту, например в кулинарии.

Рынок простых и удобных кулинарных приложений сейчас только развивается. Некоторые из них специфичны и не подойдут каждому человеку, например вегетарианские приложения. Помимо этого большинство таких приложений не переведены на русский язык, а также в них отсутствует возможность добавлять свои рецепты. В связи с этим можно сделать вывод, что на данный момент мало доступных российскому рядовому пользователю простых и удобных мобильных кулинарных приложений с возможностью добавлять свои собственные рецепты.

Целью этой квалификационной работы является разработка такого электронного мобильного приложения, задачей которого будет являться упрощения процесса готовки. В приложении будет в том числе возможность добавлять свои собственные рецепты.

1 Анализ предметной области

1.1 Анализ предметной области

Мобильное приложение представляет собой разработанную программу для планшетов и смартфонов, которая устанавливается на ту или иную платформу и имеет определенный функционал.

Процесс создания приложения проходит в несколько этапов:

- создание концепции;
- проектирование;
- разработка;
- тестирование;
- техническая поддержка.

Для разработки мобильного приложения больше всего подходят эти языки программирования:

- java;
- HTML5/JavaScript;
- C#;
- Kotlin;
- C/C++.

Главными блоками при создании мобильного приложения являются front-end и back-end.

Front-end – это клиентская часть приложения. С ней пользователь может контактировать и взаимодействовать напрямую. Во Frontend происходят отображение различных функциональных задач, пользовательского интерфейса, выполняемые на стороне клиента, а также первичная обработка пользовательских запросов.

Back-end – это программно-аппаратная часть приложения. Она отвечает за передачу данных между пользователями или ресурсами. Одной из главных функций бэкэнд является работа с пользовательской информацией, получаемой из front-end [1].

1.2 Обзор современных решений

Существует три вида мобильных приложений: нативные приложения, веб-приложения, гибридные приложения.

Нативные приложения называют так потому, что они написаны на родном (с англ. native – родной) для определённой платформы языке программирования. На Android таким языком будет являться Java, на iOS – Swift или objective-C. Нативные приложения находятся на самом устройстве, доступ к которым можно получить, нажав на иконку. Такие приложения могут использовать все возможности девайса, такие как GPS, камера, список контактов и многие другие. Нативные приложения хорошо оптимизированы и работают хорошо на любом смартфоне. Они устанавливаются через магазин приложений (Play Market для Android, App Store для iOS и др.). Примерами нативных приложений могут служить такие приложения как Shazam для поиска музыки. Преимущества нативных приложений это:

- высокий уровень безопасности;
- скорость работы и производительность;
- удобство для пользователя;
- использование возможностей гаджета;
- возможность работы без использования интернета.

Недостатки это:

- большие сроки разработки;
- доступность лишь на одной платформе;
- высокая стоимость разработки [2].

Веб-приложение не совсем является приложением в привычном смысле, а представляет собой сайт, который адаптирован и оптимизирован под любой абсолютно любой смартфон. Независимо от операционной системы устройства веб-приложение не будет использовать программное обеспечение самого смартфона, ведь запуская такое приложение пользователь по сути выполняет все те же действия, которые он выполняет при переходе на любой веб-сайт. Для того, чтобы воспользоваться веб-

приложением, нужно иметь на устройстве браузер, знать адрес и быть подключенным к интернету. Благодаря интернету в приложении производится обновления. Веб-приложения кроссплатформены, то есть они будут функционировать, независимо от платформы устройства. Кроме того, веб-приложения не используют программное обеспечение и память смартфона. Широкую известность и распространение такие приложения получили в то время, когда начал развиваться HTML5. Сейчас граница между веб-приложениями и обычными веб-страницами все более размытая и уже не всегда очевидно, где кончается одно и начинается другое. Примером таких приложений может служить last.fm для прослушивания музыки или maps.google.com с картами, которое одновременно является и веб-сайтом и приложением. Преимущества веб-приложений это:

- простота и высокая скорость разработки;
- полный охват всех платформ;
- отсутствие необходимости загрузки из магазинов приложений.

Недостатки это:

- обязательное подключение к Интернету;
- производительность и скорость работы;
- низкий уровень безопасности;
- скудный интерфейс приложения [3].

Гибридные приложения это сочетание веб и нативных приложений. Имеется в виду их кроссплатформенность и доступ к функционалу смартфона. Гибридные приложения могут быть загружены исключительно из маркетов приложений. Интернет-соединение такими приложениями используется для обновления информации. Разработчикам не приходится, как в случае с нативными приложениями, повторно загружать приложение в маркет ради устранения ошибок, ведь устранить их можно с помощью обновления. В свою очередь отсутствие интернет соединения будет мешать работать веб-функциям приложения и может привести как к ошибкам, так и к тому, что оно вообще не будет запускаться без интернета. Кроме того

гибридные приложения создаются сразу же под две платформы. Пользователю же разница между гибридным и нативным приложением скорее всего будет не замечена, так как внешне они схожи, но гибридное приложение в силу своих особенностей плохо работает со сложными анимациями и графикой, что не подходит для игр и визуальных приложений. Качество и возможности таких приложений во многом зависят от фреймворка, выбранного разработчиком. Преимущества гибридных приложений это:

- стоимость и скорость разработки;
- кроссплатформенность.

Недостатки это:

- некорректная работа при отсутствии интернет-соединения;
- разработка таких приложений требует больших усилий если нужно, чтобы гибридное приложение выглядело на одном уровне с нативным;
- средняя скорость работы на фоне нативных [4].

Нативные мобильные приложения являются самыми популярными и удобными для пользователя, что лучше всего соответствует теме. В связи с этим было решено выбрать нативное мобильное приложение под систему Android.

1.3 Анализ рынка конкурирующего программного обеспечения

Рынок мобильных приложений постоянно растет. Эта огромная отрасль расширяется с каждым днем и останавливаться пока не собирается. Резко увеличивается количество разработчиков мобильных приложений, бьет рекорды количество самих приложений. Доход, создаваемый индустрией мобильных приложений, достиг заоблачных показателей, а приложения для помощи в повседневной жизни сейчас на стадии активного развития. Конкурентоспособность приложений помогает повысить работа с рекламой.

Большинство приложений на тему еды, это приложения дающие услуги доставки («Delivery Club», «Яндекс Еда» и т.д.), рынок кулинарных приложений сильно меньше. Помимо этого, большинство кулинарных приложений не переведены на русский язык, такие как «Jamie Oliver's Recipes», «NYT Cooking». Некоторые из них специфичны и не подойдут каждому человеку, например: вегетарианское приложений «Just Vegas» [5]. Кроме того, некоторые из перечисленных приложений требуют платную подписку. После изучения рынка был сделан вывод: на данный момент крайне мало доступных российскому пользователю простых и полезных мобильных кулинарных приложений [6].

Подробное сравнение конкурирующих приложений представлено в таблице 1.

Таблица 1 – Подробное сравнение конкурирующих продуктов

№ п	Особенности	Кулинарная Книга	Just Vegan	NYT Cooking	Jamie Oliver's Recipes
1	Наличие русского языка	Да	Да	Нет	Нет
2	Возможность работы без интернета	Да	Да	Нет	Нет
3	Приложение для не вегетарианцев	Да	Нет	Да	Да
4	Бесплатное приложение	Да	Да	Нет	Да
5	Возможность добавлять свои рецепты	Да	Нет	Да	Нет

Рассмотренные приложения не предоставляют полный и удобный спектр возможностей пользователю. При разработке были учтены недостатки представленных приложений.

1.4 Анализ функциональных требований

Приложение включает в себя следующие функции:

- получить список названий рецептов;
- получить описание, каждое из которых соответствует одному рецепту;
- получить список названий составных частей рецептов (ингредиентов);
- получить список строк, представляющих собой инструкции по приготовлению;
- авторизация и аутентификация пользователя.

Исходя из рассмотренных требований разрабатываемое приложение будет содержать несколько вариантов использования представленных на рисунке 1.



Рисунок 1 – Диаграмма вариантов использования приложения

Так как приложение используется в быту одной из задач является создание удобного интерфейса. Было решено упростить доступ к информации увеличив количество главных окон.

1.5 Анализ требований к архитектуре ПО

В приложении была выбрана структура Model-View-Presenter («Модель – Представление – Презентер»). MVP – это шаблон проектирования, использующийся для разработки пользовательского интерфейса в приложении. Главным плюсом конкретно MVP является универсальность как для IOS так и для Android, а так же четкое разделение ответственности между компонентами. Компоненты приложения разделяют его данные, ПО и управляющую логику на три части. К ним относятся:

- компонент Model, который представляет собой данные для пользователя. В большинстве приложений представляет собой слой, отвечающий за получение данных с бэкэнда;
- компонент View отвечает за отображение данных. В Android приложениях это Activity или Fragment;
- компонент Presenter используется как прослойка между Model и View. Отвечает за обработку событий приходящих из View, получение данных из Model и обновление с View с полученными данными [6].

MVP работает с помощью связи модели вида и представителя. Сначала вид строит интерфейс добавляя в него данные из модели, а потом передает представителю события. После этого представитель обрабатывает данные и передает их модели. Модель выполняет операции, обновляется, а представитель получает обновленную модель и передает её виду. Вид строит интерфейс с новыми данными [7].

1.6 Выбор средств разработки

Для этапов проектирования использовались такие средства как ERwin Data Modeller и Online UML Diagram Tool. Это удобные и простые инструменты для построения uml диаграмм и разработки баз данных.

Язык Java 8 используется для разработки приложения, так как он самый подходящий и удобный для такой задачи. Кроме того используется

база данных Sqlite, в которой хранится вся необходимая информация. Так создаются большинство крупных приложений для платформы Android.

Для удобства контроля версий исходного кода была использована система контроля версий git.

Система контроля версий – это система, запоминая изменения и записывающая их в файл и позволяющая вернуться позже к определённой версии исходного кода. Она позволяет вернуть содержимое файлов к состоянию, в котором оно было до внесения изменений, а также информацию о внесенных изменениях. Предоставляет возможность вернуть проект к исходному состоянию. Предоставляет мониторинг изменений, а также информацию об участниках проекта и многое другое [8].

Для создания и редактирования базы данных были выбраны SQLite – компактная встраиваемая СУБД. DB Browser – это кроссплатформенное ПО для менеджмента баз данных для платформ Linux, Windows и MacOS. Положительными чертами этих приложений является: удобство, multifunctionality, portability, crossplatformness и принадлежность к категории свободного ПО – Open Source [9].

Было решено использовать Android Studio в качестве основной IDE для создания приложения. Android Studio – интегрированная среда разработки производства Google на которой доступны инструменты для создания приложения на платформе Android OS. В этой среде есть возможность разрабатывать приложения не только для смартфонов, но и для любых устройств на Android: на планшетах, часах и т.д.

1.7 Результаты анализа предметной области

В результате анализа была сформулирована задача на разработку системы, актуальную на рынке приложений: разработать клиентскую часть на языке Java использующего MVP структуру. Для создания приложения была выбрана среда разработки Android Studio. Для реализации базы данных была выбрана СУБД SQLite и кроссплатформенный менеджер баз

данных с открытым кодом DB Browser. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором выполняется программа.

Все рассматриваемые методики, архитектуры и технологии соответствуют актуальным требованиям рынка. Для этого требуется выполнить следующие подзадачи:

- спроектировать и реализовать базу, используя SQLite;
- сформировать классы ИС и связи между ними;
- разработать контроллеры бизнес-логики;
- реализовать слой представления в виде приложения;
- реализовать связь между приложением и базой данных.

После выполнения задач было проведено комплексное тестирование приложения. Это необходимо для последующего запуска приложения на рынок.

2 Проектирование системы

2.1 Архитектура системы

Для разработки приложения была выбрана двухзвенная архитектура. Был использован паттерн MVP для Android. Архитектура системы представлена на рисунке 2.



Рисунок 2 – Архитектура системы

Клиент является интерфейсом для взаимодействия с базой данных приложения. База данных является локальной, именно это обеспечивает работоспособность приложения в условиях отсутствия интернет соединения. Соединение с интернетом необходимо только на этапе загрузки приложения. Загруженное приложение инициализирует базу базовыми рецептами. Клиент наряду с предоставлением пользователю интерфейса для работы с БД реализует прослой бизнес логики. На стороне клиента формируются и обрабатываются пользовательские запросы к хранилищу данных. База данных упорядоченно хранит данные об объекте или группе объектов, обладающих набором свойств, которые можно категоризировать.

2.2 Логическая модель

Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Основным средством разработки логической модели данных в настоящий момент являются варианты ER-диаграмм. В результате анализа

предметной области при помощи Erwin Data Modeler была построена логическая модель данных, изображенная на рисунке 3.

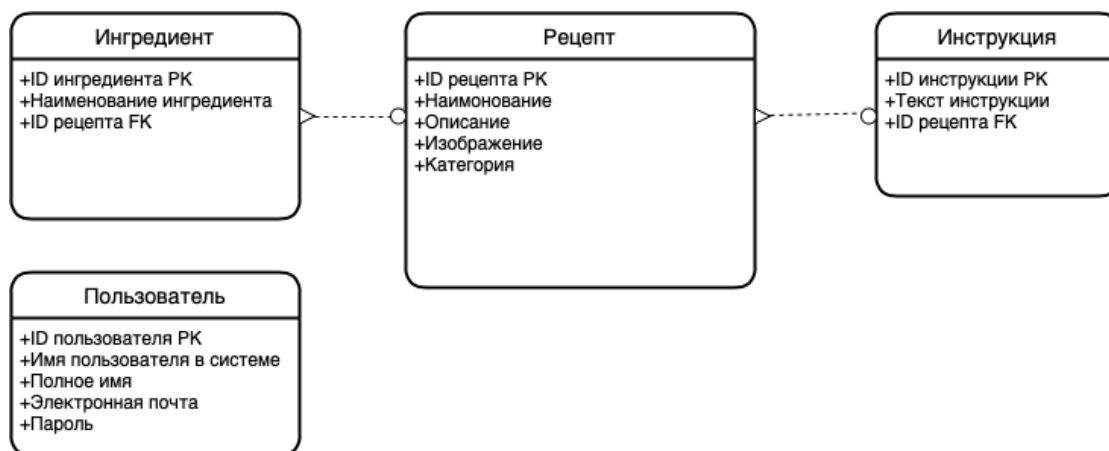


Рисунок 3 – Логическая модель

В дальнейшем будет производиться переход от логической модели БД к физической.

2.3 Физическая модель

При переходе от логической к физической модели реляционной БД нужно будет работать с объектами и ограничениями, поддерживаемыми в выбранной СУБД. Физическая модель базы данных приложения “Книга рецептов” представлено на рисунке 4.

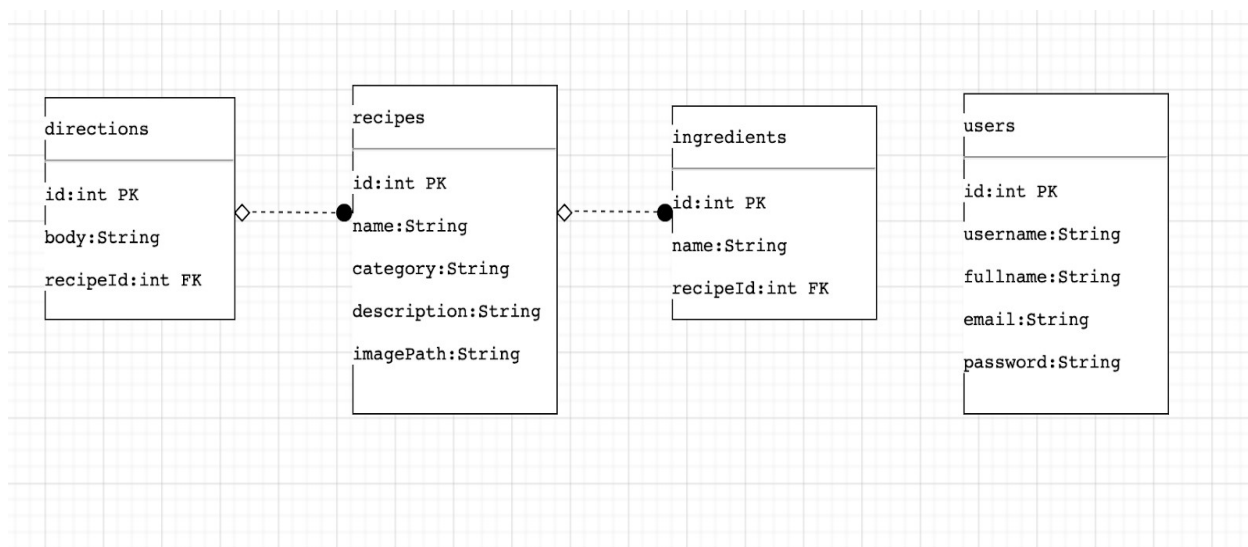


Рисунок 4 – Физическая модель

В физической модели были добавлены типы данных атрибутов, а также были добавлены такие атрибуты как:

- id – идентификатор (он присутствует в каждой из таблиц);
- body – атрибут таблицы directions, отвечает за хранение описания действия;
- recipeId – внешний ключ, используется для связи между таблицей directions и recipes;
- name – атрибут таблицы recipes, отвечает за хранение названия рецепта;
- category – атрибут таблицы recipes, отвечает за хранение наименования категории, к которой относится рецепт;
- description – атрибут таблицы recipes, отвечает за хранение описания рецепта;
- imagePath – атрибут таблицы recipes, отвечает за хранение пути до изображения на устройстве;
- name – атрибут таблицы ingredients, отвечает за хранение наименования ингредиента;
- username – атрибут таблицы users, отвечает за хранение имени пользователя в системе;
- fullname – атрибут таблицы users, отвечает за хранение полного имени пользователя;
- email – атрибут таблицы users, отвечает за хранение электронного адреса пользователя;
- password – атрибут таблицы users, отвечает за хранение пароля пользователя.

2.4 Проектирование программного интерфейса

Рассмотрим спроектированный интерфейс. Приложение имеет функционал авторизации и аутентификации. Если осуществляется первый запуск приложения, пользователь направлен на страницу регистрации. Кнопка «создать аккаунт», создает запись о пользователе в базе данных.

После прохождения этой процедуры происходит автоматический вход в приложение. Экран, на котором происходит регистрация нового клиента представлен на рисунке 5.

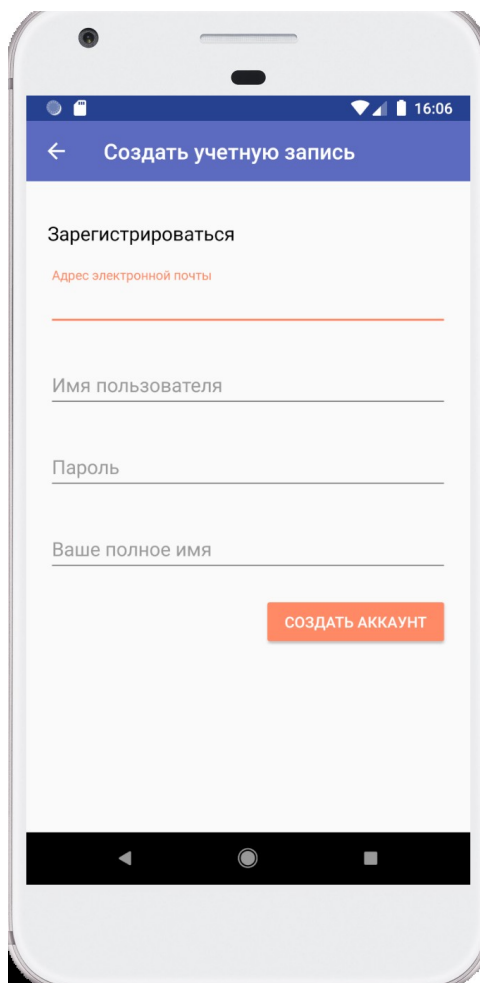


Рисунок 5 – Экран регистрации

Основные компоненты, представленные на рисунке 5:

- 1 – компонент textview;
- 2 – компоненты editText;
- 3 – компоненты button.

В том случае, если записи о пользователях существует, однако последняя сессия была прервана, пользователь направлен на экран аутентификации. На этом экране ему предложено ввести свои аутентификационные данные, а именно имя пользователя в системе и пользовательский пароль. По нажатию кнопки «войти» происходит

формирование запроса к БД. Происходит сравнение данных пользовательского ввода с данными, хранящимися в БД. Процесс аутентификации представлен на рисунке 6. На этом же экране предоставлена возможность перейти к панели регистрации. Это возможно с помощью элемента UI text-button «Если вы забыли свой логин или пароль -- зарегистрируйтесь еще раз».

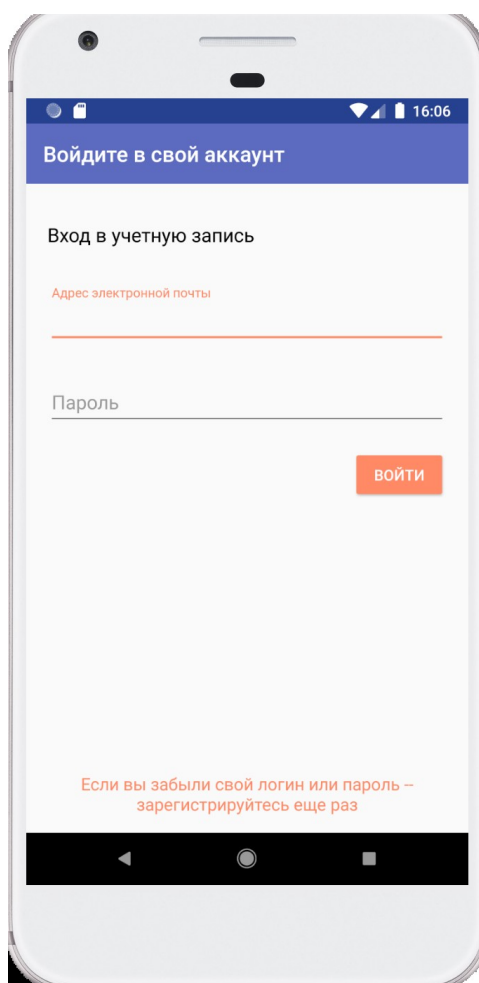


Рисунок 6 – Аутентификация

Список компонентов, представленных на рисунке 6:

- 1 – компоненты textview;
- 2 – компоненты editText;
- 3 – компонент button.

После прохождения процесса аутентификации или авторизации пользователь направлен на экран каталога рецептов. Верхний элемент toolbar

соответствует выбранной категории рецептов. Ниже представлены рецепты этой категории. С помощью выпадающего меню пользователь может добавить новый рецепт или выйти из аккаунта. Каждый элемент, представляющий изображение рецепта и название (превью), при нажатии пользователем вызывает новый экран – «Детальный просмотр рецепта». Экран каталога рецептов представлен на рисунке 7.

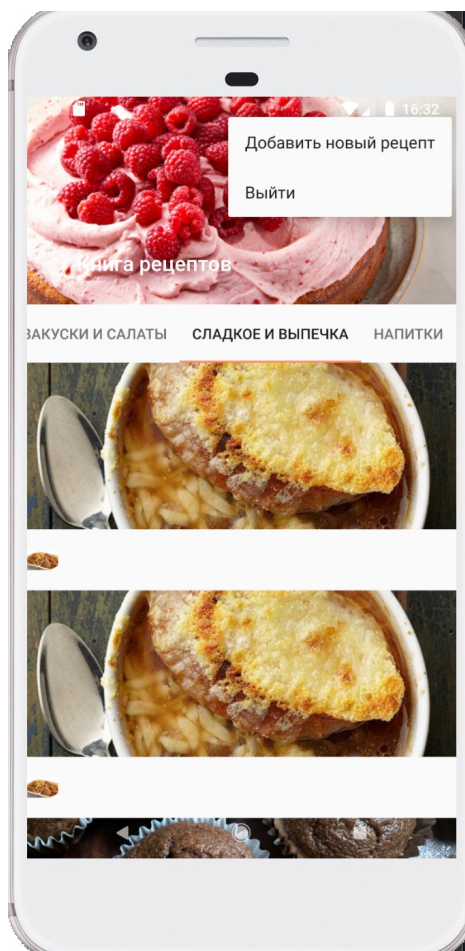


Рисунок 7 – Экран каталога рецептов

Список компонентов, представленных на рисунке 7:

- 1 – компоненты textview;
- 2 – компоненты edittext;
- 3 – компоненты image view;
- 4 – компонент menu;
- 5 – компонент button.

При нажатии на кнопку «Добавить новый рецепт» на предыдущем экране будет вызван экран добавления рецепта. На нем необходимые поля для заполнения – «Наименование», «Описание», «Изображение». Добавление изображения происходит по нажатии кнопки соответствующей кнопки. Кнопка «Далее» вызывает переход на экран добавление ингредиентов. Экран представлен на рисунке 8.

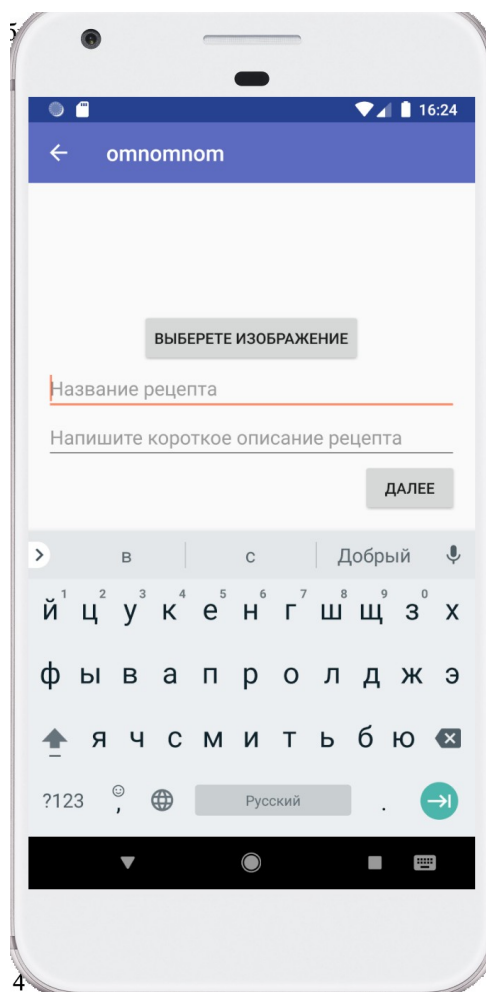


Рисунок 8 – Экран добавления рецепта

Список компонентов, представленных на рисунке 8:

- 1 – компоненты textview;
- 2 – компоненты editText;
- 3 – компоненты image view;
- 4 – компонент menu;
- 5 – компонент button.

Поля для заполнения на экране «Ингредиенты», не являются обязательными. Для добавления ингредиента реализована кнопка «Добавить», а для удаления кнопка с соответствующей иконкой. Для перехода к следующему экрану кнопка «далее». Экран представлен на рисунке 9.

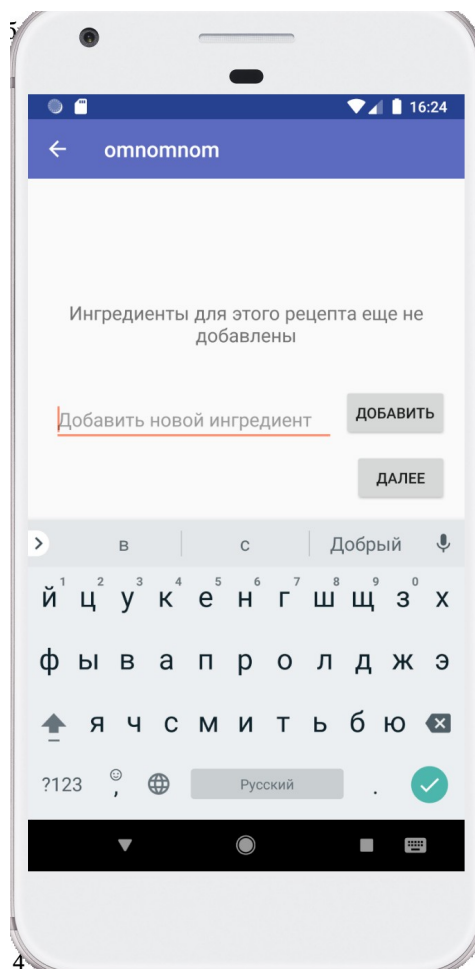


Рисунок 9 – Экран добавления ингредиентов

Список компонентов, представленных на рисунке 9:

- 1 – компоненты textview;
- 2 – компоненты edittext;
- 3 – компонент button;
- 4 – компонент image view.

Экран для добавления инструкций содержит поле, также не обязательное для заполнения и кнопку, завершения. По нажатии на нее

происходит добавления рецепта в базу данных и переход к экрану каталога, на котором будет отображен только что созданный рецепт. Экран представлен на рисунке 10.

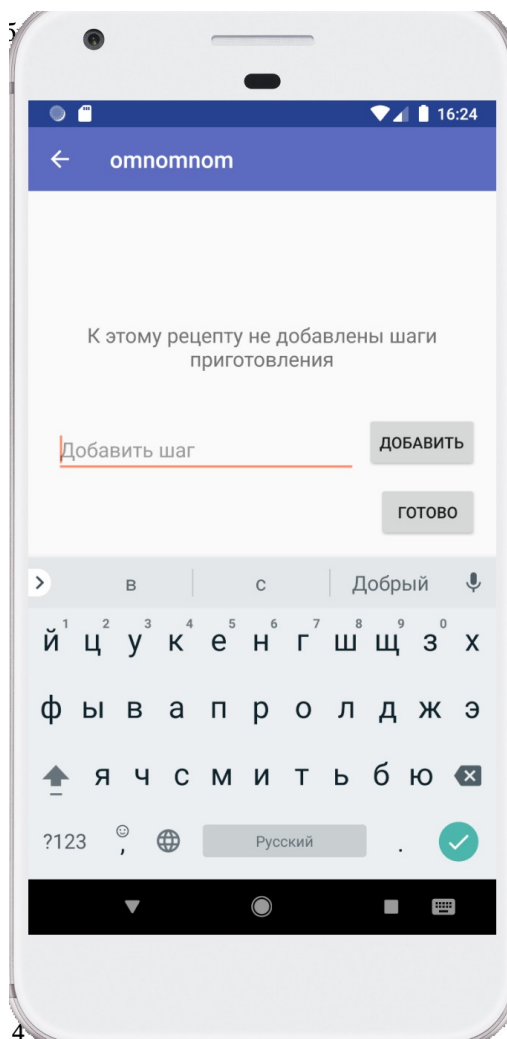


Рисунок 10 – Экран добавления инструкций и завершения процесса создания рецепта

Список компонентов, представленных на рисунке 10:

- 1 – компоненты textview;
- 2 – компоненты editText;
- 3 – компоненты button.

Для детального просмотра рецепта разработан отдельный экран. Вверху расположены иконки кнопок «удалить» и «редактировать». Экран представлен на рисунке 11.

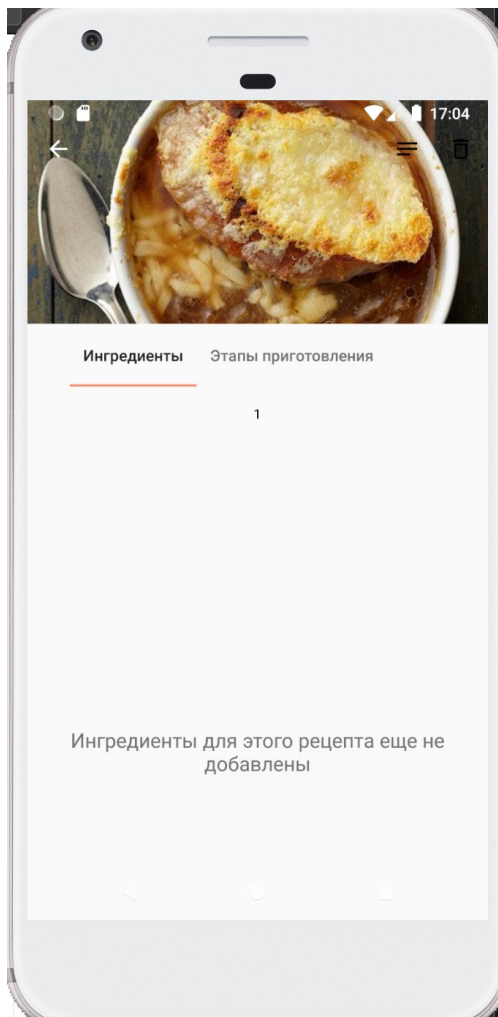


Рисунок 11 – Детальный просмотр

Список компонентов, представленных на рисунке 11:

- 1 – компоненты textview;
- 2 – компоненты image view;
- 3 – компоненты button.

Для редактирования рецептов разработан отдельный экран. Вверху расположена иконка кнопок «обновить изображение». Экран представлен на рисунке 12.

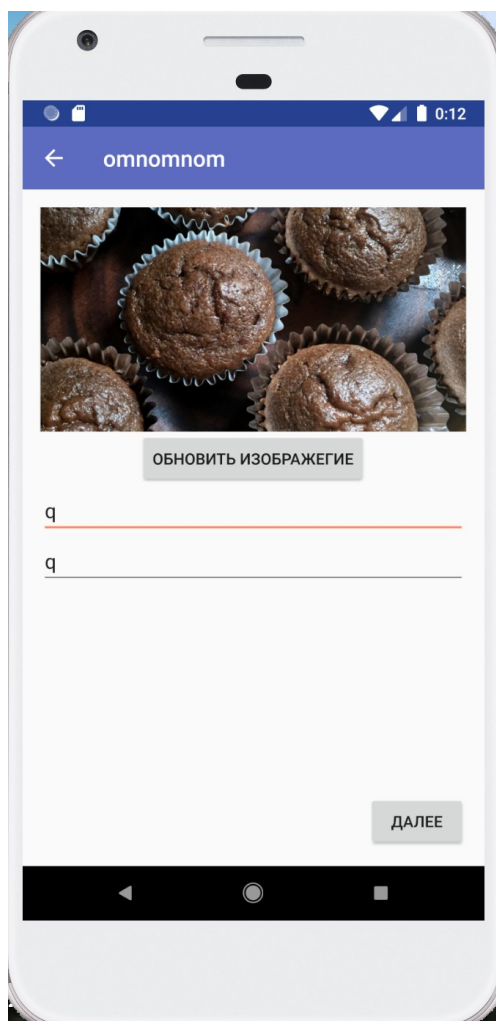


Рисунок 12 – Редактирование общей информации о рецепте

Список компонентов, представленных на рисунке 12:

- 1 – компоненты textview;
- 2 – компоненты editText;
- 3 – компоненты image view;
- 4 – компоненты button.

Интерфейс редактирования схож с интерфейсом добавления рецепта. По нажатию кнопки «далее» на предыдущем экране совершается переход на экран редактирования ингредиентов. Кнопка-иконка «удалить» позволяет удалить ранее добавленные ингредиенты. Кнопка «добавить» реализует функционал добавления нового ингредиента. По нажатии на кнопку «далее» происходит переход к следующему экрану – добавлению инструкций по

приготовлению. Экран редактирования ингредиентов представлен на рисунке 13.

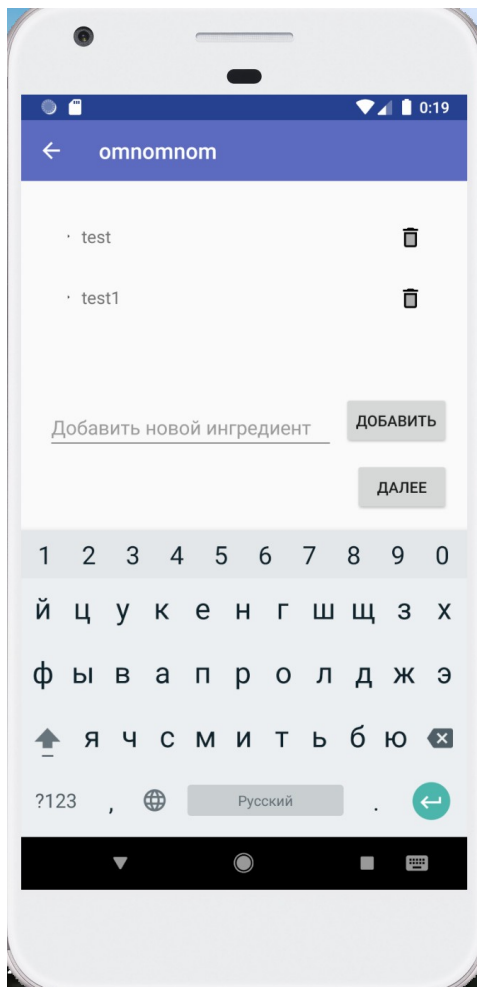


Рисунок 13 – Редактирование ингредиентов

Список компонентов, представленных на рисунке 13:

- 1 – компоненты textview;
- 2 – компоненты edittext;
- 3 – компоненты button.

Последний экран интерфейса редактирования предоставляет пользователю возможность отредактировать ранее добавленные инструкции по приготовлению и/или добавить дополнительные шаги к уже существующим. Для удаления рецепта реализована кнопка-иконка удалить, для добавления кнопка «добавить». Это последний этап редактирования рецепта. По нажатии кнопки «готово» происходит обновление данных в БД.

После происходит переход к главному экрану приложения. Экран редактирования инструкций представлен на рисунке 14.

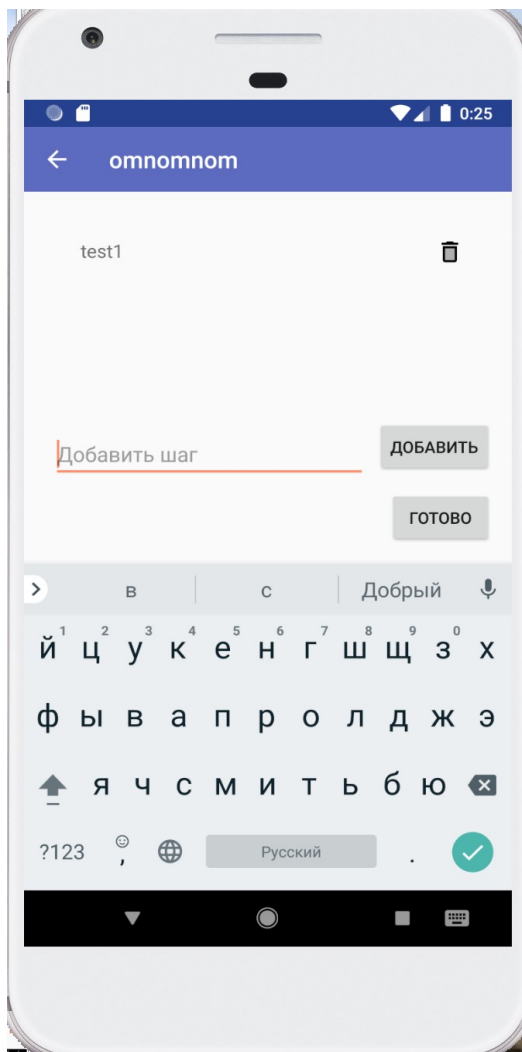


Рисунок 14 – Редактирование ингредиентов

Список компонентов, представленных на рисунке 14:

- 1 – компоненты textview;
- 2 – компоненты editText;
- 3 – компоненты button.

2.5 Проектирование программных средств

На основе анализа предметной области был разработан набор классов, приведенный на рисунке 15.

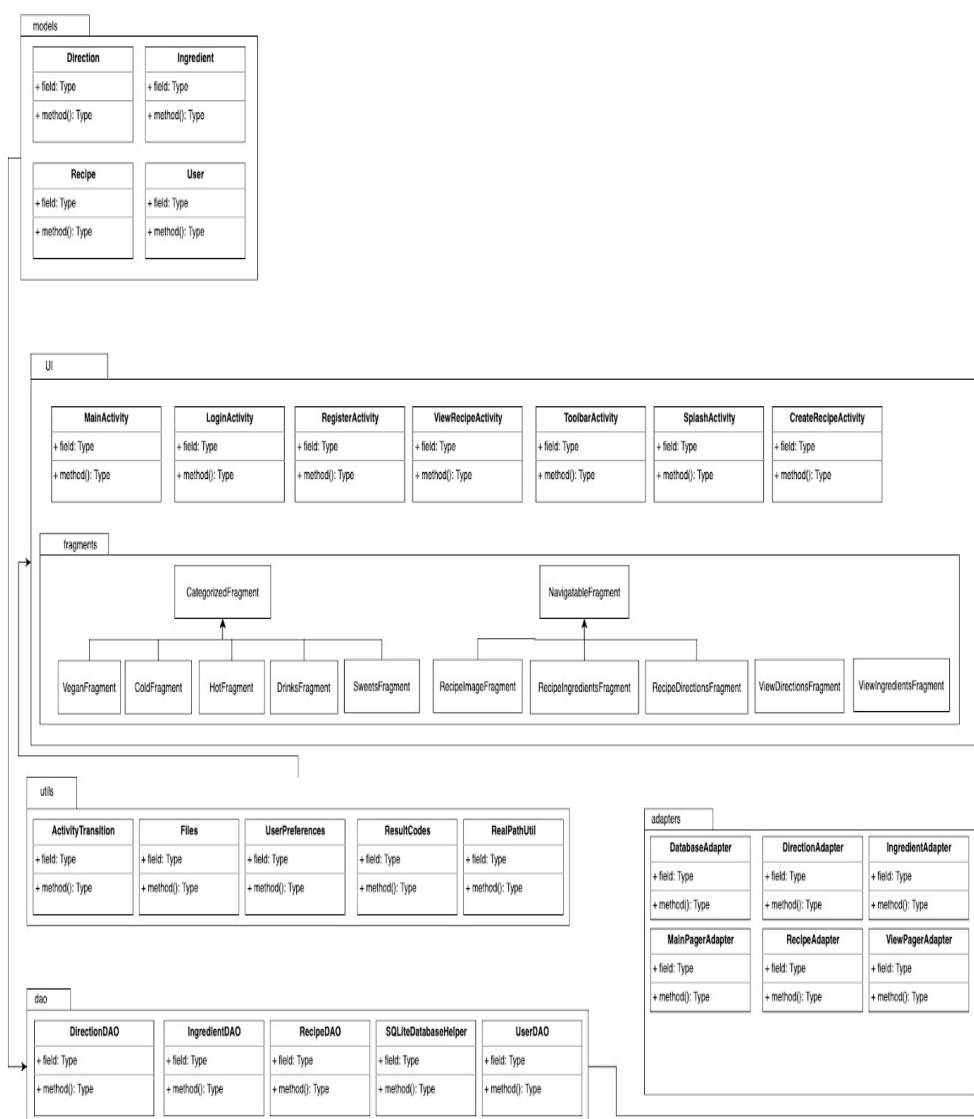


Рисунок 15 – Диаграмма классов

На диаграмме изображены следующие классы:

- LoginActivity представляет экран авторизации пользователя;
- MainActivity представляет основной экран (каталог рецептов);
- ViewRecipeActivity представляет экран просмотра рецепта;
- SplashActivity – экран, вызываемый при старте приложения, который, если приложение запущено в первый раз, инициализирует локальную БД подготовленными записями;

- RegisterActivity – экран регистрации;
- CreateRecipeActivity – экран создания рецепта;
- CategorizedFragment – фрагмент экрана, отвечающий за категории;
- ColdFragment – экран категории холодных закусок;
- DrinksFragment – экран категории напитков;
- HotFragment – экран категории горячих блюд;
- RecipeIngredientsFragment – фрагмент, отвечающий за отображение списка ингредиентов на экране просмотра рецепта;
- RecipeImageFragment – фрагмент, отвечающий за отображение изображения на экране просмотра рецепта;
- RecipeDirectionsFragment – фрагмент, отвечающий за отображение списка инструкций на экране просмотра рецепта;
- Direction – модель данных «Инструкция»;
- Ingredient – модель данных «Ингредиент»;
- Recipe – модель данных «Рецепт»;
- User – модель данных «Пользователь»;
- DirectionDAO – класс, реализующий интерфейс взаимодействия с БД, отвечает за обработку и хранение инструкций;
- IngredientDAO – класс, реализующий интерфейс взаимодействия с БД, отвечает за обработку и хранение ингредиентов;
- RecipeDAO – класс, реализующий интерфейс взаимодействия с БД, отвечает за обработку и хранение рецептов;
- SQLiteDatabaseHelper;
- UserDao – класс, реализующий интерфейс взаимодействия с БД, отвечает за обработку и хранение пользователей;
- ViewPagerAdapter – класс, отвечающий за панель выбора на экране детального просмотра рецепта;
- Files – класс, реализующий вспомогательный функционал по загрузке и последующей работе с изображениями;

– RealPathUtil – класс, реализующий вспомогательный функционал по сохранению изображений в базу данных;

– ResultCodes – класс, реализующий вспомогательный функционал, хранит в себе константы-статус коды;

Таким образом, были выделены и разработаны основные классы, а также связи между ними.

2.6 Планирование и расчет бюджета разрабатываемой системы

Для планирования проекта были выделены основные задачи и отображены с помощью Microsoft Project Professional 2010 на рисунке 16.

	Task Name	Duration	Start	Finish	Resource Names
0	▲ Разработка мобильного приложения "Книга Рецептов"	46 days	Thu 4/16/20	Thu 6/18/20	
1	▲ Анализ	4 days	Thu 4/16/20	Wed 4/22/20	
2	Анализ предметной области	8 hrs	Thu 4/16/20 8:00 AM	Thu 4/16/20 5:00 PM	Аналитик
3	Анализ бизнес-процессов предметной области	1 day	Fri 4/17/20 8:00 AM	Fri 4/17/20 5:00 PM	Аналитик
4	Анализ существующих решений	1 day	Mon 4/20/20 8:00 A	Mon 4/20/20 5:00 P	Аналитик
5	Анализ функциональных требований	1 day	Tue 4/21/20 8:00 AM	Tue 4/21/20 5:00 PM	Аналитик
6	Анализ завершен	0 days	Wed 4/22/20 8:00 A	Wed 4/22/20 8:00 A	
7	▲ Проектирование	9 days	Fri 4/24/20 8:00 A	Thu 5/7/20 8:00 A	
8	Проектирование системы	2 days	Fri 4/24/20 8:00 AM	Mon 4/27/20 5:00 P	Архитектор
9	Проектирование БД	1 day	Tue 4/28/20 8:00 AM	Tue 4/28/20 5:00 PM	Архитектор
10	Проектирование интерфейса взаимодействия с БД	1 day	Wed 4/29/20 8:00 A	Wed 4/29/20 5:00 P	Архитектор
11	Проектирование прослойки бизнес-логики	3 days	Thu 4/30/20 8:00 AM	Mon 5/4/20 5:00 PM	Архитектор
12	Проектирование пользовательского интерфейса	2 days	Tue 5/5/20 8:00 AM	Wed 5/6/20 5:00 PM	Архитектор
13	Этап проектирования завершен	0 days	Thu 5/7/20 8:00 AM	Thu 5/7/20 8:00 AM	
14	▲ Разработка	25 days	Fri 5/8/20 8:00 A	Thu 6/11/20 5:00	
15	Реализация системы	3 days	Fri 5/8/20 8:00 AM	Tue 5/12/20 5:00 PM	Программист-р
16	Реализация интерфейса взаимодействия с БД	1 day	Fri 5/15/20 8:00 AM	Fri 5/15/20 5:00 PM	Программист-р
17	Реализация прослойки бизнес-логики	1 day	Thu 5/21/20 8:00 AM	Thu 5/21/20 5:00 PM	Программист-р
18	Реализация пользовательского интерфейса	15 days	Fri 5/22/20 8:00 AM	Thu 6/11/20 5:00 PM	Программист-р
19	Отладка	0 days	Wed 5/27/20 3:00 P	Wed 5/27/20 3:00 P	Программист-р
20	Разработка завершена	0 days	Thu 5/28/20 3:00 PM	Thu 5/28/20 3:00 PM	
21	▲ Документирование	4 days	Thu 5/28/20 8:00	Wed 6/3/20 8:00	
22	Разработка руководства пользователя	2 days	Thu 5/28/20 8:00 AM	Fri 5/29/20 5:00 PM	Технический пи
23	Документирование исходного кода	1 day	Fri 5/29/20 1:00 PM	Mon 6/1/20 12:00 P	Технический пи
24	Документирование завершено	0 days	Wed 6/3/20 8:00 AM	Wed 6/3/20 8:00 AM	
25	▲ Тестирование	14 days	Mon 6/1/20 8:00 A	Thu 6/18/20 5:00	
26	Ручное тестирование	7 days	Mon 6/1/20 8:00 AM	Tue 6/9/20 5:00 PM	Специалист по
27	Внедрение инструментов автоматизации процессов ручного тестиров	1 day	Wed 6/10/20 8:00 A	Wed 6/10/20 5:00 P	Программист-р
28	Разработка средств автоматизированного тестирования	1 day	Thu 6/11/20 8:00 AM	Thu 6/11/20 5:00 PM	Программист-р
29	Интеграционные тестирование	1 wk	Fri 6/12/20 8:00 AM	Thu 6/18/20 5:00 PM	Специалист по
30	Тестирование завершено	0 days	Thu 6/18/20 8:00 AM	Thu 6/18/20 8:00 AM	
31					

Рисунок 16 – Основные задачи разработки системы

Для подробного просмотра плана задач и распределении ресурсов была составлена диаграмма Ганта, представленная на рисунке 17.

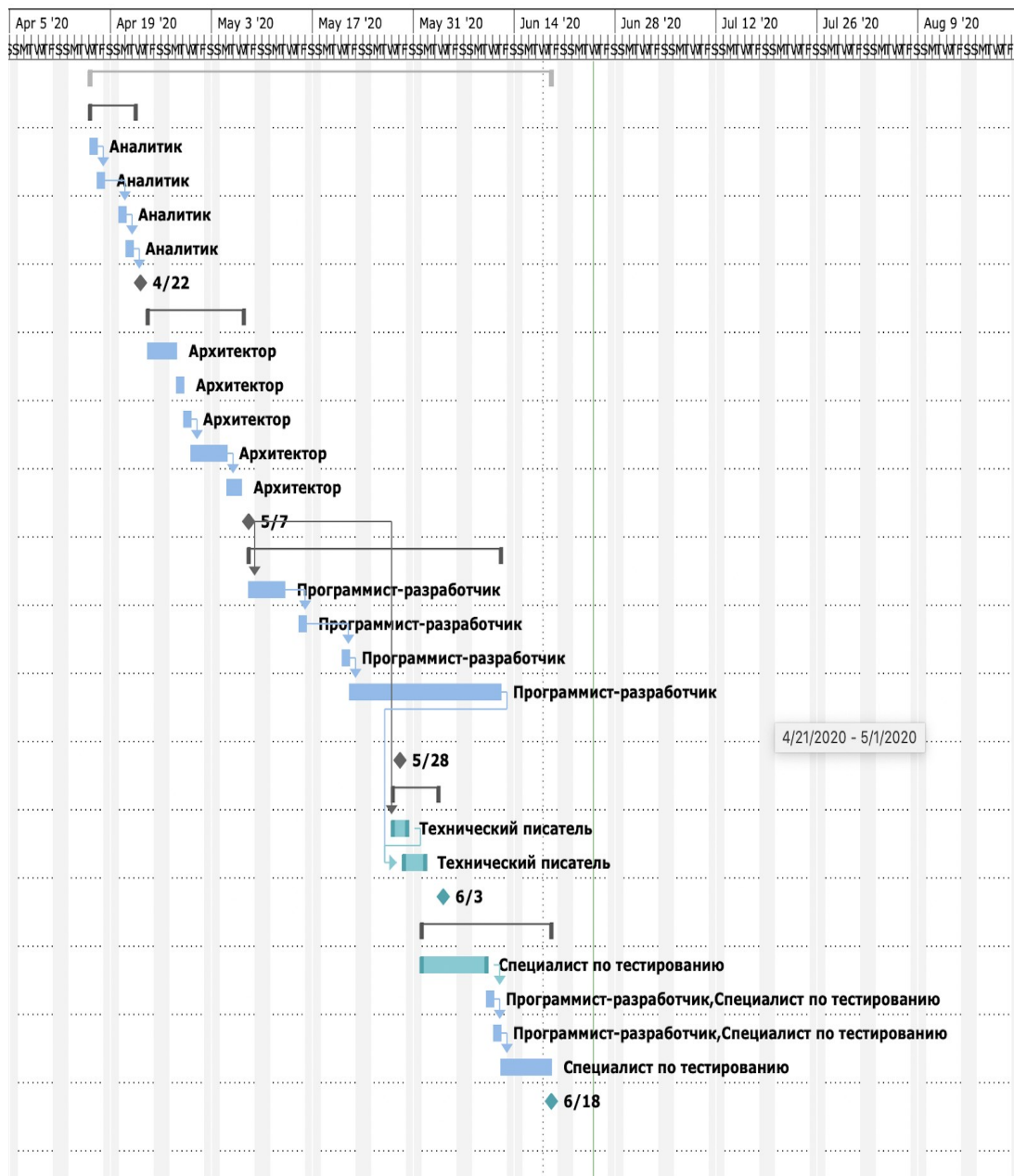


Рисунок 17 – Диаграмма Ганта

Во время проектирование и разработки необходимо следить за расходами ресурсов. Основные расходы необходимые для реализации приложения приведены на рисунке 18.

<u>Task Name</u>	<u>4/5/21</u>	<u>4/12/21</u>	<u>4/19/21</u>	<u>4/26/21</u>	<u>5/3/21</u>	<u>5/10/21</u>	<u>5/17/21</u>	<u>5/24/21</u>	<u>5/31/21</u>	<u>6/7/21</u>	<u>Total</u>
Анализ бизнес-процессов предметной области	\$0.00	\$112.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$112.00
Анализ существующих решений	\$0.00	\$112.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$112.00
Анализ функциональных требований	\$0.00	\$0.00	\$112.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$112.00
Анализ завершен	\$0.00	\$0.00	\$112.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$112.00
Проектирование интерфейса взаимодействия с БД	\$0.00	\$0.00	\$160.00	\$160.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$320.00
Проектирование прослойки бизнес-логики	\$0.00	\$0.00	\$0.00	\$160.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$160.00
Проектирование пользовательского интерфейса	\$0.00	\$0.00	\$0.00	\$160.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$160.00
Этап проектирования завершен	\$0.00	\$0.00	\$0.00	\$320.00	\$160.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$480.00
Реализация системы	\$0.00	\$0.00	\$0.00	\$0.00	\$320.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$320.00
Реализация пользовательского интерфейса	\$0.00	\$0.00	\$0.00	\$0.00	\$120.00	\$240.00	\$0.00	\$0.00	\$0.00	\$0.00	\$360.00
Отладка	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$120.00	\$0.00	\$0.00	\$0.00	\$0.00	\$120.00
Разработка завершена	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$120.00	\$0.00	\$0.00	\$0.00	\$120.00
Разработка руководства пользователя	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$120.00	\$600.00	\$600.00	\$480.00	\$1,800.00
Внедрение инструментов автоматизации процессов ручного тестирования	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$112.00	\$0.00	\$0.00	\$112.00
Разработка средств автоматизированного тестирования	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$28.00	\$28.00	\$0.00	\$56.00

Рисунок 18 – Расходы, необходимые для реализации приложения

Поставленные задачи были выполнены. Можно сделать вывод, что реализация системы экономически выгоднее, чем покупка приложений-аналогов. Можно сделать вывод, что длительность реализации проекта составит 70 дней. Затраты составят 4456 долларов, что примерно 325399 рублей.

3 Реализация системы

3.1 Диаграмма компонентов приложения

Диаграмма компонентов позволяет определить структуру программы, ее элементы и взаимоотношения этих элементов. Кроме того диаграмма компонентов позволяет установить зависимости между разными компонентами. Для программной структуры приложения «Книга рецептов» была построена диаграмма компонентов, изображенная на рисунке 18.

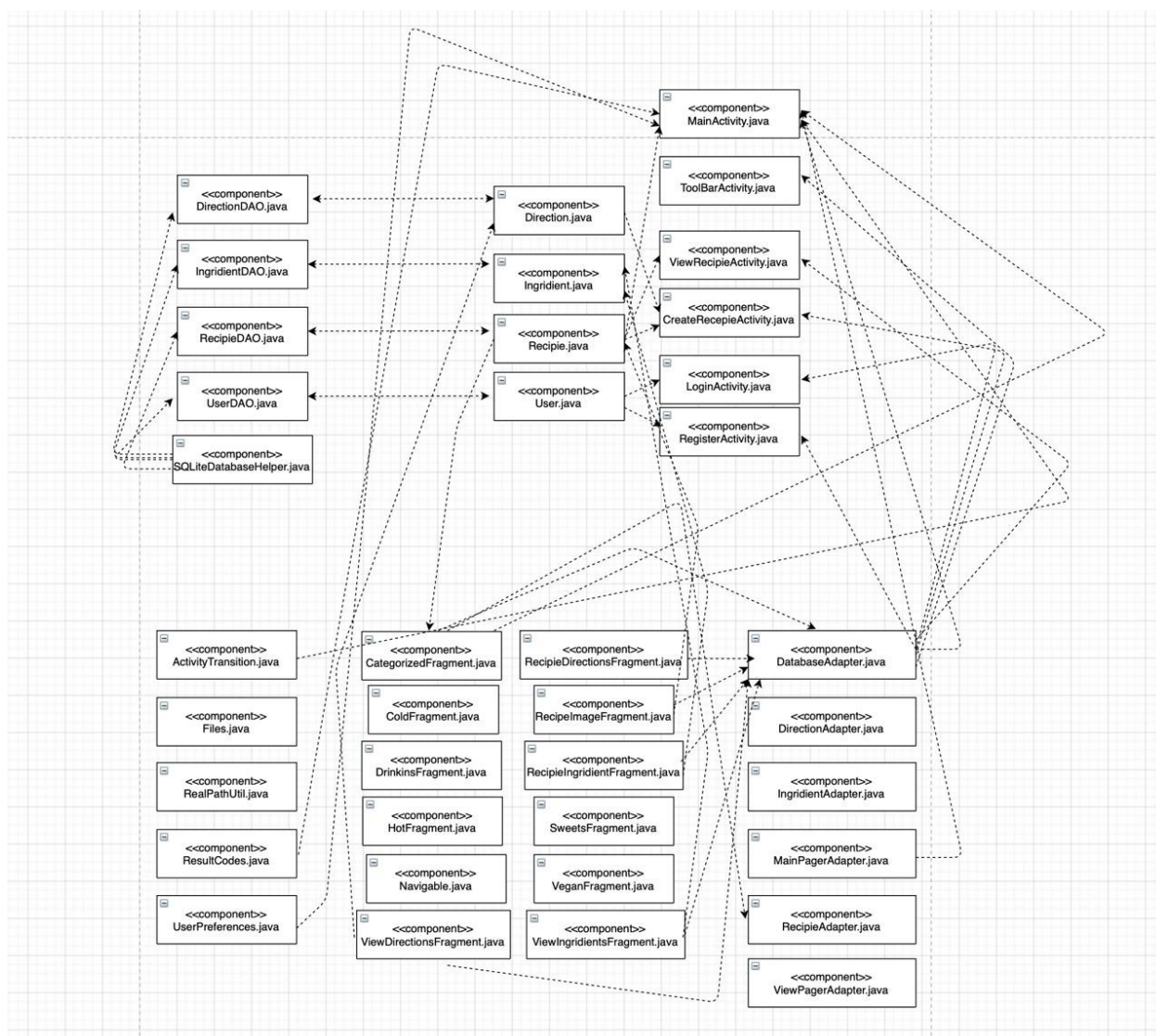


Рисунок 18 – Диаграмма компонентов приложения

В качестве компонентов выступают различные модули, файлы, библиотеки, пакеты.

3.2 Реализация программных средств

Для удобства контроля версий исходного кода была использована система контроля версий git. Была создана корневая директория дипломного проекта. Далее была выполнена команда `git init` (Команда создает пустой гит-репозиторий). В результате выполнение команды было получено сообщение «Инициализирован пустой репозиторий Git в /Users/merk/dipl/.git/». Таким образом был создан пустой локальный репозиторий. Далее была начата разработка. Был создан базовый шаблон android приложения.

Для хранения данных в удаленном репозитории был выбран сервис `bitbucket.org`. Репозиторий был создан как приватный, поэтому изменения были доступны только для одного пользователя. Был сделан инициализирующий коммит, который включал в себя добавление файла `.gitignore` для android проекта (листинг файла в приложении) и файла `README.md`, в котором содержатся заметки по проекту. Файлы были добавлены в локальный репозиторий с помощью команды `git add`. Был сделан коммит. Коммиты – основные конструктивные элементы временной шкалы проекта Git. Их можно рассматривать как снимки состояния или контрольные точки на временной шкале проекта Git. К каждому коммиту было добавлено сообщение о внесенных изменениях в проект. Со списком коммитов можно ознакомиться путем выполнения команды `git log` в директории, хранящей репозиторий. Вывод команды приведен в приложении.

Были выбраны название приложения, домен, путь к проекту и название пакета. Опционально, была предоставлена возможность включить поддержку языков программирования таких как C++ и Kotlin. Было принято решение не использовать поддерживаемые опции.

Была задана целевая платформа для сборки проекта. Для этого были использованы SDK и AVD, менеджер виртуальных устройств Android. Инструмент позволяет устанавливать в SDK пакеты, которые поддерживают несколько версий ОС Android и несколько уровней API (интерфейсов программирования приложений).

Была выбрана основная активность, которая запускается при нажатии на иконку приложения. Это MainActivity. Классы Activity отвечают за создание экрана. Класс Activity является важнейшим компонентом Android-приложения, а способ запуска и компоновки является фундаментальной частью платформы Android. В отличие от парадигм программирования, где приложение запускается в методе main(), система Android инициализирует код в экземпляре Activity, вызывая специальные коллбэки, которые соответствуют конкретным этапам жизненного цикла активности. Это окно обычно полностью заполняет экран, но может быть меньше экрана или находиться поверх других окон. Как правило, одна активность представляет собой один экран в приложении. Например, одна активность может реализовывать окно настроек, в то время как другая – окно с выбором фото. Большинство приложений имеют несколько экранов, из чего можно сделать вывод, что они содержат несколько активностей. Как правило, одна из активностей определяется как главная и является первым экраном, появляющимся при запуске пользователем приложения. Затем каждая активность может запустить другую активность для выполнения каких-либо действий. Несмотря на то, что активности работают вместе, чтобы организовать хороший пользовательский интерфейс в приложении, каждая активность слабо связана с другими: обычно в приложении есть минимальные зависимости между активностями [10].

Файл AndroidManifest.xml был автоматически создан средством разработки Android Studio. Он описывает фундаментальные характеристики приложения, а также определяет все его компоненты. Далее автоматически сгенерированные файлы, хранящиеся в директории app/res/layout, были отредактированы. Также были добавлены новые файлы в директорию. Это XML-файлы с макетом для UI основной активности. Фрагмент кода файла AndroidManifest.xml представлен на рисунке 19.

```

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="omnomnom"
    android:roundIcon="@drawable/ic_launcher"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".ui.main.RegisterActivity" />
    <activity android:name=".ui.main.LoginActivity" />
    <activity android:exported="true" android:name=".ui.main.MainActivity"/>
    <activity
        android:name=".ui.main.CreateRecipeActivity"
        android:configChanges="orientation|screenSize" />
    <activity android:name=".ui.main.ViewRecipeActivity"/>
    <activity
        android:name=".ui.main.SplashActivity"
        android:theme="@style/SplashTheme">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>

```

Рисунок 19 – Фрагмент кода AndroidManifest.xml

Пользовательский интерфейс Android-приложения создаётся через иерархию макетов (layouts, объекты ViewGroup) и виджетов (объекты View [11]. Макеты управляют расположением дочерних виджетов на экране. Сами виджеты – это непосредственно компоненты UI: кнопки, текстовые поля на экране и т.п. Интерфейс активностей создаётся в Android Studio в редакторе макетов (Layout Editor) в основном хранится в формате XML.

Далее были добавлены действия, активности и навигация. Полное описание кода находится в приложении А.

4 Тестирование системы и анализ кода

4.1 Функциональное тестирование методом черного ящика

Функциональное тестирование предназначено для проверки способности разрабатываемого ПО решать необходимые пользователю задачи в определенных условиях.

При проведении функционального тестирования для построения тестовых наборов использовались метод эквивалентного разделения и метод анализа граничных значений.

Была использована техника классов эквивалентности. В качестве тестируемого поля было выбрано поле ввода текста «Описание рецепта». Оно доступно при вызове окна «Добавить рецепт». Допустимыми значениями являются все строки, длина которых не менее 1 символа и не превышает 200 символов. Таким образом, все значения, входящие в этот диапазон, эквивалентны. Все недопустимые значения также являются эквивалентными. Исходя из условий, разбиение на классы эквивалентности делится на две категории: допустимые значения и недопустимые значения. К допустимым значениям или к блоку позитивных тестов относятся:

1 – любые символы, составляющие строку, размер которой не выходит за диапазон заданных значений;

2 – допускается использование пробельных символов;

3 – допускается использование специальных символов.

К недопустимым значениям или к блоку негативных тестов относятся:

1 – пустая строка или null;

2 – любой текст, выходящий за рамки заданного диапазона [12].

Исходя из установленных границ, приведен тестовый набор в таблице 2.

Таблица 2 – Классы эквивалентности

№	Тест	Ожидаемый результат	Результат
1	+!=!'--\$^\$\$ \$1111111@@@	Успешное создание записи.	Тест выполнен, запись в базе данных создана.
2	Табуляция	Успешное создание записи.	Тест выполнен, запись в базе данных создана.
3	Тест тест тест тест	Успешное создание записи.	Тест выполнен, запись в базе данных создана.
4	Пустая строка	Вывод сообщения об ошибке.	Тест выполнен, выведено сообщение об ошибке.
5	Текст, состоящий из более чем 200 символов	Вывод сообщения об ошибке.	Тест выполнен, выведено сообщение об ошибке.

Была использована техника анализа граничных значений. В качестве тестируемого поля было выбрано поле ввода текста «Описание рецепта». Оно доступно при вызове окна «Добавить рецепт».

Логической границей установим, что текст не может быть пустым. Технологической границей установим, что текст не должен превышать 65535 байт (максимальный размер переменной строкового типа в java). Произвольной границей установим, что количество символов должно находиться в пределах [1,200]. Исходя из установленных границ, приведем тестовый набор в таблице 3.

Таблица 3 – Анализ граничных значений

№	Тест	Ожидаемый результат	Результат
1	В поле ничего не вводится.	Вывод сообщения об ошибке.	Тест выполнен, выведено сообщение об ошибке.
2	Вводится текст, в котором более 200 символов.	Вывод сообщения об ошибке.	Тест выполнен, выведено сообщение об ошибке.

Было произведено функциональное тестирование методом черного ящика.

4.2 Структурное тестирование

Был рассмотрен метод, реализующий запись рецепта в базу данных. Блок-схема представлена на рисунке 20.

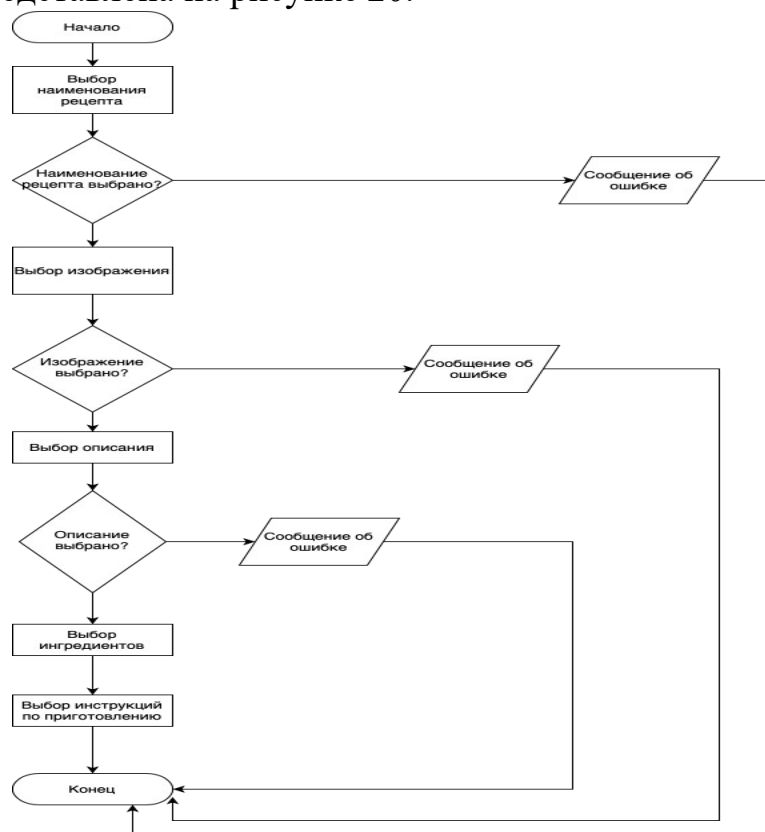


Рисунок 20 – Блок-схема метода создания рецепта

На основании блок-схемы сформирован потоковый граф, узлы которого включают операторы программы, а дуги отображают поток управления. Граф представлен на рисунке 21.

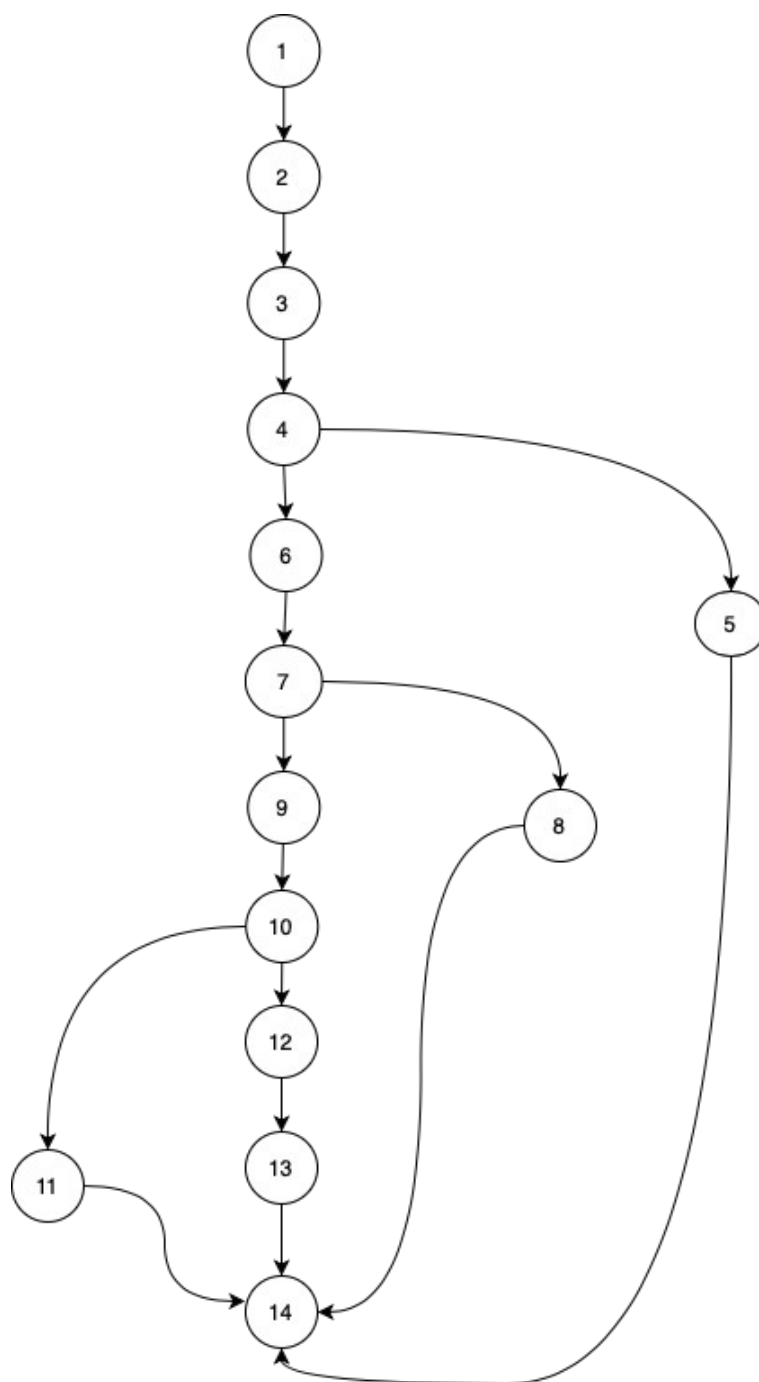


Рисунок 21 – Потоковый граф для создания рецепта

Тестовые наборы приведены в таблице 4.

Таблица 4 – Анализ граничных значений

№	Операторы	Входные данные	Выходные данные	Результат
1	1, 2, 3, 4, 5, 14	null	Сообщение об ошибке	Тест выполнен, выведено сообщение об ошибке
2	1, 2, 3, 4, 6, 7, 8, 14	ТестНазвание2, null	Сообщение об ошибке	Тест выполнен, выведено сообщение об ошибке
3	1, 2, 3, 4, 6, 7, 9, 10, 11, 14	ТестНазвание3, file:///path/to/file/test 3. png, null	Сообщение об ошибке	Тест выполнен, выведено сообщение об ошибке
4	1, 2, 3, 4, 6, 7, 9, 10, 12, 13, 14	ТестНазвание4, file:///path/to/file/test4 .. png, ТестОписание, [ТестИнгредиенты4, [ТестИнструкции4]	Запись в базе данных	Тест выполнен, запись успешно создана в базе, созданная запись появляется в интерфейсе

Было произведено тестирование условий. Тестовый набор был построен таким образом, чтобы все возможные результаты каждого условия в решении были выполнены по крайней мере один раз. Тестовый набор приведен в таблице 5.

Таблица 5 – Тестовый набор для тестирования условий

№	Операторы	Входные данные	Выходные данные	Результат
1	4 -- нет	Не выбрано наименование	Сообщение об ошибке,	Тест выполнен,

		рецепта	связанной с отсутствием наименования	выведено сообщение об ошибке
--	--	---------	--	------------------------------------

Продолжение таблицы 5

№	Операторы	Входные данные	Выходные данные	Результат
2	2 – да, 7 – да, 10 – нет	Наименование: ТестНазвание	Сообщение об ошибке, связанной с отсутствием изображения	Тест выполнен, выведено сообщение об ошибке
3	4 — да, 7 — да, 10 — нет	Наименование: ТестНазвание, Изображение: Изображение: file:///path/to/file /test4..png,	Сообщение об ошибке, связанной с отсутствием описания	Тест выполнен, выведено сообщение об ошибке
4	4 — да, 7 — да, 10 — да	Наименование: ТестНазвание, Изображение: Изображение: file:///path/to/file /test4..png, Описание: ТестОписание	Вывод добавленного в бд рецепта	Тест выполнен, запись успешно создана в базе, созданная запись появляется в интерфейсе

Было произведено тестирование путей. Тестовый набор был построен таким образом, чтобы каждый путь выполнялся по крайней мере один раз. Тестовый набор приведен в таблице 6.

Таблица 6 – Тестовый набор для тестирования путей

№	Операторы	Входные данные	Выходные данные	Результат
1	1, 2, 3, 4, 5, 14	Наименование: null	Сообщение об ошибке, связанной с отсутствием наименования	Тест выполнен, выведено сообщение об ошибке

2	1, 2, 3, 4, 6, 7, 8, 14	Наименование: Test1, Изображение: null	Сообщение об ошибке, связанной с отсутствием изображения	Тест выполнен, выведено сообщение об ошибке
---	-------------------------	---	--	--

Продолжение таблицы 6

№	Операторы	Входные данные	Выходные данные	Результат
3	1, 2, 3, 4, 6, 7, 9, 10, 11, 14	Наименование: Test2, Изображение: Изображение: file:///path/to/file/ test2.png, Описание: null	Сообщение об ошибке, связанной с отсутствием описания	Тест выполнен, выведено сообщение об ошибке
4	1, 2, 3, 4, 6, 7, 9, 10, 12, 13, 14	Наименование: Test3, Изображение: Изображение: file:///path/to/file/ test3.png, Описание: Test3	Добавление в базу данных записи и последующий её вывод в интерфейсе пользователя	Тест выполнен, запись успешно создана в базе, созданная запись появляется в интерфейсе

Тестовый набор был построен таким образом, чтобы каждый путь выполнялся по крайней мере один раз.

Заключение

В ходе выполнения выпускной квалификационной работы была изучена предметная область и выполнена задача разработки системы актуальной на рынке приложений: разработать клиентскую часть на языке Java с MVP структурой. Для создания приложения была выбрана среда разработки Android Studio. Для реализации базы данных была выбрана СУБД SQLite и кроссплатформенный менеджер баз данных с открытым кодом DB Browser.

Все рассматриваемые методики, архитектуры и технологии соответствуют актуальным требованиям рынка. Для этого реализованы следующие подзадачи:

- спроектирована и реализована база, используя SQLite;
- сформированы классы ИС и связи между ними;
- разработаны контроллеры бизнес-логики;
- реализован слой представления в виде приложения;
- реализована связь между приложением и базой данных.

После выполнения задач было проведено комплексное тестирование приложения. Длительность проекта составляет 70 дней. Бюджет проекта составляет 4456 долларов или примерно 325399 рублей.

После выполнения задач было проведено комплексное тестирование приложения. Это необходимо для последующего запуска приложения на рынок.

Список использованных источников

1. Разработка со стороны Front end [Электронный ресурс]. – что это такое и чем отличается от back end? URL: <https://dan-it.com.ua/razrabotka-so-storony-front-end-chto-jeto-takoe-i-chem-otlichaetsja-ot-back-end/> (дата обращения: 10.04.2021).
2. Что такое нативное мобильное приложение? Отличие натива от гибрида [Электронный ресурс]. URL: <https://artjoker.ua/ru/blog/chto-takoe-nativnoe-mobilnoe-prilozhenie-otlichie-nativa-ot-gibrida/> (дата обращения: 16.04.2021).
3. Типы мобильных приложений [Электронный ресурс]. URL: <https://qastart.by/mainterms/59-tipy-mobilnykh-prilozhenij> (дата обращения: 19.04.2021).
4. Нативные vs. гибридные приложения [Электронный ресурс]. URL: <https://umbrellait.com/ru/blog/native-vs-hybrid-app/> (дата обращения: 19.04.2021).
5. Мобильная кухня: лучшие кулинарные приложения [Электронный ресурс]. URL: <https://style.rbc.ru/impressions/571791519a7947760a920d9e> (дата обращения: 20.04.2021).
6. Исследование рынка мобильных приложений в России и в мире [Электронный ресурс]. URL: <https://www.sfx-tula.ru/news/infoblog/9574/> (дата обращения: 27.04.2021).
7. Model-View-Presenter – компромисс и универсальный рецепт [Электронный ресурс]. URL: <https://habr.com/ru/post/343438/> (дата обращения: 03.05.2021).
8. О системы контроля версий [Электронный ресурс]. URL: <https://git-scm.com/book/ru/v2/Введение-О-системе-контроля-версий> (дата обращения: 11.05.2021).
9. SQLite – замечательная встраиваемая БД (часть 1) [Электронный ресурс]. URL: <https://habr.com/ru/post/149356/> (дата обращения: 19.05.2021).

10. Активность и её жизненный цикл [Электронный ресурс]. URL: <https://android-tools.ru/coding/aktivnost-i-eyo-zhiznennyj-cikl/> (дата обращения: 28.05.2021).

11. Как сделать приложение для Android самостоятельно [Электронный ресурс]. URL: https://skillbox.ru/media/code/kak_sdelat_prilozhenie_dlya_android/ (дата обращения: 02.06.2021).

12. Особенности тестирования «черного ящика» [Электронный ресурс]. URL: <https://quality-lab.ru/blog/key-principles-of-black-box-testing/> (дата обращения: 09.06.2021).

Приложение А
ЛИСТИНГ ПРОГРАММЫ НА ЯЗЫКЕ JAVA
(рекомендуемое)

Листинг А.1 – Код класса DatabaseAdapter

```

package com.cul.omnomnom.adapters;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;

import com.cul.omnomnom.dao.RecipeDAO;
import com.cul.omnomnom.dao.SQLiteDatabaseHelper;
import com.cul.omnomnom.dao.UserDAO;
import com.cul.omnomnom.models.Recipe;
import com.cul.omnomnom.models.User;
import com.cul.omnomnom.utils.UserPreferences;

import java.util.List;

public class DatabaseAdapter {

    private static DatabaseAdapter instance;
    private static final String DATABASE_NAME = "recipesDatabase";
    private static final int DATABASE_VERSION = 1;

    private SQLiteDatabaseHelper dbHelper;
    private SQLiteDatabase db;
    private Context mContext;

    private UserDAO userDAO;
    private RecipeDAO recipeDAO;

    public DatabaseAdapter(Context context, SQLiteDatabaseHelper dbHelper,
        SQLiteDatabase db) {
        this.db = db;
        this.dbHelper = dbHelper;
    }

    public static DatabaseAdapter getInstance(Context context) {
        if (instance == null) {
            synchronized (DatabaseAdapter.class) {
                if (instance == null)
                    instance = new DatabaseAdapter(context).open();
            }
        }
        return instance;
    }

    private DatabaseAdapter(Context context) {
        mContext = context;
        dbHelper = new SQLiteDatabaseHelper(context, DATABASE_NAME,
        DATABASE_VERSION);
    }

    private DatabaseAdapter open() {
        db = dbHelper.getWritableDatabase();
        userDAO = new UserDAO(db);
        recipeDAO = new RecipeDAO(db);
        return this;
    }

    public boolean signIn(String email, String password) {
        User currentUser = userDAO.getUserByEmailAndPassword(email,
        password);
        UserPreferences.saveCurrentUser(mContext, currentUser);
        return currentUser != null;
    }
}

```

```

public void addNewUser(User user) {
    userDao.insert(user);
}

public long addNewRecipe(Recipe recipe) {
    return recipeDAO.insert(recipe);
}

public void updateRecipe(Recipe recipe) {
    recipeDAO.update(recipe);
}

public void deleteRecipe(long recipeId) {
    recipeDAO.deleteById(recipeId);
}

public List<Recipe> getAllRecipesByCategory(String category) {
    return recipeDAO.selectAllByCategory(category);
}
}

```

Листинг А.2 – Код класса DirectionAdapter

```

package com.cul.omnomnom.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.models.Direction;

import org.jetbrains.annotations.NotNull;

import java.util.List;

public class DirectionAdapter extends
RecyclerView.Adapter<DirectionAdapter.DirectionViewHolder> {

    private List<Direction> directionList;
    private boolean isEditable = true;
    private Context mContext;
    private DirectionListener directionListener;

    public DirectionAdapter(Context context, List<Direction>
directionList) {
        mContext = context;
        this.directionList = directionList;
    }

    public DirectionAdapter(Context context, List<Direction>
directionList, boolean isEditable) {
        mContext = context;
        this.directionList = directionList;
        this.isEditable = isEditable;
    }

    @Override
    public int getItemViewType(int position) {

```

```

        return isEditable ? 0 : 1;
    }

    @NotNull
    @Override
    public DirectionViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View v = LayoutInflater.from(parent.getContext())
            .inflate(viewType == 0 ? R.layout.direction_item_row :
R.layout.direction_item_row_non_editable,
                parent, false);
        return new DirectionViewHolder(v);
    }

    @Override
    public void onBindViewHolder(final DirectionViewHolder holder, int
position) {
        Direction direction = directionList.get(position);
        holder.bind(direction);
    }

    @Override
    public int getItemCount() {
        return directionList.size();
    }

    public class DirectionViewHolder extends RecyclerView.ViewHolder {

        TextView directionText;
        ImageView wasteBin;

        public DirectionViewHolder(View itemView) {
            super(itemView);

            directionText = itemView.findViewById(R.id.directionText);
            if (isEditable) {
                wasteBin = itemView.findViewById(R.id.wasteBin);
                wasteBin.setOnClickListener(v -> {
                    if (directionListener != null)
                        directionListener.onDeleteDirection(getAdapterPosi
tion());
                });
            }
        }

        public void bind(Direction direction) {
            directionText.setText(direction.getBody());
        }
    }

    public void setDirectionListener(DirectionListener directionListener)
{
        this.directionListener = directionListener;
    }

    public interface DirectionListener {
        void onDeleteDirection(int position);
    }
}

```

Листинг А.3 – Код класса IngredientAdapter

```
package com.cul.omnomnom.adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.models.Ingredient;

import org.jetbrains.annotations.NotNull;

import java.util.List;

public class IngredientAdapter extends
RecyclerView.Adapter<IngredientAdapter.IngredientViewHolder> {

    private List<Ingredient> ingredientList;
    private boolean isEditable = true;
    private Context mContext;
    private IngredientListener ingredientListener;

    public IngredientAdapter(Context context, List<Ingredient>
ingredientList) {
        mContext = context;
        this.ingredientList = ingredientList;
    }

    public IngredientAdapter(Context context, List<Ingredient>
ingredientList, boolean isEditable) {
        mContext = context;
        this.ingredientList = ingredientList;
        this.isEditable = isEditable;
    }
}
```

```

@Override
public int getItemViewType(int position) {
    return isEditable ? 0 : 1;
}

@NotNull
@Override
public IngredientViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
    View v = LayoutInflater.from(parent.getContext())
        .inflate(viewType == 0 ? R.layout.ingredient_item_row :
R.layout.ingredient_item_row_non_editable,
        parent, false);
    return new IngredientViewHolder(v);
}

@Override
public void onBindViewHolder(final IngredientViewHolder holder, int
position) {
    Ingredient ingredient = ingredientList.get(position);
    holder.bind(ingredient);
}

@Override
public int getItemCount() {
    return ingredientList.size();
}

public class IngredientViewHolder extends RecyclerView.ViewHolder {

    TextView ingredientText;
    ImageView wasteBin;

    public IngredientViewHolder(View itemView) {
        super(itemView);

        ingredientText = itemView.findViewById(R.id.ingredientText);
        if (isEditable) {
            wasteBin = itemView.findViewById(R.id.wasteBin);
            wasteBin.setOnClickListener(v -> {
                if (ingredientListener != null)
                    ingredientListener.onDeleteIngredient(getAdapterPo
sition()));
            }
        }
    }
}

```

```

        });
    }
}

public void bind(Ingredient ingredient) {
    ingredientText.setText(ingredient.getName());
}

}

        public void setIngredientListener(IngredientListener
ingredientListener) {
    this.ingredientListener = ingredientListener;
}

public interface IngredientListener {
    void onDeleteIngredient(int position);
}
}

```

Листинг А.4 – Код класса MainPagerAdapter

```

package com.cul.omnomnom.adapters;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentStatePagerAdapter;

import com.cul.omnomnom.ui.main.fragments.DrinksFragment;
import com.cul.omnomnom.ui.main.fragments.SweetsFragment;
import com.cul.omnomnom.ui.main.fragments.HotFragment;
import com.cul.omnomnom.ui.main.fragments.ColdFragment;
import com.cul.omnomnom.ui.main.fragments.VeganFragment;

import org.jetbrains.annotations.NotNull;

public class MainPagerAdapter extends FragmentStatePagerAdapter {

    private static final int TAB_COUNT = 5;
    private String[] tabTitles = {"Первые блюда", "Вторые блюда", "Закуски
и салаты", "Сладкое и выпечка", "Напитки"};

    public MainPagerAdapter(FragmentManager fm) {

```

```

        super(fm);
    }

    @NotNull
    @Override
    public Fragment getItem(int position) {

        Fragment frag = null;
        switch (position) {
            case 0:
                frag = DrinksFragment.newInstance(tabTitles[position]);
                break;
            case 1:
                frag = VeganFragment.newInstance(tabTitles[position]);
                break;
            case 2:
                frag = SweetsFragment.newInstance(tabTitles[position]);
                break;
            case 3:
                frag = ColdFragment.newInstance(tabTitles[position]);
                break;
            case 4:
                frag = HotFragment.newInstance(tabTitles[position]);
                break;
        }
        return frag;
    }

    @Override
    public int getCount() {
        return TAB_COUNT;
    }

    @Override
    public int getItemPosition(@NotNull Object object) {
        return POSITION_NONE;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        return tabTitles[position];
    }
}

```

Листинг А.5 – Код класса RecipeAdapter

```

package com.cul.omnomnom.adapters;

import android.content.Context;
import android.net.Uri;
import android.util.Pair;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.widget.PopupMenu;
import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.models.Recipe;

import org.jetbrains.annotations.NotNull;

import java.io.File;
import java.util.List;

public class RecipeAdapter extends
RecyclerView.Adapter<RecipeAdapter.RecipeViewHolder> {
    private List<Recipe> recipeList;
    private Context mContext;
    private RecipeListener recipeListener;

    public RecipeAdapter(Context context, List<Recipe> recipeList) {
        mContext = context;
        this.recipeList = recipeList;
    }

    @NotNull
    @Override
    public RecipeViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {
        View v = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.recipe_item_row, parent, false);

```

```

        return new RecipeViewHolder(v);
    }

    @Override
    public void onBindViewHolder(final RecipeViewHolder holder, int
position) {
        Recipe recipe = recipeList.get(position);
        holder.bind(recipe);
    }

    @Override
    public int getItemCount() {
        return recipeList.size();
    }

    public class RecipeViewHolder extends RecyclerView.ViewHolder {

        TextView titleLabel;
        ImageView thumbnail;
        ImageView overflow;

        public RecipeViewHolder(View itemView) {
            super(itemView);

            titleLabel = itemView.findViewById(R.id.titleLabel);
            thumbnail = itemView.findViewById(R.id.thumbnail);
            overflow = itemView.findViewById(R.id.overflow);

            itemView.setOnClickListener(v -> {
                if (recipeListener != null)
                    recipeListener.onShowRecipe(recipeList.get(getAdapterP
osition()), new Pair[]{
                        Pair.create(thumbnail, "image_shared")
                    });
            });
            overflow.setOnClickListener(this::showPopupMenu);
        }

        public void bind(Recipe recipe) {
            titleLabel.setText(recipe.getName());
            thumbnail.setImageURI(Uri.fromFile(new
File(recipe.getImagePath())));
        }
    }

```

```

private void showPopupMenu(View view) {
    PopupMenu popup = new PopupMenu(view.getContext(), view);
    popup.inflate(R.menu.grid_popup_menu);
    popup.setOnMenuItemClickListener(
new
MyMenuItemClickListener(recipeList.get(getAdapterPosition())));
    popup.show();
}

class MyMenuItemClickListener implements
PopupMenu.OnMenuItemClickListener {

    private Recipe recipe;

    public MyMenuItemClickListener(Recipe recipe) {
        this.recipe = recipe;
    }

    @Override
    public boolean onMenuItemClick(MenuItem menuItem) {
        switch (menuItem.getItemId()) {
            case R.id.action_edit_recipe:
                if (recipeListener != null)
                    recipeListener.onEditRecipe(recipe);
                break;
            case R.id.action_delete_recipe:
                if (recipeListener != null)
                    recipeListener.onDeleteRecipe(recipe.getId());
                return true;
            default:
        }
        return false;
    }
}

public void setRecipeListener(RecipeListener recipeListener) {
    this.recipeListener = recipeListener;
}

public interface RecipeListener {
    void onShowRecipe(Recipe recipe, Pair<View, String>[] pairs);
}

```

```

        void onEditRecipe(Recipe recipe);
        void onDeleteRecipe(long recipeId);
    }
}

```

Листинг А.6 – Код класса ViewPagerAdapter

```

package com.cul.omnomnom.adapters;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentStatePagerAdapter;

import com.cul.omnomnom.models.Recipe;
import com.cul.omnomnom.ui.main.fragments.ViewDirectionsFragment;
import com.cul.omnomnom.ui.main.fragments.ViewIngredientsFragment;

public class ViewPagerAdapter extends FragmentStatePagerAdapter {

    private static final int TAB_COUNT = 2;
    private final Recipe recipe;
    private String[] tabTitles = {"Ingredients", "Directions"};

    public ViewPagerAdapter(FragmentManager fm, Recipe recipe) {
        super(fm);
        this.recipe = recipe;
    }

    @Override
    public Fragment getItem(int position) {
        Fragment frag = null;
        switch (position) {
            case 0:
                frag =
ViewIngredientsFragment.newInstance(recipe.getIngredients());
                break;
            case 1:
                frag =
ViewDirectionsFragment.newInstance(recipe.getDirections());
                break;
        }
        return frag;
    }
}

```

```

@Override
public int getCount() {
    return TAB_COUNT;
}

@Override
public CharSequence getPageTitle(int position) {
    return tabTitles[position];
}
}

```

Листинг А.7 – Код класса DirectionDAO

```

package com.cul.omnomnom.dao;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

import com.cul.omnomnom.models.Direction;

import java.util.ArrayList;
import java.util.List;

public class DirectionDAO {

    private SQLiteDatabase db;

    public DirectionDAO(SQLiteDatabase db) {
        this.db = db;
    }

    public void insert(Direction direction) {
        insert(direction.getBody(), direction.getRecipeId());
    }

    public void insert(String direction, long recipeId) {
        ContentValues values = new ContentValues();
        values.put(Config.KEY_BODY, direction);
        values.put(Config.KEY_RECIPE_ID, recipeId);
        db.insert(Config.TABLE_NAME, null, values);
    }
}

```

```

public List<Direction> selectAllByRecipeId(long recipeId) {
    List<Direction> directions = new ArrayList<>();
    try (Cursor cursor = db.query(Config.TABLE_NAME,
        new String[]{Config.KEY_ID, Config.KEY_BODY,
Config.KEY_RECIPE_ID},
        Config.KEY_RECIPE_ID + " = ?", new String[]{recipeId +
""}, null, null, null)) {
        if (cursor.moveToFirst()) {
            do {
                directions.add(new Direction(
                    cursor.getLong(0), cursor.getString(1),
cursor.getLong(2)));
            } while (cursor.moveToNext());
        }
    }

    Log.i("DAO", "DirectionDAO returning: " + directions + " for
recipe ID: " + recipeId);
    return directions;
}

public boolean deleteAllByRecipeId(long recipeId) {
    return db.delete(Config.TABLE_NAME, Config.KEY_RECIPE_ID + "=" +
recipeId, null) > 0;
}

public static class Config {
    public static final String TABLE_NAME = "directions";
    public static final String KEY_ID = "id";
    public static final String KEY_BODY = "body";
    public static final String KEY_RECIPE_ID = "recipeId";

    public static final String CREATE_TABLE_STATEMENT =
        "CREATE TABLE " + TABLE_NAME + " (" +
            KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            KEY_BODY + " TEXT NOT NULL, " +
            KEY_RECIPE_ID + " TEXT NOT NULL, " +
            "FOREIGN KEY(" + KEY_RECIPE_ID + ") REFERENCES " +
RecipeDAO.Config.TABLE_NAME + "(" + RecipeDAO.Config.KEY_ID + ")");
}
}

```

Листинг А.8 – Код класса IngredientDAO

```
package com.cul.omnomnom.dao;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

import com.cul.omnomnom.models.Ingredient;

import java.util.ArrayList;
import java.util.List;

public class IngredientDAO {

    private SQLiteDatabase db;

    public IngredientDAO(SQLiteDatabase db) {
        this.db = db;
    }

    public void insert(Ingredient ingredient) {
        insert(ingredient.getName(), ingredient.getRecipeId());
    }

    public void insert(String ingredient, long recipeId) {
        ContentValues values = new ContentValues();
        values.put(Config.KEY_NAME, ingredient);
        values.put(Config.KEY_RECIPE_ID, recipeId);
        db.insert(Config.TABLE_NAME, null, values);
    }

    public List<Ingredient> selectAllByRecipeId(long recipeId) {
        List<Ingredient> ingredients = new ArrayList<>();
        try (Cursor cursor = db.query(Config.TABLE_NAME,
                                     new String[]{Config.KEY_ID, Config.KEY_NAME,
Config.KEY_RECIPE_ID},
                                     Config.KEY_RECIPE_ID + " = ?", new String[]{recipeId +
""}, null, null, null)) {
            if (cursor.moveToFirst()) {
```

```

        do {
            ingredients.add(new Ingredient(
                cursor.getLong(0), cursor.getString(1),
cursor.getLong(2)));
        } while (cursor.moveToNext());
    }
}

    Log.i("DAO", "IngredientDAO returning: " + ingredients + " for
recipe ID: " + recipeId);
    return ingredients;
}

    public boolean deleteAllByRecipeId(long recipeId) {
        return db.delete(Config.TABLE_NAME, Config.KEY_RECIPE_ID + "=" +
recipeId, null) > 0;
    }

    public static class Config {
        public static final String TABLE_NAME = "ingredients";
        public static final String KEY_ID = "id";
        public static final String KEY_NAME = "name";
        public static final String KEY_RECIPE_ID = "recipeId";

        public static final String CREATE_TABLE_STATEMENT =
            "CREATE TABLE " + TABLE_NAME + " (" +
                KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                KEY_NAME + " TEXT, " +
                KEY_RECIPE_ID + " TEXT, " +
                "FOREIGN KEY(" + KEY_RECIPE_ID + ") REFERENCES " +
RecipeDAO.Config.TABLE_NAME + "(" + RecipeDAO.Config.KEY_ID + ")";
    }
}

```

Листинг А.9 – Код класса RecipeDAO

```
package com.cul.omnomnom.dao;
```

```
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
```

```
import com.cul.omnomnom.models.Recipe;
```

```

import org.jetbrains.annotations.NotNull;

import java.util.ArrayList;
import java.util.List;

public class RecipeDAO {

    private SQLiteDatabase db;

    private IngredientDAO ingredientDAO;
    private DirectionDAO directionDAO;

    public RecipeDAO(SQLiteDatabase db) {
        this.db = db;

        ingredientDAO = new IngredientDAO(db);
        directionDAO = new DirectionDAO(db);
    }

    public long insert(Recipe recipe) {

        long newRecipeId = insert(recipe.getName(), recipe.getCategory(),
recipe.getDescription(), recipe.getImagePath());

        recipe.getIngredients()
            .forEach(ingredient -> {
                ingredient.setRecipeId(newRecipeId);
                ingredientDAO.insert(ingredient);
                Log.i("DAO", "Inserted " + ingredient);
            });
        recipe.getDirections()
            .forEach(direction -> {
                direction.setRecipeId(newRecipeId);
                directionDAO.insert(direction);
                Log.i("DAO", "Inserted " + direction);
            });

        return newRecipeId;
    }

    @NotNull

```

```

        private long insert(String name, String category, String description,
String imagePath) {
            ContentValues values = new ContentValues();
            values.put(Config.KEY_NAME, name);
            values.put(Config.KEY_CATEGORY, category);
            values.put(Config.KEY_DESCRIPTION, description);
            values.put(Config.KEY_IMAGE_PATH, imagePath);
            return db.insert(Config.TABLE_NAME, null, values);
        }

        public List<Recipe> selectAllByCategory(String category) {
            List<Recipe> recipes = new ArrayList<>();
            try (Cursor cursor = db.query(Config.TABLE_NAME,
                new String[]{Config.KEY_ID, Config.KEY_NAME,
Config.KEY_CATEGORY, Config.KEY_DESCRIPTION, Config.KEY_IMAGE_PATH},
                Config.KEY_CATEGORY + " = ?", new String[]{category},
null, null, null)) {
                if (cursor.moveToFirst()) {
                    do {
                        Recipe recipe = new Recipe(
                            cursor.getLong(0),
                            cursor.getString(1),
                            cursor.getString(2),
                            cursor.getString(3),
                            cursor.getString(4));
                        recipe.setIngredients(ingredientDAO.selectAllByRecipeI
d(recipe.getId()));
                        recipe.setDirections(directionDAO.selectAllByRecipeId(
recipe.getId()));
                        recipes.add(recipe);

                    } while (cursor.moveToNext());
                }
            }

            Log.i("DAO", "RecipeDAO returning: " + recipes);
            return recipes;
        }

        public void update(Recipe recipe) {
            ingredientDAO.deleteAllByRecipeId(recipe.getId());
            directionDAO.deleteAllByRecipeId(recipe.getId());
            recipe.getIngredients()

```

```

        .forEach(ingredient -> {
            ingredient.setRecipeId(recipe.getId());
            ingredientDAO.insert(ingredient);
        });
recipe.getDirections()
    .forEach(direction -> {
        direction.setRecipeId(recipe.getId());
        directionDAO.insert(direction);
    });

ContentValues values = new ContentValues();
values.put(Config.KEY_NAME, recipe.getName());
values.put(Config.KEY_CATEGORY, recipe.getCategory());
values.put(Config.KEY_DESCRIPTION, recipe.getDescription());
values.put(Config.KEY_IMAGE_PATH, recipe.getImagePath()==null ||
recipe.getImagePath().isEmpty() ? "" : recipe.getImagePath());
int update = db.update(Config.TABLE_NAME, values, Config.KEY_ID +
"=" + recipe.getId(), null);
    }

public boolean deleteById(long id) {
    ingredientDAO.deleteAllByRecipeId(id);
    directionDAO.deleteAllByRecipeId(id);
    return db.delete(Config.TABLE_NAME, Config.KEY_ID + "=" + id,
null) > 0;
}

public static class Config {
    public static final String TABLE_NAME = "recipes";
    public static final String KEY_ID = "id";
    public static final String KEY_NAME = "name";
    public static final String KEY_CATEGORY = "category";
    public static final String KEY_DESCRIPTION = "description";
    public static final String KEY_IMAGE_PATH = "imagePath";

    public static final String CREATE_TABLE_STATEMENT =
        "CREATE TABLE " + TABLE_NAME + " (" +
            KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            KEY_NAME + " TEXT, " +
            KEY_CATEGORY + " TEXT, " +
            KEY_DESCRIPTION + " TEXT, " +
            KEY_IMAGE_PATH + " TEXT)";
}

```

```
}
```

Листинг А.10 – Код класса SQLiteDatabaseHelper

```
package com.cul.omnomnom.dao;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class SQLiteDatabaseHelper extends SQLiteOpenHelper {

    private static final String TAG =
SQLiteDatabaseHelper.class.getSimpleName();

    public SQLiteDatabaseHelper(Context context, String dbName, int
databaseVersion) {
        super(context, dbName, null, databaseVersion);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(UserDAO.Config.CREATE_TABLE_STATEMENT);
        db.execSQL(RecipeDAO.Config.CREATE_TABLE_STATEMENT);
        db.execSQL(DirectionDAO.Config.CREATE_TABLE_STATEMENT);
        db.execSQL(IngredientDAO.Config.CREATE_TABLE_STATEMENT);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        Log.w(TAG, "Upgrading database from version " + oldVersion +
            " to " + newVersion + ", which will destroy all old
data.");
        db.execSQL("DROP TABLE IF EXISTS " + UserDAO.Config.TABLE_NAME);
        db.execSQL("DROP TABLE IF EXISTS " +
DirectionDAO.Config.TABLE_NAME);
        db.execSQL("DROP TABLE IF EXISTS " +
IngredientDAO.Config.TABLE_NAME);
        db.execSQL("DROP TABLE IF EXISTS " + RecipeDAO.Config.TABLE_NAME);
        onCreate(db);
    }
}
```

```
}
```

Листинг А.11 – Код класса UserDAO

```
package com.cul.omnomnom.dao;

import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;

import com.cul.omnomnom.models.User;

public class UserDAO {

    private SQLiteDatabase db;

    public UserDAO(SQLiteDatabase db) {
        this.db = db;
    }

    public void insert(User user) {
        ContentValues values = new ContentValues();
        values.put(Config.KEY_USERNAME, user.getUsername());
        values.put(Config.KEY_FULLNAME, user.getFullname());
        values.put(Config.KEY_EMAIL, user.getEmail());
        values.put(Config.KEY_PASSWORD, user.getPassword());

        insert(values);
    }

    private long insert(ContentValues values) {
        return db.insert(Config.TABLE_NAME, null, values);
    }

    public User getUserByEmailAndPassword(String email, String password) {
        try (Cursor cursor = db.query(
            true,
            Config.TABLE_NAME,
            new String[]{Config.KEY_ID, Config.KEY_USERNAME,
Config.KEY_FULLNAME, Config.KEY_EMAIL, Config.KEY_PASSWORD},
            Config.KEY_EMAIL + "=? AND " + Config.KEY_PASSWORD + "=?",
            new String[]{email, password},
            null, null, null, null)) {
            return extractUserFromCursor(cursor);
        }
    }
}
```

```

    }

    private User extractUserFromCursor(Cursor cursor) {
        if (cursor.moveToFirst()) {
            return new User(
                cursor.getInt(0),
                cursor.getString(1),
                cursor.getString(2),
                cursor.getString(3),
                cursor.getString(4)
            );
        }

        return null;
    }

    public static class Config {
        public static final String TABLE_NAME = "users";
        public static final String KEY_ID = "id";
        public static final String KEY_USERNAME = "username";
        public static final String KEY_FULLNAME = "fullname";
        public static final String KEY_EMAIL = "email";
        public static final String KEY_PASSWORD = "password";

        public static final String CREATE_TABLE_STATEMENT =
            "CREATE TABLE " + TABLE_NAME + " (" +
                KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                KEY_USERNAME + " TEXT NOT NULL, " +
                KEY_FULLNAME + " TEXT NOT NULL, " +
                KEY_EMAIL + " TEXT NOT NULL, " +
                KEY_PASSWORD + " TEXT NOT NULL)";
    }
}

```

Листинг А.12 – Код класса Direction

```

package com.cul.omnomnom.models;

import android.os.Parcel;
import android.os.Parcelable;

public class Direction implements Parcelable {
    private long id;
    private String body;
}

```

```
private long recipeId;

public Direction(String body) {
    this.body = body;
}

public Direction(long id, String body, long recipeId) {
    this.id = id;
    this.body = body;
    this.recipeId = recipeId;
}

private Direction(Parcel in) {
    id = in.readLong();
    body = in.readString();
    recipeId = in.readLong();
}

public long getId() {
    return id;
}

public String getBody() {
    return body;
}

public long getRecipeId() {
    return recipeId;
}

public void setId(long id) {
    this.id = id;
}

public void setBody(String body) {
    this.body = body;
}

public void setRecipeId(long recipeId) {
    this.recipeId = recipeId;
}

@Override
```

```

    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel parcel, int i) {
        parcel.writeLong(id);
        parcel.writeString(body);
        parcel.writeLong(recipeId);
    }

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

        @Override
        public Direction createFromParcel(Parcel source) {
            return new Direction(source);
        }

        @Override
        public Direction[] newArray(int size) {
            return new Direction[size];
        }
    };

    @Override
    public String toString() {
        return "Direction{" +
            "id=" + id +
            ", body='" + body + '\'' +
            ", recipeId=" + recipeId +
            '\'';
    }
}

```

Листинг А.13 – Код класса Ingredient

```

package com.cul.omnomnom.models;

import android.os.Parcel;
import android.os.Parcelable;

public class Ingredient implements Parcelable {
    private long id;

```

```
private String name;
private long recipeId;

public Ingredient(String name) {
    this.name = name;
}

public Ingredient(long id, String name, long recipeId) {
    this(name);
    this.id = id;
    this.recipeId = recipeId;
}

private Ingredient(Parcel in) {
    id = in.readLong();
    name = in.readString();
    recipeId = in.readLong();
}

public long getId() {
    return id;
}

public String getName() {
    return name;
}

public long getRecipeId() {
    return recipeId;
}

public void setId(long id) {
    this.id = id;
}

public void setName(String name) {
    this.name = name;
}

public void setRecipeId(long recipeId) {
    this.recipeId = recipeId;
}
```

```

@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel parcel, int i) {
    parcel.writeLong(id);
    parcel.writeString(name);
    parcel.writeLong(recipeId);
}

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

    @Override
    public Ingredient createFromParcel(Parcel source) {
        return new Ingredient(source);
    }

    @Override
    public Ingredient[] newArray(int size) {
        return new Ingredient[size];
    }
};

@Override
public String toString() {
    return "Ingredient{" +
        "id=" + id +
        ", name='" + name + '\'' +
        ", recipeId=" + recipeId +
        '}';
}
}

```

Листинг А.14 – Код класса Recipe

```

package com.cul.omnomnom.models;

import android.os.Parcel;
import android.os.Parcelable;

import org.jetbrains.annotations.NotNull;

```

```

import java.util.ArrayList;
import java.util.List;

public class Recipe implements Parcelable {
    private long id;
    private String name;
    private String category;
    private String description;
    private List<Ingredient> ingredients;
    private List<Direction> directions;
    private String imagePath;

    public Recipe() {
        ingredients = new ArrayList<>();
        directions = new ArrayList<>();
    }

    public Recipe(String category) {
        this();
        this.category = category;
    }

    public Recipe(String name, String category, String description, String
imagePath) {
        this(category);
        this.name = name;
        this.description = description;
        this.imagePath = imagePath;
    }

    public Recipe(long id, String name, String category, String
description, String imagePath) {
        this(name, category, description, imagePath);
        this.id = id;
    }

    public Recipe(String name, String category, String description,
                    List<Ingredient> ingredients, List<Direction>
directions, String imagePath) {
        this(name, category, description, imagePath);
        this.ingredients = ingredients;
        this.directions = directions;
    }

```

```

    }

    public Recipe(long id, String name, String category, String
description,
                    List<Ingredient> ingredients, List<Direction>
directions, String imagePath) {
        this(name, category, description, ingredients, directions,
imagePath);
        this.id = id;
    }

    private Recipe(Parcel in) {
        ingredients = new ArrayList<>();
        directions = new ArrayList<>();

        id = in.readLong();
        name = in.readString();
        category = in.readString();
        description = in.readString();
        in.readTypedList(ingredients, Ingredient.CREATOR);
        in.readTypedList(directions, Direction.CREATOR);
        imagePath = 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 long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

```

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getCategory() {
    return category;
}

public void setCategory(String category) {
    this.category = category;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public List<Ingredient> getIngredients() {
    return ingredients;
}

public void setIngredients(List<Ingredient> ingredients) {
    this.ingredients = ingredients;
}

public List<Direction> getDirections() {
    return directions;
}

public void setDirections(List<Direction> directions) {
    this.directions = directions;
}

public String getImagePath() {
    return imagePath;
}
```

```

    }

    public void setImagePath(String imagePath) {
        this.imagePath = imagePath;
    }

    @NotNull
    @Override
    public String toString() {
        return "Recipe{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", category='" + category + '\'' +
            ", description='" + description + '\'' +
            ", ingredients=" + ingredients +
            ", directions=" + directions +
            ", imagePath='" + imagePath + '\'' +
            '}';
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel parcel, int i) {
        parcel.writeLong(id);
        parcel.writeString(name);
        parcel.writeString(category);
        parcel.writeString(description);
        parcel.writeTypedList(ingredients);
        parcel.writeTypedList(directions);
        parcel.writeString(imagePath);
    }
}

```

Листинг А.15 – Код класса User

```

package com.cul.omnomnom.models;

public class User {
    private int id;
    private String username;

```

```
private String fullname;
private String email;
private String password;

    public User(String username, String fullname, String email, String
password) {
        this.username = username;
        this.fullname = fullname;
        this.email = email;
        this.password = password;
    }

    public User(int id, String username, String fullname, String email,
String password) {
        this(username, fullname, email, password);
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getFullname() {
        return fullname;
    }

    public void setFullname(String fullname) {
        this.fullname = fullname;
    }

    public String getEmail() {
```

```

        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}

```

Листинг А.16 – Код класса CategorizedFragment

```

package com.cul.omnomnom.ui.main.fragments;

import android.app.Activity;
import android.os.Bundle;
import android.util.Pair;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.LayoutRes;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.cul.omnomnom.adapters.RecipeAdapter;
import com.cul.omnomnom.models.Recipe;

import java.util.ArrayList;
import java.util.List;

```

```

public abstract class CategorizedFragment extends Fragment {

    private CategorizedFragmentListener categorizedFragmentListener;
    protected RecyclerView recipeRecyclerView;
    private TextView emptyView;
    protected RecipeAdapter recipeAdapter;
    protected DatabaseAdapter databaseAdapter;
    protected String currentCategory;
    protected List<Recipe> recipes;

    public CategorizedFragment() {
        // Required empty public constructor
        recipes = new ArrayList<>();
        databaseAdapter = DatabaseAdapter.getInstance(getActivity());
    }

    public static Fragment newInstance(String category) {
        Fragment fragment;
        switch (category) {
            case "DrinksFragment":
                fragment = new DrinksFragment();
                break;
            case "ColdFragment":
                fragment = new ColdFragment();
                break;
            case "HotFragment":
                fragment = new HotFragment();
                break;
            case "SweetsFragment":
                fragment = new SweetsFragment();
                break;
            default:
                fragment = new VeganFragment();
                break;
        }

        Bundle args = new Bundle();
        args.putString("category", category);
        fragment.setArguments(args);

        return fragment;
    }
}

```

```

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    try {
        categorizedFragmentListener = (CategorizedFragmentListener)
activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
            + " must implement CategorizedFragmentListener");
    }
}

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {

    View rootView = inflater.inflate(getFragmentLayout(), container,
false);

    Bundle args = getArguments();
    currentCategory = args.getString("category");

    recipeRecyclerView = rootView.findViewById(R.id.recyclerView);
    emptyView = rootView.findViewById(R.id.empty_view);
    recipeRecyclerView.setHasFixedSize(true);

                                recipeRecyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));

    return rootView;
}

@Override
public void onStart() {
    super.onStart();
    refresh();
}

@Override
public void onDetach() {
    super.onDetach();
    categorizedFragmentListener = null;
}

```

```

    }

    private void refresh() {

        recipes =
databaseAdapter.getAllRecipesByCategory(currentCategory);
        toggleEmptyView();
        recipeAdapter = new RecipeAdapter(getActivity(), recipes);
        recipeAdapter.setRecipeListener(new RecipeAdapter.RecipeListener()
{
            @Override
            public void onShowRecipe(Recipe recipe, Pair<View, String>[]
pairs) {

                categorizedFragmentManager.onShowRecipe(recipe, pairs);
            }

            @Override
            public void onEditRecipe(Recipe recipe) {
                categorizedFragmentManager.onEditRecipe(recipe);
            }

            @Override
            public void onDeleteRecipe(long recipeId) {
                categorizedFragmentManager.onDeleteRecipe(recipeId);
                refresh();
            }
        });
        recipeRecyclerView.setAdapter(recipeAdapter);
    }

    private void toggleEmptyView() {
        if (recipes.size() == 0) {
            emptyView.setVisibility(View.VISIBLE);
            recipeRecyclerView.setVisibility(View.GONE);
        } else {
            emptyView.setVisibility(View.GONE);
            recipeRecyclerView.setVisibility(View.VISIBLE);
        }
    }

    protected abstract @LayoutRes
    int getFragmentLayout();

    public interface CategorizedFragmentManager {

```

```

        void onShowRecipe(Recipe recipe, Pair<View, String>[] pairs);

        void onEditRecipe(Recipe recipe);

        void onDeleteRecipe(long recipeId);
    }
}

```

Листинг А.17 – Код класса ColdFragment

```

package com.cul.omnomnom.ui.main.fragments;
import androidx.fragment.app.Fragment;
import com.fuckyoushitbitch.omnomnom.R;

/**
 * A simple {@link Fragment} subclass.
 */
public class ColdFragment extends CategorizedFragment {
    @Override
    protected int getFragmentLayout() {
        return R.layout.fragment_cold;
    }
}

```

Листинг А.18 – Код класса DrinksFragment

```

package com.cul.omnomnom.ui.main.fragments;
import androidx.fragment.app.Fragment;
import com.cul.omnomnom.R;

/**
 * A simple {@link Fragment} subclass.
 */
public class DrinksFragment extends CategorizedFragment {
    @Override
    protected int getFragmentLayout() {
        return R.layout.fragment_drinks;
    }
}

```

Листинг А.19 – Код класса HotFragment

```

package com.cul.omnomnom.ui.main.fragments;
import androidx.fragment.app.Fragment;
import com.cul.omnomnom.R;

```

```

/**
 * A simple {@link Fragment} subclass.
 */
public class HotFragment extends CategorizedFragment {
    @Override
    protected int getFragmentLayout() {
        return R.layout.fragment_hot;
    }
}

```

Листинг А.20 – Код класса NavigableFragment

```

package com.cul.omnomnom.ui.main.fragments;

import androidx.fragment.app.Fragment;

public abstract class NavigableFragment extends Fragment {
    public abstract void onNext();
}

```

Листинг А.21 – Код класса RecipeDirectionsFragment

```

package com.cul.omnomnom.ui.main.fragments;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.cul.omnomnom.adapters.DirectionAdapter;
import com.cul.omnomnom.models.Direction;
import com.cul.omnomnom.models.Recipe;

```

```

import java.util.ArrayList;
import java.util.List;

/**
 * A simple {@link Fragment} subclass.
 */
public class RecipeDirectionsFragment extends NavigableFragment {
    private DirectionsListener mListener;
    private List<Direction> directionList;
    private DirectionAdapter directionAdapter;
    private DatabaseAdapter databaseAdapter;

    private RecyclerView directionRecyclerView;
    private TextView emptyView;
    private Button addButton;
    private EditText directionField;

    public RecipeDirectionsFragment() {
        // Required empty public constructor
    }

    public static RecipeDirectionsFragment newInstance(Recipe recipe) {
        RecipeDirectionsFragment fragment = new
RecipeDirectionsFragment();

        if (recipe != null) {
            Bundle args = new Bundle();
            args.putParcelableArrayList("directions",
(ArrayList<Direction>) recipe.getDirections());
            fragment.setArguments(args);
        }

        return fragment;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_recipe_directions,
container, false);
        databaseAdapter = DatabaseAdapter.getInstance(getActivity());

```

```

Bundle args = getArguments();
if (args != null)
    directionList = args.getParcelableArrayList("directions");
if (directionList == null)
    directionList = new ArrayList<>();

directionRecyclerView = view.findViewById(R.id.recyclerView);
emptyView = view.findViewById(R.id.empty_view);
addButton = view.findViewById(R.id.add_button);
directionField = view.findViewById(R.id.directionField);
    directionAdapter = new DirectionAdapter(getActivity(),
directionList);
    directionAdapter.setDirectionListener(position -> {
        directionList.remove(position);
        toggleEmptyView();
        directionAdapter.notifyDataSetChanged();
    });

toggleEmptyView();

directionRecyclerView.setHasFixedSize(true);
                                directionRecyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));
    directionRecyclerView.setAdapter(directionAdapter);

addButton.setOnClickListener(v -> {
    Log.i("DAO", "Add button pressed.");
    String newDirection = directionField.getText().toString();
    Log.i("DAO", "New direction: " + newDirection);
    if (!newDirection.isEmpty()) {
        Log.i("DAO", "Directions list BEFORE: " + directionList);
        directionField.setText("");
        directionList.add(new Direction(newDirection));
        Log.i("DAO", "Directions list AFTER: " + directionList);
        toggleEmptyView();
        directionAdapter.notifyDataSetChanged();
    }
});

return view;
}

```

```

private void toggleEmptyView() {
    if (directionList.size() == 0) {
        emptyView.setVisibility(View.VISIBLE);
        directionRecyclerView.setVisibility(View.GONE);
    } else {
        emptyView.setVisibility(View.GONE);
        directionRecyclerView.setVisibility(View.VISIBLE);
    }
}

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    try {
        mListener = (DirectionsListener) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
            + " must implement DirectionsListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
}

@Override
public void onNext() {
    if (mListener != null) {
        Log.i("DAO", "Steps finished: " + directionList);
        mListener.onStepsFinished(directionList);
    }
}

public interface DirectionsListener {
    void onStepsFinished(List<Direction> directions);
}
}

```

Листинг А.22 – Код класса RecipeImageFragment

```
package com.cul.omnomnom.ui.main.fragments;
```

```

import android.app.Activity;
import android.content.Context;
import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.fragment.app.Fragment;

import com.cul.omnomnom.R;
import com.cul.omnomnom.models.Recipe;

import java.io.File;

/**
 * A simple {@link Fragment} subclass.
 */
public class RecipeImageFragment extends NavigableFragment {
    private static final int MAX_DESCRIPTION_LENGTH = 200;
    private String currentRecipeImage;
    private ImageListener mListener;

    private ImageView recipeImage;
    private Button selectImageBtn;
    private EditText recipeName;
    private EditText recipeDescription;
    private Recipe recipe;
    private Context context;

    public static RecipeImageFragment newInstance(Recipe recipe) {
        RecipeImageFragment fragment = new RecipeImageFragment();

        Bundle args = new Bundle();
        if (recipe.getDescription() != null) {
            args.putString("imagePath", recipe.getImagePath());
            args.putString("description", recipe.getDescription());
            args.putString("name", recipe.getName());
            fragment.setArguments(args);

```

```

    }

    return fragment;
}

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    try {
        mListener = (ImageListener) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
            + " must implement ImageListener");
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_recipe_image,
container, false);

    recipeImage = view.findViewById(R.id.recipe_image);
    selectImageBtn = view.findViewById(R.id.choose_image);
    recipeDescription = view.findViewById(R.id.recipe_description);
    recipeName = view.findViewById(R.id.recipe_name);

    Bundle args = getArguments();
    if (args != null) {
        String imagePath = args.getString("imagePath");
        String description = args.getString("description");
        String name = args.getString("name");
        onImageSelected(imagePath);
        recipeDescription.setText(description);
        recipeName.setText(name);
    }

    selectImageBtn.setOnClickListener(v -> {
        if (mListener != null)
            mListener.onSelectImage();
    });

    return view;
}

```

```

    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    @Override
    public void onNext() {
        if (currentRecipeImage == null) {
            Toast.makeText(getActivity(), "Выберете изображение",
Toast.LENGTH_LONG).show();
            return;
        }

        String name = recipeName.getText().toString();
        String description = recipeDescription.getText().toString();

        if (name.isEmpty()) {
            Toast.makeText(getActivity(), "Укажите наименование рецепта",
Toast.LENGTH_LONG).show();
            return;
        }

        if (description.isEmpty()) {
            Toast.makeText(getActivity(), "Дайте краткое описание
рецепта", Toast.LENGTH_LONG).show();
            return;
        } else {
            if (description.length() > 200) {
                Toast.makeText(getActivity(), "Количество символов в
поле \"описание\" превышает максимально возможную длину " +
MAX_DESCRIPTION_LENGTH + " characters.", Toast.LENGTH_LONG).show();
                return;
            }
        }

        if (mListener != null)
            mListener.navigateToIngredientsFragment(name, description);
    }

```

```

        public void onImageSelected(String imagePath) {
            currentRecipeImage = imagePath;
            if (!currentRecipeImage.isEmpty()) {
                recipeImage.setImageURI(Uri.fromFile(new
File(currentRecipeImage)));
                selectImageBtn.setText("Обновить изображение");
                //                TODO WHY CRASH???
                //                Drawable attachmentIcon =
this.context.getDrawable(R.drawable.attachment24px);
                //                Resources r = context.getResources();
                //                int resourceId =
this.getResources().getIdentifier("attachment24px", "android:drawable"
,
context.getPackageName());
                //                PointerIcon p = PointerIcon.load(r, resourceId);
                //                selectImageBtn.setPointerIcon(p);

            }
        }

        public interface ImageListener {
            void onSelectImage();
            void navigateToIngredientsFragment(String name, String
description);
        }
    }

```

Листинг А.23 – Код класса RecipeIngredientsFragment

```
package com.cul.omnomnom.ui.main.fragments;
```

```

import android.app.Activity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;

```

```

import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.cul.omnomnom.adapters.IngredientAdapter;
import com.cul.omnomnom.models.Ingredient;
import com.cul.omnomnom.models.Recipe;

import java.util.ArrayList;
import java.util.List;

public class RecipeIngredientsFragment extends NavigableFragment {
    private IngredientListener mListener;
    private List<Ingredient> ingredientList;
    private IngredientAdapter ingredientAdapter;

    private RecyclerView ingredientRecyclerView;
    private TextView emptyView;
    private Button addButton;
    private EditText ingredientField;

    public RecipeIngredientsFragment() {
        // Required empty public constructor
    }

    public static RecipeIngredientsFragment newInstance(Recipe recipe) {
        RecipeIngredientsFragment fragment = new
RecipeIngredientsFragment();

        if (recipe.getIngredients() != null) {
            Bundle args = new Bundle();
            args.putParcelableArrayList("ingredients",
(ArrayList<Ingredient>) recipe.getIngredients());
            fragment.setArguments(args);
        }

        return fragment;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_recipe_ingredients,
container, false);

```

```

DatabaseAdapter databaseAdapter =
DatabaseAdapter.getInstance(getActivity());

Bundle args = getArguments();
if (args != null)
    ingredientList = args.getParcelableArrayList("ingredients");
if (ingredientList == null)
    ingredientList = new ArrayList<>();

ingredientRecyclerView = view.findViewById(R.id.recyclerView);
emptyView = view.findViewById(R.id.empty_view);
addButton = view.findViewById(R.id.add_button);
ingredientField = view.findViewById(R.id.ingredientField);
    ingredientAdapter = new IngredientAdapter(getActivity(),
ingredientList);
    ingredientAdapter.setIngredientListener(position -> {
        ingredientList.remove(position);
        toggleEmptyView();
        ingredientAdapter.notifyDataSetChanged();
    });

toggleEmptyView();

ingredientRecyclerView.setHasFixedSize(true);
    ingredientRecyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));
    ingredientRecyclerView.setAdapter(ingredientAdapter);

addButton.setOnClickListener(v -> {
    String newIngredient = ingredientField.getText().toString();
    if (!newIngredient.isEmpty()) {
        ingredientField.setText("");
        ingredientList.add(new Ingredient(newIngredient));
        toggleEmptyView();
        ingredientAdapter.notifyDataSetChanged();
    }
});

return view;
}

private void toggleEmptyView() {
    if (ingredientList.size() == 0) {

```

```

        emptyView.setVisibility(View.VISIBLE);
        ingredientRecyclerView.setVisibility(View.GONE);
    } else {
        emptyView.setVisibility(View.GONE);
        ingredientRecyclerView.setVisibility(View.VISIBLE);
    }
}

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    try {
        mListener = (IngredientListener) activity;
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()
            + " must implement IngredientListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
}

@Override
public void onNext() {
    if (mListener != null)
        mListener.navigateToDirectionsFragment(ingredientList);
}

public interface IngredientListener {
    void navigateToDirectionsFragment(List<Ingredient> ingredients);
}
}

```

Листинг А.24 – Код класса SweetsFragment

```

package com.cul.omnomnom.ui.main.fragments;
import androidx.fragment.app.Fragment;
import com.cul.omnomnom.R;

/**
 * A simple {@link Fragment} subclass.

```

```

*/
public class SweetsFragment extends CategorizedFragment {
    @Override
    protected int getFragmentLayout() {
        return R.layout.fragment_sweets;
    }
}

```

Листинг А.25 – Код класса VeganFragment

```

package com.cul.omnomnom.ui.main.fragments;
import androidx.fragment.app.Fragment;
import com.cul.omnomnom.R;

/**
 * A simple {@link Fragment} subclass.
 */
public class VeganFragment extends CategorizedFragment {
    @Override
    protected int getFragmentLayout() {
        return R.layout.fragment_vegan;
    }
}

```

Листинг А.26 – Код класса ViewDirectionsFragment

```

package com.cul.omnomnom.ui.main.fragments;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DirectionAdapter;
import com.cul.omnomnom.models.Direction;

import java.util.ArrayList;
import java.util.List;

public class ViewDirectionsFragment extends Fragment {

```

```

private List<Direction> directionList;
private DirectionAdapter directionAdapter;

private RecyclerView directionRecyclerView;
private TextView emptyView;

public ViewDirectionsFragment() {
    // Required empty public constructor
}

public static ViewDirectionsFragment newInstance(List<Direction>
directions) {
    ViewDirectionsFragment fragment = new
ViewDirectionsFragment();
    if (directions == null)
        directions = new ArrayList<>();
    Bundle args = new Bundle();
        args.putParcelableArrayList("directions",
(ArrayList<Direction>) directions);
    fragment.setArguments(args);
    return fragment;
}

@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view =
inflater.inflate(R.layout.fragment_view_directions, container, false);

        Bundle args = getArguments();
        if (args != null)
            directionList = args.getParcelableArrayList("directions");

        directionRecyclerView = view.findViewById(R.id.recyclerView);
        emptyView = view.findViewById(R.id.empty_view);

        directionAdapter = new DirectionAdapter(getActivity(),
directionList, false);
        toggleEmptyView();

```

```

        directionRecyclerView.setHasFixedSize(true);
        directionRecyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));
        directionRecyclerView.setAdapter(directionAdapter);

        return view;
    }

    private void toggleEmptyView() {
        if (directionList.size() == 0) {
            emptyView.setVisibility(View.VISIBLE);
            directionRecyclerView.setVisibility(View.GONE);
        } else {
            emptyView.setVisibility(View.GONE);
            directionRecyclerView.setVisibility(View.VISIBLE);
        }
    }
}

```

Листинг А.27 – Код класса ViewIngredientsFragment

```

package com.cul.omnomnom.ui.main.fragments;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.IngredientAdapter;
import com.cul.omnomnom.models.Ingredient;

import java.util.ArrayList;
import java.util.List;

/**
 * A simple {@link Fragment} subclass.
 */

```

```

public class ViewIngredientsFragment extends Fragment {

    private List<Ingredient> ingredientList;
    private IngredientAdapter ingredientAdapter;

    private RecyclerView ingredientRecyclerView;
    private TextView emptyView;

    public ViewIngredientsFragment() {
        // Required empty public constructor
    }

    public static ViewIngredientsFragment newInstance(List<Ingredient>
ingredients) {
        ViewIngredientsFragment fragment = new ViewIngredientsFragment();
        if (ingredients == null)
            ingredients = new ArrayList<>();
        Bundle args = new Bundle();
        args.putParcelableArrayList("ingredients", (ArrayList<Ingredient>)
ingredients);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_view_ingredients,
container, false);

        Bundle args = getArguments();
        if (args != null)
            ingredientList = args.getParcelableArrayList("ingredients");

        ingredientRecyclerView = view.findViewById(R.id.recyclerView);
        emptyView = view.findViewById(R.id.empty_view);

        ingredientAdapter = new IngredientAdapter(getActivity(),
ingredientList, false);
        toggleEmptyView();

        ingredientRecyclerView.setHasFixedSize(true);

```

```

        ingredientRecyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));
        ingredientRecyclerView.setAdapter(ingredientAdapter);

        return view;
    }

    private void toggleEmptyView() {
        if (ingredientList.size() == 0) {
            emptyView.setVisibility(View.VISIBLE);
            ingredientRecyclerView.setVisibility(View.GONE);
        } else {
            emptyView.setVisibility(View.GONE);
            ingredientRecyclerView.setVisibility(View.VISIBLE);
        }
    }
}

```

Листинг А.28 – Код класса CreateRecipeActivity

```

package com.cul.omnomnom.ui.main;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;

```

```

import com.cul.omnomnom.models.Direction;
import com.cul.omnomnom.models.Ingredient;
import com.cul.omnomnom.models.Recipe;
import com.cul.omnomnom.ui.main.fragments.NavigableFragment;
import com.cul.omnomnom.ui.main.fragments.RecipeDirectionsFragment;
import com.cul.omnomnom.ui.main.fragments.RecipeImageFragment;
import com.cul.omnomnom.ui.main.fragments.RecipeIngredientsFragment;
import com.cul.omnomnom.utils.Files;
import com.cul.omnomnom.utils.ResultCodes;

import java.util.List;
import java.util.Objects;

public class CreateRecipeActivity extends AppCompatActivity implements
                                RecipeImageFragment.ImageListener,
RecipeDirectionsFragment.DirectionsListener,
                                RecipeIngredientsFragment.IngredientListener {
    private static final int REQUEST_OPEN_GALLERY = 10;
    private static final int REQUEST_TO_ACCESS_GALLERY = 11;

    private Recipe currentRecipe;
    private boolean isUpdating;
    private String currentCategory;
    private DatabaseAdapter databaseAdapter;

    private Button nextButton;
    private Toolbar mToolbar;

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

        databaseAdapter = DatabaseAdapter.getInstance(this);

        Intent intent = getIntent();
        isUpdating = intent.getBooleanExtra("isUpdating", false);
        if (isUpdating)
            currentRecipe = intent.getParcelableExtra("recipe");
        else {
            currentCategory = intent.getStringExtra("category");
            currentRecipe = new Recipe(currentCategory);
        }
    }

```

```

        initializeUI();
        displayFragment(0);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                onBackPressed();
                break;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onBackPressed() {
        Fragment fragment =
getSupportFragmentManager().findFragmentById(R.id.frame_container);
        if (fragment instanceof RecipeIngredientsFragment)
            displayFragment(0);
        else if (fragment instanceof RecipeDirectionsFragment)
            displayFragment(1);
        else
            super.onBackPressed();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case REQUEST_OPEN_GALLERY:
                switch (resultCode) {
                    case RESULT_OK:
                        Uri imageData = data.getData();
                        String imageSrc = Files.getRealPathFromURI(this,
imageData);

                        ((RecipeImageFragment)
Objects.requireNonNull(getSupportFragmentManager().findFragmentById(R.id.frame_c
ontainer)))

```

```

        .onImageSelected(imageSrc);

        currentRecipe.setImagePath(imageSrc);
        break;
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
    switch (requestCode) {
        case REQUEST_TO_ACCESS_GALLERY:
            if (grantResults.length > 0
                && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
                openGallery();
            else
                Toast.makeText(this, "Permission denied to access the
gallery.", Toast.LENGTH_LONG)
                    .show();
            break;
    }
}

private void displayFragment(int position) {
    Fragment fragment = null;
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction ft = fragmentManager.beginTransaction();

    String nextButtonText = "";

    switch (position) {
        case 0:
            fragment = RecipeImageFragment.newInstance(currentRecipe);
            ft.setCustomAnimations(R.animator.slide_in_left,
R.animator.slide_out_right);
            nextButtonText = getString(R.string.Next);
            break;
        case 1:
            fragment =
RecipeIngredientsFragment.newInstance(currentRecipe);

```

```

        ft.setCustomAnimations(R.animator.slide_in_right,
R.animator.slide_out_left);
        nextButtonText = getString(R.string.Next);
        break;
    case 2:
        fragment =
RecipeDirectionsFragment.newInstance(currentRecipe);
        ft.setCustomAnimations(R.animator.slide_in_right,
R.animator.slide_out_left);
        nextButtonText = getString(R.string.fin);
        break;
    }

    nextButton.setText(nextButtonText);

    ft.replace(R.id.frame_container, fragment, "fragment" + position);
    ft.commit();
}

public void onNext(View view) {
    ((NavigableFragment)
getSupportFragmentManager().findFragmentById(R.id.frame_container)).onNext();
}

private void initializeUI() {
    mToolbar = findViewById(R.id.toolbar);
    setSupportActionBar(mToolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

    nextButton = findViewById(R.id.nextButton);
}

@Override
    public void navigateToIngredientsFragment(String name, String
description) {
        currentRecipe.setName(name);
        currentRecipe.setDescription(description);
        displayFragment(1);
    }

@Override

```

```

        public void navigateToDirectionsFragment(List<Ingredient> ingredients)
    {
        currentRecipe.setIngredients(ingredients);
        displayFragment(2);
    }

    @Override
    public void onSelectImage() {
        if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.READ_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
                new String[]
{Manifest.permission.READ_EXTERNAL_STORAGE},
                REQUEST_TO_ACCESS_GALLERY);
        } else {
            openGallery();
        }
    }

    @Override
    public void onStepsFinished(List<Direction> directions) {
        currentRecipe.setDirections(directions);
        if (isUpdating)
            databaseAdapter.updateRecipe(currentRecipe);
        else
            databaseAdapter.addNewRecipe(currentRecipe);

        Log.i("CreateRecipeActivity", "Final recipe: " + currentRecipe);
        setResult(isUpdating ? ResultCodes.RECIPE_EDITED :
ResultCodes.RECIPE_ADDED);
        finish();
    }

    private void openGallery() {
        Intent gallery = new Intent(Intent.ACTION_GET_CONTENT);
        gallery.setType("image/*");
        startActivityForResult(gallery, REQUEST_OPEN_GALLERY);
    }
}

```

Листинг А.29 – Код класса LoginActivity

```
package com.cul.omnomnom.ui.main;
```

```

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Toast;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.google.android.material.textfield.TextInputLayout;

public class LoginActivity extends ToolbarActivity {

    private TextInputLayout mLoginEmail;
    private TextInputLayout mLoginPassword;

    private DatabaseAdapter databaseAdapter;

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

        databaseAdapter = DatabaseAdapter.getInstance(this);

        mToolbar = findViewById(R.id.toolbar);
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("Войдите в свой аккаунт");

        mLoginEmail = findViewById(R.id.login_email);
        mLoginPassword = findViewById(R.id.login_password);
    }

    public void onLoginPressed(View view) {
        String email = mLoginEmail.getEditText().getText().toString();
        String password = mLoginPassword.getEditText().getText().toString();

        if (!TextUtils.isEmpty(email) || !TextUtils.isEmpty(password)) {
            loginUser(email, password);
        }
    }
}

```

```

        private void loginUser(String email, String password) {
            boolean status = databaseAdapter.signIn(email, password);
            if (status) {
                Intent intent = new Intent(this, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intent);
                finish();
            } else {
                Toast.makeText(this, "Пароль или логин был введен
некорректно", Toast.LENGTH_LONG).show();
            }
        }

        public void onCreateAccountPressed(View view) {
            startActivity(new Intent(this, RegisterActivity.class));
        }
    }

```

Листинг А.30 – Код класса MainActivity

```

package com.cul.omnomnom.ui.main;

import android.content.Intent;
import android.os.Bundle;
import android.util.Pair;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.ViewSwitcher;

import androidx.annotation.DrawableRes;
import androidx.viewpager.widget.ViewPager;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.cul.omnomnom.adapters.MainPagerAdapter;
import com.cul.omnomnom.models.Recipe;
import com.cul.omnomnom.ui.main.fragments.CategorizedFragment;
import com.cul.omnomnom.utils.ActivityTransition;

```

```

import com.cul.omnomnom.utils.ResultCodes;
import com.cul.omnomnom.utils.UserPreferences;
import com.google.android.material.appbar.CollapsingToolbarLayout;
import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.tabs.TabLayout;

import java.util.Objects;

public class MainActivity extends ToolbarActivity implements
CategorizedFragment.CategorizedFragmentListener {

    private static final int REQUEST_ADD_RECIPE = 1;
    private static final int REQUEST_VIEW_RECIPE = 2;
    private DatabaseAdapter databaseAdapter;
    private MainPagerAdapter mAdapter;

    private ViewPager mViewPager;
    private TabLayout mTabLayout;
    private CollapsingToolbarLayout mCollapsingToolbarLayout;
    private ImageView first;
    private ImageView second;
    private ViewSwitcher mViewSwitcher;

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

        Window w = getWindow();
        w.setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);

        databaseAdapter = DatabaseAdapter.getInstance(this);

        mToolbar = findViewById(R.id.toolbar);
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("Книга рецептов");

        mViewPager = findViewById(R.id.viewpager);
        mTabLayout = findViewById(R.id.tablayout);
        mCollapsingToolbarLayout = findViewById(R.id.collapsing_toolbar);

```

```

first = findViewById(R.id.first);
second = findViewById(R.id.second);

mViewSwitcher = findViewById(R.id.switcher);

mTabLayout.bringToFront();

mAdapter = new MainPagerAdapter(getSupportFragmentManager());

mViewPager.setAdapter(mAdapter);
mTabLayout.setupWithViewPager(mViewPager);
                                mViewPager.addOnPageChangeListener(new
TabLayout.TabLayoutOnPageChangeListener(mTabLayout));

                                mViewPager.addOnPageChangeListener(new
ViewPager.OnPageChangeListener() {
    @Override
    public void onPageScrolled(int position, float positionOffset,
int positionOffsetPixels) {
        }

    /**
     * Метод для переключения изображения категории в зависимости
от выбора пользователя
     * @param position
     */
    @Override
    public void onPageSelected(int position) {
        @DrawableRes int image = -1;
        switch (position) {
            case 0:
                image = R.drawable.soup; //TODO
                break;
            case 1:
                image = R.drawable.hot;
                break;
            case 2:
                image = R.drawable.salad_meat;
                break;
            case 3:
                image = R.drawable.cake_top_sweet;
                break;

```

```

        case 4:
            image = R.drawable.drinks;
            break;
    }

    if (first.getVisibility() == View.VISIBLE) {
        second.setImageResource(image);
        mViewSwitcher.showNext();
    } else {
        first.setImageResource(image);
        mViewSwitcher.showPrevious();
    }
}

@Override
public void onPageScrollStateChanged(int state) {

}

});
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.new_recipe:
            Intent intent = new Intent(this,
CreateRecipeActivity.class);
            intent.putExtra("category",
getCurrentlyDisplayedCategory());
            startActivityForResult(intent, REQUEST_ADD_RECIPE);
            break;
        case R.id.sign_out:
            UserPreferences.clear(this);
            navigateToLogin();
            break;
    }
    return super.onOptionsItemSelected(item);
}

```

```

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case REQUEST_ADD_RECIPE:
                switch (resultCode) {
                    case ResultCodes.RECIPE_ADDED:
                        Snackbar.make(getWindow().getDecorView(), "Рецепт
успешно добавлен", Snackbar.LENGTH_LONG)
                            .show();
                        break;
                    case ResultCodes.RECIPE_EDITED:
                        Snackbar.make(getWindow().getDecorView(), "Рецепт
успешно изменен", Snackbar.LENGTH_LONG)
                            .show();
                        break;
                }
                break;
            case REQUEST_VIEW_RECIPE:
                switch (resultCode) {
                    case ResultCodes.RECIPE_SHOULD_BE_EDITED:
                        Recipe recipe = data.getParcelableExtra("recipe");
                        Intent intent = new Intent(this,
CreateRecipeActivity.class);
                        intent.putExtra("recipe", recipe);
                        intent.putExtra("category", recipe.getCategory());
                        intent.putExtra("isUpdating", true);
                        startActivityForResult(intent,
REQUEST_ADD_RECIPE);
                        break;
                    case ResultCodes.RECIPE_SHOULD_BE_DELETED:
                        long recipeId = data.getLongExtra("recipeId", -1);
                        if (recipeId != -1) {
                            onDeleteRecipe(recipeId);
                            mViewPager.getAdapter().notifyDataSetChanged();
                        }
                        break;
                }
                break;
        }
    }
}

```

```

    }
}

private String getCurrentlyDisplayedCategory() {
    return
Objects.requireNonNull(mAdapter.getPageTitle(mViewPager.getCurrentItem()))
.toString();
}

private void navigateToLogin() {
    Intent startIntent = new Intent(this, LoginActivity.class);
    startActivity(startIntent);
    finish();
}

@Override
public void onShowRecipe(Recipe recipe, Pair<View, String>[] pairs) {
    Intent intent = new Intent(this, ViewRecipeActivity.class);
    intent.putExtra("recipe", recipe);

    ActivityTransition.startActivityForResultWithSharedElement(
        this, intent, pairs[0].first, pairs[0].second,
REQUEST_VIEW_RECIPE);
}

@Override
public void onEditRecipe(Recipe recipe) {
    Intent intent = new Intent(this, CreateRecipeActivity.class);
    intent.putExtra("recipe", recipe);
    intent.putExtra("category", getCurrentlyDisplayedCategory());
    intent.putExtra("isUpdating", true);
    startActivityForResult(intent, REQUEST_ADD_RECIPE);
}

@Override
public void onDeleteRecipe(long recipeId) {
    databaseAdapter.deleteRecipe(recipeId);
    Snackbar.make(getWindow().getDecorView(), "Рецепт удален",
Snackbar.LENGTH_LONG).show();
}
}

```

Листинг А.31 – Код класса RegisterActivity

```

package com.cul.omnomnom.ui.main;

import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;

import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.cul.omnomnom.models.User;
import com.google.android.material.textfield.TextInputLayout;

public class RegisterActivity extends ToolbarActivity {
    private DatabaseAdapter databaseAdapter;

    private TextInputLayout mUsername;
    private TextInputLayout mFullname;
    private TextInputLayout mEmail;
    private TextInputLayout mPassword;

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

        databaseAdapter = DatabaseAdapter.getInstance(this);

        mToolbar = findViewById(R.id.toolbar);
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("Создать учетную запись");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        mUsername = findViewById(R.id.reg_display_name);
        mFullname = findViewById(R.id.login_fullname);
        mEmail = findViewById(R.id.login_email);
        mPassword = findViewById(R.id.login_password);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {

```

```

        case android.R.id.home:
            finish();
            break;
    }

    return super.onOptionsItemSelected(item);
}

public void onCreateAccountPressed(View view) {
    String username = mUsername.getText().getText().toString();
    String fullname = mFullname.getText().getText().toString();
    String email = mEmail.getText().getText().toString();
    String password = mPassword.getText().getText().toString();

    registerUser(new User(username, fullname, email, password));
}

private void registerUser(User user) {
    databaseAdapter.addNewUser(user);

    Intent intent = new Intent(this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
    startActivity(intent);
}
}

```

Листинг А.32 – Код класса ToolbarActivity

```

package com.cul.omnomnom.ui.main;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

public abstract class ToolbarActivity extends AppCompatActivity {
    protected Toolbar mToolbar;
}

```

Листинг А.33 – Код класса ViewRecipeActivity

```

package com.cul.omnomnom.ui.main;

import android.content.Intent;
import android.graphics.Typeface;
import android.net.Uri;
import android.os.Bundle;

```

```

import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.viewpager.widget.ViewPager;

import com.elmargomez.typer.Font;
import com.elmargomez.typer.Typer;
import com.cul.omnomnom.R;
import com.cul.omnomnom.adapters.DatabaseAdapter;
import com.cul.omnomnom.adapters.ViewPagerAdapter;
import com.cul.omnomnom.models.Recipe;
import com.cul.omnomnom.utils.ResultCodes;
import com.google.android.material.appbar.CollapsingToolbarLayout;
import com.google.android.material.tabs.TabLayout;

import java.io.File;
import java.util.Objects;

public class ViewRecipeActivity extends ToolbarActivity {

    private ViewPagerAdapter mAdapterter;
    private Recipe currentRecipe;
    private DatabaseAdapter databaseAdapter;

    private ImageView mRecipeImage;
    private TextView mRecipeDescription;
    private ViewPager mViewPager;
    private TabLayout mTabLayout;
    private CollapsingToolbarLayout mCollapsingToolbarLayout;

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

        Window w = getWindow();
        w.setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);

```

```

        currentRecipe = getIntent().getParcelableExtra("recipe");
        databaseAdapter = DatabaseAdapter.getInstance(this);
        findViewsById();

        setSupportActionBar(mToolbar);
        Objects.requireNonNull(getSupportActionBar()).setTitle(currentRecipe.getName());
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        mRecipeImage.setImageURI(Uri.fromFile(new
File(currentRecipe.getImagePath())));
        mRecipeDescription.setText(currentRecipe.getDescription());

        mTabLayout.bringToFront();
        mAdapter = new ViewPagerAdapter(getSupportFragmentManager(),
currentRecipe);

        Typeface font = Typer.set(this).getFont(Font.ROBOTO_MEDIUM);
        mCollapsingToolbarLayout.setCollapsedTitleTypeface(font);
        mCollapsingToolbarLayout.setExpandedTitleTypeface(font);

        mViewPager.setAdapter(mAdapter);
        mTabLayout.setupWithViewPager(mViewPager);
        mViewPager.addOnPageChangeListener(new
TabLayout.TabLayoutOnPageChangeListener(mTabLayout));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.view_recipe_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.edit_recipe:
                Intent intent = new Intent();
                intent.putExtra("recipe", currentRecipe);
                setResult(ResultCodes.RECIPE_SHOULD_BE_EDITED, intent);
                finish();
            }
    }

```



```

        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP)
            ((Activity) context).startActivityForResult(intent,
requestCode);
        else {
            ActivityOptions transitionActivityOptions =
                ActivityOptions.makeSceneTransitionAnimation(
                    (Activity) context, sharedView,
transitionName);
            ((Activity) context).startActivityForResult(
                intent, requestCode,
transitionActivityOptions.toBundle());
        }
    }

    public static void startActivityForResultWithMultipleSharedElements(
        Context context, Intent intent, int requestCode, Pair<View,
String>... sharedElements) {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP)
            ((Activity) context).startActivityForResult(intent,
requestCode);
        else {
            ActivityOptions transitionActivityOptions =
                ActivityOptions.makeSceneTransitionAnimation((Activity
) context, sharedElements);
            ((Activity) context).startActivityForResult(
                intent, requestCode,
transitionActivityOptions.toBundle());
        }
    }

    @SafeVarargs
    public static void startActivityWithMultipleSharedElements(
        Context context, Intent intent, Pair<View, String>...
sharedElements) {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP)
            context.startActivity(intent);
        else {
            ActivityOptions transitionActivityOptions =
                ActivityOptions.makeSceneTransitionAnimation((Activity
) context, sharedElements);
            context.startActivity(intent,
transitionActivityOptions.toBundle());
        }
    }

```

```
    }
}
```

Листинг А.35 – Код класса Files

```
package com.cul.omnomnom.utils;

import android.content.Context;
import android.net.Uri;

public class Files {
    /**
     * @param contentURI of the file
     * @return путь до файла на диске
     */
    public static String getRealPathFromURI(Context context, Uri
contentURI) {
        return RealPathUtil.getRealPathFromURI_API19(context, contentURI);
    }
}
```

Листинг А.36 – Код класса RealPathUtil

```
package com.cul.omnomnom.utils;

import android.annotation.SuppressLint;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.DocumentsContract;
import android.provider.MediaStore;

import androidx.loader.content.CursorLoader;

public class RealPathUtil {

    public static String getRealPathFromURI_API19(Context context, Uri
uri) {

        String filePath = "";
        String wholeID = DocumentsContract.getDocumentId(uri);

        // Split at colon, use second item in the array
        String id = wholeID.split(":")[1];

        String[] column = {MediaStore.Images.Media.DATA};
```

```

        // where id is equal to
        String sel = MediaStore.Images.Media._ID + "=?";

        Cursor cursor =
context.getContentResolver().query(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
        column, sel, new String[]{id}, null);

        int columnIndex = cursor.getColumnIndex(column[0]);

        if (cursor.moveToFirst()) {
            filePath = cursor.getString(columnIndex);
        }
        cursor.close();
        return filePath;
    }

    @SuppressWarnings("NewApi")
    public static String getRealPathFromURI_API11to18(Context context, Uri
contentUri) {
        String[] proj = {MediaStore.Images.Media.DATA};
        String result = null;

        CursorLoader cursorLoader = new CursorLoader(
            context,
            contentUri, proj, null, null, null);
        Cursor cursor = cursorLoader.loadInBackground();

        if (cursor != null) {
            int column_index =
                cursor.getColumnIndexOrThrow(MediaStore.Images.Media.D
ATA);
            cursor.moveToFirst();
            result = cursor.getString(column_index);
        }
        return result;
    }

    public static String getRealPathFromURI_BelowAPI11(Context context,
Uri contentUri) {
        String[] proj = {MediaStore.Images.Media.DATA};

```

```

        Cursor cursor = context.getContentResolver().query(contentUri,
proj, null, null, null);
        int column_index
=
cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
        cursor.moveToFirst();
        return cursor.getString(column_index);
    }
}

```

Листинг А.37 – Код класса ResultCodes

```

package com.cul.omnomnom.utils;

public class ResultCodes {
    public static final int REQUEST_ADD_RECIPE = 1;
    public static final int REQUEST_VIEW_RECIPE = 2;
    public static final int RECIPE_ADDED = 100;
    public static final int RECIPE_EDITED = 101;
    public static final int RECIPE_SHOULD_BE_DELETED = 102;
    public static final int RECIPE_SHOULD_BE_EDITED = 103;
}

```

Листинг А.38 – Код класса UserPreferences

```

package com.cul.omnomnom.utils;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.util.Log;

import com.cul.omnomnom.models.User;

public class UserPreferences {
    private static final String TAG =
UserPreferences.class.getSimpleName();
    private static final String ID = "id";
    private static final String USERNAME = "username";
    private static final String FULLNAME = "fullname";
    private static final String EMAIL = "email";
    private static final String PASSWORD = "password";
    private static final String FIRST_RUN = "firstRun";
}

```

```

public static void saveCurrentUser(Context context, User currentUser)
{
    if (currentUser != null) {
        PreferenceManager.getDefaultSharedPreferences(context)
            .edit()
            .putInt(ID, currentUser.getId())
            .putString(USERNAME, currentUser.getUsername())
            .putString(FULLNAME, currentUser.getFullname())
            .putString(EMAIL, currentUser.getEmail())
            .putString(PASSWORD, currentUser.getPassword())
            .apply();
    } else {
        Log.i(TAG, "cannot save current user: user is NULL.");
    }
}

public static boolean isUserLoggedIn(Context context) {
    return PreferenceManager.getDefaultSharedPreferences(context)
        .contains(ID);
}

public static User loadUser(Context context) {
    SharedPreferences sharedPreferences =
        PreferenceManager.getDefaultSharedPreferences(context);
    if (sharedPreferences.contains(ID)) {
        return new User(
            sharedPreferences.getInt(ID, -1),
            sharedPreferences.getString(USERNAME, ""),
            sharedPreferences.getString(FULLNAME, ""),
            sharedPreferences.getString(EMAIL, ""),
            sharedPreferences.getString(PASSWORD, "")
        );
    }

    return null;
}

public static void setIsFirstRun(Context context, boolean isFirstRun)
{
    PreferenceManager.getDefaultSharedPreferences(context)
        .edit()
        .putBoolean(FIRST_RUN, isFirstRun)
        .apply();
}

```

```
}

public static boolean isFirstRun(Context context) {
    return PreferenceManager.getDefaultSharedPreferences(context)
        .getBoolean(FIRST_RUN, true);
}

public static void clear(Context context) {
    PreferenceManager.getDefaultSharedPreferences(context)
        .edit()
        .remove(ID)
        .remove(USERNAME)
        .remove(FULLNAME)
        .remove(EMAIL)
        .remove(PASSWORD)
        .apply();
}
}
```