ЧАСТНА ПРОФЕСИОНАЛНА ГИМНАЗИЯ ЗА ДИГИТАЛНИ НАУКИ „СОФТУНИ БУДИТЕЛ“, гр. София

**ДИПЛОМЕН ПРОЕКТ**

на Александър Дамянов Медаров

ученик/ученичка от XIIА клас

професия- код: **481030**, **“Приложен програмист”**

специалност- код: **4810301**, **“Приложно програмиране”**

**Тема: Разработване на мобилна апликация за запазване на готварски рецепти за любители-готвачи**

**Ръководител-консултант: Антони Добренов**

Сесия: май-юни 2024г.

Дата:.........................

**СЪДЪРЖАНИЕ**

[1. Увод 3](#_Toc162805980)

[1.1. Описание 3](#_Toc162805981)

[1.2. Мотивация 4](#_Toc162805982)

[1.3. Основна цел 5](#_Toc162805983)

[1.4. Очаквания 6](#_Toc162805984)

[1.5. Предизвикателства 6](#_Toc162805985)

[2. Изложение 7](#_Toc162805986)

[2.1. Описание на софтуерните технологии 7](#_Toc162805987)

[2.1.1. Мобилно native приложение 7](#_Toc162805988)

[2.1.2. Аndroid Studio 8](#_Toc162805989)

[2.1.3. Java 11](#_Toc162805990)

[2.1.4. XML(eXtensible Markup Language) 13](#_Toc162805991)

[2.1.5. Firebase(RealTime Database) 14](#_Toc162805992)

[2.1.6. Google-services.json 17](#_Toc162805993)

[2.1.7. Room 19](#_Toc162805994)

[2.1.8. Room DAO 21](#_Toc162805995)

[2.1.9. Адаптери 23](#_Toc162805996)

[2.1.11. Модели/POJOS 25](#_Toc162805997)

[2.1.12. Build.gradle/project 27](#_Toc162805998)

[2.1.13. Build.gradle/app 28](#_Toc162805999)

[2.1.14. Android Manifest 30](#_Toc162806000)

[2.2. Реализация на приложението 33](#_Toc162806001)

[2.3. Поставени цели за проекта 33](#_Toc162806002)

[2.4. Ръководство на потребителя 34](#_Toc162806003)

[2.4.1. Крайния потребител 34](#_Toc162806004)

[2.4.2. Потребителя като разработчик 34](#_Toc162806005)

[3. Заключение 35](#_Toc162806006)

[4. Информационни източници 37](#_Toc162806007)

[5. Приложения 39](#_Toc162806008)

# Увод

## Описание

Проектът RecipeBook се откроява като уникална платформа, пригодена за хора, запалени по готвенето, предлагаща безпроблемно пространство за подготвяне, подбиране и организиране на кулинарни творения. Той доизгражда конвенционалните приложения за управление на рецепти, насърчавайки обща атмосфера, в която потребителите могат да изследват разнообразни кухни и да експериментират с рецепти.

В рамките на интуитивния интерфейс на RecipeBook, потребителите могат щателно да документират любимите си рецепти, от ценни семейни реликви до новооткрити кулинарни вдъхновения. Всяка рецепта може да бъде подробно описана със съставки, етапи на приготвяне, продължителност на готвене, размери на сервиране и снимка на полученото ястие, като се гарантира изчерпателна документация за успешно повторение.

Организационните инструменти на приложението рационализират процеса на управление на рецепти, позволявайки на потребителите да ги категоризират въз основа на кухня. Освен това, възможността за маркиране на рецепти като любими улеснява лесния достъп до любимите ястия, когато се появи желание.

Обширната база данни с рецепти на RecipeBook служи като център за кулинарни изследвания, включвайки принос от разнообразна общност от ентусиасти в готвенето.

RecipeBook предлага както потребителски акаунти за безпроблемна синхронизация между устройствата, така и режим на гост за безпроблемно изследване. Създадено с native технологии за Android като Java, Room DB, Room DAO и Firebase, приложението осигурява отзивчиво и надеждно потребителско изживяване.

Като проект с отворен код, RecipeBook приканва към сътрудничество и иновации в общността на разработчиците, предоставяйки ценен ресурс за учене и експериментиране. Той е виден пример за използване на нови и иновативни технологии в разработката на мобилни приложения.

По същество RecipeBook е инструмент за управление на рецепти който. Независимо дали сте опитен готвач или начинаещ в кухнята, RecipeBook ви приветства да се впуснете в пътешествие на кулинарни открития и вдъхновение.

## Мотивация

Идеята за RecipeBook се ражда от простото желание да направя управлението на рецепти по-лесно и по-приятно. Като човек, който обича готвенето като хоби, често се затруднявах да проследя всички рецепти, които исках да опитам или вече бях изпробвал. Бележки, надраскани на хартия, отметки, разпръснати из различни уебсайтове, и неорганизирана бъркотия от запазени рецепти - всичко това беше твърде познато.

Реших да демонстрирам силата на новите технологии за разработване на native приложения и се заех да създам решение, което да рационализира процеса на съхранение, организиране и достъп до рецепти. RecipeBook се превърна в мой любим проект, подхранван от вярата, че технологията може да подобри кулинарното изживяване и да събере хората на споделени ястия.

Мотивацията ми беше проста: да разработя практичен инструмент за ентусиасти в готвенето, като същевременно допринасям за общността на разработчиците чрез проект с отворен код. Исках RecipeBook да бъде повече от просто приложение за рецепти – трябваше да бъде гостоприемно пространство, където потребителите да изследват нови кулинарни идеи и да намират вдъхновение за следващите си любителски рецепти

По време на процеса на разработка използвах собствения си опит като домашен готвач, стремейки се да създам удобен за потребителя интерфейс и използваеми функции, които наистина биха променили кулинарния живот на хората. RecipeBook е моят начин да слея двете си страсти – готвенето и технологиите – и да споделя този ентусиазъм с други, които обичат да готвят, експериментират и творят в кухнята.

В крайна сметка целта ми с RecipeBook е проста: да направя готвенето по-приятно, достъпно и общо за всички и да запозная любителите готвачи(и не само) с технологиите за създаване на мобилни приложения.

## Основна цел

Основните цели на проекта RecipeBook се въртят около опростяване на управлението на рецепти, насърчаване на кулинарни изследвания и изграждане на жизнена общност от ентусиасти в готвенето. Основната цел е да предостави на потребителите безпроблемна платформа за съхранение, организиране и достъп до любимите им рецепти. Като предлага интуитивни инструменти за категоризиране, маркиране и търсене, RecipeBook има за цел да рационализира процеса на управление на колекциите от рецепти, като улеснява потребителите да намират и използват рецепти, когато имат нужда от тях.

Освен това, RecipeBook се стреми да вдъхнови потребителите да изследват нови вкусове, кухни и техники за готвене. Чрез разнообразна база данни с рецепти, предоставени от общността, приложението насърчава потребителите да излязат от своите кулинарни зони на комфорт и да открият вълнуващи нови ястия, които да опитат.

Освен ролята си на инструмент за управление на рецепти, RecipeBook има за цел да насърчи чувството за общност сред ентусиастите в готвенето. Чрез функции като потребителски акаунти и любими рецепти. Чрез изграждането на подкрепяща и съвместна общност, RecipeBook обогатява кулинарното изживяване за потребителите и създава пространство, където ентусиастите на готвенето могат да се съберат, за да отпразнуват споделената си страст към храната.

Освен това, като проект с отворен код, RecipeBook се ангажира да насърчава сътрудничеството и иновациите в общността на разработчиците. Чрез споделяне на своя изходен код и документация, проектът дава възможност на други разработчици да се учат от и да надграждат върху основата му. Чрез приноси, обратна връзка и постоянни усилия за развитие, RecipeBook се развива и подобрява с течение на времето, като гарантира, че остава ценен ресурс както за потребителите, така и за разработчиците.

## Очаквания

Очакванията за проекта RecipeBook са съсредоточени около способността му да отговаря на нуждите и предпочитанията на своите потребители, да насърчава процъфтяваща общност и да допринася за напредъка на разработката с отворен код.

RecipeBook се очаква да бъде изграден с мисъл за устойчивостта, както по отношение на техническата инфраструктура, така и на усилията за ангажиране на потребителската база. Това включва осигуряване на мащабируемост, надеждност и сигурност на приложението, както и култивиране на поддържаща и устойчива общност, която може да поддържа непрекъснато развитие и растеж в дългосрочен план.

Основното очакване е да се предостави удобна за потребителя и надеждна платформа, която опростява управлението на рецептите, вдъхновява хоби кулинарите и подобрява цялостното готварско изживяване за потребителите. Проектът трябва да отговаря на очакванията на потребителите по отношение на функционалност, използваемост и производителност, като гарантира високи нива на удовлетворение и ангажираност на потребителската база.

## Предизвикателства

Разработката на дипломната работа беше маркирана от няколко основни затруднения. Които бяха най-често свързани с разработката на приложението. Най-често срещаните проблеми се срещат проектирането на интуитивен потребителски интерфейс, който балансира простотата с функционалността, се оказа предизвикателство, както и внедряването на ефективна структура на база данни за управление на рецепти, като същевременно се гарантира мащабируемост и цялост на данните.

Гарантирането на надеждността, стабилността и сигурността на приложението чрез цялостно тестване и процеси за осигуряване на качество беше от решаващо значение.

# Изложение

## Описание на софтуерните технологии

### 2.1.1. Мобилно native приложение

Мобилно native приложение е софтуерна програма, разработена специално за конкретна мобилна операционна система, като Android или iOS, използваща „родните“ езици за програмиране, инструменти и библиотеки на платформата. За разлика от междуплатформените или хибридните приложения, които използват уеб технологии, native приложенията са изградени с помощта на специфични за платформата езици като Java или Kotlin за Android и Objective-C или Swift за iOS. Избора на език и платформа за проекта Recipe Book e напълно стратегически. Android е по-разпространена от останалите операционни системи и предлага по-голям достъп до клиентска база. Java е основният език използван за разработката на дипломната работа понеже за момента има по-добра поддръжка от алтернативата му Kotlin понеже този език все още се води за сравнително нов.

Native приложенията предлагат висока производителност и безпроблемна интеграция с функции на устройството като камери, GPS и сензори, благодарение на достъпа до специфични за платформата API-ове. Те осигуряват удобно и интуитивно потребителско изживяване, като се придържат към насоките за дизайн на всяка платформа и стандартите за UI/UX, включително компоненти на собствения UI и модели за навигация.

Native приложенията също така често могат да осигурят стабилна офлайн поддръжка чрез кеширане на данни локално на устройството или използване на локална база данни, както в случая на нашия проект Room DB, което позволява на потребителите да имат достъп до съдържание и да изпълняват задачи, дори когато са офлайн или имат ограничена връзка.

Разпространени чрез официални магазини за приложения като Google Play Store и Apple App Store, native приложенията предлагат сигурност и доверие чрез прегледи и процеси на одобрение в магазина за приложения. Те също така поддържат стабилна офлайн функционалност и безпроблемно се интегрират с екосистемите на платформата за функции като насочени известия, покупки в приложението и удостоверяване. Въпреки че разработването на native приложения изисква експертни познания в специфични за платформата езици и инструменти, те осигуряват превъзходна производителност и потребителско изживяване, пригодени за всяка платформа.

### 2.1.2. Аndroid Studio

Android Studio служи като основна интегрирана среда за разработка (IDE) за изграждане на приложения за Android, предлагани от Google. Той предоставя на програмистите изчерпателен набор от инструменти, пригоден специално за разработка на приложения за Android. С удобен за потребителя интерфейс, той включва основни компоненти като редактор на код, редактор на визуално оформление и навигатор на проекти. Тези инструменти оптимизират процеса на кодиране, проектиране на потребителски интерфейси и управление на проекти.

Редакторът на кода улеснява ефективното кодиране с функции като подчертаване на синтаксис, допълване на код и инструменти за преработване. Той се интегрира безпроблемно със системи за контрол на версиите като Git за съвместна разработка. Освен това редакторът на визуално оформление на Android Studio дава възможност на разработчиците да проектират потребителски интерфейси чрез интуитивна функционалност за плъзгане и пускане(drag and drop), с визуализации в реално време в различни конфигурации на устройства.

Android Studio се интегрира безпроблемно също така с Firebase, платформата за мобилна разработка на Google, предлагайки набор от инструменти и услуги, които подобряват функционалността и възможностите на приложенията за Android. Интегрирането на Firebase в рамките на Android Studio предоставя на разработчиците лесен достъп до услуги като анализи, автентикация, облачни услуги и други.

Използвайки услугите на Firebase директно в Android Studio, разработчиците могат да подобрят своите приложения с функции като анализи в реално време, за да получат представа за поведението на потребителите, стабилни решения за удостоверяване за сигурно удостоверяване на потребителите и облачни съобщения за ангажиране на потребителите с насочени известия.

Интеграцията с Firebase опростява процеса на включване на тези мощни функции в приложения за Android, позволявайки на разработчиците да се съсредоточат върху изграждането на иновативни и удобни за потребителя изживявания без сложността на управлението на задната инфраструктура.

Зад кулисите Android Studio използва системата за изграждане Gradle за автоматизиране на задачи като компилиране, пакетиране и внедряване на приложения за Android. Разработчиците могат да персонализират конфигурациите за изграждане, да управляват зависимости и да дефинират персонализирани задачи, за да отговарят на нуждите на проекта. Освен това Android Studio предоставя и емулатор за тестване и отстраняване на грешки в приложения на симулирани устройства с Android, предлагайки различни конфигурации, за да се гарантира съвместимост и производителност.

За отстраняване на грешки и анализ на производителността Android Studio предлага надеждни инструменти като дебъгер и профилиращ инструмент. Тези инструменти помагат на разработчиците да идентифицират и разрешават проблеми ефективно, като подобряват качеството и производителността на техните приложения.

Важен аспект на Android Studio е неговата поддръжка за Kotlin, модерен език за програмиране, напълно одобрен от Google за разработка на Android. Разработчиците имат гъвкавостта да избират между Java и Kotlin или дори да използват и двата езика в един и същи проект, благодарение на тяхната оперативна съвместимост.

A logo of a compass

Description automatically generatedВ обобщение, Android Studio е многофункционално и мощно IDE, предлагащо на разработчиците цялостен набор от инструменти и функции за ефективно създаване на висококачествени приложения за Android. Неговият удобен за потребителя интерфейс, надеждни инструменти за разработка и безпроблемна интеграция с услугите на Google го правят предпочитан избор за разработка на приложения за Android.

**Фигура 1**

### 2.1.3. Java

В разработката на приложения за Android Java отдавна служи като стандарт на езика за програмиране на мобилни приложения , предлагайки на разработчиците мощен и гъвкав набор от инструменти за изграждане на собствени приложения за Android. Известността на Java в екосистемата на Android се корени в нейната здрава парадигма за обектно-ориентирано програмиране (OOP), която се привежда в съответствие безпроблемно с компонентно-базираната архитектура на Android framework.

Една ключова разлика между традиционната Java и специфичната за Android Java е в съответните им среди и библиотеки. Докато и двете споделят един и същ основен синтаксис и принципи, Android Java включва допълнителни класове и библиотеки, специално пригодени за мобилно развитие. Тези специфични за Android библиотеки предоставят достъп до функции на устройството като сензори, камери и услуги за местоположение, както и функционалности, специфични за платформата, като управление на жизнения цикъл на екраните и изобразяване на потребителския интерфейс.

Богатата екосистема на Java допълнително подобрява нейната полезност при разработването на Android, предлагайки на разработчиците достъп до широк набор от библиотеки, framework-ове и инструменти, оптимизирани за изграждане на мобилни приложения. От работа в мрежа и анализ на JSON до инжектиране на зависимости и обвързване на изгледи с логика, обширната екосистема на библиотеките на Java дава възможност на разработчиците да рационализират работните потоци за разработка и да изграждат ефективно приложения за Android, богати на функции.

Освен това съвместимостта на Java с платформата Android гарантира широка поддръжка на устройства и постоянна производителност при различни хардуерни конфигурации и версии на операционна система. Тази съвместимост е от съществено значение за достигане до разнообразна потребителска база и предоставяне на безпроблемно потребителско изживяване на устройства с Android по целия свят.

Въпреки появата на Kotlin като алтернативен език, одобрен от Google за разработка на Android, Java продължава да бъде популярен избор сред програмистите, особено в проекти с установени кодови бази. Докато Kotlin предлага модерни езикови функции и подобрен синтаксис, зрелостта, стабилността и широкото приемане на Java я правят надеждна опция за изграждане на стабилни и мащабируеми приложения за Android.

В обобщение, Java остава основен компонент на разработката на приложения за Android, предоставяйки на разработчиците доказан и многофункционален език за изграждане на native приложения за Android. Неговата съвместимост, обширната екосистема и безпроблемната интеграция с Android framework го правят ценен инструмент за програмисти, които искат да създадат висококачествени мобилни изживявания за потребителите на Android.

**Фигура 2**

### 2.1.4. XML(eXtensible Markup Language)

В разработването на приложения за Android XML (eXtensible Markup Language) играе жизненоважна роля при дефинирането на оформлението на потребителския интерфейс и конфигурирането на различни аспекти на приложенията. Файловете с XML оформление, обикновено съхранявани в директорията "res/layout" на проект на Android, йерархично описват елементи на потребителския интерфейс като изгледи, изпълними модули и оформления. Неговият декларативен синтаксис позволява на разработчиците да определят визуални и поведенчески характеристики, като размер, позиция, външен вид и поведение, използвайки XML тагове и атрибути.

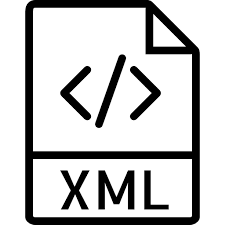
Разделянето на дефинициите на оформлението на потребителския интерфейс от логиката на приложението придава чиста, модулна структура на кода, улесняваща по-лесна поддръжка и актуализации независимо от основната кодова база. Системата за управление на ресурсите на Android използва XML, за да дефинира UI елементи и други ресурси на приложението(като в този проект например икони), опростявайки локализацията, тематизирането и персонализирането.

XML файловете с оформление поддържат ресурсни класификатори, което позволява създаването на алтернативни оформления, оптимизирани за различни размери на екрана, плътност и ориентация, осигурявайки адаптивност към различни устройства. При разработката на Android, базирана на Java, XML файловете с оформление се раздуват по време на изпълнение, като се използва класът "LayoutInflater", за да се създаде екземпляр на изгледните обекти.

XML улеснява обвързването на изгледи, позволявайки директно препращане на UI елементи, дефинирани в XML от Java код чрез автоматично генерирани обвързващи класове. Разработчиците могат да декларират манипулатори на събития за UI елементи в XML, свързвайки интерактивно поведение с конкретни UI елементи. XML също така поддържа прилагане на стилове и теми към елементи на потребителския интерфейс, позволявайки персонализиране и стилизиране чрез базирани на XML дефиниции, насърчавайки повторното използване на кода и поддръжката.

Като цяло XML играе основна роля в разработването на приложения за Android, особено при дефиниране на оформления на потребителския интерфейс и конфигуриране на ресурсите на приложението. Комбиниран с Java, XML позволява създаването на динамични, отзивчиви и визуално привлекателни приложения за Android с изчистени и модулни кодови архитектури.

**Фигура 3**



### 2.1.5. Firebase(RealTime Database)

Firebase е цялостна платформа, предлагана от Google за изграждане на мобилни и уеб приложения, предоставяща широка гама от услуги, за да помогне на разработчиците да разработват, развиват и осигуряват функционалности на своите приложения. Една от ключовите услуги, предлагани от Firebase, е Realtime Database, хоствана в облака NoSQL база данни, която позволява синхронизиране на данни в реално време между клиенти и облака.

RecipeBook се възползва от Firebase Realtime Database, за да запазва информация за рецепти, потребители и дадени категории с рецепти в реално време.

Firebase Realtime Database предлага синхронизиране на данни в реално време, като гарантира, че промените, направени в базата данни от един клиент, се отразяват незабавно на всички свързани клиенти. Тази възможност позволява на разработчиците да създават отзивчиви и съвместни приложения, без да управляват сложна сървърна инфраструктура.

Базата данни съхранява данни във формат JSON (JavaScript Object Notation), което улеснява лесното организиране и манипулиране на структурирани данни. Той поддържа вложени структури от данни, масиви и примитивни стойности, което позволява гъвкаво моделиране на данни, за да отговарят на различни случаи на използване на приложения.

Офлайн възможностите са вградени в Firebase Realtime Database, което позволява на приложенията да продължат да функционират дори когато устройството е офлайн. Промените, направени в базата данни, докато сте офлайн, автоматично се синхронизират с облака, след като устройството възстанови връзката, като се гарантира съгласуваност на данните между устройствата.

Firebase предоставя мощна система от правила за сигурност за Firebase Realtime Database, позволявайки на разработчиците да дефинират контрол на достъпа и правила за валидиране на своите данни. Тези правила могат да бъдат конфигурирани въз основа на удостоверяване на потребителя, структура на данните и потребителски роли, като се гарантира поверителност и цялост на данните.

Изградена върху мащабируемата инфраструктура на Google, Firebase Realtime Database предлага висока достъпност, надеждност и автоматично мащабиране, за да се справи с нарастващите натоварвания на приложенията. Тази мащабируемост гарантира, че приложенията, изградени на Firebase, могат безпроблемно да посрещнат нарастващото потребителско търсене.

Firebase Realtime Database се интегрира безпроблемно с други услуги на Firebase, като Firebase Authentication, Cloud Functions и Cloud Messaging. Тази интеграция позволява на разработчиците да създават решения от край до край за удостоверяване на потребители, изчисления без сървър, насочени известия и др.

Поддържайки множество клиентски платформи, включително Android, iOS и уеб, Firebase Realtime Database позволява на разработчиците да създават синхронизирани приложения, които работят безпроблемно на различни устройства и платформи.

В обобщение, Firebase Realtime Database предоставя мощно и мащабируемо решение за изграждане на приложения за съвместна работа в реално време. Неговата синхронизация на данни в реално време, офлайн възможности, правила за сигурност и интеграция с други услуги на Firebase го правят популярен избор за разработчиците, които искат да създават отзивчиви и богати на функции приложения без сложността на управлението на сървърната инфраструктура.

**Фигура 4**



### 2.1.6. Google-services.json

Файлът `google-services.json` е основен компонент в разработването на приложението, специално използван за конфигуриране и свързване на проекта с услугите на Firebase, предоставяни от Google. Този файл съдържа основни настройки за конфигурация и метаданни, необходими за безпроблемното интегриране на приложението с услугите на Firebase.

Първо, той съдържа подробности за проекта Firebase, свързан с приложението, като ID на проекта, API ключ и специфични за проекта настройки. Тези подробности позволяват на приложението да комуникира ефективно с услугите на Firebase и да се удостоверява с бекенда на Firebase.

Освен това във файла са включени специфични за услугата конфигурации за всяка услуга на Firebase, използвана от приложението. Тези конфигурации включват идентификатори, API ключове и други параметри, които са от съществено значение за правилното функциониране на услугите.

Освен това са включени идентификационни данни за удостоверяване и настройки за оторизация, за да се гарантира сигурно удостоверяване на потребителя и достъп до услугите на Firebase, като се ограничава достъпът само до оторизирани потребители.

За приложения, използващи Firebase Cloud Messaging (FCM) за насочени известия, файлът `google-services.json` включва настройки за конфигурация, свързани с FCM, като ID на изпращача и ключ на сървъра, улесняващи получаването и обработката на насочени известия.

Освен това, ако приложението използва динамични връзки на Firebase за създаване и обработка на дълбоки връзки, файлът може да включва настройки за конфигурация, специфични за динамичните връзки, като определя тяхното поведение и ги свързва с проекта на Firebase.

A screenshot of a computer program

Description automatically generatedКато цяло файлът `google-services.json` служи като централен конфигурационен файл, рационализирайки настройката и конфигурацията на услугите на Firebase в приложение за Android. Той осигурява гладка комуникация с бекенда на Firebase и позволява на приложението да използва пълния потенциал на услугите на Firebase.

**Фигура 5**

**Фигура 6**

### 2.1.7. Room

Room е библиотека, предоставена от Google като част от компонентите на архитектурата на Android Jetpack, предназначена да опрости работата с бази данни в приложения за Android. Той предлага абстрактен слой върху SQLite бази данни, улеснявайки операциите с бази данни с кратки SQL заявки и проверки по време на компилиране.

В основата си Room използва модела за връзка между обекти, позволявайки на разработчиците да дефинират обекти с данни, използвайки обикновени Java обекти (POJO), анотирани с анотации `@Entity`. Тези обекти представляват таблици на база данни и разработчиците могат да определят свойства и връзки между тях точно както в обикновена SQL база данни.

За достъп и манипулиране на данни, Room предоставя DAO (Data Access Object) интерфейс, където разработчиците анотират методи със SQL заявки, използвайки анотации като `@Insert`, `@Update`, `@Delete` и `@Query`. След това Room генерира съответния код за достъп до базата данни по време на компилиране, като гарантира коректността на кода и улавя потенциални грешки в началото на процеса на разработка.

Room също така поддържа преобразуватели на типове, което позволява на разработчиците да дефинират персонализирани съпоставки между сложни типове данни и техните представяния на бази данни, опростявайки съхранението на нестандартни типове данни.

Класът `RoomDatabase` служи като основна точка за достъп до основната база данни SQLite, предоставяйки единичен екземпляр и абстрактни методи за достъп до DAO. Този клас се интегрира безпроблемно с компонентите на Android, съобразени с жизнения цикъл, позволявайки на разработчиците да наблюдават промените в базата данни и да актуализират ефективно UI компонентите.

В RecipeBook основната цел на на Room е да интегрира локална база данни в приложението за по лесно запазване и зареждане на рецептите.

Като цяло, Room рационализира управлението на бази данни в приложенията за Android, като предлага удобен и ефективен слой за абстракция, с поддръжка за проверки по време на компилиране, кратки SQL заявки и интеграция с компонентите на архитектурата на Android, което го прави ценен инструмент за локално съхранение и A screen shot of a computer

Description automatically generatedуправление на данни.

**Фигура 7**

### 2.1.8. Room DAO

Room DAO (обект за достъп до данни) е ключов интерфейс в рамките на библиотеката за постоянство Room в разработката на Android, улесняващ взаимодействието с основната база данни SQLite. Той действа като централен център за извършване на различни CRUD (Създаване, четене, актуализиране, изтриване) операции върху данни, съхранявани в базата данни.

Чрез Room DAO разработчиците дефинират методи за операции с бази данни като вмъкване, актуализиране, изтриване и запитване към данни. Тези методи са анотирани с подходящи анотации на Room като `@Insert`, `@Update`, `@Delete` и `@Query`, които предоставят инструкции на Room как да изпълни съответните SQL заявки.

Едно от значителните предимства на Room DAO е неговата поддръжка за проверки по време на компилиране на SQL заявки. Room проверява синтаксиса и семантиката на SQL заявките, дефинирани в DAO методите по време на фазата на компилация, като по този начин открива грешки и потенциални проблеми в началото на процеса на разработка. Това гарантира коректността на кода и намалява вероятността от грешки по време на изпълнение.

Разработчиците могат да дефинират персонализирани SQL заявки, като използват анотацията `@Query` в методите Room DAO, което позволява гъвкаво и ефективно извличане и манипулиране на данни. Параметризирани заявки също се поддържат, позволявайки динамично предаване на параметри към SQL заявки, подобрявайки гъвкавостта и повторното използване.

Room DAO методите могат да връщат различни типове данни, включително единични обекти, списъци с обекти, примитивни типове данни или персонализирани класове данни. Room автоматично обработва преобразуването на резултатите от заявката в посочения тип връщане, опростявайки извличането и обработката на данни.

Транзакциите, анотирани с `@Transaction`, позволяват множество операции с бази данни да се изпълняват атомарно като една единица. Това гарантира последователност и цялост на данните чрез извършване на всички промени или връщане назад на цялата транзакция в случай на грешки.

Освен това методите на Room DAO поддържат асинхронно изпълнение с помощта на съпрограмми на Kotlin, RxJava или LiveData, което позволява операциите на базата данни да се извършват във фонови нишки, без да се блокира основната UI нишка. Това гарантира отзивчив потребителски интерфейс и подобрява цялостната производителност на приложението.

A computer screen shot of text

Description automatically generatedВ обобщение, Room DAO служи като мощен интерфейс за взаимодействие с бази данни SQLite в приложения за Android. Неговата поддръжка за персонализирани SQL заявки, проверки по време на компилация, параметризирани заявки, управление на транзакции и асинхронно изпълнение го прави незаменим инструмент за управление на данни и постоянство в разработката на Android.

**Фигура 8**

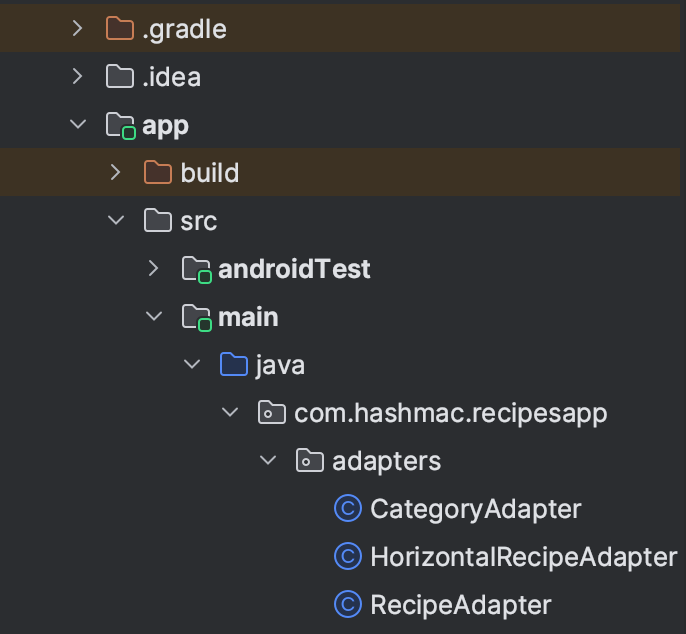
### 2.1.9. Адаптери

В Android Studio адаптерите служат като жизненоважни компоненти при свързването на източници на данни към елементи на потребителския интерфейс като ListView, RecyclerView, Spinner и GridView. Действайки като посредници, адаптерите улесняват попълването на UI компоненти с данни и обработват безпроблемно потребителските взаимодействия.

ArrayAdapter, например, опростява процеса, като работи с масиви или списъци с данни, преобразувайки всеки елемент в изглед за показване в UI компоненти като ListView или Spinner. От друга страна, по-персонализираният BaseAdapter, абстрактен клас, изисква от разработчиците да прилагат методи за обвързване на данни, създаване на изгледи и други функционалности на адаптера, което го прави подходящ за сложни сценарии.

За ефективно боравене с големи набори от данни, RecyclerView.Adapter се сдвоява с RecyclerView, използвайки модел на притежател на изглед за управление и рециклиране на изгледи, докато потребителят превърта през списъка. CursorAdapter, пригоден за Cursor обекти от SQLite бази данни или доставчици на съдържание, автоматично управлява жизнения цикъл на курсора и актуализира потребителския интерфейс, когато данните се променят.

И накрая, PagerAdapters работят с ViewPager, за да създават екрани с възможност за плъзгане, като например галерии с изображения или екрани за включване, предоставяйки съдържание за всяка страница и управлявайки създаването и унищожаването на изгледи на страници, докато потребителят навигира. Адаптерите в Android Studio предлагат жизненоважен механизъм за динамично свързване на източници на данни към UI компоненти, което позволява на разработчиците да създават динамични и интерактивни потребителски интерфейси безпроблемно. Изборът на подходящия адаптер въз основа на UI компонента и източника на данни гарантира ефективно попълване на UI елементи и осигурява сплотено потребителско изживяване в приложенията за Android. 2.1.10. Фрагменти

React Router е популярна библиотека, използвана за обработка на маршрутизиране в React приложения. Той позволява на разработчиците да създават приложения с една страница с множество изгледи или страници, което позволява на потребителите да навигират между различни части на приложението, без да причиняват пълно презареждане на страницата. React Router предоставя декларативен начин за дефиниране на маршрути и компоненти за свързване, което улеснява създаването на навигационна структура за приложението. Той използва подход, базиран на компоненти, където всеки маршрут е представен от React компонент. Основните компоненти, предоставени от React Router, включват BrowserRouter, Route, Switch, Link и Redirect. BrowserRouter използва API за история на HTML5, за да поддържа потребителския интерфейс в синхрон с URL адреса и осигурява основата за маршрутизиране. Маршрутът изобразява потребителския интерфейс въз основа на текущия URL адрес, което позволява на разработчиците да определят кой компонент да се изобразява за даден URL път. Превключвателят изобразява първия дъщерен маршрут или пренасочване, който съответства на текущия URL адрес, полезен за изобразяване само на единичен маршрут. Link предоставя начин за навигация между маршрути в приложението. Пренасочването изобразява ново местоположение на маршрута, когато пътят съвпада. React Router също поддържа разширени функции като вложени маршрути, параметри на маршрута, защита на маршрута и други, което го прави подходящ за изграждане на сложни навигационни структури и потребителски интерфейси.

**Фигура 9**

### 2.1.11. Модели/POJOS

В Android Studio моделите представляват структурите от данни, използвани за организиране и управление на данни в приложение. Тези модели капсулират свойствата и поведението на обекти от реалния свят или абстрактни концепции за данни, осигурявайки структуриран начин за манипулиране и взаимодействие с данни.

POJO (обикновени стари Java обекти) са прости Java класове, които обикновено се използват като модели в разработката на Android. Те капсулират полета с данни и предоставят методи за получаване и настройка за достъп и модифициране на данните. POJO обикновено представляват обекти като потребители, продукти или събития, със свойства, съответстващи на атрибути на обекта и методи за извършване на операции върху данните.

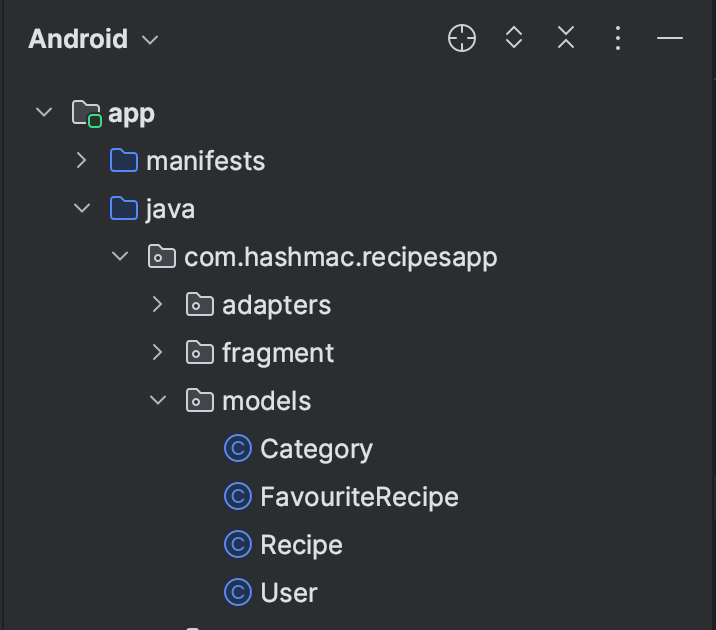
Класовете на обекти, особено в приложения, управлявани от бази данни, представляват структури от данни, нанесени към таблици на база данни. Тези класове използват анотации като `@Entity`, `@PrimaryKey` и `@ColumnInfo`, за да дефинират схемата на таблицата и да съпоставят полетата на класа към колоните на базата данни. Room, библиотека за постоянство в Android Jetpack, обикновено използва класове обекти за моделиране на данни за бази данни SQLite.

ViewModels, част от Android Architecture Components, съхраняват и управляват данни, свързани с потребителския интерфейс, по начин, съобразен с жизнения цикъл. Те абстрахират данни от UI контролери и оцеляват при промени в конфигурацията като завъртане на екрана. ViewModel често съдържа препратки към модели на данни или хранилища и излага LiveData или RxJava потоци, за да наблюдава промените в данните.

Обектите за прехвърляне на данни (DTO) са леки обекти, използвани за прехвърляне на данни между различни слоеве на приложение или между клиента и сървъра. Те представляват данни, извлечени от отдалечени API или сериализират/десериализират данни за мрежова комуникация. DTO обикновено съдържат само полета с данни и не включват бизнес логика.

Моделът на хранилището абстрахира източниците на данни от останалата част от приложението, осигурявайки унифициран интерфейс за достъп до данни. Репозиториите действат като посредници между ViewModels и източници на данни (напр. локални бази данни, отдалечени API), като често използват модели за представяне на данни, извлечени от или съхранени в източници на данни.

В обобщение, моделите в Android Studio са от съществено значение за организиране, манипулиране и управление на данни в приложенията. Те капсулират свойствата и поведението на обекти или концепции за данни, осигурявайки структуриран подход за работа с данни в разработката на Android. В зависимост от архитектурата и изискванията на приложението могат да се използват различни типове модели, включително POJO, класове на обекти, ViewModels, DTO и хранилища..



**Фигура 10**

### 2.1.12. Build.gradle/project

Файлът „build.gradle“ за проекта в Android Studio е конфигурационен файл, използван за определяне на настройките за изграждане и зависимостите за целия проект. Обикновено се намира в главната директория на проекта и е от съществено значение за конфигуриране на различни аспекти на процеса на изграждане. Ето общ преглед на това за какво се използва файлът `build.gradle` за проекта:

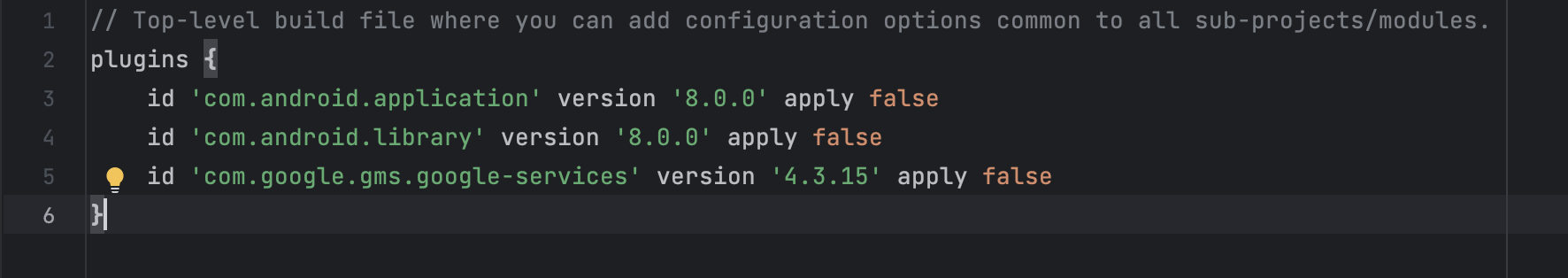
Конфигурация на проекта - Файлът `build.gradle` дефинира настройки за целия проект, като например името на проекта, версията и версиите на инструментите за изграждане. Той също така определя зависимостите и хранилищата на скрипта за изграждане, необходими за изграждането на проекта.

Приложение за приставки - Файлът `build.gradle` прилага приставки, които предоставят функционалност за изграждане и управление на проекта. За проекти с Android плъгинът `com.android.application` или `com.android.library` обикновено се прилага, за да се активират специфични за Android задачи за изграждане.

Хранилища - Файлът `build.gradle` указва хранилищата, от които се разрешават зависимостите. По подразбиране той включва хранилища за Android SDK и хранилището Maven на Google. Могат да се добавят допълнителни хранилища за разрешаване на зависимости от трети страни.

Управление на зависимостите - Файлът `build.gradle` дефинира зависимостите, изисквани от проекта, включително библиотеки, рамки и други модули. Зависимостите се декларират с помощта на блока `dependencies`, като се посочват групата, ID на артефакта и версията на всяка зависимост.

Типове компилация и вкусове на продукти - Файлът `build.gradle` конфигурира типове компилации (напр. отстраняване на грешки, издание) и вкусове на продукта (напр. безплатни, платени) за проекта. Той позволява на разработчиците да персонализират настройките за изграждане, конфигурациите на ресурсите и поведението на кода за различни варианти на приложението.

Конфигуриране на задачи - Файлът `build.gradle` конфигурира задачи за изграждане, тестване и пакетиране на проекта. Разработчиците могат да персонализират поведението на задачата, зависимостите и реда на изпълнение, за да отговорят на изискванията на проекта.

**Фигура 11**

### 2.1.13. Build.gradle/app

Файлът „build.gradle“ в директорията на модула „app“ на проект за Android е конфигурационен файл, специфичен за този модул. Той съдържа настройки и зависимости, свързани със създаването на самото приложение за Android. Ето за какво се използва файлът `build.gradle` в модула `app`:

Приложение на приставката - Файлът `build.gradle` прилага приставката `com.android.application`, което показва, че този модул е модул за приложение за Android. Този плъгин предоставя задачи и функции, специфични за създаване на приложения за Android.

Конфигурация на Android - Блокът `android` във файла `build.gradle` конфигурира различни настройки, свързани с приложението за Android, като ID на приложението, код на версията, име на версията и минимална версия на SDK, изисквана от приложението.

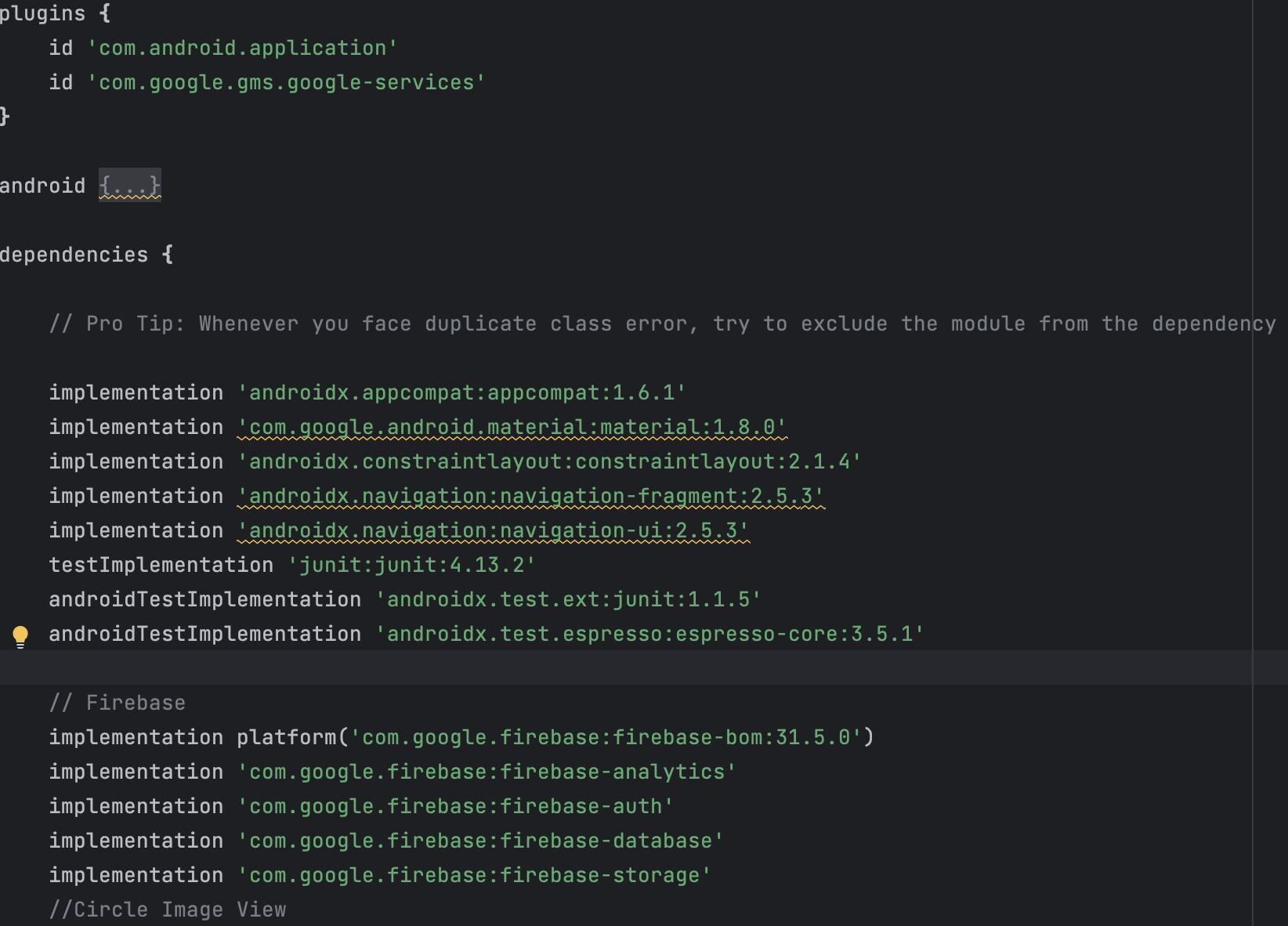
Зависимости - Блокът `dependencies` определя зависимостите, изисквани от приложението за Android, включително библиотеки, рамки и модули. Тези зависимости се декларират с помощта на ключовите думи `implementation`, `api` или `compileOnly`, заедно с групата, ID на артефакта и версията на всяка зависимост.

Типове компилации – Блокът `buildTypes` конфигурира различни типове компилации за приложението за Android, като например `debug` и `release`. Всеки тип компилация може да има свой собствен набор от опции за конфигурация, като например конфигурации за подписване, правила на proguard и варианти на компилация.

Вкусове на продукта – Блоковете `flavorDimensions` и `productFlavors` позволяват на разработчиците да дефинират вкусове на продукта за приложението за Android. Продуктовите вкусове представляват различни версии или варианти на приложението с уникални функции, брандиране или конфигурации.

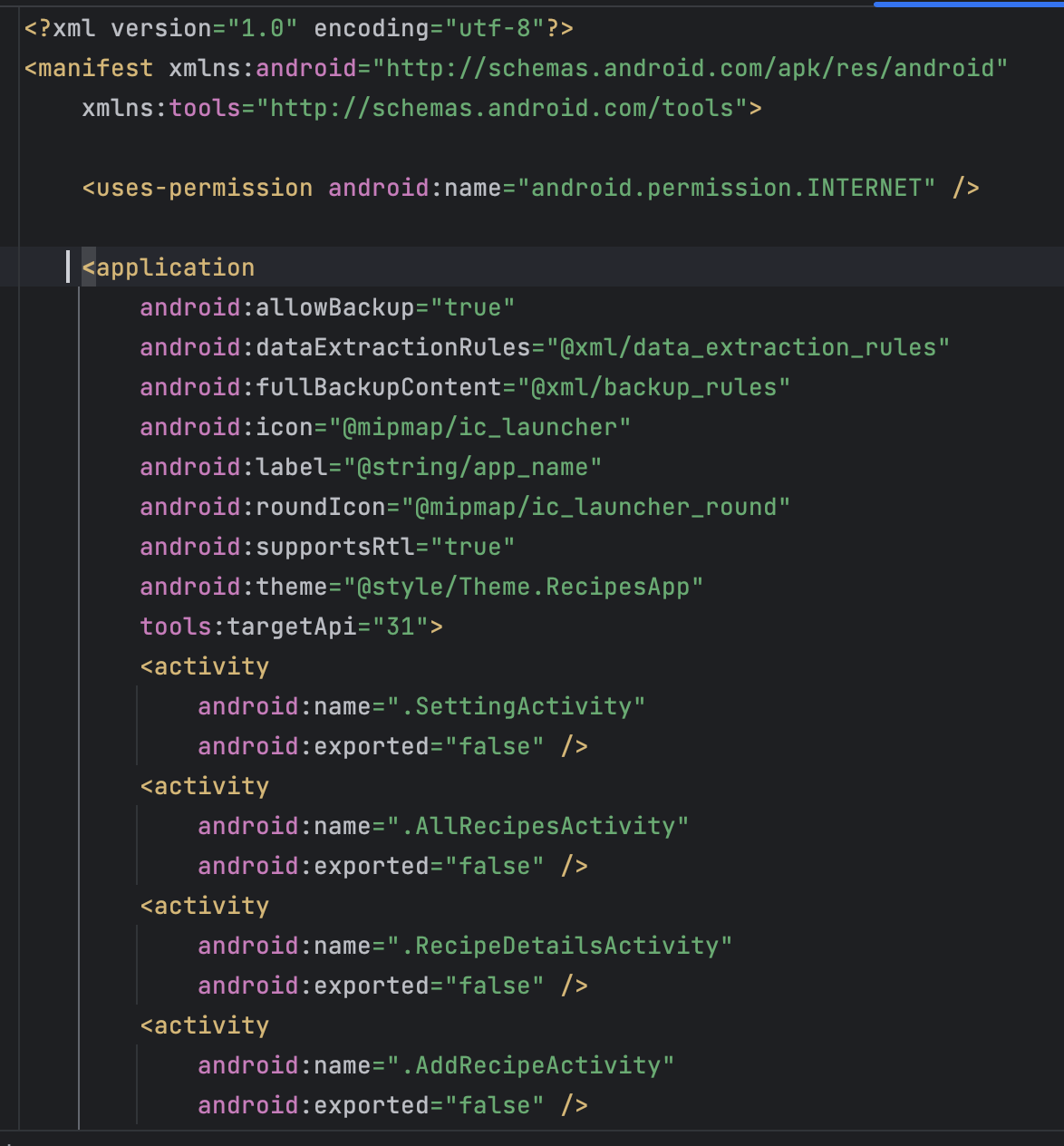
Опции за опаковане – Блокът `packagingOptions` определя опции, свързани с ресурсите и активите за опаковане в приложението за Android. Може да се използва за изключване или обединяване на определени файлове или ресурси, разрешаване на конфликти или персонализиране на поведението на опаковките.

Правила на Proguard/R8 – Файлът `build.gradle` може да включва опции за конфигурация за Proguard или R8, които са инструменти, използвани за обфускация и оптимизация на кода. Правилата на Proguard/R8 могат да бъдат зададени, за да се определи как трябва да се обработват кодът и ресурсите по време на процеса на изграждане.

Друга конфигурация – Освен това файлът „build.gradle“ може да съдържа други опции за конфигурация и персонализации, специфични за приложението за Android, като персонализирани задачи, проверки на мъх, конфигурации на ресурси или настройки за генериране на код.

**Фигура 12**

### 2.1.14. Android Manifest

Файлът `AndroidManifest.xml` е жизненоважен компонент на приложение за Android, предоставящ съществена информация за приложението на системата Android. Той служи за няколко важни цели. Първо, той декларира фундаментални характеристики на приложението, като името на неговия пакет, кода на версията, името на версията и идентификатора на приложението, като еднозначно идентифицира приложението на устройството и в Google Play Store. Второ, той определя разрешенията, изисквани от приложението за достъп до определени функции или ресурси на устройството, като информира потребителите за тези разрешения по време на инсталацията. Освен това файлът на манифеста декларира компонентите на приложението, включително дейности (екрани на потребителския интерфейс), услуги (фонови задачи) и приемници за излъчване (компоненти, които отговарят на събития в цялата система), заедно с техните имена, типове и конфигурации. Той също така дефинира филтри за намерения, позволявайки на компонентите да реагират на специфични типове намерения, което позволява взаимодействие с други приложения. Освен това файлът на манифеста съдържа допълнителни опции за конфигуриране, като например тема на приложението, поддържани ориентации на екрана, хардуерни и софтуерни изисквания и други, което помага да се определи поведението и външния вид на приложението. Освен това, той установява връзки между различни компоненти на приложението, като уточнява основната входна точка (лаунчер) дейност и услуги, които да се изпълняват при конкретни събития. И накрая, метаданните и ресурсите, свързани с приложението, също могат да бъдат дефинирани във файла на манифеста, включително етикета, иконата, темата и други свързани с ресурсите конфигурации на приложението. Като цяло файлът `AndroidManifest.xml` служи като план за системата Android, като влияе върху начина, по който системата взаимодейства с приложението и го управлява по време на инсталиране, време на изпълнение и деинсталиране.

**Фигура 13**

## Реализация на приложението

Приложението и неговите функционалности макар и семпли дават много добра идея за силата на native технологиите и предоставените от тях възможности. В едно такова приложение с отворен код се дават нагледни примери за работа с няколко вида база данни, native технологии, работа с BaaS и още много други неща. Проектът е успешно реализиран спрямо поставените задачи в началото.

## Поставени цели за проекта

* Намиране на актуална тема и нуждата от подобен софтуер
* Избор на рентабилни и лесни за поддръжка технологии
* Избор на back end за приложението (лесен начин за съхраняване на данни като рецепти)
* Избор на сигурни и защитени начини за регистриране на потребителите
* Запознаване с native технологии
* Проектиране архитектурата на приложението
* Публикуване на приложението в тестова версия

## Ръководство на потребителя

### 2.4.1. Крайния потребител

- Изтегляне на APK файла на приложението на съвместимо устройство

- Изберете една от трите опции : Регистриране, влизане в профил или продължете като гост

- Попълнете поисканата от вас информация и продължете към началната страница

- От тук вече може да записвате собствени рецепти, да редактирате профила си и да видите още много други рецепти.

### 2.4.2. Потребителя като разработчик

Понеже RecipeBook е приложение с отворен код, всеки един потребител който иска да се запознае с кода може да го инсталира и да го настрои за неговите нужди.

* След инсталиране на проекта от Гитхъб репозиторията и отварянето и в подходяща среда за разработка трябва да се изчака синхронизацията на gradle файловете и да се намери нужния Аndroid SDK.
* След успешно инсталиране приложението може да бъде пуснато на емулатор или на външно устройство.

# Заключение

Приложението RecipeBook е удобен за потребителя инструмент, предназначен за хоби готвачи да организират, откриват и споделят любимите си рецепти. Той е изработен с внимание към детайла и акцент върху простотата, което улеснява потребителите да изследват кулинарни вдъхновения.

По време на развитието му се сблъскахме с предизвикателства, но с отдаденост и постоянство ги преодоляхме, за да създадем надеждно и интуитивно приложение.

Сега приложението RecipeBook е готово да служи като полезен спътник за готварски ентусиасти по целия свят. Ние се ангажираме да поддържаме качеството му и непрекъснато да го подобряваме, за да отговорим на нуждите на нашите потребители.

Приложението RecipeBook е не само удобен инструмент за организиране на рецепти, но и проект с отворен код, приканващ към сътрудничество и иновации от разработчици от цял свят. Със своята кодова база, достъпна за общността, разработчиците могат да изследват, допринасят и персонализират приложението, за да отговаря на техните нужди или дори да надграждат върху основата му, за да създават нови кулинарни приложения.

Възприемайки принципите на разработката с отворен код, RecipeBook насърчава култура на споделяне и учене, като дава възможност на разработчиците да се възползват от неговите функции, да интегрират нови функционалности или да решават проблеми съвместно. Този отворен подход не само насърчава прозрачността и отчетността, но също така култивира жизнена екосистема, където идеите могат да процъфтяват и да се развиват.

Освен това природата на RecipeBook с отворен код насърчава ангажираността на общността и обратната връзка, като гарантира, че приложението остава подходящо и отговаря на променящите се нужди на своите потребители. Чрез приноси, дискусии и споделени прозрения, проектът може да расте и да се адаптира, превръщайки се в ценен ресурс както за хоби готвачи, така и за разработчици.

# Информационни източници

[Meet Android Studio](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://developer.android.com/studio/intro&ved=2ahUKEwjf4pPg9Z6FAxWnQPEDHUUeDAUQjBB6BAgNEAE&usg=AOvVaw2NJuxm9eqwR3pL_VYT-gsx)

[Android StudioWikipediahttps://en.wikipedia.org › wiki › Android\_Studio](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://en.wikipedia.org/wiki/Android_Studio&ved=2ahUKEwjf4pPg9Z6FAxWnQPEDHUUeDAUQFnoFCNYBEAE&usg=AOvVaw22JMQaX3bMBam8AHjLRjzn)

[Develop for AndroidAndroid Developershttps://developer.android.com › develop](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://developer.android.com/develop&ved=2ahUKEwiZu7H99Z6FAxWcZ_EDHRZMCl0QFnoECA4QAQ&usg=AOvVaw0_ItpWapX8bGMKaPA2BN4f)

[Save data in a local database using RoomAndroid Developershttps://developer.android.com › training › data-storage](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://developer.android.com/training/data-storage/room&ved=2ahUKEwip6ruD9p6FAxXAQ_EDHWTAD3MQFnoECCAQAQ&usg=AOvVaw3zNIzWq32_4C5U2awzrPZ9)

[**Firebase DocumentationGooglehttps://firebase.google.com › docs**](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://firebase.google.com/docs&ved=2ahUKEwjQmvOghp2FAxXHYPEDHW9vAbQQFnoECAcQAQ&usg=AOvVaw0PvZiBaWaogN0S0Q7uY2lT)

[Android Developers: Android Mobile App Developer ToolsAndroid Developershttps://developer.android.com](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://developer.android.com/&ved=2ahUKEwi6wLqp9p6FAxW2cfEDHT2hCysQFnoECBkQAQ&usg=AOvVaw12eEvU93yNv2rl5Tzxmb_u)

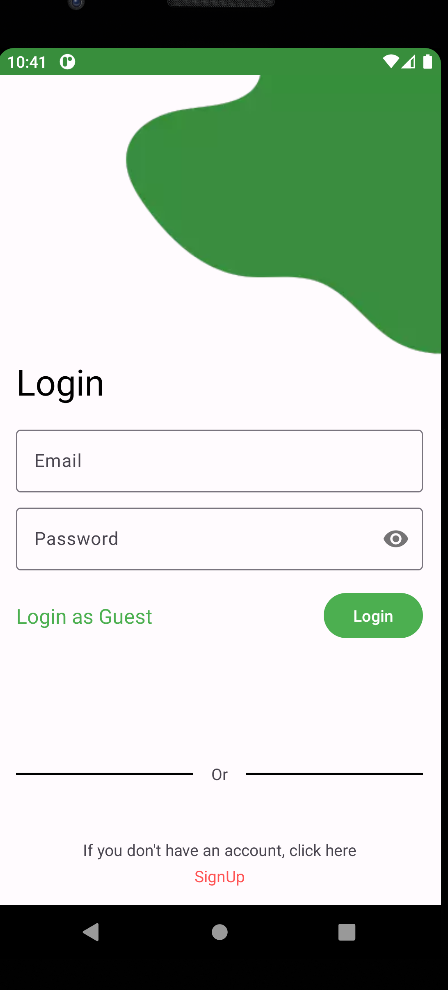
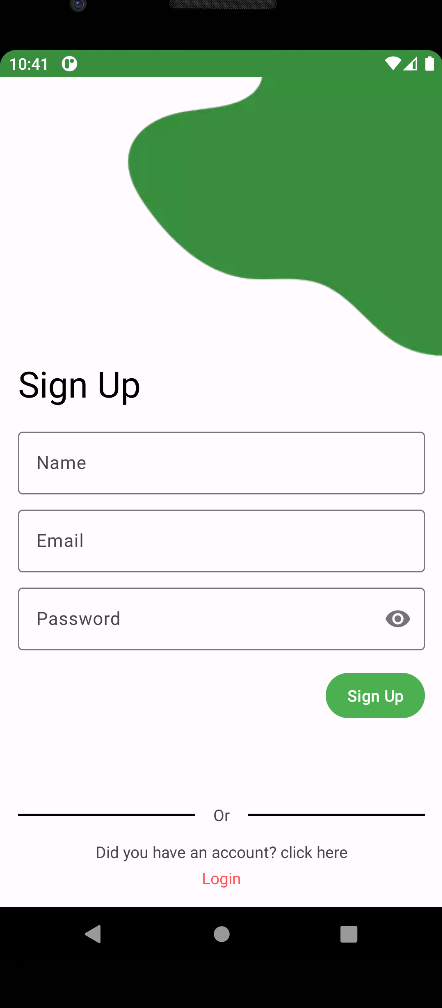
# Приложения

A screenshot of a menu

Description automatically generated

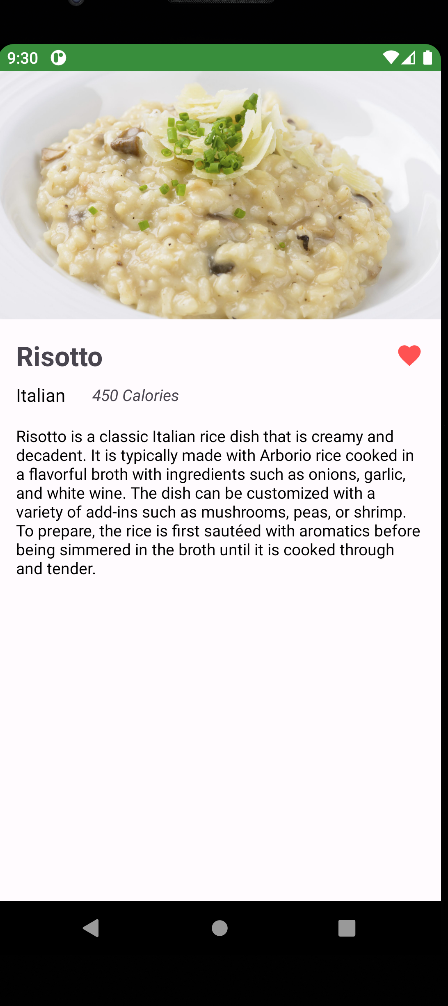
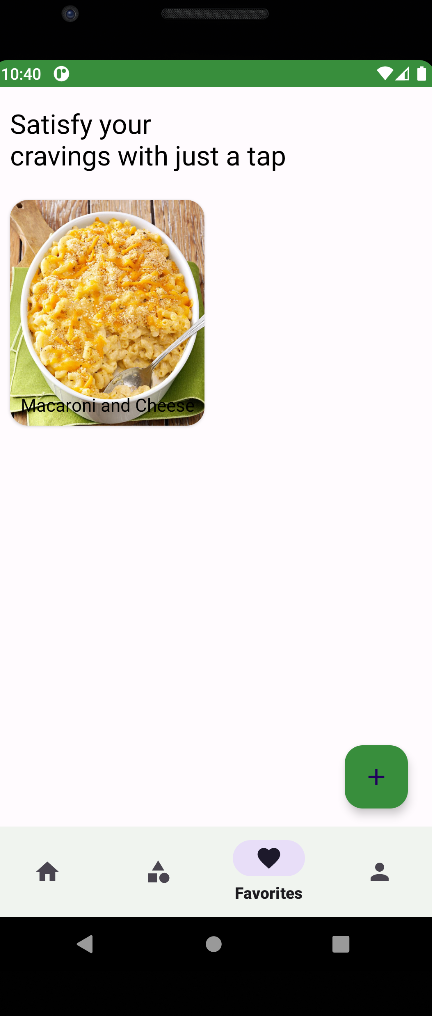
Фигура 17

Фигура 16



Фигура 19

Фигура 18



Фигура 21

Фигура 20