Министерство образования Новосибирской области ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

**Разработка мобильного приложения для ведения дневника питания**

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

МДК01.03 Разработка мобильных приложений

НАТКиГ.210300.043.000ПЗ

Выполнил:

студент группы ПР315

Бурцев В.В.

2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc104133636)

[1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ 4](#_Toc104133637)

[1.1 Описание предметной области 4](#_Toc104133638)

[1.2 Образ клиента 4](#_Toc104133639)

[1.3 Сценарии 5](#_Toc104133640)

[1.4 Сбор и анализ прототипов 5](#_Toc104133641)

[2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 7](#_Toc104133642)

[2.1 UI/UX дизайн проекта 7](#_Toc104133643)

[2.2 Выбор технологии, языка и среды программирования 11](#_Toc104133644)

[3 РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 12](#_Toc104133645)

[3.1 Разработка базы данных 12](#_Toc104133646)

[3.2 Разработка мультимедийного контента 15](#_Toc104133647)

[3.3 Описание используемых плагинов 17](#_Toc104133648)

[3.4 Описание разработанных процедур и функций 18](#_Toc104133649)

[4 Тестирование 22](#_Toc104133650)

[4.1 Протокол тестирования дизайна приложения 22](#_Toc104133651)

[4.2 Протокол тестирования функционала приложения 24](#_Toc104133652)

[ЗАКЛЮЧЕНИЕ 27](#_Toc104133653)

[СПИСОК ИСТОЧНИКОВ 28](#_Toc104133654)

[Приложение А 29](#_Toc104133655)

# ВВЕДЕНИЕ

По данным федеральной службы государственной статистики с2018 по 2021 год проблемы с лишним весом испытывает 40 и 50 процентов россиян соответственно. Основываясь на эти данные, можно сделать вывод, что этот процент растёт с каждым годом. Чтобы избавиться от лишнего веса необходимо настроить здоровое питание. Важно отметить, здоровое питание это не просто контроль за калориями и жесткими диетами, а в обеспечении организма полноценным рационом, в который входят все необходимые для него продукты: мясо, злаки, овощи, продукты. Здоровое питание важная составляющая здорового образа жизни человека. Именно поэтому, вопросы по темам лишнего веса и здорового питания являются как никогда актуальными.

Целью курсового проекта является разработка мобильного приложения для ведения дневника питания.

Для достижения поставленной цели определены следующие задачи:

1. изучить предметную область;
2. определить основной функционал приложения;
3. выбрать технологии, язык и среду программирования;
4. разработать дизайн дневника питания;
5. разработать базу данных для приложения;
6. разработать функционал приложения;
7. протестировать полученный продукт.

Объект исследования −рацион питания людей.

Предметом исследования является способ контроля рациона питания

Практическая значимость курсового проекта заключается в пользе данного приложения для людей, желающих поменять привычки питания. При помощи данного приложения человек начнёт обращать своё внимание на пищу, которую он употребляет, что поспособствует формированию здорового питания.

# ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

## Описание предметной области

На данный момент в России все больше людей становятся сторонниками здорового образа жизни. Для ведения здорового образа жизни важно следить за питанием. По данным исследования Всероссийского центра изучения общественного мнения от 30 мая 2020 года почти 60% опрошенных россиян следят за своим питанием.

Для определения нормы питания человеку необходимо сначала взвесить употребляемую пищу. Далее человек должен подсчитать вручную количество всех питательных веществ и калорий. И только после этого полученные значения занести в дневник. В целях экономии времени необходимо автоматизировать данные процессы.

Современный человек ценит своё время, а процесс подсчета употреблённых белков, жиров, углеводов и калорий весьма затратный по времени. Дабы сделать этот процесс менее трудоёмким и времязатратным, можно воспользоваться приложением, нацеленным на решение этой проблемы. При помощи данного приложения пользователь сможет вести ежедневный дневник питания для контроля над своим питанием без особых затрат по времени.

## Образ клиента

Данное приложение нацелено на российскую аудиторию, которая интересуется здоровым образом жизни, и обращает внимание на свой рацион питания. Как правило, здоровым питанием человек начинает интересоваться уже в осознанном возрасте, начиная с 16-18 лет. Данным приложением могут пользоваться как мужчины, так женщины.

Разрабатываемое мобильное приложение предназначено для людей, которые хотят ежедневно вести контроль над своим питанием

## Сценарии

Алина уже продолжительное время ведёт здоровый образ жизни и следит за своим питанием. У менеджера Алины рабочий день начинается с 8 часов утра. Утром Алине необходимо привести себя в порядок, подготовиться к работе и позавтракать. На все это ей бы хотелось тратить как можно меньше времени, чтобы была возможность по утрам выполнять домашние дела.

Программист Владимир несколько раз пытался начать следить за своим рационом питания. Владимир очень медлительный человек, он ничего не может выполнить вовремя. Каждый раз он забрасывал это дело, потому что Владимир все время опаздывает и времени на ведение дневника питания у него не остается.

## Сбор и анализ прототипов

Рассмотрим несколько приложений для ведения ежедневного дневника питания.

«Healthy» – разрабатываемое мобильное приложение, которое предназначено для ведения ежедневного дневника питания.

«Lifesum» – приложение, которое помогает перейти на здоровый образ жизни, начать правильно питаться и сбросить вес.

«Мой здоровый рацион» – многофункциональное приложение для ведения дневника питания. В данном приложении имеется собственная база данных на 150 тысяч продуктов с содержанием белков, жиров, углеводов.

В таблице 1 представлено сравнение данных приложений по следующим критериям: платное/бесплатное, необходимость иметь подключение к интернету, перегруженность интерфейса, сложность в освоении, поддержка русского языка, возможность самостоятельно добавлять пищу.

Таблица 1 – Сравнение прототипов

|  |  |  |  |
| --- | --- | --- | --- |
| Название приложения | Healthy | Lifesum | Мой здоровый рацион |
| Платное/Бесплатное | Бесплатное | Бесплатное со встроенными покупками | Бесплатное |
| Подключение к интернету | Не нужно | Нужен | Нужен |
| Перегруженность интерфейса | Не перегружен | Перегружен | Перегружен |
| Сложность в освоении | Легко | Легко | Сложно |
| Поддержка русского языка | Есть | Нет | Есть |
| Возможность самостоятельно добавлять пищу | Можно | Нельзя | Нельзя |

# ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

## UI/UX дизайн проекта

Для проектирования дизайна приложения выбран онлайн-сервис для разработки интерфейсов «Figma».

В качестве элемента изображенным на логотипе выбрано яблоко, потому что у человека этот фрукт ассоциируется с полезной пищей.

Для мобильного приложения разработан минималистичный логотип, представленный на рисунке 1.



Рисунок 1 – Логотип приложения

Определена основная цветовая схема приложения для светлой темы, которая представлена на рисунке 2. Именно эти цвета у человека ассоциируются со здоровьем, здоровым питанием, потому что зелёный цвет и его оттенки − это естественный цвет, который человек часто встречает в природе.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| #298043 | #9EFFBB | #52FF86 | #4F805E | #41CC6A |

Рисунок 2 – Цветовая схема приложения

На основе данной цветовой схемы разработан дизайн, который состоит из 6 экранов. При запуске программы отображается загрузочный экран (Рисунок 3).



Рисунок 3 – Загрузочный экран приложения

Основным экраном приложения является экран, на котором отображается список записей употребляемой пищи (Рисунок 4).

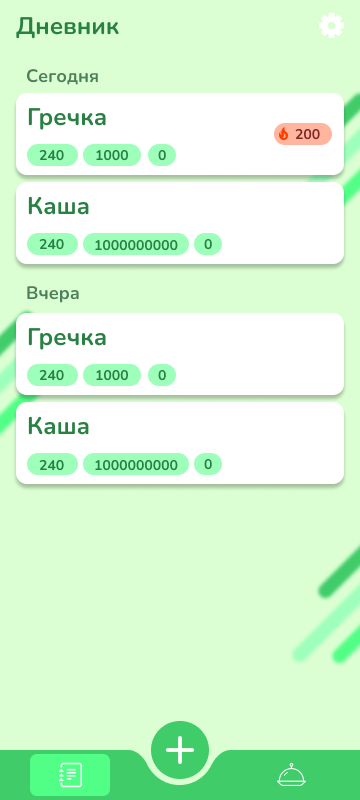


Рисунок 4 – Экран записей употребляемой пищи

Для добавления записи на основной экран, необходимо перейти на экран добавления записи (Рисунок 5). На данном экране пользователь может выбрать из списка ранее добавленных блюд нужное, после чего вписать вес употреблённой пищи. Далее, для добавления записи нужно нажать на кнопку.

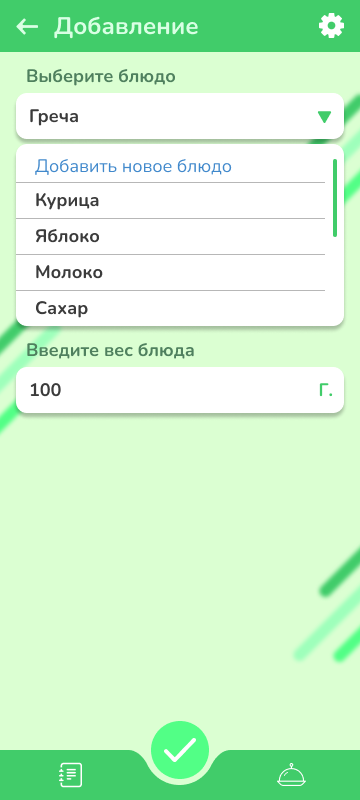


Рисунок 5 – Экран добавления записи в дневник

Чтобы пользователь мог вести записи личного дневника, питания разработан экран для вывода списка добавленных блюд (Рисунок 6) и экран добавление этих самых блюд (Рисунок 7).



Рисунок 6 – Экран списка добавленных блюд

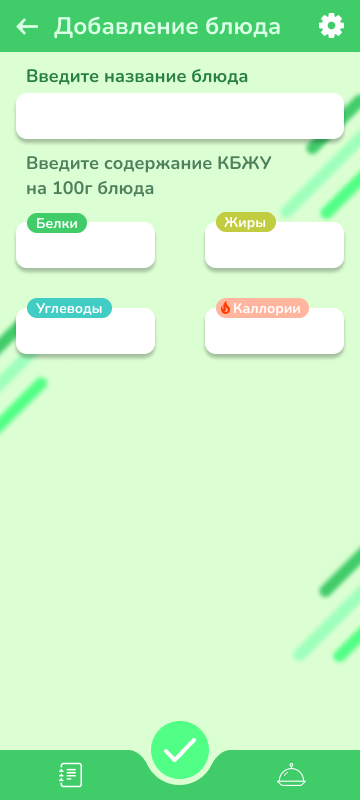


Рисунок 7 – Экран добавления блюда

На экране добавления блюда (Рисунок 7) пользователю необходимо ввести уникальное название блюда, а также содержание белков, жиров, углеводов и калорий в блюде на 100 грамм.

Последним экраном в приложении является экран настроек   
(Рисунок 8), где пользователь сможет выбрать цветовую схему приложения, которая ему более комфортна и информацию о приложении.

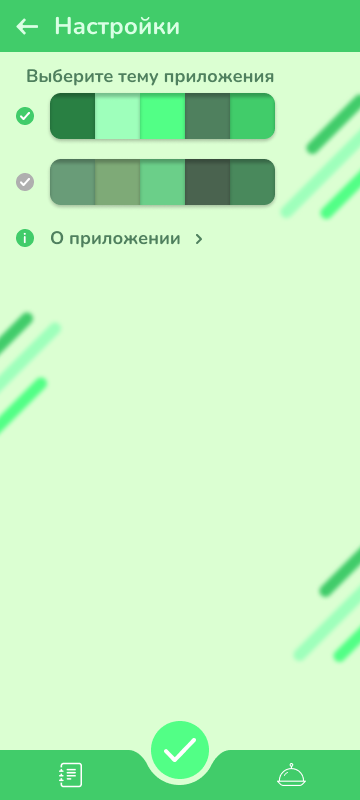


Рисунок 8 – Экран настроек

Для улучшения пользовательского опыта в использовании данного приложения предусмотрены плавные анимации перехода на каждый экран.

## Выбор технологии, языка и среды программирования

Вся разработка проекта ведётся в IDE Android Studio. Используется именно эта среда разработки, потому что необходимо разработать нативное мобильное приложение.

Для решения поставленной задачи выбран язык программирования Kotlin, потому что на данный момент является современным и обладает всеми необходимыми технологиями и библиотеками, которые могут решить данные задачи.

# РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ

## Разработка базы данных

В качестве разрабатываемой базы выбрана локальная система управления базами данных SQLite. В данном приложении для удобства работы с SQLiteиспользуется библиотека Room. Room – библиотека, которая является оберткой для работы с локальной базой данных SQLite. Первым шагом в разработке базы данных является реализация сущностей «FoodDbEntity» и «JournalDbEntity». Реализации данных сущностей представлены на рисунках 9 и 10 соответственно.

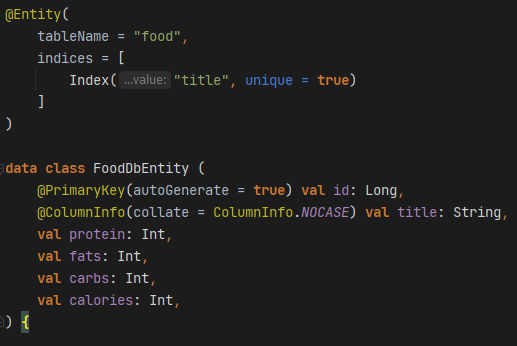


Рисунок 9 – Сущность базы данных «FoodDbEntity»



Рисунок 10 – Сущность базы данных «JournalDbEntity»

После создания сущностей базы данных необходимо прописать к ним запросы, чтобы иметь возможность манипулировать данными из приложения. Для реализации данной функции необходимо прописать следующие запросы: «FoodDao» для «FoodDbEntity» (Рисунок 11) и «JournalDao» для «JouranlDbEntity» (Рисунок 12).

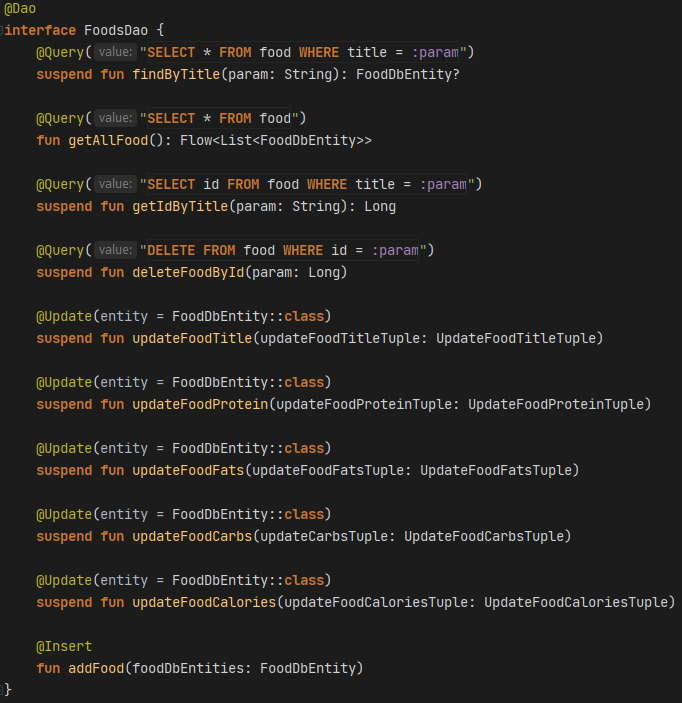


Рисунок 11 –Запросы «FoodDao»

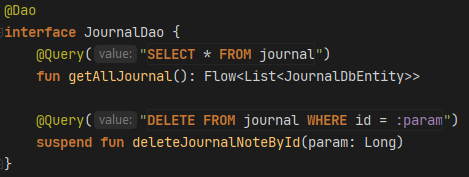


Рисунок 12 − Запросы «JournalDao»

Далее необходимо написать реализацию этих запросов (Рисунок 13).

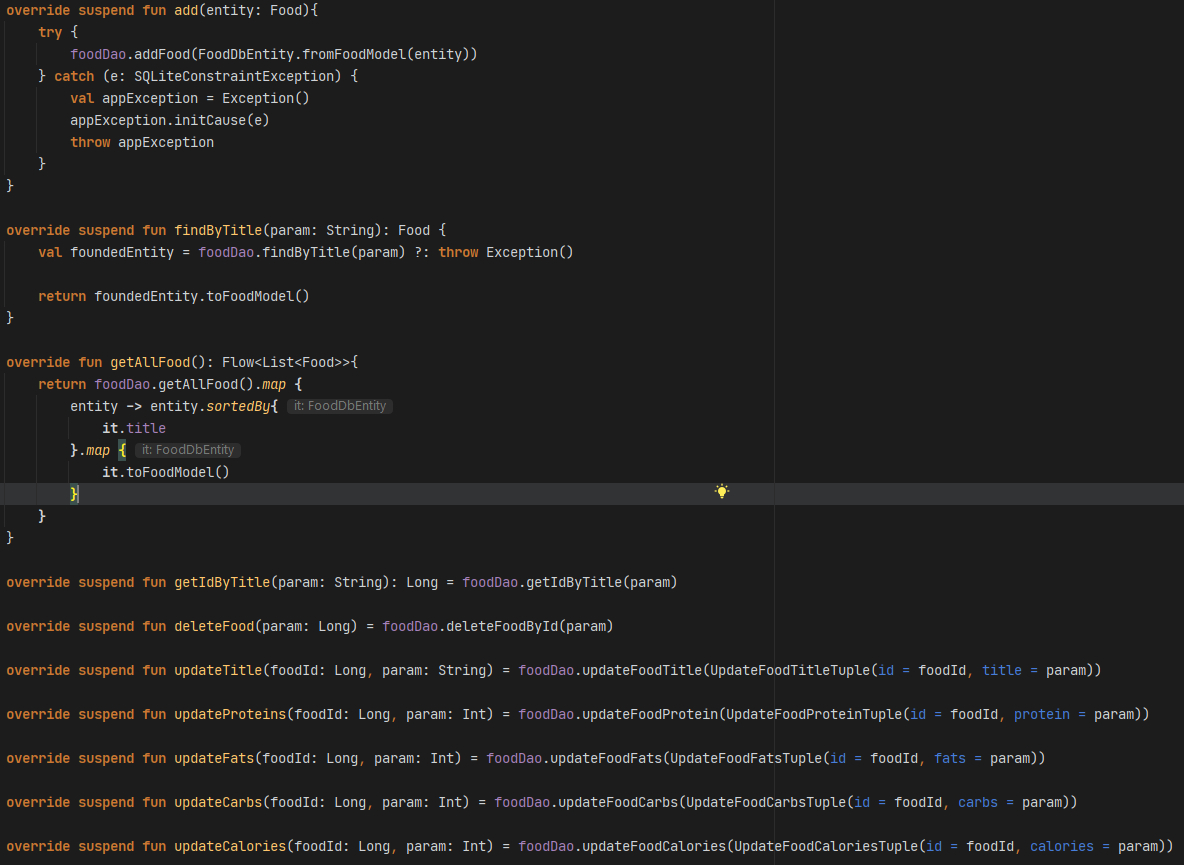


Рисунок 13 – Реализация запросов

Завершающим шагом в разработке базы данных является реализация класса, которая представлена на рисунке 14. Этот класс даёт возможность получать данные из разработанной локальной базы данных.

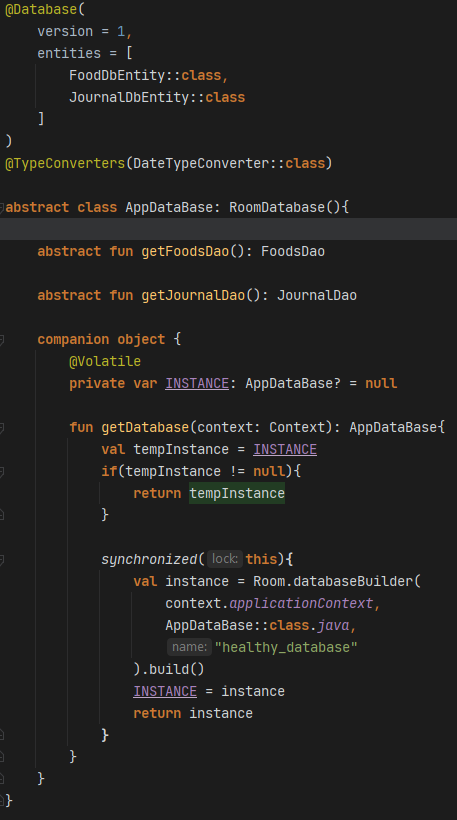
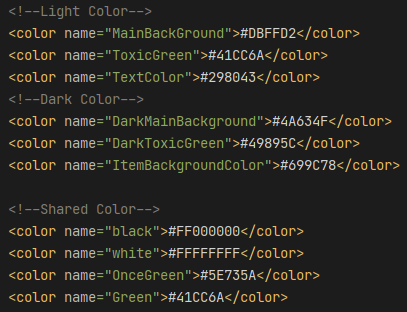


Рисунок 14 – Реализация класса «AppDataBase»

## Разработка мультимедийного контента

Вся верстка приложения ведётся на языке разметки «Xml». Перед началом разработки дизайна все цвета приложения, размер шрифта необходимо вынести в отдельные файлы, как представлено на рисунке 15.



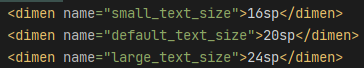


Рисунок 15 – Вынесение основных значений

Для использования иконок в приложении и задних фонов для элементов изображение необходимо хранить в отдельных файлах с расширением «Xml» в папке drawable (Рисунок 16).

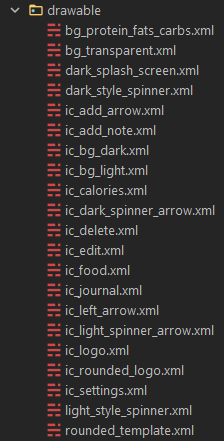


Рисунок 16 –Расположение иконок и задних фонов в отдельных файлах

Каждый экран приложения также хранится в отдельном файле (Рисунок 17) с расширением «Xml». Все экраны находятся в одной папке с название layout.

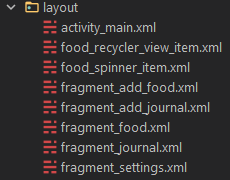


Рисунок 17 – Экраны приложения

Для реализации светлой и темной темы в приложении необходимо прописывать стили для каждого файла themes, представленных на   
рисунке 18.



Рисунок 18 – Файлы themes для светлой и темной темы

Примеры реализации стиля для задней поверхности элемента светлой и темной темы представлены на рисунках 19 и 20 соответственно.

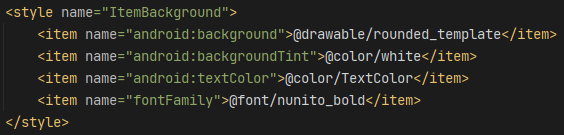


Рисунок 19 – Стиль задней поверхности элемента для светлой темы приложения

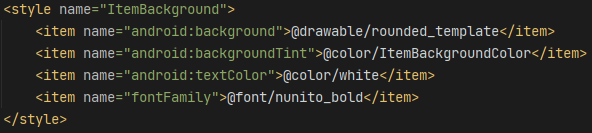


Рисунок 20 – Стиль задней поверхности элемента для темной темы приложения

## Описание используемых плагинов

При создании проекта по умолчанию в «build.gradle (:app)» устанавливаются следующие зависимости, представленные на рисунке 21.

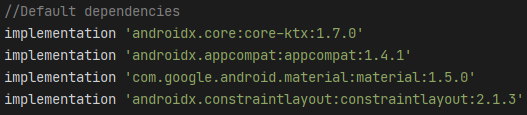


Рисунок 21 – Предустановленные зависимости

Для проведения тестов приложения важно установить зависимости, представленные на рисунке 22.

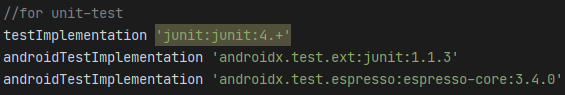


Рисунок 22 – Зависимости для проведения Unit-тестов

Чтобы реализовать локальную базу данных, необходимо установить библиотеку Room (Рисунок 23).

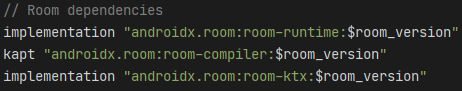


Рисунок 23 – Установка библиотеки Room

Для реализации паттерна проектирования MVVM нужно прописать зависимости, представленные на рисунке 24.

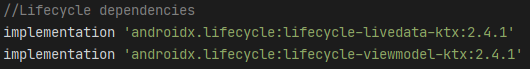


Рисунок 24 – Библиотеки для реализации паттерна проектирования

Чтобы успешно реализовать заданный интерфейс приложения, а также сделать его масштабируемым под разные расширения экранов необходимо установить следующие зависимости (Рисунок 25)

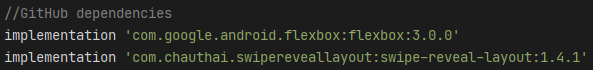


Рисунок 25 – Зависимости для реализации интерфейса

Для реализации асинхронности в мобильном приложении необходимо установить библиотеку Coroutines (Рисунок 26)



Рисунок 26 – Установка зависимости Coroutines

## Описание разработанных процедур и функций

В данном проекте используется чистая архитектура. В данной архитектуре вся бизнес-логика выносится в отдельные классы (Рисунок 27), т.к. это позволит улучшить читаемость кода, тестируемость и самое главное упростит процесс нахождения ошибок в приложении.

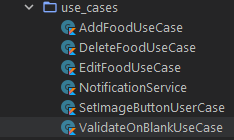


Рисунок 27 − Бизнес-логика приложения

Класс «AddFoodUseCase» отвечает за внесение данных о еде в локальную базу данных. Реализация данного класса представлена на   
рисунке 28.

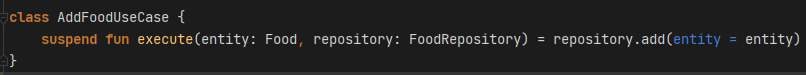


Рисунок 28 – Реализация класса «AddFoodUseCase»

Для редактирования данных о еде необходимо разработать класс «EditFoodUseCase» (Рисунок 29).

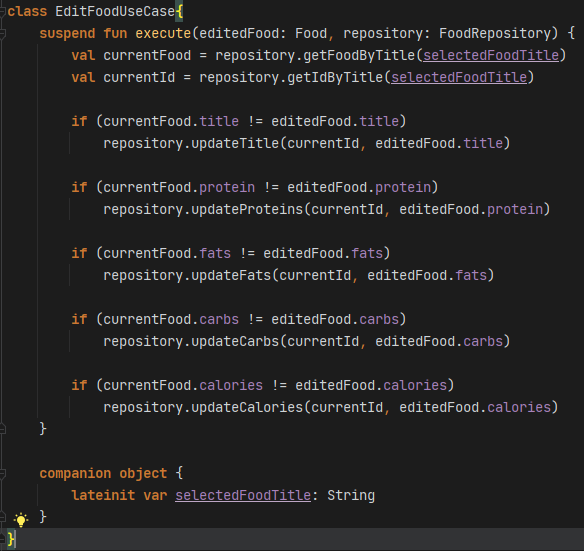


Рисунок 29 – Реализация класса «EditFoodUseCase»

Чтобы была возможность удалять еду из списка нужно разработать класс «DeleteFoodUseCase», реализация которого представлена на рисунке 30

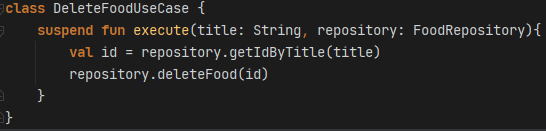


Рисунок 30 – Реализация класса «DeleteFoodUseCase»

Для проверки введенных данных на наличие не заполненных полей нужно написать реализацию для класса «ValidateOnBlank». Реализация данного класса представлена на рисунке 31.

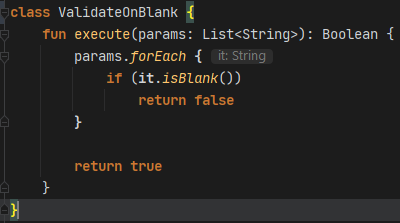


Рисунок 31 – Реализация класса ValidateOnBlank

Класс «NotificationService» (Рисунок 32) отвечает за уведомления. Это необходимо, чтобы уведомлять пользователя об ошибках ввода данных, добавлении и изменении данных в локальной базе.

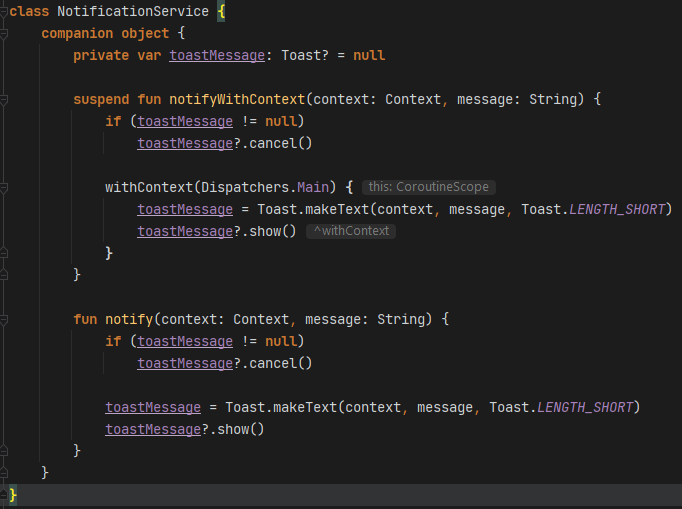


Рисунок 32 – Реализация класса «NotificationService»

Класс «SetImageButton» отвечает за изменение картинки на кнопке добавления при изменении фрагмента. Реализация данного класса представлена на рисунке 33.

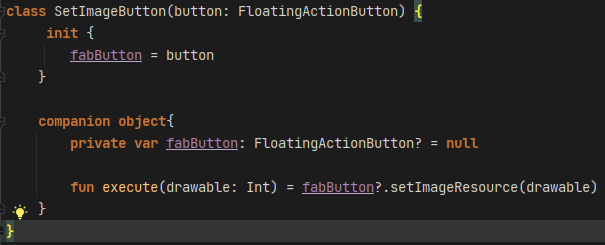


Рисунок 33 – Реализация класса «SetImageButton»

Также для ведения дневника питания необходимо реализовать классы «AddJournalNoteUseCase», «DeleteJournalNoteUseCase». Реализации данных классов представлены на рисунках 34, 35 соответственно.

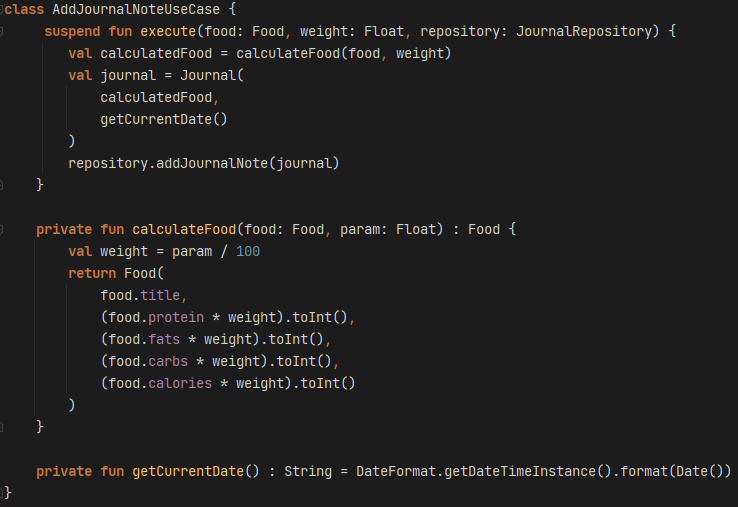


Рисунок 34 – Реализация класса «AddJournalNoteUseCase»

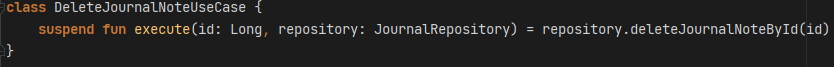


Рисунок 35 – Реализация класса «DeleteJournalNoteUseCase»

# Тестирование

## Протокол тестирования дизайна приложения

Тестирование дизайна будет проводиться на самом минимальном (Android SDK 24) и на более позднем (Android SDK 29) с увеличенным масштабом дабы проверить масштабируемость интерфейса.

Каждый экран успешно прошёл проверку на наличие грамматических ошибок. Также каждый экран прошёл проверку на разных SDK на корректное отображение элементов соответствующих экранов. Примеры проверок отображения элементов на экране представлены на   
рисунках 36, 37, 38 соответственно.

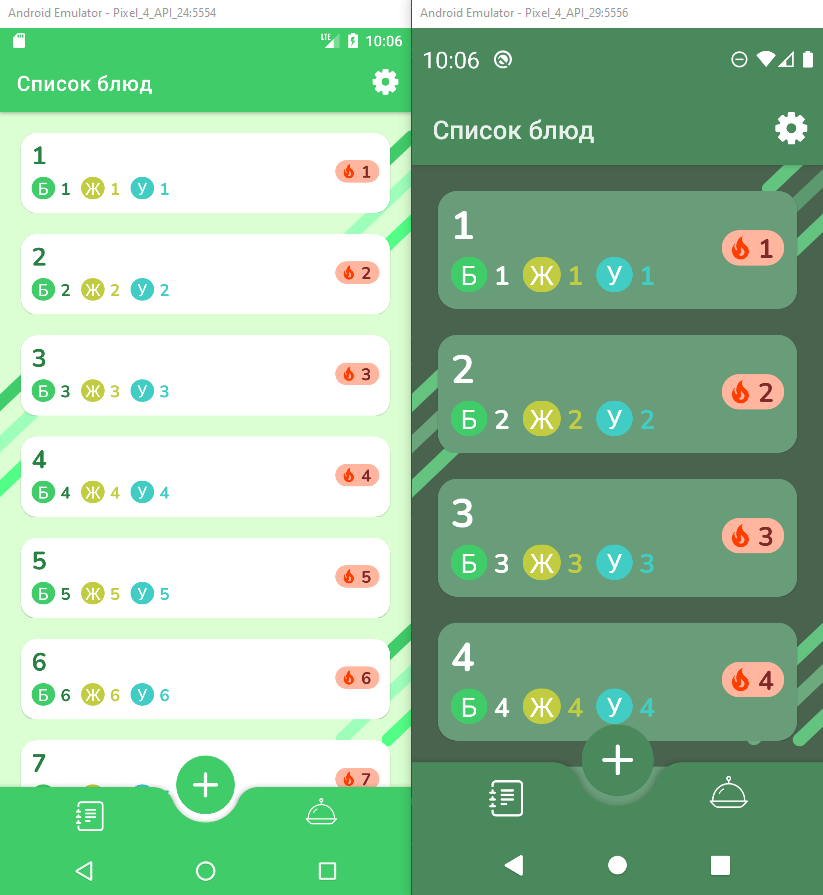


Рисунок 36 – Экран «Список блюд» на 24 и 29 SDK соответственно

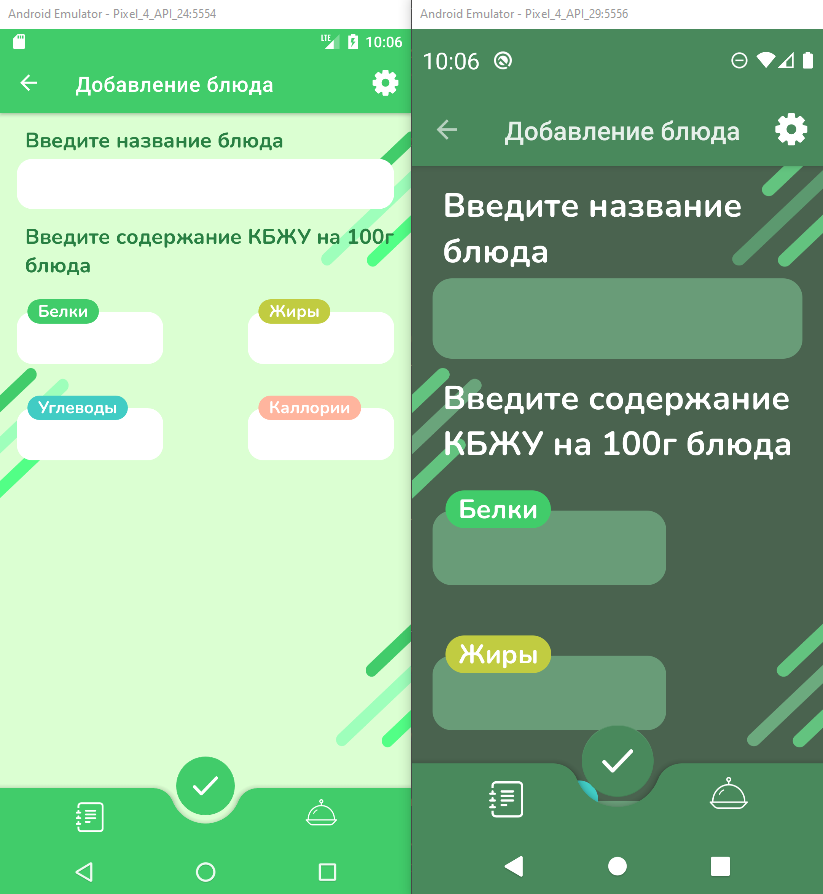


Рисунок 37 – Экран «Добавление блюда» на 24 и 29 SDK соответственно

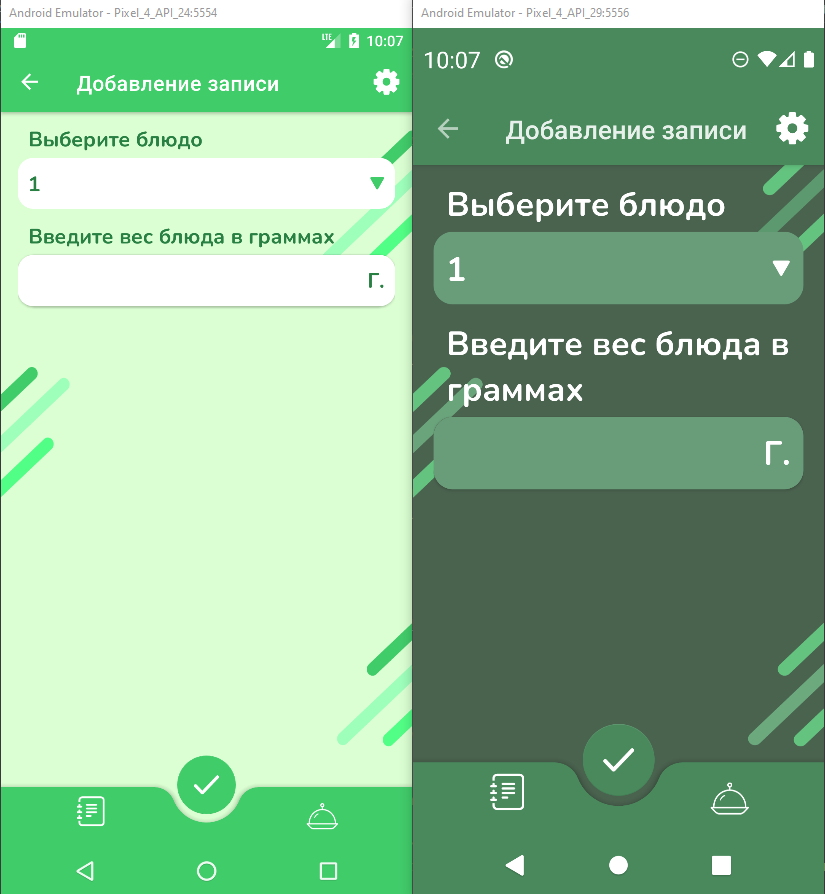


Рисунок 38 − Экран «Добавление записи» на 24 и 29 SDK соответственно

В процессе тестирование не было выявлено критических ошибок в дизайне приложения ни на 24, ни на 29 SDK.

## Протокол тестирования функционала приложения

Чтобы убедиться в правильности работы разрабатываемой   
бизнес-логики, необходимо провести тестирование разработанных классов: «AddFoodUseCase», «EditFoodUseCase», «DeleteFoodUseCase», «ValidateOnBlank». Реализация Unit-тестов для данных классов представлены на рисунках 39, 40, 41, 42.



Рисунок 39 – Unit-тест для «AddFoodUseCase»

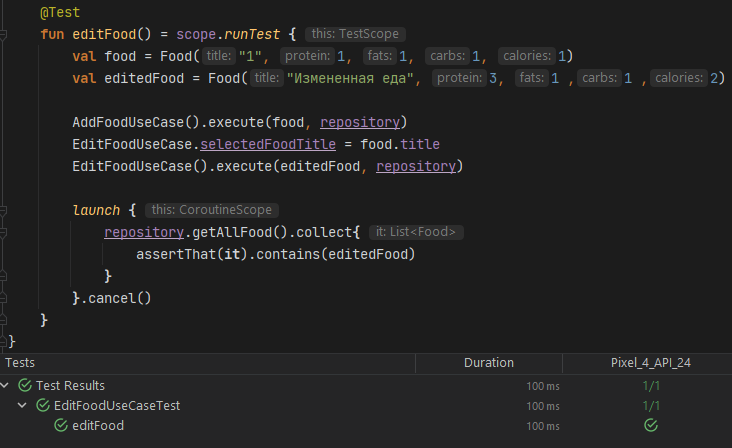


Рисунок 40 – Unit-тест для «EditFoodUseCase»



Рисунок 41 – Unit-тест для «DeleteFoodUseCase»

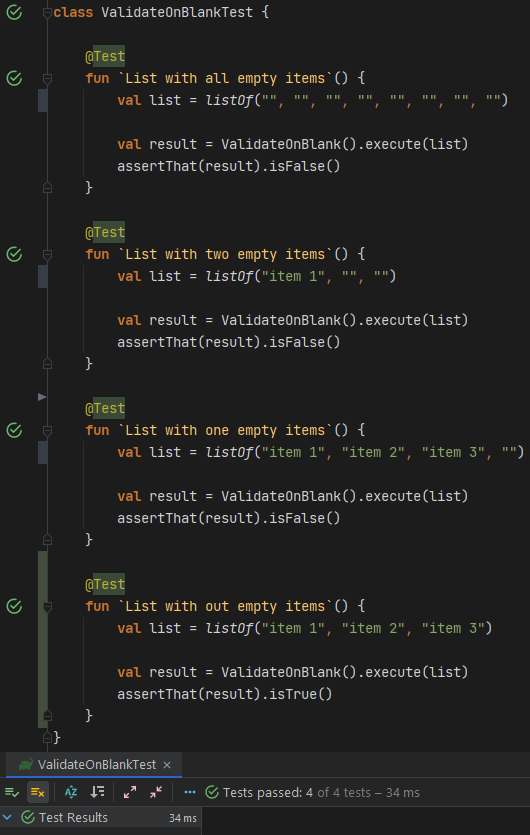


Рисунок 42 – Unit-тест для «ValidateOnBlank»

Аналогичные тесты также успешно прошли «AddJournalNoteUseCase», «DeleteJournalNoteUseCase» на правильность запросов к базе данных.

# ЗАКЛЮЧЕНИЕ

В ходе изучения предметной области было выявлено, что экономия времени при ведении ежедневного дневника питания является преобладающим параметром, на который было акцентировано все внимание при разработке мобильного приложения. Чтобы данный процесс являлся менее трудоемким и времязатратным определен основной функционал приложения, который позволит вести ежедневный дневник питания. Чтобы реализовать поставленный функционал мобильного приложения, была выбрана среда разработки Android Studio и язык программирования Kotlin.

С целью реализации не перегруженного дизайна определены и разработаны основные экраны приложения. Данные экраны имеют лаконичный и минималистичный дизайн без лишних элементов, которые могли бы отвлечь внимание пользователя.

Для реализации возможности сохранения всех необходимых записей пользователя разработана локальная база данных SQLite при помощи библиотеки Room.

В целях проверки функционала мобильного приложения необходимо провести Unit-тесты для бизнес-логики приложения. Все разработанные функции успешно прошли Unit-тесты.

Таким образом, поставленная цель курсовой работы была достигнута.

# СПИСОК ИСТОЧНИКОВ

1. Developers Android [Библиография] Сайт с документацией поAndroid Studio. − Режим доступа: [developer.android.com/studio/intro](https://developer.android.com/studio/intro).
2. Kotlinlang [Библиография] Сайт с документацией по Kotlin. − Режим доступа: [kotlinlang.org/docs/basic-syntax.html#functions](https://kotlinlang.org/docs/basic-syntax.html#functions).
3. Material Design [Библиография] Bottom navigation. − Режим доступа: [material.io/components/bottom-navigation](https://material.io/components/bottom-navigation).
4. Material Design [Библиография] Floating Action Button. − Режим доступа: [material.io/components/buttons-floating-action-button](https://material.io/components/buttons-floating-action-button).
5. Metanit [Библиография] Сайт о программировании. − Режим доступа: [metanit.com/kotlin/tutorial](https://metanit.com/kotlin/tutorial/).
6. StackOverflow [Электронный ресурс] Система вопросов и ответов о программировании. − Режим доступа: [stackoverflow.com/questions](https://stackoverflow.com/questions).
7. YouTube [Электронный ресурс]: Плейлист. Уроки по архитектуре. − Режим доступа: [www.youtube.com/playlist?list=PLeF3l86ZMVkLQbdRL6Ra4cr  
   \_cmPROj94y](http://www.youtube.com/playlist?list=PLeF3l86ZMVkLQbdRL6Ra4cr_cmPROj94y).
8. YouTube [Электронный ресурс]: Плейлист. [Основы разработки Android-приложений](https://www.youtube.com/playlist?list=PLRmiL0mct8WnodKkGLpBN0mfXIbAAX-Ux). − Режим доступа:[www.youtube.com/playlist?list  
   =PLRmiL0mct8WnodKkGLpBN0mfXIbAAX-Ux](https://www.youtube.com/playlist?list=PLRmiL0mct8WnodKkGLpBN0mfXIbAAX-Ux).
9. YouTube [Электронный ресурс] How to Make a Clean Architecture Cryptocurrency App (MVVM, Use Cases, Compose) − Android Studio. −Режим доступа: [www.youtube.com/watch?v=EF33KmyprEQ](http://www.youtube.com/watch?v=EF33KmyprEQ).

# Приложение А

Введение

1 Основания для разработки

2 Назначения для разработки

3 Требование к приложению

3.1 Требования к функциональным характеристикам

3.2 Требования к надёжности

3.3 Условия эксплуатации

3.4 Требования к составу и параметрам технических средств

3.5 Требования к информационной и программной совместимости

3.6 Требования к защите информации

3.7 Требования к маркировке и упаковке

4 Требования к программной документации

5 Технико-Экономические показатели

6 Стадии и этапы разработки

7 Порядок контроля и приемки