**Федеральное государственное образовательное бюджетное учреждение высшего образования**

**«ФинансовЫЙ УНИВЕРСИТЕТ при Правительстве Российской Федерации»**

Факультет информационных технологий и анализа больших данных

**КУРСОВАЯ работа**

на тему:

«Фирма грузоперевозок»

Руководитель:

Старший преподаватель

Гущин С. И.

Москва 2023

Оглавление

[ВВЕДЕНИЕ 3](#_Toc133976149)

[ОСНОВНАЯ ЧАСТЬ 5](#_Toc133976150)

[Глава 1. Постановка задачи 5](#_Toc133976151)

[Глава 2. Описание предметной области 6](#_Toc133976152)

[Глава 3. Актуальность автоматизации 7](#_Toc133976153)

[Глава 4. Описание программы 8](#_Toc133976154)

[4.1. Алгоритмические решения 8](#_Toc133976155)

[4.2. Описание интерфейса программы 22](#_Toc133976156)

[ЗАКЛЮЧЕНИЕ 42](#_Toc133976157)

# ВВЕДЕНИЕ

В современном мире транспортировка грузов является неотъемлемой частью бизнеса, и компании, занимающиеся этой сферой, постоянно ищут новые способы улучшения своих услуг. Одним из ключевых аспектов успешной транспортировки является контроль за грузом на каждом этапе доставки. Для этого необходимы инструменты, позволяющие отправителям и получателям отслеживать перемещение своих посылок и получать информацию о статусе доставки.

В данной курсовой работе будет рассмотрено создание мобильного приложения для отслеживания грузов по трек-номеру. Это приложение позволит пользователям контролировать перемещение своих посылок на каждом этапе доставки, получать уведомления о изменении статуса доставки и получать информацию о сроках доставки и условиях хранения груза.

Основная цель данной работы – разработка функционального и удобного мобильного приложения, которое будет полезным инструментом для компаний, занимающихся транспортировкой грузов, а также для обычных пользователей, отправляющих или получающих посылки.

Для достижения этой цели необходимо рассмотреть основные функциональные возможности приложения, такие как отслеживание груза по трек-номеру, уведомления о статусе доставки, информация о доставке и контакты службы поддержки. Также необходимо уделить внимание техническим особенностям реализации приложения, таким как использование API, разработка серверной части и мобильного приложения, а также тестирование приложения на различных устройствах и операционных системах.

В результате данной работы будет создано полноценное мобильное приложение для отслеживания грузов по трек-номеру, которое будет полезным инструментом для компаний и обычных пользователей. Разработка приложения позволит повысить уровень доверия клиентов и обеспечить более эффективный контроль за доставкой грузов.

# ОСНОВНАЯ ЧАСТЬ

## Глава 1. Постановка задачи

В рамках курсовой работы требуется разработать приложение (Клиент), которое является графическим интерфейсом и инструментом управления приложения, и программу, которая занимается хранением и обработкой информации (Сервер), выполняющее добавление, изменения и отображение объектов предметной области, хранящихся в базе данных (БД), и предоставляющее справочную информацию.

Для реализации клиентской части был выбран язык Kotlin, язык, работающий поверх Java, язык позволяет создавать мобильные высоконагруженные приложения под платформу Android. Интерфейс клиентской части реализован на основе языка разметки XML. Главный инструмент разработки клиентской части – это Android Studio. Android Studio – это **интегрированная среда разработки (IDE), основанная на IntelliJ IDEA, которая предоставляет множество инструментов и функций для упрощения создания, отладки и тестирования Android-приложений.**

Для реализации серверной части был выбран фреймворк Ktor на основе языка Kotlin, который позволяет создавать веб-приложения, использующий корутины для асинхронности и предлагающий идиоматический API. В качестве Для реализации серверной части была использована **IntelliJ IDEA, которая является мощной и удобной средой разработки для языка Kotlin, поддерживающей автодополнение кода, рефакторинг, отладку и тестирование.**

Для реализации слоя данных была выбранна PosgreSQL база данных, которая является открытым и надежным решением для хранения и обработки разнообразных данных, поддерживающим SQL и NoSQL запросы, а также множество расширений и инструментов для аналитики и масштабирования.

Далее необходимо определить функционал будущего приложения. После изучения предметной области темы, возникла необходимость в составлении следующих функций приложения:

* Безопасная авторизация и регистрация пользователей;
* Отслеживание груза по трек-номеру.
* Удаление груза по трек-номеру
* Создание и редактирование информации о трек-номере
* Статистика по прогрессу подзадач;

## Глава 2. Описание предметной области

Грузовые перевозки являются предметом исследования данной курсовой работы. Объектом анализа служит фирма “АлпоТрэк”, которая располагает собственным автопарком из 20 автомобилей, задействованных в доставке грузов. Все грузы, отправляемые фирмой, проходят процесс упаковки и подготовки к транспортировке. Курьеры, ответственные за перевозку грузов, обязаны соблюдать требования к качеству и срокам доставки. Однако в условиях динамично растущего рынка грузоперевозок возникает необходимость контроля и мониторинга текущего статуса грузов, находящихся в пути. Для решения этой проблемы разработано специальное приложение, которое позволяет каждому получателю отслеживать свою посылку в режиме реального времени. Это приложение обеспечивает прозрачность и надежность в доставке грузов, а также повышает удобство для клиентов. Они могут легко связаться с курьером или диспетчером, чтобы получить информацию о своей посылке или решить любые возможные проблемы. Благодаря этому фирма “АлпоТрэк” может обеспечить высокое качество сервиса и удовлетворенность клиентов.

## Глава 3. Актуальность автоматизации

Приложение помогает оптимизировать маршруты доставки и распределение грузов по автомобилям, что позволяет снизить затраты на топливо и увеличить эффективность работы автопарка. Такой подход повышает конкурентоспособность фирмы на рынке грузоперевозок. Кроме того, приложение предоставляет возможность отслеживать грузы в режиме реального времени. Клиенты могут узнать, в каком состоянии находится их посылка. Это дает им больше контроля над процессом доставки и уменьшает вероятность потери груза. Кроме того, приложение может быть использовано для автоматизации процессов управления складом и инвентаризации. Оно позволяет быстро и точно отслеживать поступление и отгрузку товаров, а также контролировать остатки на складе. Это повышает эффективность работы склада и уменьшает вероятность ошибок.

Кроме того, приложение обеспечивает безопасность грузов и минимизирует риски их повреждения или потери. Система отслеживания и контроля позволяет оперативно реагировать на любые непредвиденные ситуации и принимать меры по предотвращению возможных проблем. Важным преимуществом приложения является его удобство и простота использования. Клиенты могут легко отслеживать статус заказа и получать информацию о доставке. Это экономит время и упрощает процесс сотрудничества с фирмой.

В целом, использование современных технологий и инструментов позволяет любой фирме грузоперевозок улучшить свою работу, повысить качество обслуживания и удержать лидерство на рынке грузоперевозок.

## Глава 4. Описание программы

### 4.1. Алгоритмические решения

Основная часть работы заключается в хранении и обработки данных. Для целей хранения данных была выбрана PostgreSQL база данных. Аутентификация также реализована с помощью комбинации логин/пароль/токен.

PostgreSQL – это**свободная объектно-реляционная система управления базами данных (СУБД), которая поддерживает широкий спектр функций и стандартов SQL.** PostgreSQL является одной из самых популярных и надежных СУБД в мире.

Для создания базы данных и для создания таблиц в этой базе данных был исопользован инструмент PgAdmin 4. Все созданные в ходе выполнения курсовой работы таблицы в базе данных представлены на рисунке 1.

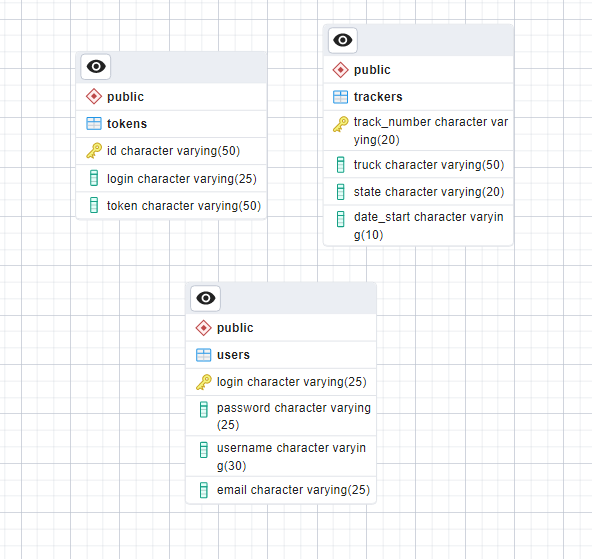


Рисунок 1 – Таблицы в PosgreSQL базе данных

Таблицы, которые были использованы в ходе выполнения курсовой работы:

* Users – данная таблица необходима для хранения информации, связанной с авторизацией и регистрацией пользователей. Хранится логин, пароль, никнейм и почта пользователя;
* Tokens – эта таблица нужна для хранения данных об уникальных устройствах, через которые пользователь проходит авторизацию в приложение. Хранится id токена, логин пользователя и токен устройства, через которое он зашёл;
* Trackers – в данной таблице содержится вся информация о грузах, обращение проходит за счёт уникальных трек-номеров. Хранится трек-номер груза, состояние груза (В пути, прибыл, завершён, на складе), грузовая машина, которая перевозит груз и дата формирования заказа на перевозку груза.

Далее следовало сделать взаимодействие базы данных и серверной части. Для этой реализации был использован фреймворк Ktor и его методы взаимодействия с базой данных. После подключения и обработки инфомарции с базы данных были сформированы JSON файлы в соответствии с REST API архитектурой.

REST API запросы – это **HTTP-запросы, которые используются для взаимодействия с сервером по определенному архитектурному стилю**. REST API позволяет получать, создавать, обновлять и удалять данные с сервера с помощью стандартных методов HTTP: **GET, POST, PUT**и**DELETE**

**JSON формат -** это **текстовый формат обмена данными, основанный на синтаксисе Java Script.**

Для проверки работоспособности REST API, а именно GET и POST запросов я использовал инструмент Postman. Для считывания информации при регистрации и авторизации пользователя, и добавлении, удалении, изменении и просмотра данных о трекере с базы данных была использована библиотека Jetbrains SQL.

Для асинхронного обращения к базе данных была использована библиотека Kotlin Coroutines. Kotlin Coroutines - это **способ писать асинхронный и неблокирующий код на языке Kotlin, который позволяет выполнять множество задач параллельно и эффективно.** Kotlin Coroutines используют концепцию сопрограмм, которые являются легковесными потоками, которые могут быть приостановлены и возобновлены по мере необходимости. Пример suspend функции на этапе авторизации изображён на рисунке 2



Рисунок 2 – suspend функция авторизации пользователя

Для формирования REST API запросов и ответов, возникла необходимость переводить данные в JSON формат, для таких нужд была использована технология Serialization. Serialization на примере регистрации пользователя показана на рисунке 3

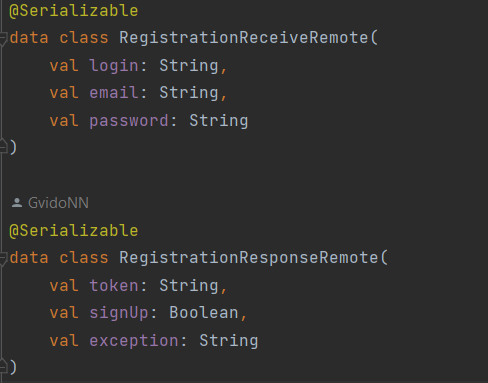


Рисунок 3 – пример технологии Serialization

Далее, для создания пути регистрации мы использовали функцю routing, пример использования данной функции на этапе формирования регистрации изображён на рисунке 4.

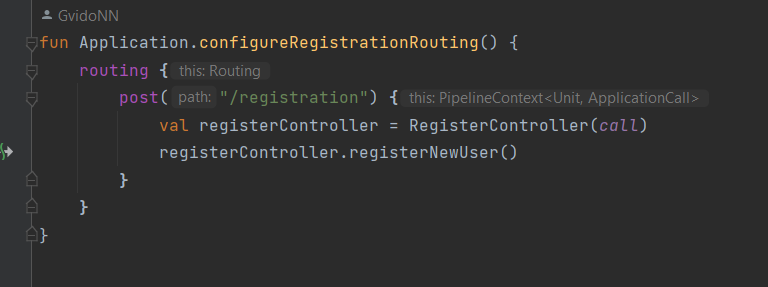


Рисунок 4 – пример routing в регистрации

В случае регистрации пользователя на сервер, подаётся логин, пароль и почта пользователя. В ответ мы получаем токен пользователя, проверку состояния регистрации, и в случае, если при регистрации возникает ошибка, сервер вставляет ошибку в ответ. Для регистрации используем функцию insert библиотеки Jetbrains SQL.

Проверка работоспособности запросов к базе данных, а далее к серверной части при регистрации показано на рисунке 5.

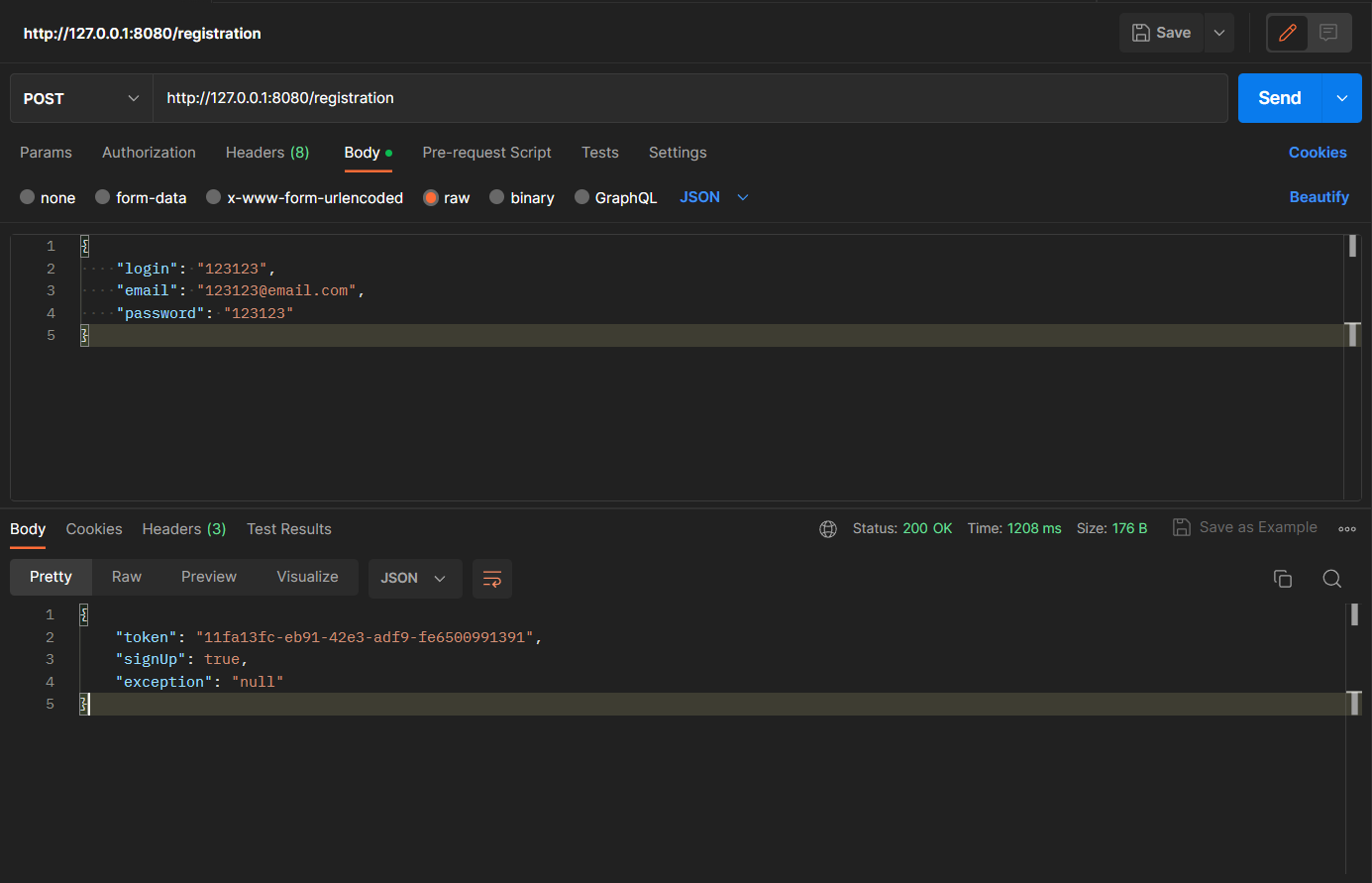


Рисунок 5 – запрос к серверу при регистрации пользователя

Далее был сформирован алгоритм для авторизации пользователя, он также работал через REST API запросы. В запросе подаётся логин и пароль, в ответ сервер отдаёт токен устройства, состояние авторизации, и в случае ошибки, выдаёт эту ошибку. Пример проверки авторизации в Postman изображён на рисунке 6.

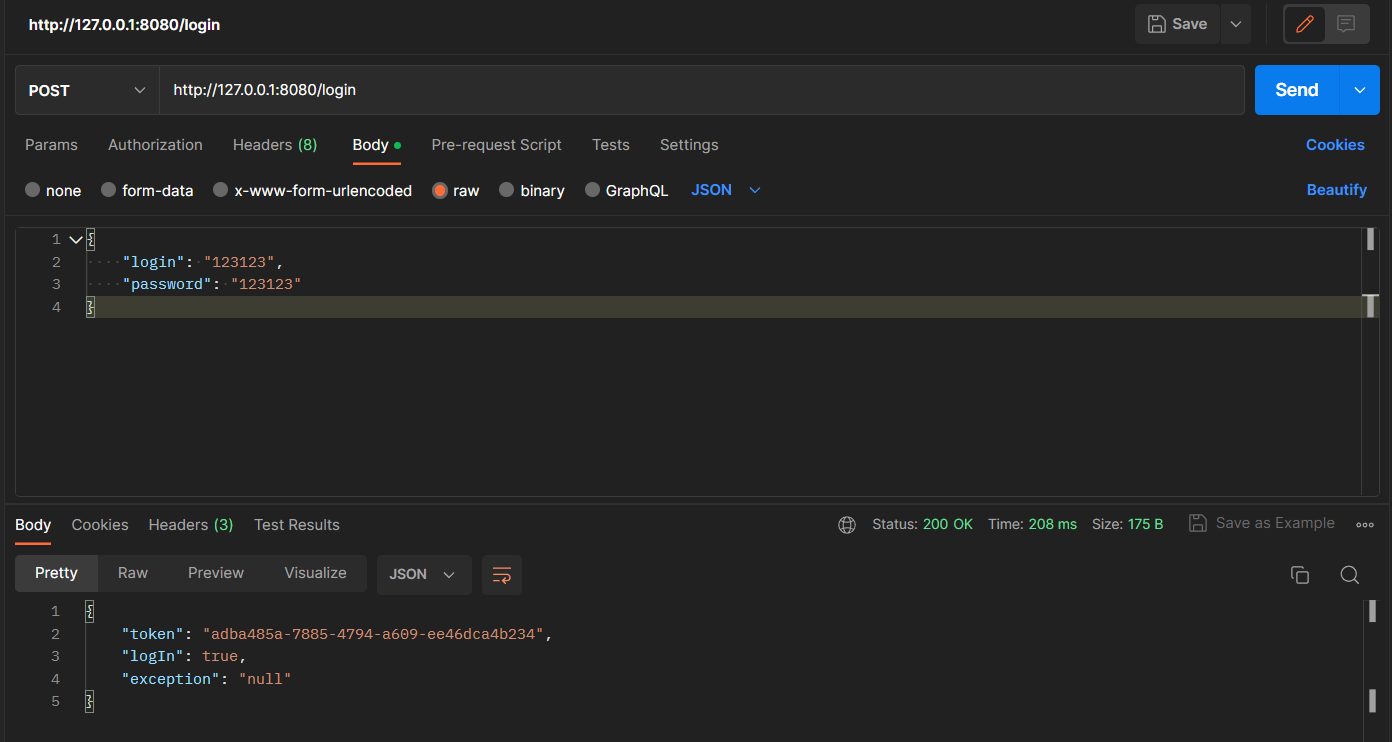


Рисунок 6 – проверка работоспособности авторизации

Далее был реализован алгоритм для добавления трек-номера груза. Для этого мы подаём трек-номер, модель грузовой машины, состояние груза и дату формирования заказа на транспортировку. В ответ получаем состояние добавления и ошибку, если вдруг она есть. Проверка работоспособности добавления приведена на рисунке 7.

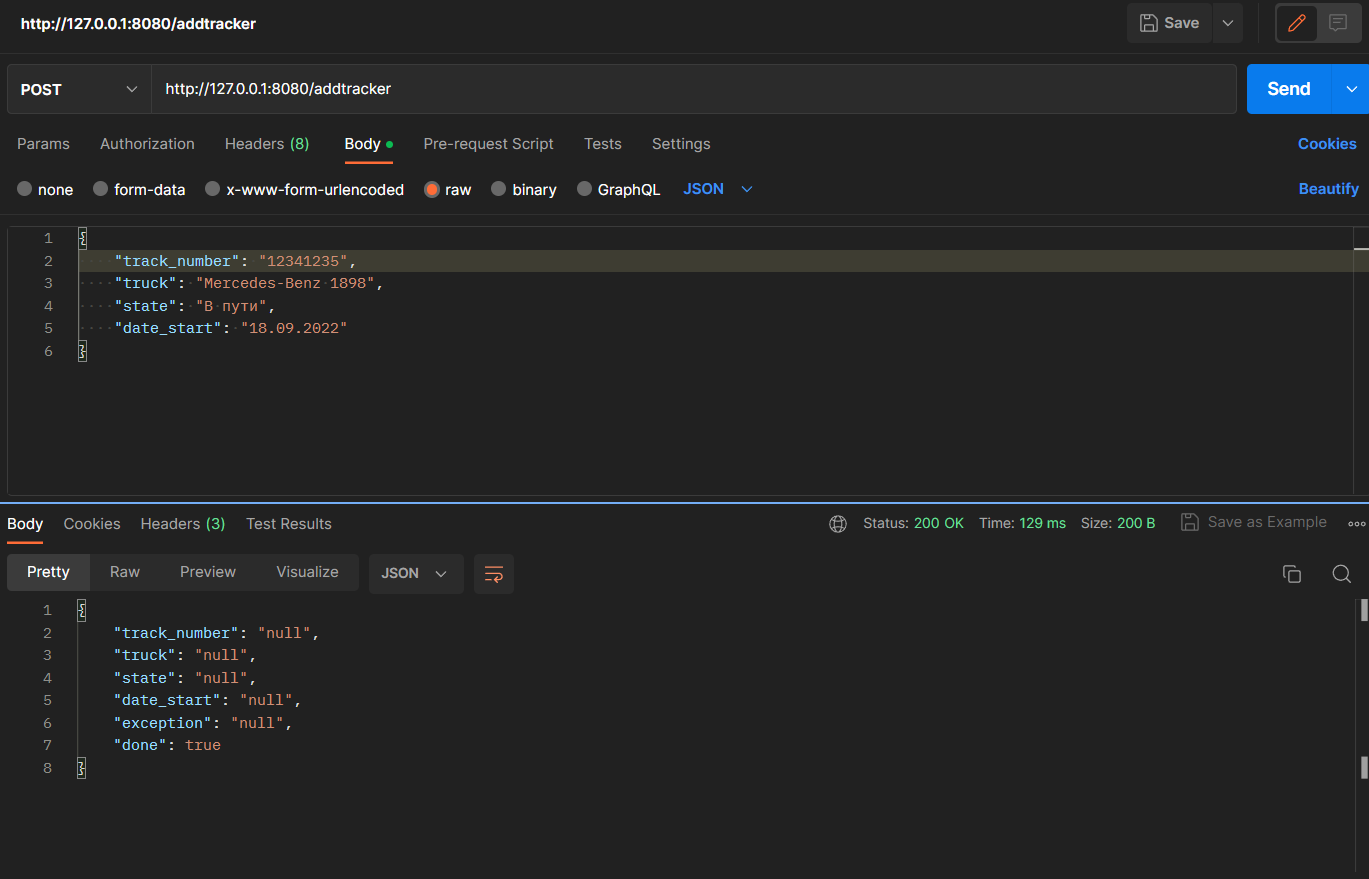


Рисунок 7 – проверка работоспособности добавления трек-номера

Помимо добавления, реализована функция изменения данных о грузе. Реализация функции updateTracker() показана на рисунке 8

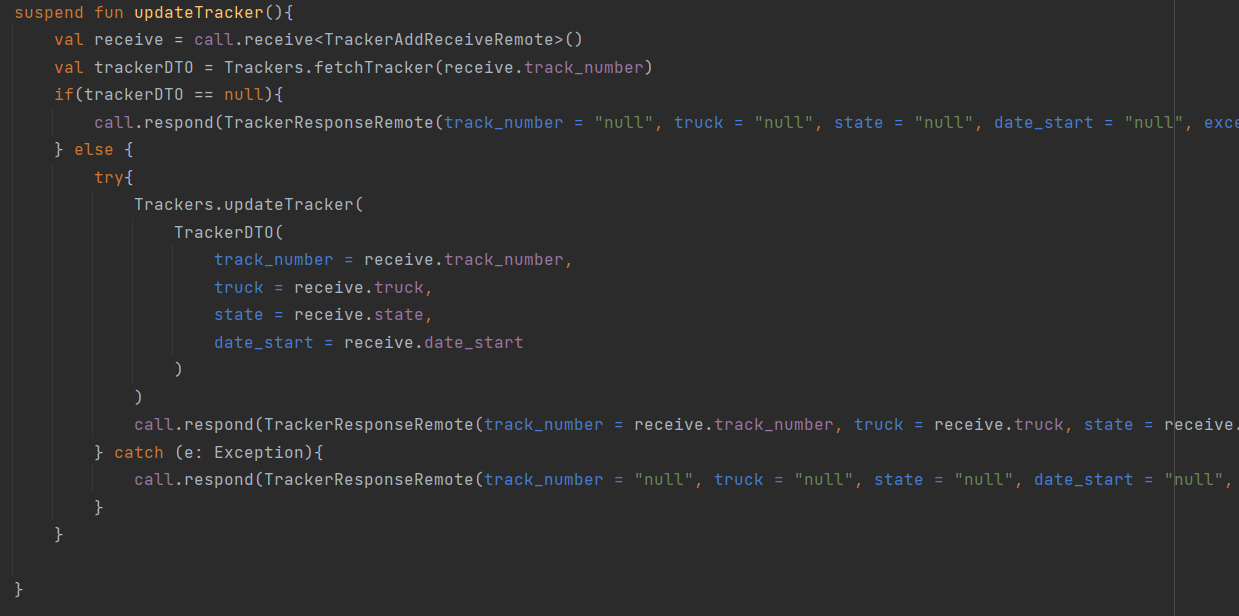


Рисунок 8 – реализация функции updateTracker()

На вход при запросе подаётся вся информация о трек-номере: трек-номер, модель грузовика, состояние, дата формирования заказа. Проверка работоспособности изменения данных приведена на рисунке 9.

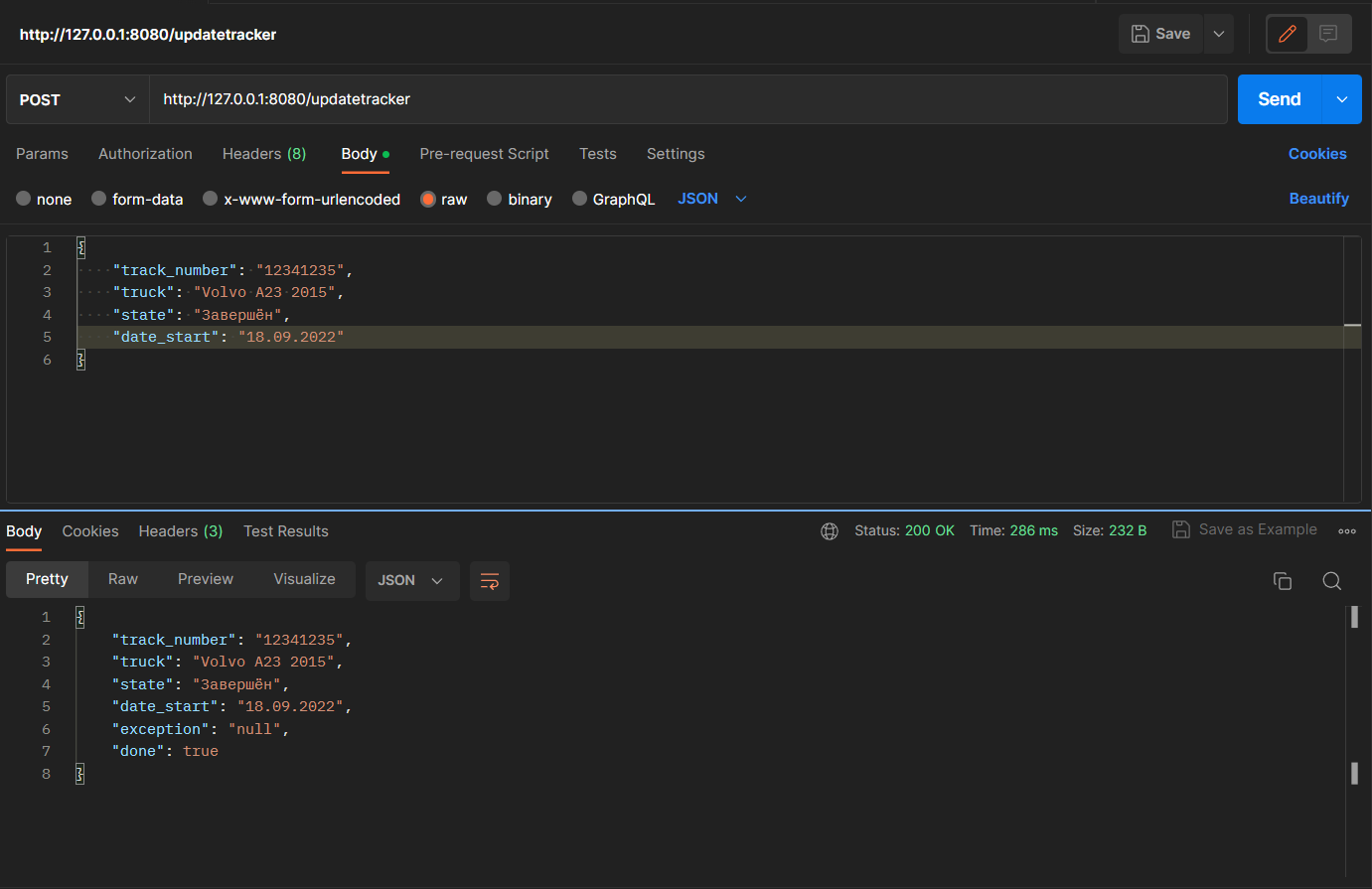


Рисунок 9 – проверка работоспособности изменения данных

Осуществлена реализация отслеживания по трек-номеру, для этого использована функция checkTracker(). Реализация данной функции показана на рисунке 10.

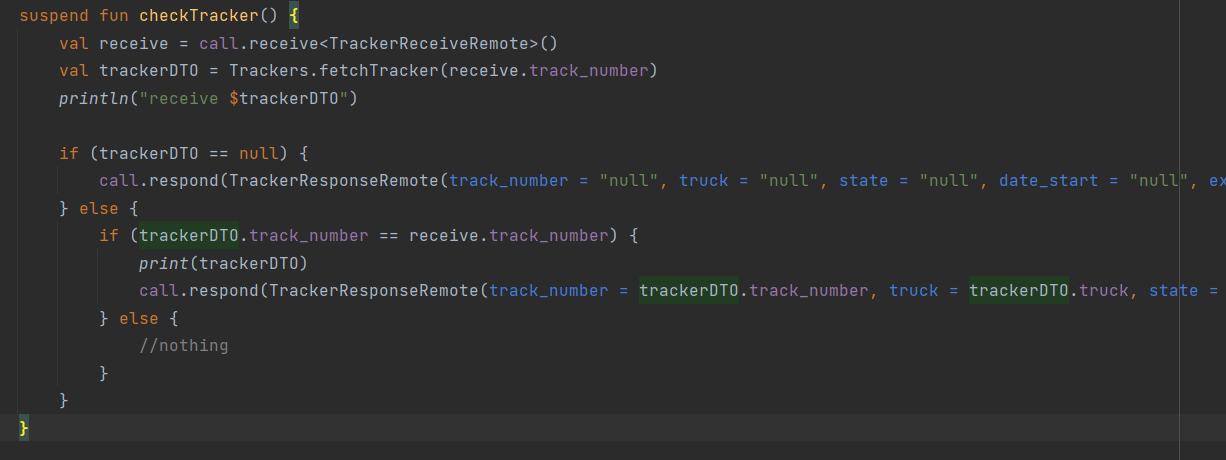


Рисунок 10 – реализация функция отслеживания по трек-номеру

На вход для отслеживания подаётся трек-номер, на выход выходит информация о грузе, состояние ответа, и в случае ошибки, выходит и сама ошибка. Проверка работоспособности отслеживания приведена на рисунке 11

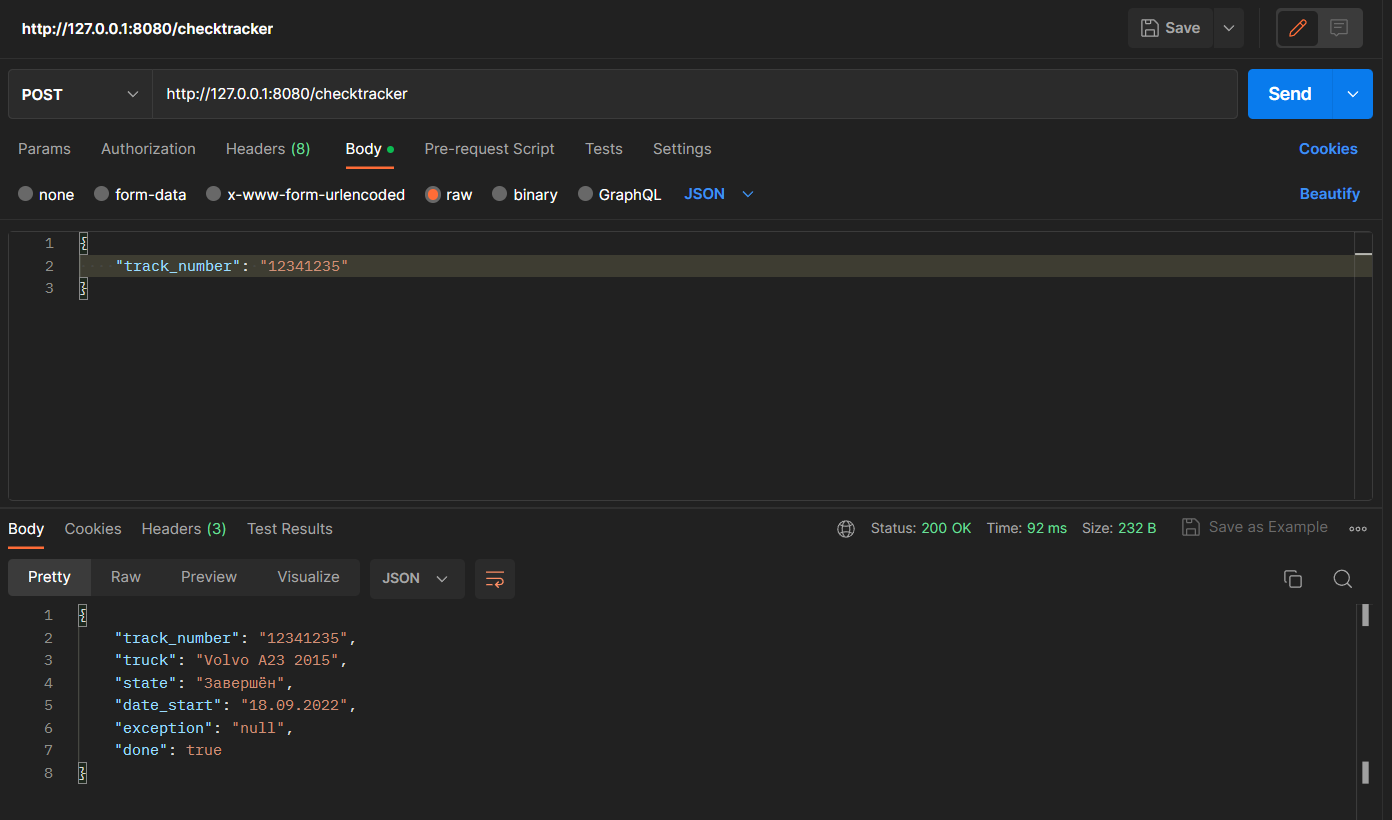


Рисунок 11 – проверка работоспособности отслеживания по трек-номеру

Помимо добавления, реализован алгоритм удаления груза. Для осуществления удаления использована функция deleteTracker(), пример реализации удаления на рисунке 12.

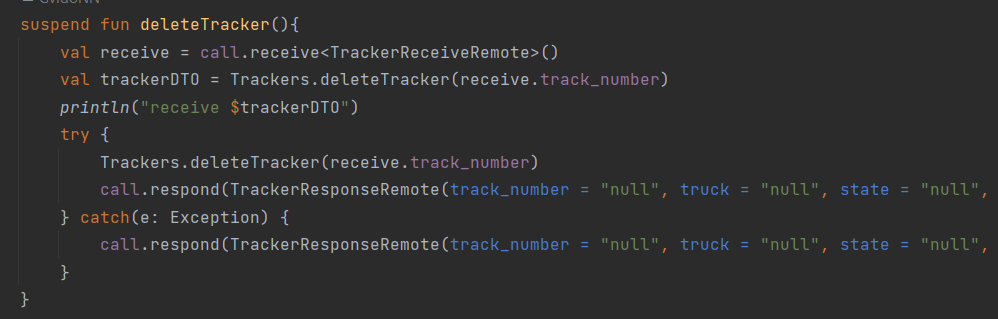
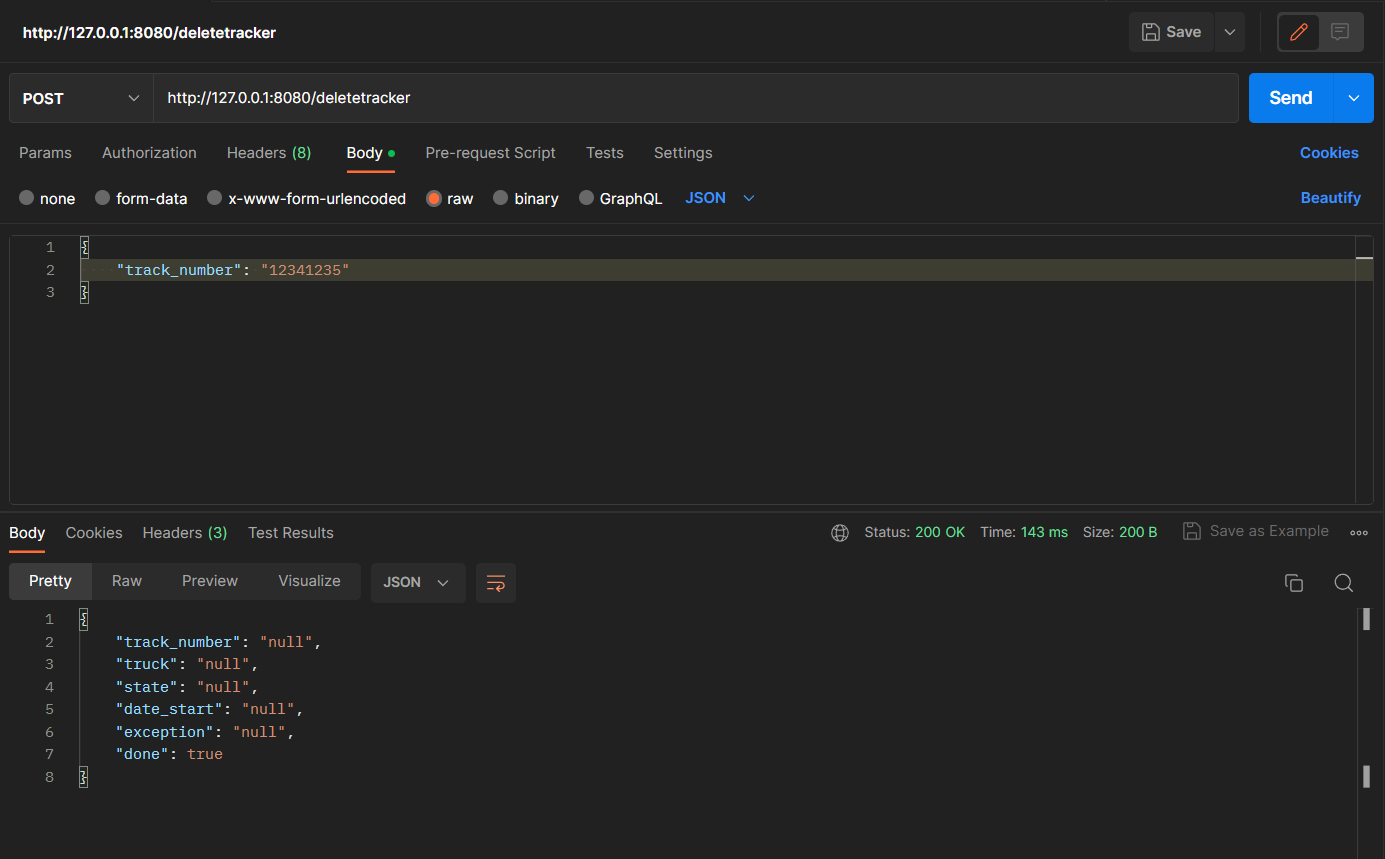


Рисунок 12 – реализация удаления трек-номера

На вход для удаления подаётся трек-номер, на выходе выходит состояние удаления, и в случае ошибки, выдаёт ошибку. Проверка работоспособности удаления трек-номера из базы данных на рисунке 13.

 Рисунок 13 – проверка работоспособности удаления трек-номера

Далее переходим на слой клиента, слой клиента представляет из себя мобильное приложение под платформу Android. Алгоритмические решения почти во всех случаях аналогичны, поэтому покажу реализацию на примере отслеживания, удаления, и изменения информации о грузе.

Для получения данных по REST API технологии была использована библиотека retrofit2 в связке с библиотекой GSONConverter – эти библиотеки позволяют получать данные по прокотоку HTTPS и обрабатывать JSON файлы для своей нужды. Пример использования этой библиотеки на рисунке 14 и 15.

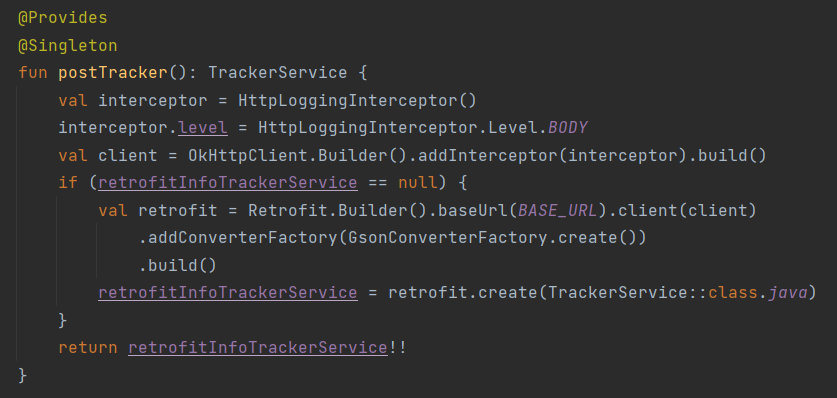


Рисунок 14 – пример использования библиотеки Retrofit

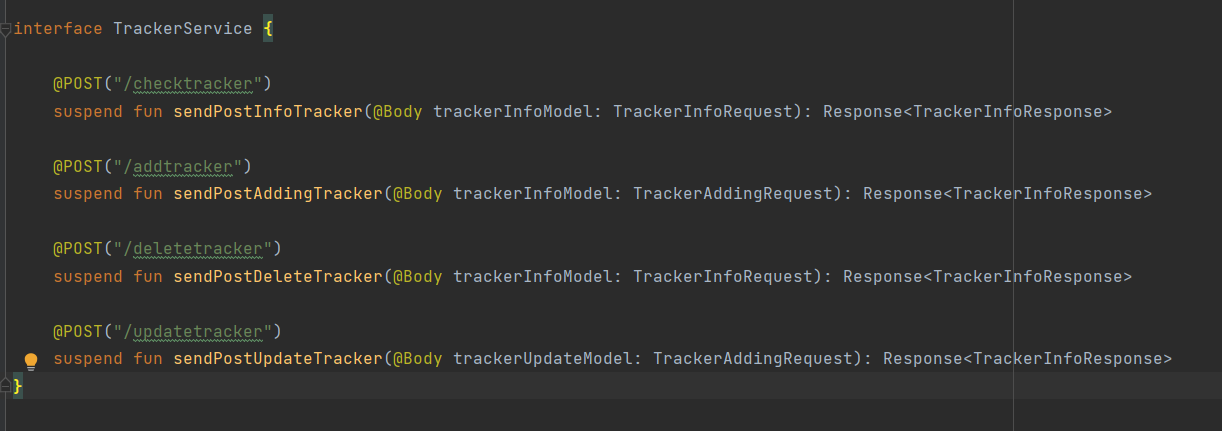


Рисунок 15 – пример использования библиотеки Retrofit

Для обращения к REST API запросам, на слое клиента также использовались корутины (Coroutines), они позволяют сделать обращение асинхронным, а значит, не останавливать основной поток, пока идёт запрос на сервер, пример реализации suspend функций показан на рисунке 16

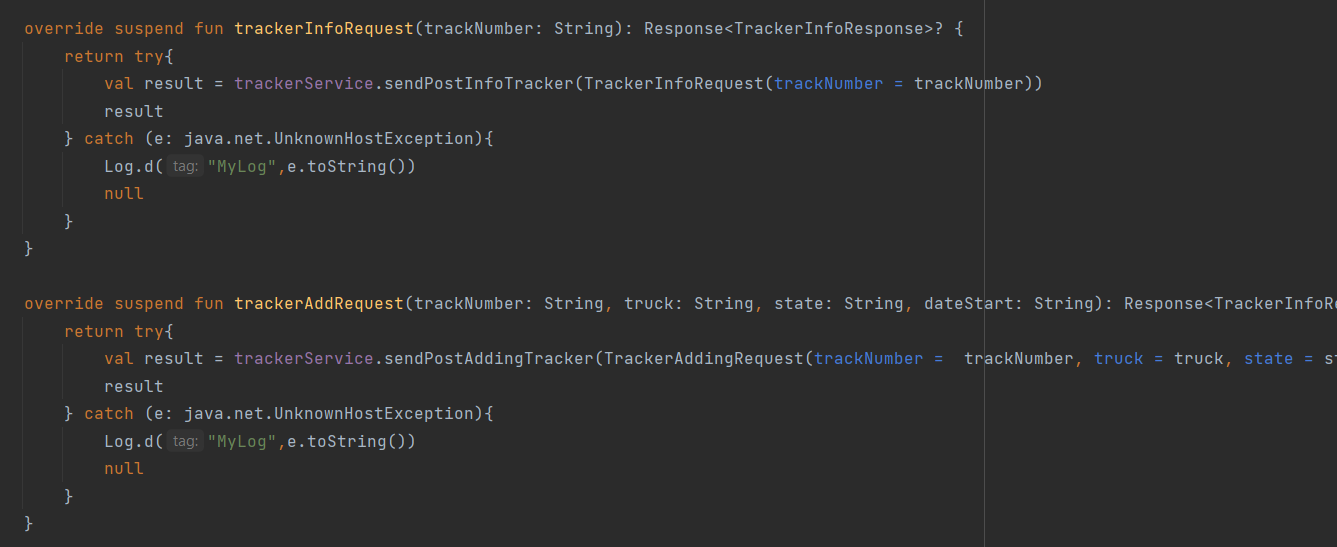


Рисунок 16 – пример реализации корутин в проекте

При реализации слоя клиента был использован паттерн MVVM(Model-View-View-Model) - это **архитектурный паттерн, который позволяет отделить логику приложения от визуальной части (представления).** Паттерн MVVM состоит из трех компонентов: модели (Model), представления (View) и модели представления (ViewModel). Модель описывает данные и бизнес-логику приложения. Представление определяет визуальный интерфейс, с которым взаимодействует пользователь.

Так же, в коде видны аннотации вида @HiltViewModel – данные нотации принадлежат библиотеке Dagger Hilt - это **библиотека для внедрения зависимостей(Depenecy Injection) в Android-приложения, основанная на кодогенерации**. @HiltViewModel - это аннотация, которая помечает классы ViewModel, которые будут внедрены в представления с помощью Hilt.

Для быстрой передачи данных внутри приложения, и отслеживания изменений этих данных используется компонент LiveData, LiveData  - это **класс, который хранит данные и позволяет наблюдать за их изменениями**. В отличие от обычного наблюдателя, LiveData учитывает жизненный цикл других компонентов приложения, таких как активности, фрагменты или сервисы.

Пример реализации ViewModel, Hilt и LiveData показаны на рисунках 17 и 18.

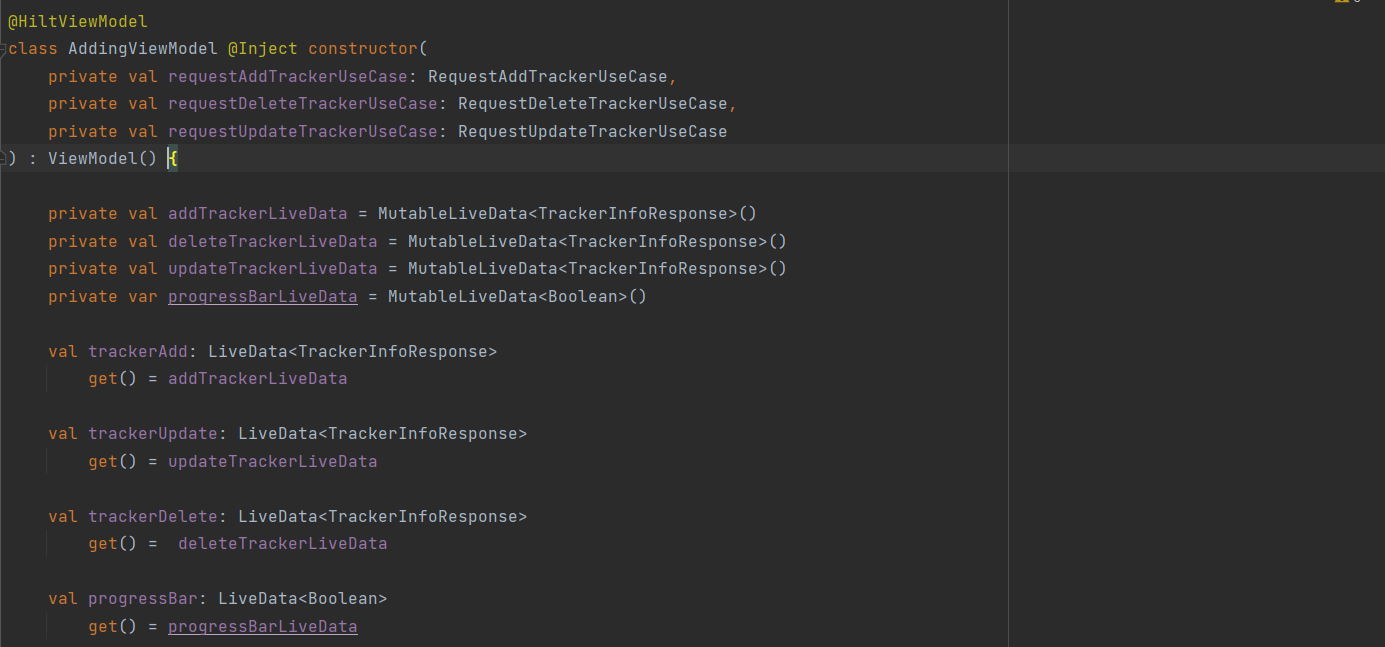


Рисунок 17 – пример реализации ViewModel, Hilt, LiveData

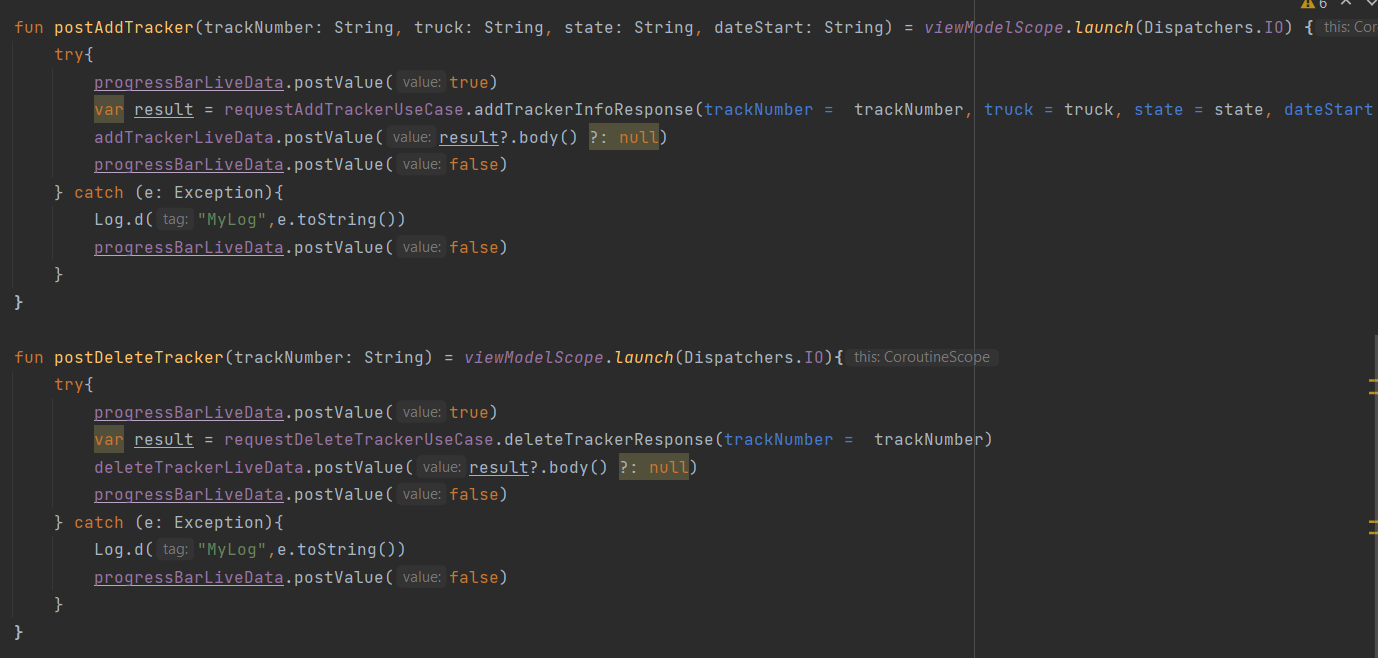


Рисунок 18 – пример реализации ViewModel, Hilt, LiveData

Далее следует слой интерфейса, для обращения к элементам интерфейса я использовал ViewBinding, данная технология позволяет намного быстрее обращаться к представлениям, и в то же время занимает меньше кода при написании. Пример данной технологии в приложении показан на примере реализации функционала кнопки изменения данных о грузе в базе данных, реализация отображена на рисунке 19

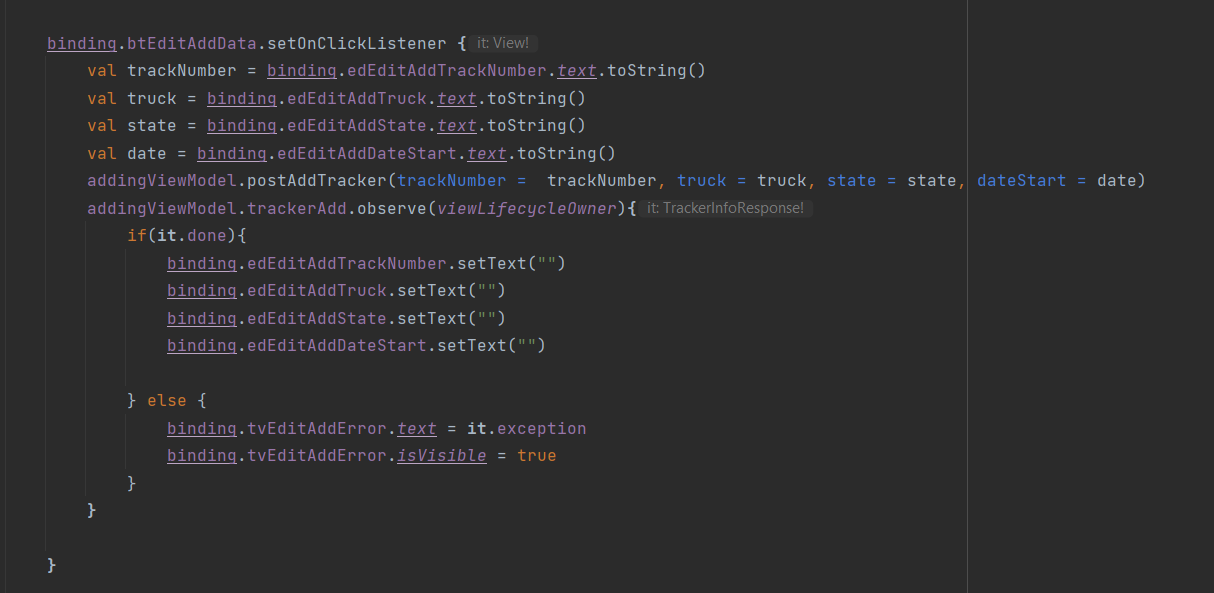


Рисунок 19 – пример использования ViewBinding в приложении

### 4.2. Описание интерфейса программы

Для реализации навигации внутри приложения была использована технология Jetpack Navigation от Google, она позволяет быстро и легко переключаться между экранами в приложении.

Так же, в приложении реализовано сразу два языка, русский и английский, язык автоматически переключается в зависимости от настроек телефона.

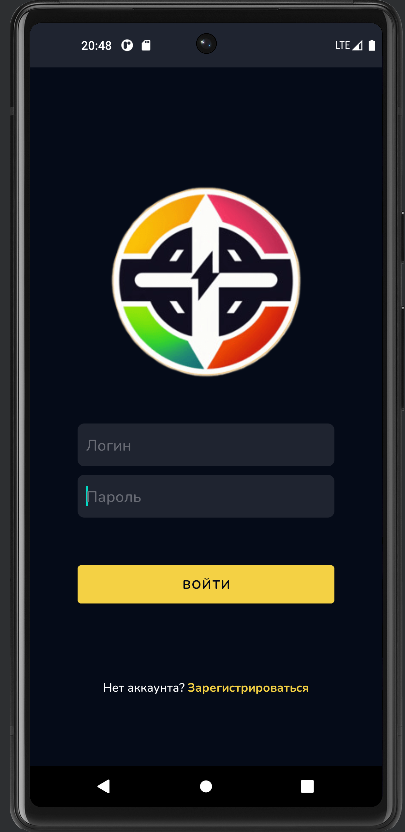
При запуске программы, пользователь сразу попадает на экран авторизации (рисунок 20), там ему предлагается ввести логин и пароль, при успешном вводе и успешной проверке он попадает на экран основного функционала приложения. При отсутствии аккаунта, у пользователя сразу есть возможность попасть на экран регистрации (рисунок 21), на нём пользователю предлагается ввести логин, пароль и почту, после успешной регистрации пользователю снова нужно авторизоваться, а после успешной авторизации, пользователю открывается весь функционал приложения. Схема навигации этапа авторизации показана на рисунке 22. 

Рисунок 20 – экран авторизации

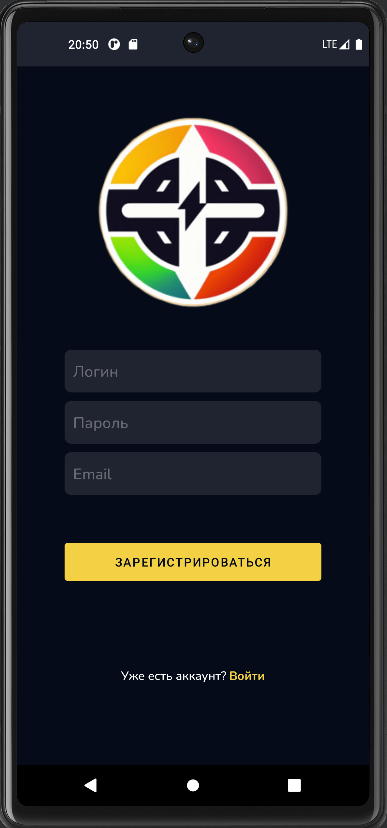


Рисунок 21 – экрана регистрации

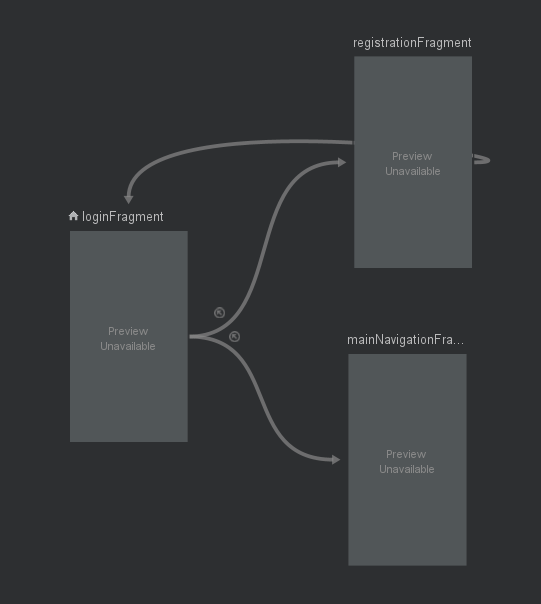


Рисунок 22 – схема навигации при открытии приложения

После прохождения авторизации пользователь попадает на экран трекинга (рисунок 23), который представляет из себя поле, где пользователю предлагается ввести трек-номер для отслеживания груза. При нажатии на кнопку и успешном поиске трек-номера высвечивается информация о грузе (рисунок 24). В случае, если возникает ошибка, то она так же отображается над полем ввода трек-номера (рисунок 25).

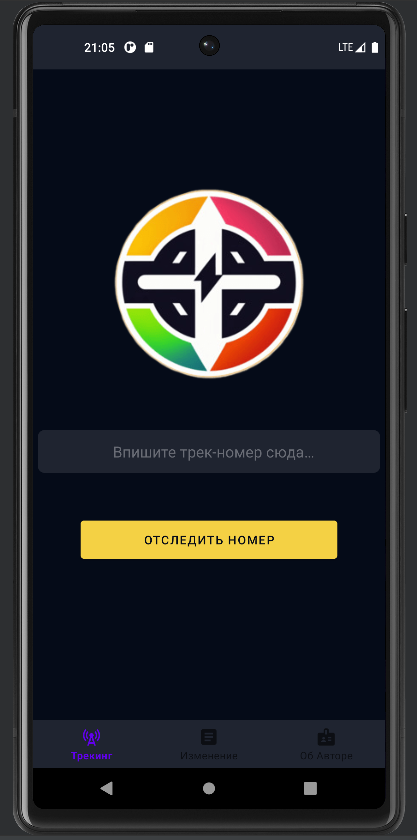


Рисунок 23 – экран трекинга

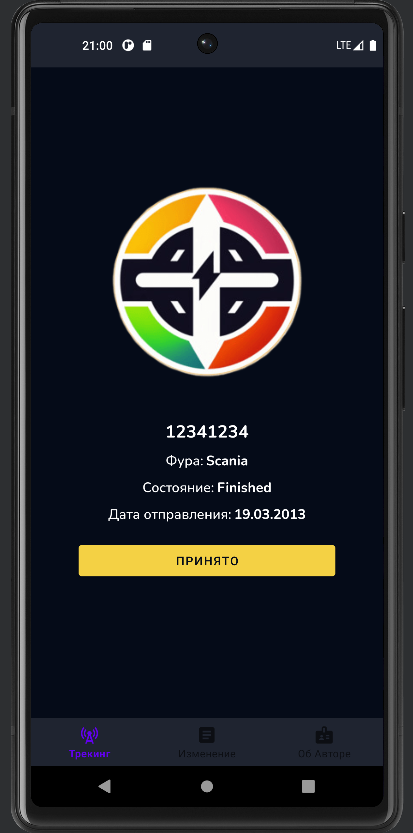


Рисунок 24 – информация о грузе

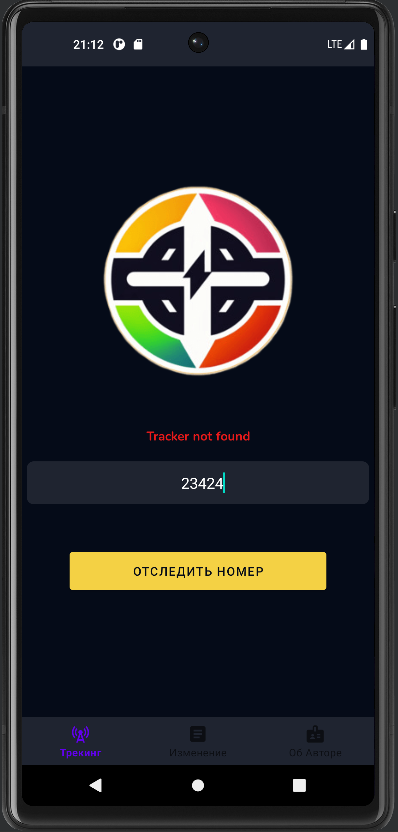


Рисунок 25 – вывод ошибки трек-номера

Далее пользователь с помощью нижнего меню (BottomNavigationView) может переключиться на экран изменения данных о грузе и может переключиться на экран с информацией об авторе.

Экран с информацией об авторе содержит информацию об авторе, экран изображён на рисунке 26.

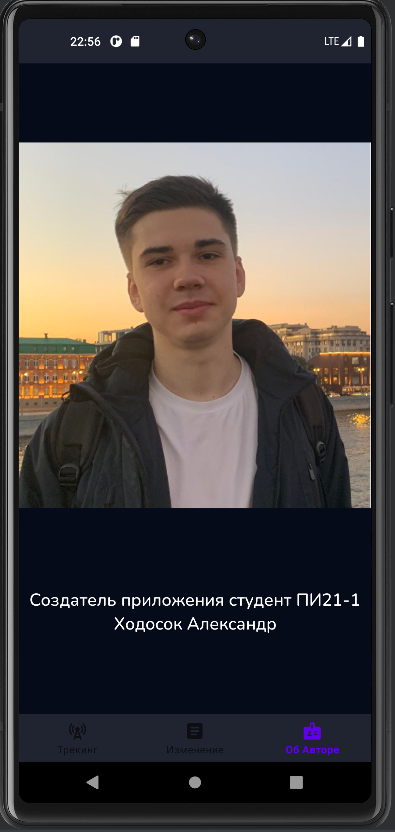


Рисунок 26 – экран об авторе

На экране изменение располагается три блока, в одном из них пользователю предлагают добавить трек-номер в базу данных, введя при этом трек-номер, грузовик, состояние и дату отправления. В следующем блоке пользователю предлагают изменить данные, введя все данные в поля по указаниям. И в последнем блоке есть возможность удалить данные, введя только трек-номер. Экран с изменениями изображён на рисунке 27. Все ошибки учтены, на примере ошибки в изменении данных о грузе показан вывод ошибки на экран (рисунок 28).

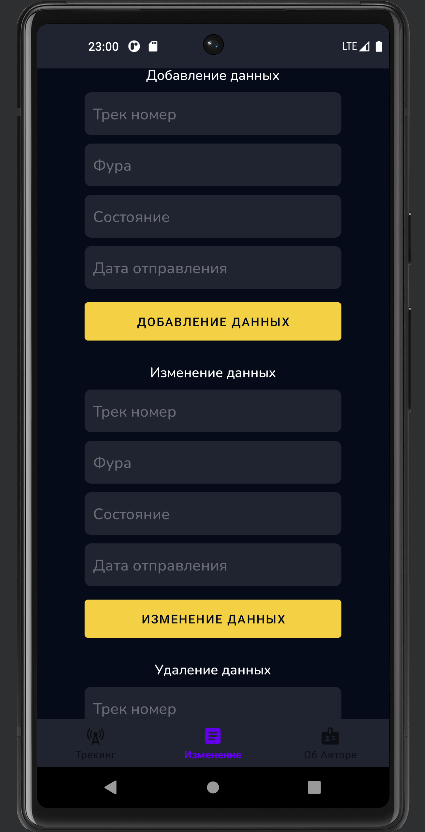


Рисунок 27 – экран с изменениями

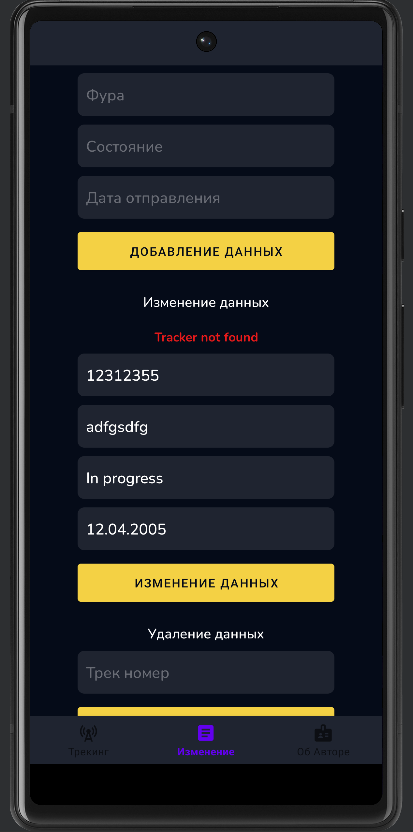


Рисунок 28 – ошибка при изменении данных

4.3. Состав приложения

При создании клиентской части был использован метод SingleActivity SingleActivity - это **подход к разработке Android-приложений, при котором используется только одна Activity и несколько фрагментов для отображения разных экранов**.

Рассмотрим файловую структуру серверной части:

Приложение запускается с файла Application.kt, в котором происходит авторизация в базу данных и объявляются все пути для формирования ссылок для REST API запросов, в основной папке содержится несколько других папок, изображённых на рисунке 29.

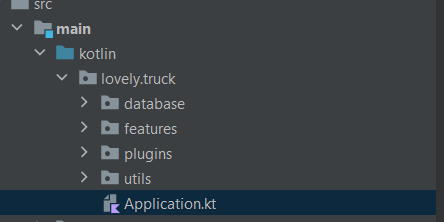


Рисунок 29 – Основная папка приложения

Рассмотрим папку database, изображённую на рисунке 30, в ней находится ещё несколько папок, которые названы в точности как таблицы в базе данных PostgreSQL, каждый из этих файлов позволяет серверу получать информацию из таблиц и формировать разные запросы в базу данных.

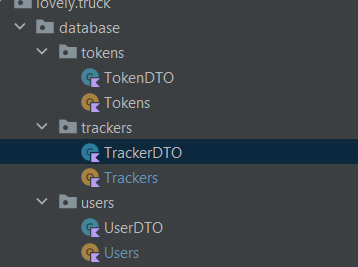


Рисунок 30 – Содержимое папки database

* TokenDTO – файл содержит структуру и типы данных колонок из таблицы tokens.
* Tokens – файл, отвечающий за запросы в базу данных, для формирования новых токенов при авторизации
* TrackerDTO – файл содержит структуру и типы данных колонок из таблицы trackers.
* Trackers – файл, отвечающий за запросы в базу данных, для создания, редактирования и удаления трек-номеров.
* UserDTO – файл содержит структуру и типы данных колонок из таблицы users
* Users – файл, отвечающий за запросы в базу данных, для авторизация и регистрации пользователей.

Далее рассмотрим папку features, изображённую на рисунке 31. В данной папке есть ещё несколько папок, каждая из которых необходима для формирования данных для REST API. К примеру, папка login содержит файлы, необходимые для формирования проверки введённого логина и пароля при авторизации и необходимые для формирования REST API запроса функции.

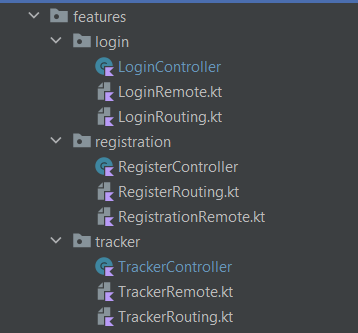


Рисунок 31 – содержимое папки features

* LoginController – файл, проверяющий данные, полученные с POST запроса, и формирующий данные для авторизации, добавление данных в таблицу tokens.
* LoginRemote – файл, который хранит модели для чтения и записи JSON файла с авторизацией.
* LoginRouting – файл, формирующий ссылку с окончанием /login для запроса с авторизацией.
* RegisterController - файл, проверяющий данные, полученные с POST запроса при регистрации, и формирующий данные для регистрации.
* RegisterRouting – файл формирует ссылку /registrations для запроса с регистрацией.
* TrackerController – файл, который позволяет проводить все манипуляции с информацией о грузах: удаление, изменение, добавление.
* TrackerRouting – файл, формирующий несколько ссылок для запросов с изменением, удалением, просмотром, добавлением трек-номеров.

Файлы, build.gradle.kts, изображённые на рисунке 32, формируют и компилируют проект, добавляя к нему зависимости и библиотеки.

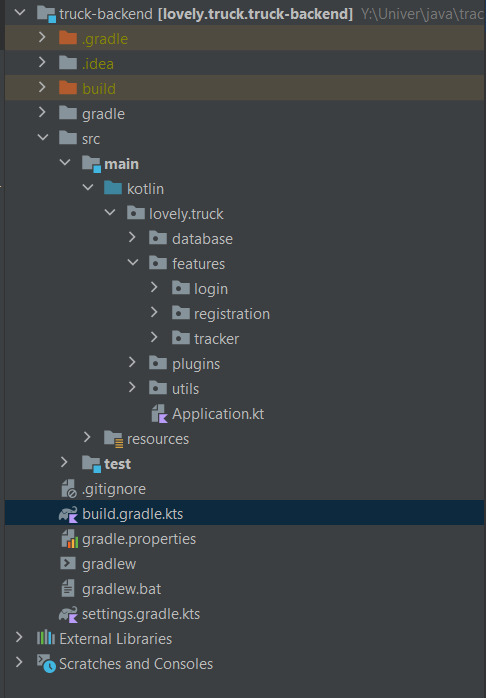


Рисунок 32 – вся файловая структура проекта

Далее рассмотрим файловую структуру логики клиентской части приложения, изображённую на рисунке 33.

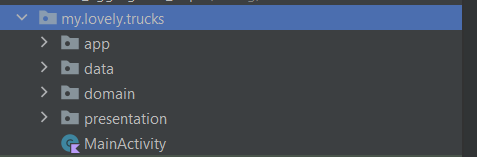


Рисунок 33 – файловая структура логики клиентского приложения

В клиентской части приложения была использована технология Clean Architecture, которая является подходом к проектированию приложений, который основан на концепциях чистого кода реализует принципы SOLID.

В папке App содержится файл App.kt, который запускает приложение с определёнными настройками.

Благодаря чистой архитектуре, можно выделить три слоя: data, domain, presentation. Рассмотрим папку data на рисунке 34.

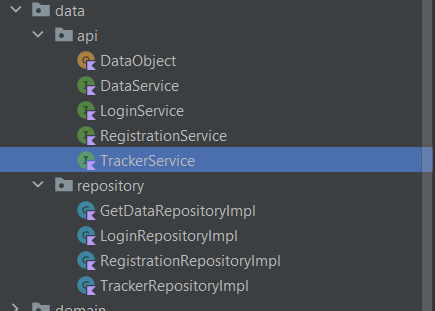


Рисунок 34 – содержимое папки data

Рассмотрим папку api:

* DataObject – объект, который конвертирует JSON файлы, полученные с GET и POST запросов, в читабельный для программы и человека формат.
* LoginService – файл, который формирует POST - запросы на сервер по определённой ссылке для авторизации.
* RegistrationService - файл, который формирует POST - запросы на сервер по определённой ссылке для регистрации.
* TrackerService - файл, который формирует POST - запрос на сервер по определённой ссылке для удаления, редактирования, добавления и просмотра информации о грузах по трек-номеру.

Рассмотрим папку repository, эта папка создана по правилам чистой архитектуры, именно на этом слое данные заходят и хранятся в приложении.

* LoginRepositoryImpl – класс, который реализует получения данных с REST API запроса для авторизации.
* RegistrationRepositoryImpl – класс, который реализует получения данных с REST API запроса для регистрации.
* TrackerRepositoryImpl – класс, который реализует получения данных с REST API запроса для взаимодействия с трек-номерами.

Далее рассмотрим папку domain, отвечающую за domain слой в чистой архитектуре. Содержимое папки изображено на рисунке 35.

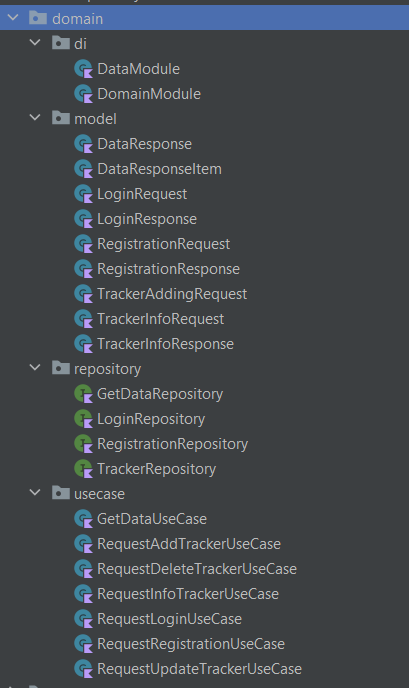


Рисунок 35 – содержимое папки domain

* di – папка с файлами, отвечающая за технологию Depenecy Injection в приложении, там объявляются все классы, которые необходимы приложению для корректной работы .
* model – папка, в которой содержатся data классы, отвечающие за модели данных, которые поступают с сервера и которые отправляются на сервер.
* repository – папка, в которой содержатся интерфейсы для всех классов, которые реализуют получение и хранение данных в приложении.
* RequestAddTrackerUseCase – это класс, который получает данные о добавлении трек-номера с интерфейса пользователя и передаёт их в data – слой для дальнейшей манипуляции.
* RequestDeleteTrackerUseCase – это класс, который получает данные о удалении трек-номера с интерфейса пользователя и передаёт их в data – слой для дальнейшей манипуляции.
* RequestInfoTrackerUseCase – это класс, который получает данные из data – слоя о грузе и передаёт эти данные на интерфейс пользователя.
* RequestLoginUseCase – класс, который отправляет логин и пароль пользователя в data – слой, который в свою очередь отправит на сервер для проверки.
* RequestRegistrationUseCase – класс, который принимает данные о регистрации пользователя и отправляет их на data-слой для передачи на сервер.
* RequestUpdateTrackerUseCase - это класс, который получает данные об изменении трек-номера с интерфейса пользователя и передаёт их в data – слой для дальнейшей передачи на сервер.

Рассмотрим следующий слой – это presentation слой, отвечающий за логику интерфейса пользователя. Содержимое папки изображено на рисунке 36.

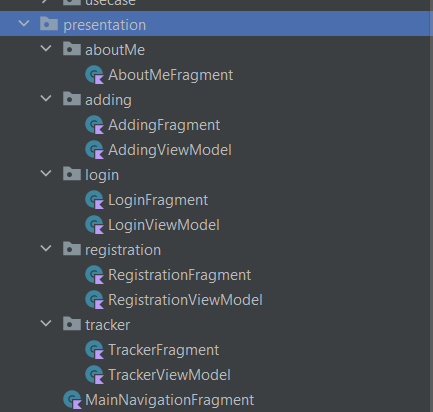


Рисунок 36 – содержимое папки presentation

* aboutMe – папка, которая содержит фрагмент, содержащий интерфейс окна “Об Авторе”.
* adding – папка, содержащая фрагмент и его ViewModel для окна “Изменение”.
* login – папка, которая содержит фрагмент и его ViewModel для окна авторизации пользователя.
* registration – папка, в которой содержится фрагмент и его ViewModel для окна регистрации пользователя.
* MainNavigationFragment – фрагмент, который является основным навигационным фрагментом по функционалу приложения.

Далее рассмотрим файлы с ресурсами, содержимое папки res изображено на рисунке 37.

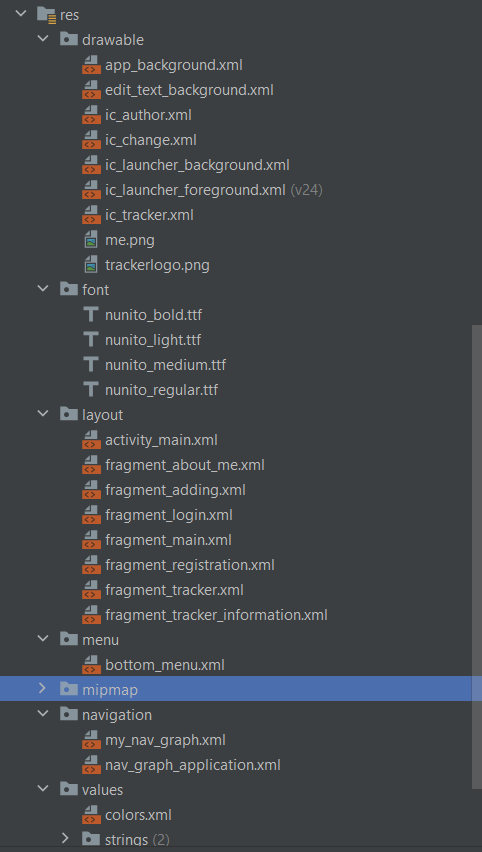


Рисунок 37 – Содержимое папки res

* drawable – папка, в которой находятся все иконки и картинки, которые используются в интерфейсе приложения.
* font – папка, в которой хранятся шрифты для текста, используемого в интерфейсе приложения.
* layout – папка, в которой хранятся экраны всех фрагментов приложения.
* menu – папка, в которой хранится главное нижнее меню приложения.
* mipmap – папка, в которой хранится логотип приложения в разных разрешениях.
* navigation – папка, в которой хранятся схемы и настройки навигации во всём приложении.
* values – папка с настройками значений для разных параметров значений.
* colors – файл, в котором хранятся все цвета, используемые в приложении.
* strings – в данной папке хранится два файла, которые содержат весь текст на двух языках - английском и русскомю.

# ЗАКЛЮЧЕНИЕ

В заключении курсовой работы по теме "Фирма грузоперевозок" можно отметить, что создание мобильного приложения для отслеживания грузов по трек-номерам является актуальной и востребованной задачей в современном бизнесе. Разработка такого приложения позволяет повысить уровень доверия клиентов, обеспечить более эффективный контроль за доставкой грузов и улучшить качество услуг компании.

В ходе выполнения курсовой работы были рассмотрены основные функциональные возможности приложения, такие как отслеживание груза по трек-номеру, изменение, удаление и добавление данных в базу данных трек-номеров. Также были уделены внимание техническим особенностям реализации приложения, таким как использование API, разработка серверной части и мобильного приложения, а также тестирование приложения на различных устройствах.

В результате работы было создано полноценное мобильное приложение для отслеживания грузов по трек-номеру под Android, которое представляет собой удобный и функциональный инструмент для контроля за доставкой грузов. Такое приложение может быть полезным как для компаний, занимающихся транспортировкой грузов, так и для обычных пользователей, отправляющих или получающих посылки.

В целом, разработка мобильного приложения для отслеживания грузов по трек-номеру является важным шагом в развитии современных технологий транспортировки грузов и может стать ключевым фактором в повышении качества услуг компании и удовлетворенности клиентов.