Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «Задание 8. Индивидуальные задания по теме: базы данных в Android Studio»

Выполнил: Горовой В.С

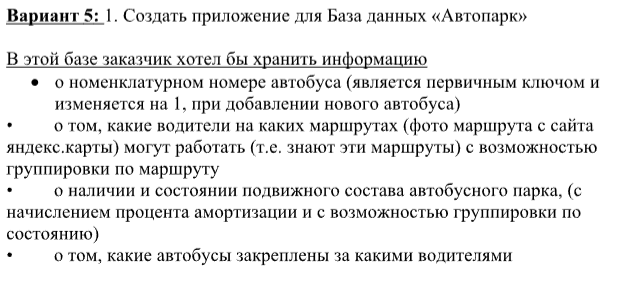
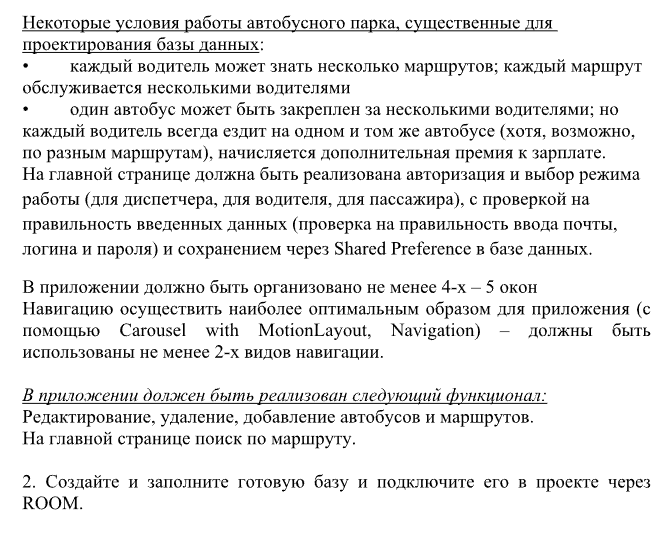
Группа: ПР-32

Преподаватель: Мирошниченко Г.В

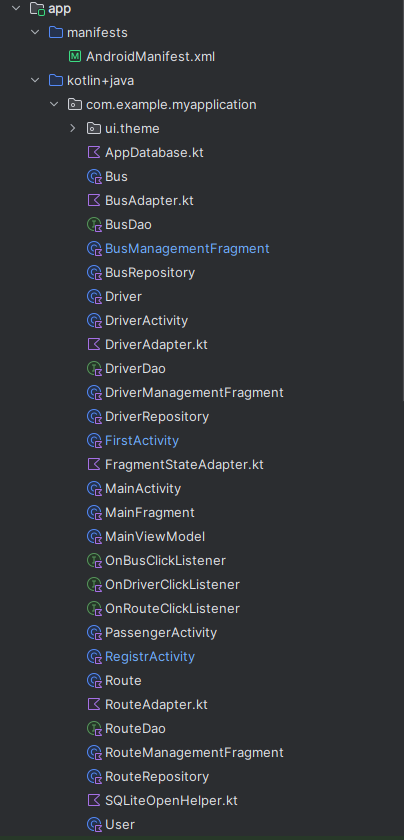
2025

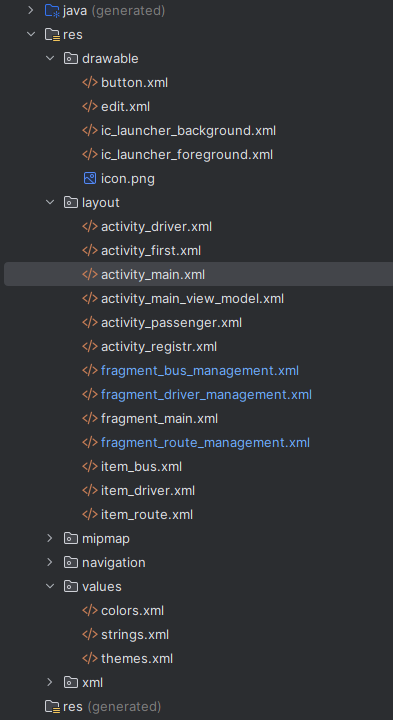
# Задание

## Описание задачи

* 1. Структура проекта

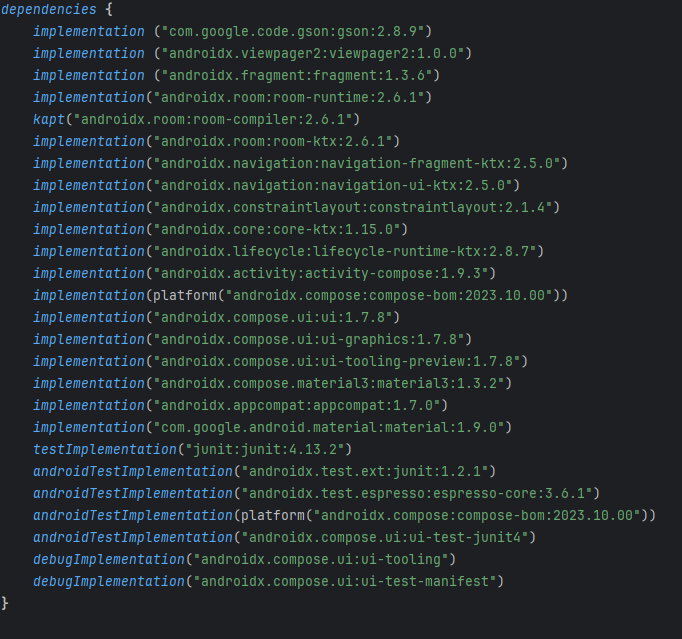




* **AppDatabase.kt** – главный класс для работы с базой данных, содержит объявление таблиц и DAO.
* **Bus** – класс, представляющий сущность автобуса (номер, модель, вместимость).
* **BusAdapter.kt** – адаптер для отображения списка автобусов в RecyclerView.
* **BusDao** – интерфейс для операций с таблицей автобусов (добавление, выборка, обновление, удаление).
* **BusManagementFragment** – фрагмент для управления автобусами (добавление, редактирование, удаление).
* **BusRepository** – репозиторий для работы с данными автобусов, связывает BusDao и ViewModel.
* **Driver** – класс, представляющий сущность водителя (имя, контакты, права).
* **DriverActivity** – активность, связанная с функционалом водителя (просмотр маршрутов, управление рейсами).
* **DriverAdapter.kt** – адаптер для отображения списка водителей в RecyclerView.
* **DriverDao** – интерфейс для операций с таблицей водителей.
* **DriverManagementFragment** – фрагмент для управления водителями (добавление, редактирование, удаление).
* **DriverRepository** – репозиторий для работы с данными водителей, связывает DriverDao и ViewModel.
* **FirstActivity** – первая активность при запуске приложения, может содержать начальный экран или логику авторизации.
* **FragmentStateAdapter.kt** – адаптер для управления фрагментами в ViewPager2 (например, для переключения между вкладками).
* **MainActivity** – основная активность, управляющая главным интерфейсом приложения и навигацией между фрагментами.
* **MainFragment** – главный фрагмент, отображаемый в MainActivity, может содержать основную информацию или навигационные элементы.
* **MainViewModel** – ViewModel для MainActivity или MainFragment, содержит бизнес-логику и данные для главного экрана.
* **OnBusClickListener** – интерфейс для обработки нажатий на элементы списка автобусов.
* **OnDriverClickListener** – интерфейс для обработки нажатий на элементы списка водителей.
* **OnRouteClickListener** – интерфейс для обработки нажатий на элементы списка маршрутов.
* **PassengerActivity** – активность для пассажиров (просмотр доступных автобусов или маршрутов).
* **RegistrActivity** – активность для регистрации новых пользователей.
* **Route** – класс, представляющий сущность маршрута (начальная и конечная точки, время отправления).
* **RouteAdapter.kt** – адаптер для отображения списка маршрутов в RecyclerView.
* **RouteDao** – интерфейс для операций с таблицей маршрутов.
* **RouteManagementFragment** – фрагмент для управления маршрутами (добавление, редактирование, удаление).
* **RouteRepository** – репозиторий для работы с данными маршрутов, связывает RouteDao и ViewModel.
* **SQLiteOpenHelper.kt** – класс для работы с SQLite (если не используется Room), содержит логику создания и обновления БД.
* **User** – класс, представляющий сущность пользователя (логин, пароль, роль).
  1. Описание разработанных функции

Entity – это объект (Bus, Driver, Route)

* createDatabase(context: Context) – создание базы данных, если приложение было скачено первый раз, иначе он будет подгружать существующую базу данных с данными  
  **onCreateViewHolder()** – создает новый ViewHolder для элемента списка.
* **onBindViewHolder()** – связывает данные с ViewHolder, обрабатывает клики и выделение элементов.
* **getItemCount()** – возвращает количество элементов в списке.
* **update[Entity]List()** – обновляет список данных в адаптере (например, updateBusList, updateDriverList, updateRouteList).
* **clearSelection()** – сбрасывает выделенный элемент.
* **getSelectedEntity** – возвращает выбранный элемент (например, getSelectedBus, getSelectedDriver, getSelectedRoute).
* **bind()** (в ViewHolder) – заполняет View данными объекта (автобуса, водителя или маршрута).
* **insert()** – добавляет объект в базу данных.
* **update()** – обновляет существующий объект.
* **delete()** – удаляет объект.
* **getAllEntities** – возвращает все объекты из таблицы (например, getAllBuses, getAllDrivers, getAllRoutes).
* **on[Entity]Click()** – обработка клика по элементу списка (обновляет поля ввода).
* **addEntity** – добавляет новый объект (автобус, водителя или маршрут) с валидацией данных.
* **deleteEntity** – удаляет выбранный объект.
* **editEntity** – редактирует выбранный объект.
* **clearInputFields()** – очищает поля ввода.
* **setupSpinners()** – заполняет выпадающие списки данными (например, типами топлива или списком водителей).
* **validateInput()** (в DriverManagementFragment) – проверяет логин/пароль при регистрации водителя.
* **saveUserData()** – сохраняет данные пользователя в SQLite.
* **updateRouteList()** (в RouteManagementFragment) – обновляет список маршрутов.
* **addEntity** – вызывает DAO для добавления объекта.
* **getAllEntities** – возвращает все объекты из таблицы.
* **updateEntity** – обновляет объект.
* **deleteEntity** – удаляет объект.
* **validateLogin()** (в FirstActivity) – проверяет логин/пароль и перенаправляет пользователя в зависимости от роли.
* **validateInput()** (в RegistrActivity) – валидирует данные при регистрации.
* **saveUserData()** – сохраняет нового пользователя.
* **editBus()** (в DriverActivity) – обновляет дату последнего ТО автобуса с проверкой корректности даты.
* **addEntity** / **updateEntity** / **deleteEntity** – методы для работы с данными через репозитории.
* **getAllEntities** – возвращает LiveData со списком объектов.  
  1. Используемые библиотеки



* 1. Тестовые ситуцаии

**Тестирование работы с автобусами (BusManagementFragment)**

**Сценарий 1: Добавление автобуса**

* **Действия:**
  1. Ввести корректные данные:
     + Год выпуска: 2015
     + Количество мест: 30
     + Тип топлива: Дизельное
     + Макс. срок службы: 15
  2. Нажать кнопку **"Добавить автобус"**.
* **Ожидаемый результат:**
  1. Автобус добавлен в список.
  2. Появляется уведомление: "Автобус добавлен".
  3. Поля ввода очищаются.

**Сценарий 2: Добавление автобуса с некорректными данными**

* **Действия:**
  1. Ввести недопустимые значения:
     + Год выпуска: 1800 (меньше 1990)
     + Количество мест: 5 (меньше 15)
     + Макс. срок службы: 50 (больше 25)
  2. Нажать кнопку **"Добавить автобус"**.
* **Ожидаемый результат:**
  1. Появляется сообщение: "Введите корректные данные".
  2. Автобус не добавляется.

**Сценарий 3: Удаление автобуса**

* **Действия:**
  1. Выбрать автобус из списка.
  2. Нажать кнопку **"Удалить автобус"**.
* **Ожидаемый результат:**
  1. Автобус исчезает из списка.
  2. Появляется уведомление: "Автобус удален".

**Сценарий 4: Редактирование автобуса**

* **Действия:**
  1. Выбрать автобус из списка.
  2. Изменить данные (например, тип топлива на Сжатый газ).
  3. Нажать кнопку **"Изменить автобус"**.
* **Ожидаемый результат:**
  1. Данные автобуса обновляются.
  2. Появляется уведомление: "Автобус обновлен".

**Тестирование работы с водителями (DriverManagementFragment)**

Сценарий 1: Попытка добавить водителя

* **Действия:**
  1. Ввести корректные данные:
     + Имя: Иван
     + Фамилия: Петров
     + Стаж: 10
     + Логин: driver1
     + Пароль: password123 (не менее 8 символов)
  2. Нажать кнопку **"Добавить водителя"**.
* **Ожидаемый результат:**
  1. Водитель добавлен в список.
  2. Появляется уведомление: "Водитель добавлен".

**Сценарий 2: Попытка добавления водителя с коротким паролем**

* **Действия:**
  1. Ввести пароль: 123 (менее 8 символов).
  2. Нажать кнопку **"Добавить водителя"**.
* **Ожидаемый результат:**
  1. Появляется сообщение: "Пароль должен содержать не менее 8 символов".
  2. Водитель не добавляется.

**Сценарий 3: Удаление водителя**

* **Действия:**
  1. Выбрать водителя из списка.
  2. Нажать кнопку **"Удалить водителя"**.
* **Ожидаемый результат:**
  1. Водитель удаляется из списка.
  2. Появляется уведомление: "Водитель удален".

**Тестирование работы с маршрутами (RouteManagementFragment)**

**Сценарий 1: Добавление маршрута**

* **Действия:**
  1. Выбрать водителя и автобус из списков.
  2. Ввести:
     + Начальная станция: Москва
     + Конечная станция: Санкт-Петербург
  3. Нажать кнопку **"Добавить маршрут"**.
* **Ожидаемый результат:**
  1. Маршрут добавлен в список.
  2. Появляется уведомление: "Маршрут добавлен".

**Сценарий 2: Попытка добавления маршрута с одинаковыми станциями**

* **Действия:**
  1. Ввести:
     + Начальная станция: Москва
     + Конечная станция: Москва
  2. Нажать кнопку **"Добавить маршрут"**.
* **Ожидаемый результат:**
  1. Появляется сообщение: "Начальная и конечная станции не должны быть одинаковые".
  2. Маршрут не добавляется.

**Сценарий 3: Удаление маршрута**

* **Действия:**
  1. Выбрать маршрут из списка.
  2. Нажать кнопку **"Удалить маршрут"**.
* **Ожидаемый результат:**
  1. Маршрут удаляется из списка.
  2. Появляется уведомление: "Маршрут удален".

**Тестирование авторизации и регистрации  
Сценарий 1: Успешная авторизация диспетчера**

* **Действия:**
  1. Ввести логин и пароль существующего диспетчера.
  2. Нажать кнопку **"Войти"**.
* **Ожидаемый результат:**
  1. Открывается MainActivity с доступом к управлению автобусами, водителями и маршрутами.

**Сценарий 2: Неудачная авторизация (неверный пароль)**

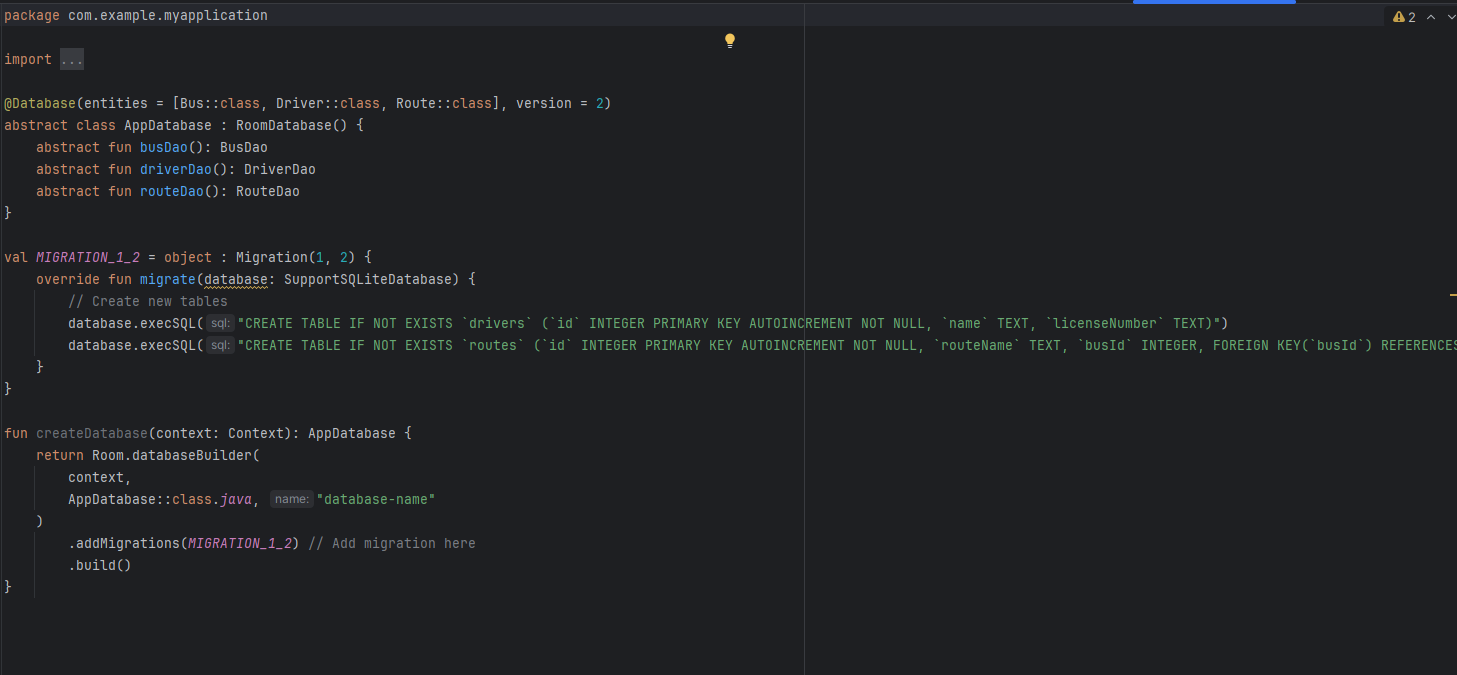
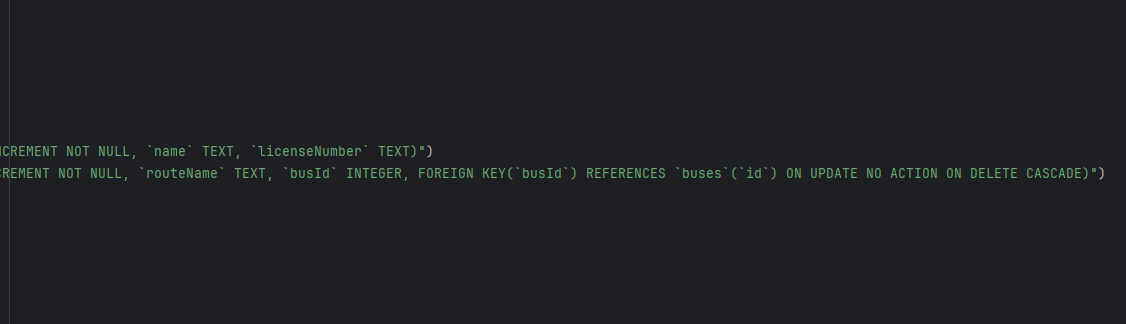
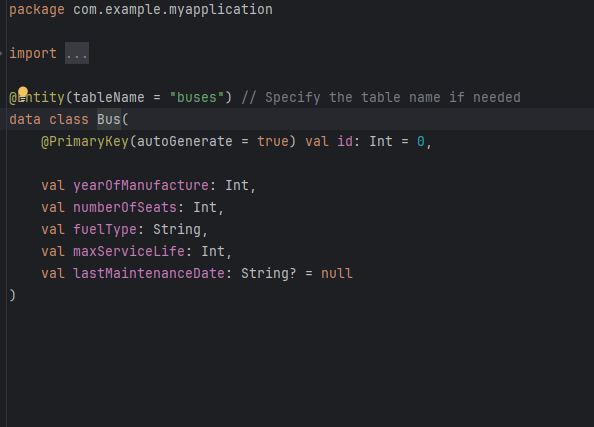
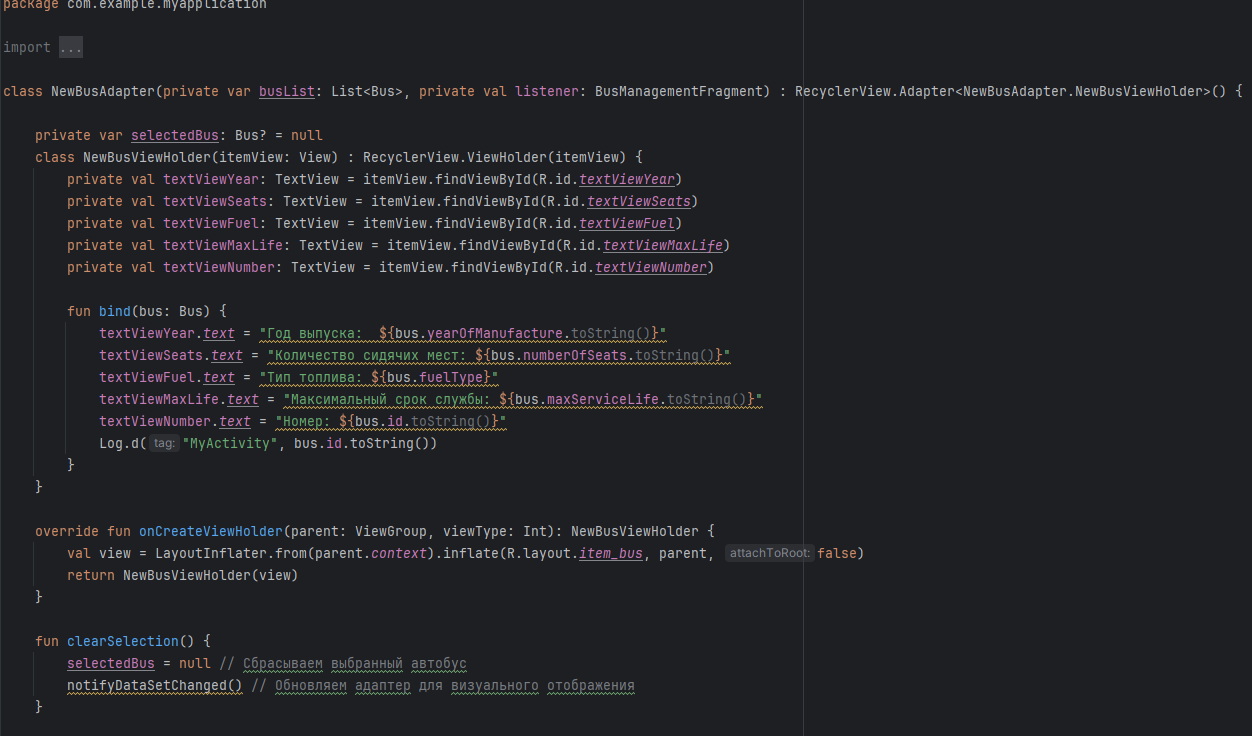
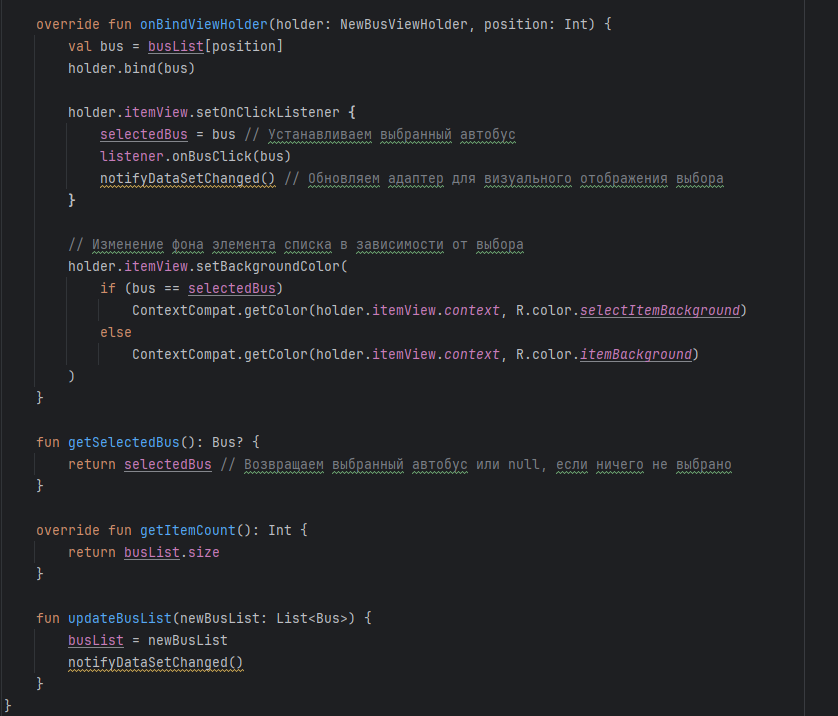
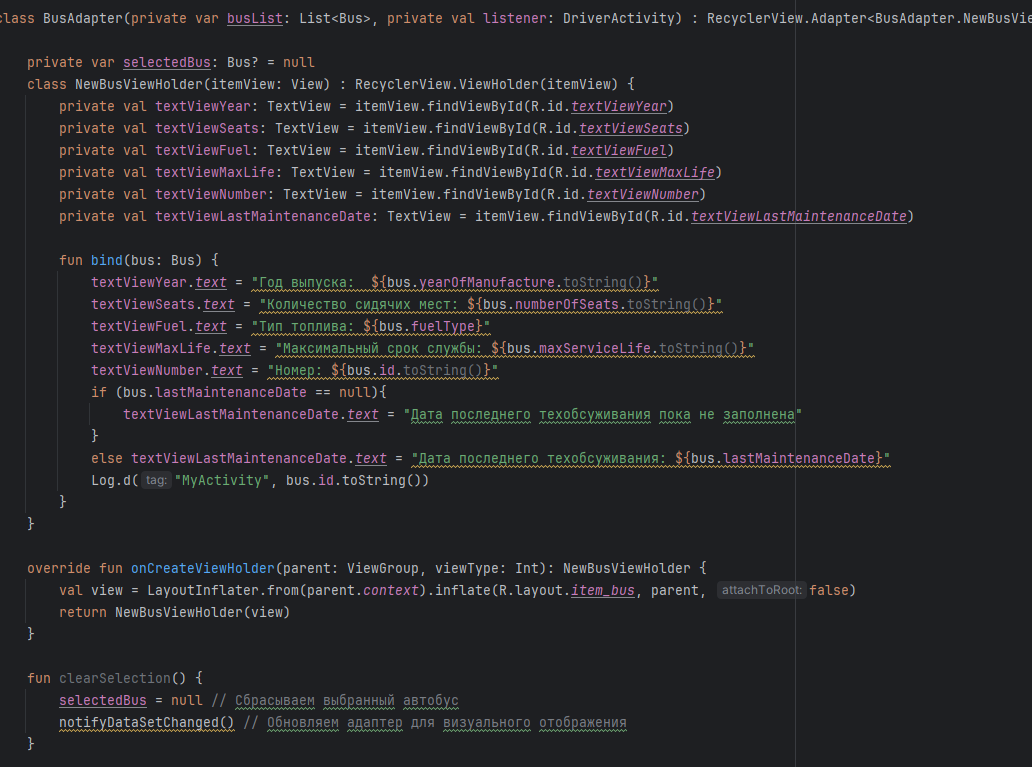
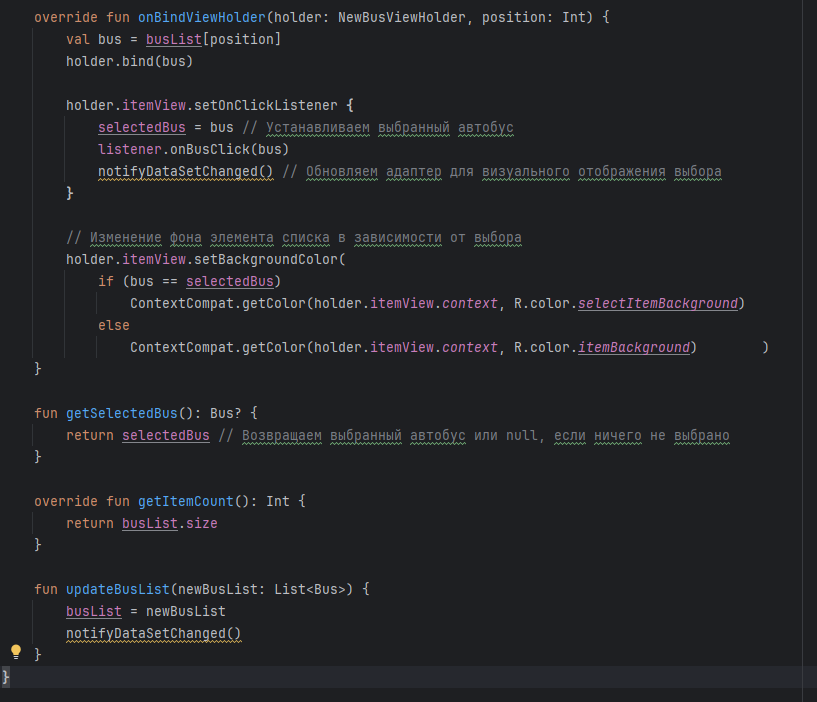
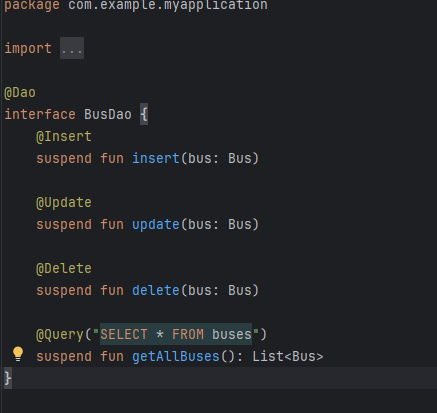
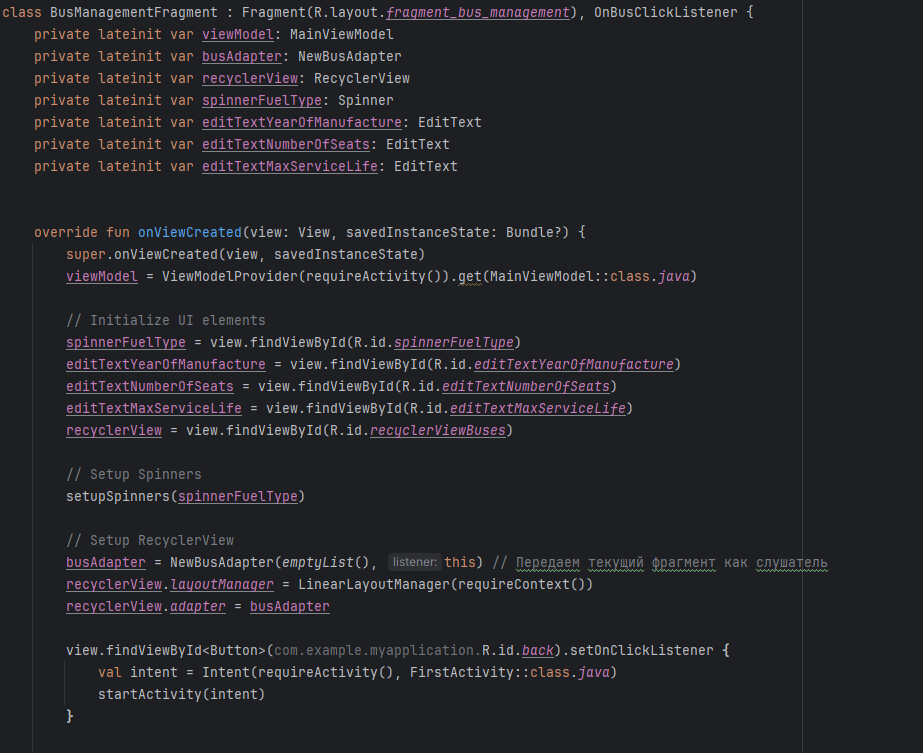
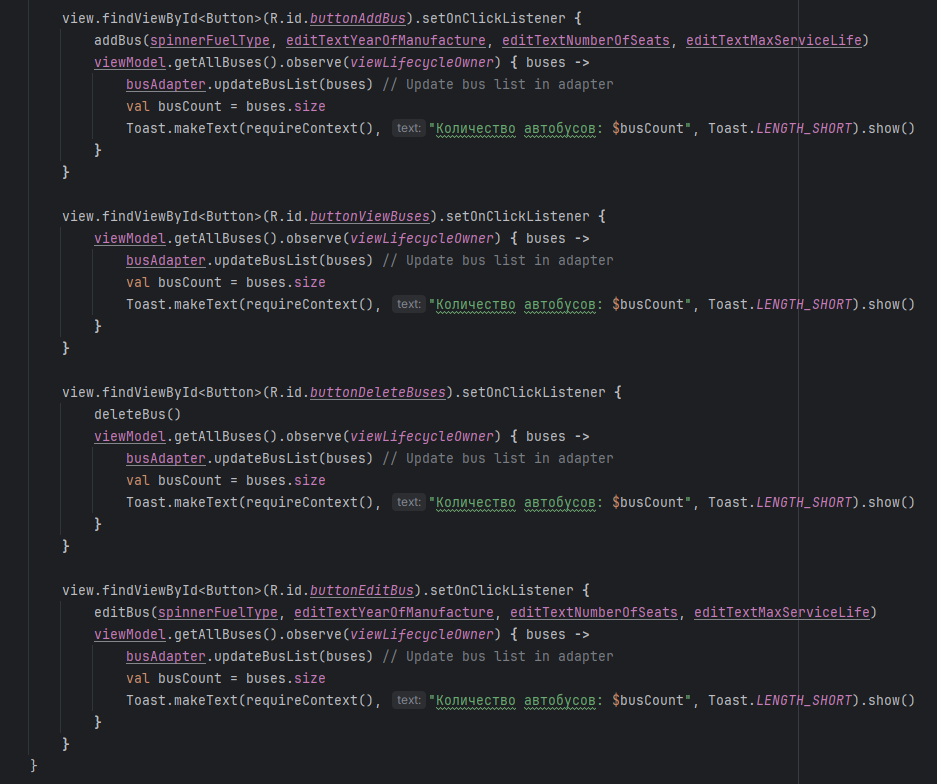
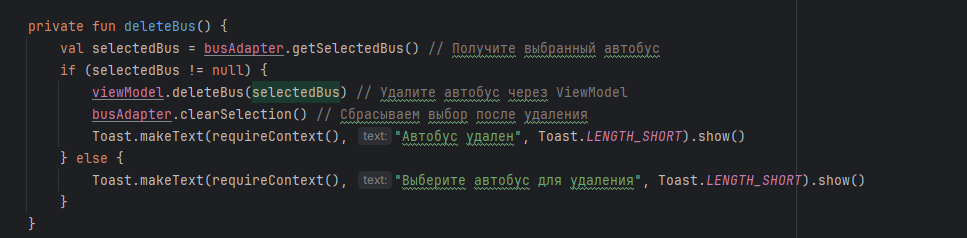
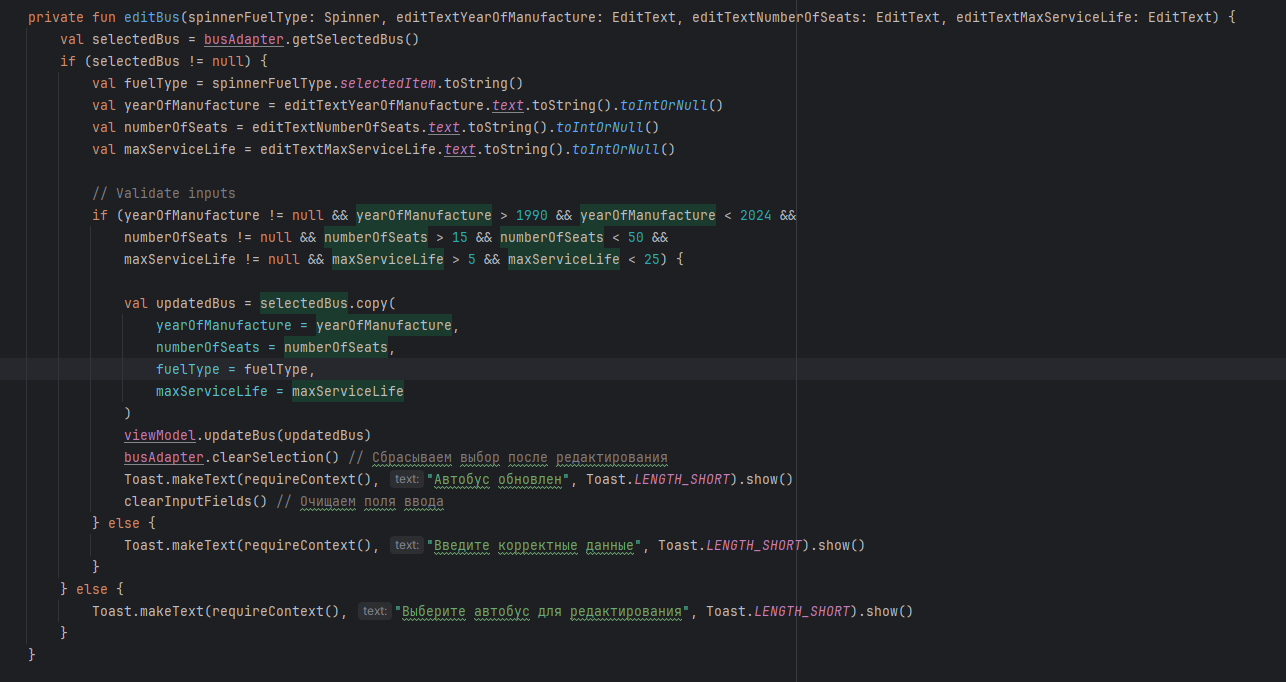
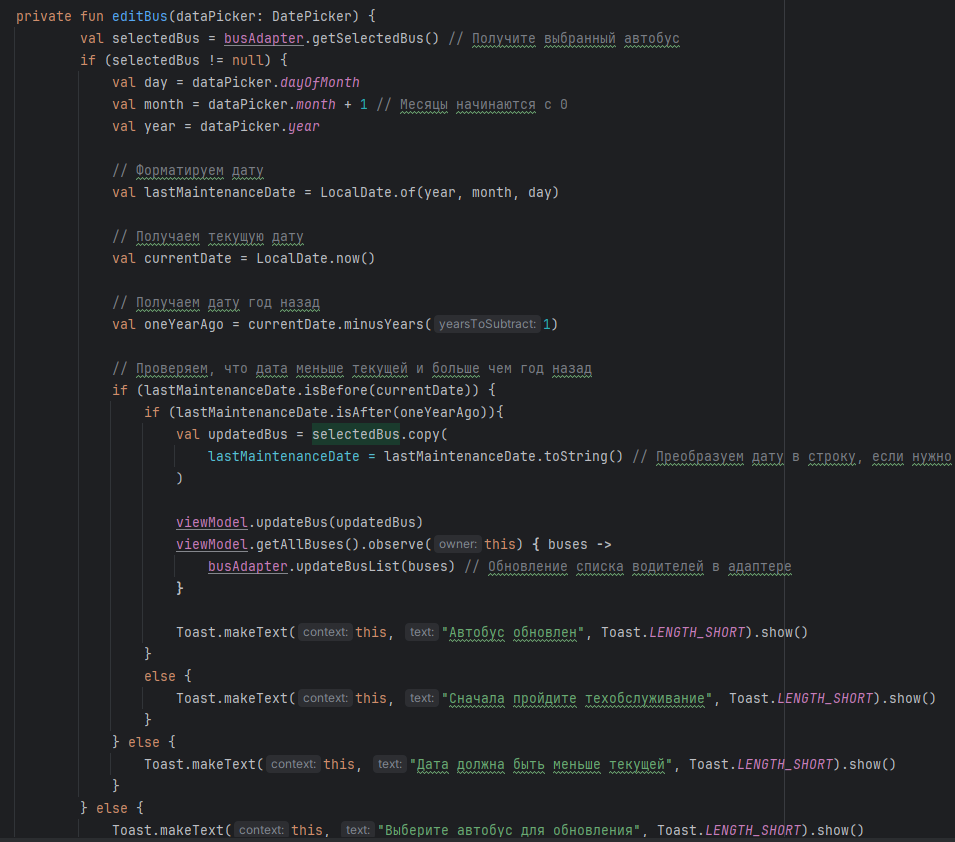
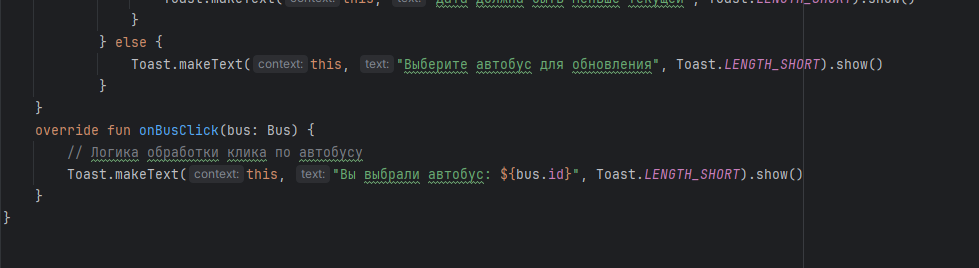
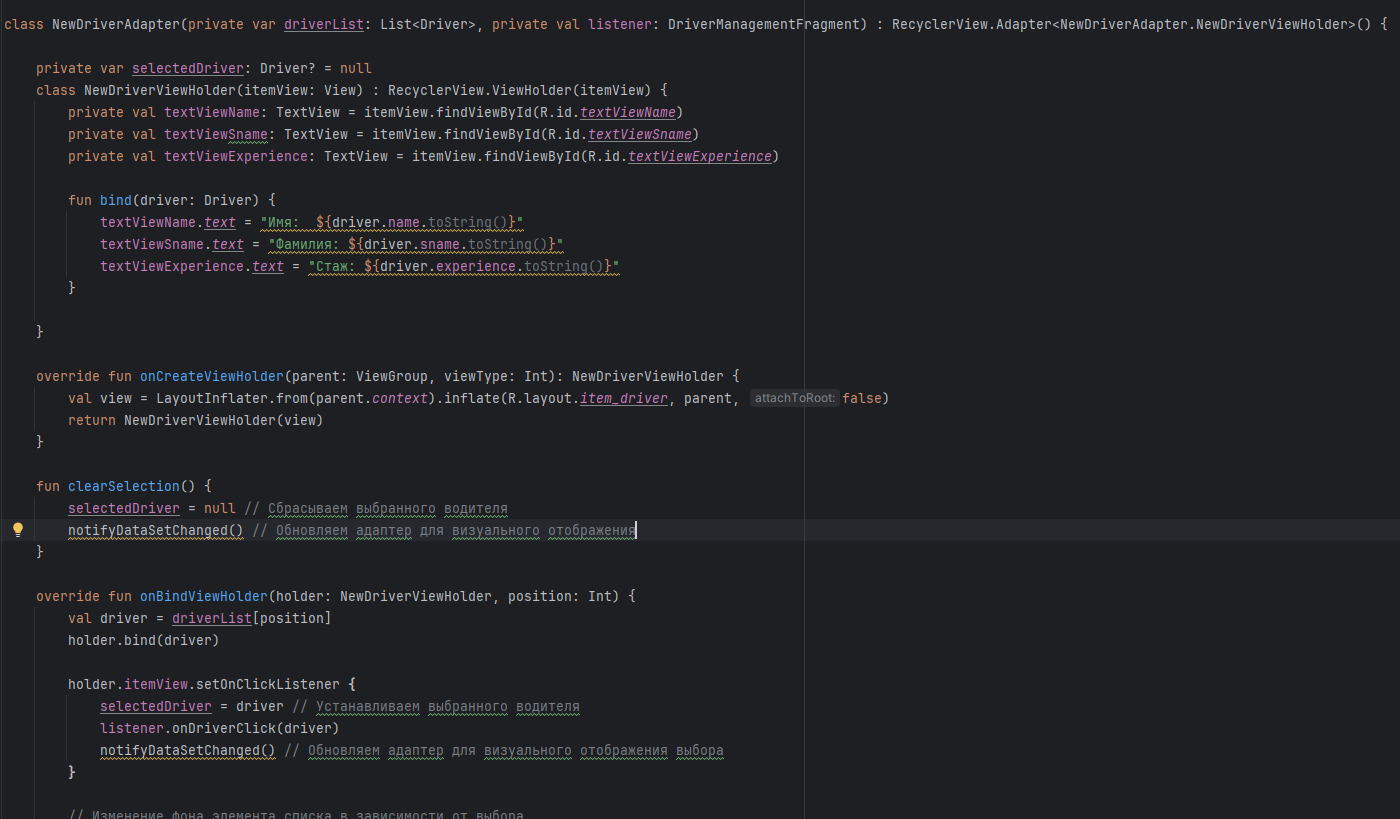
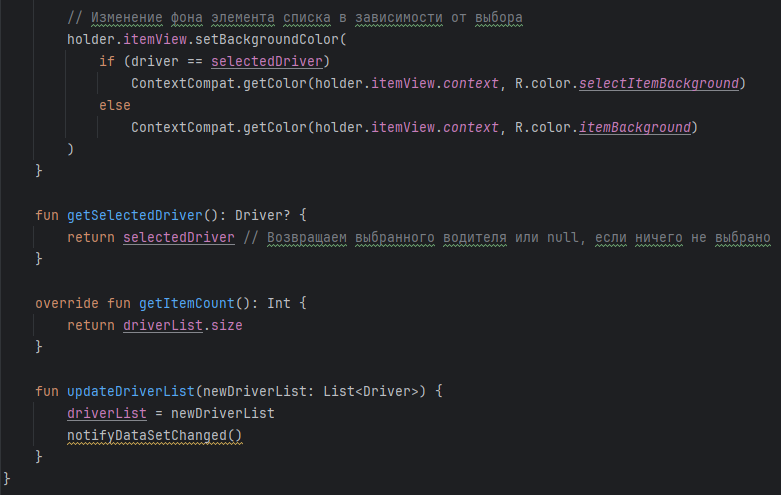
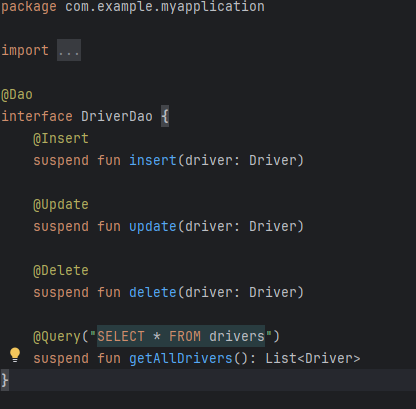
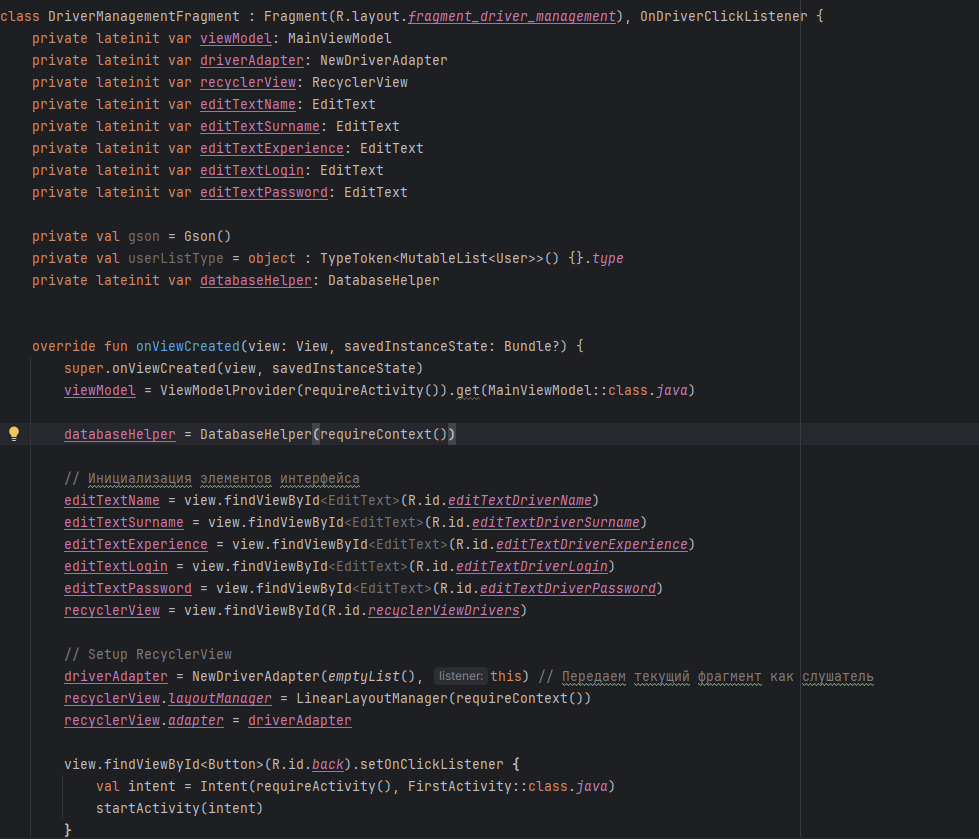
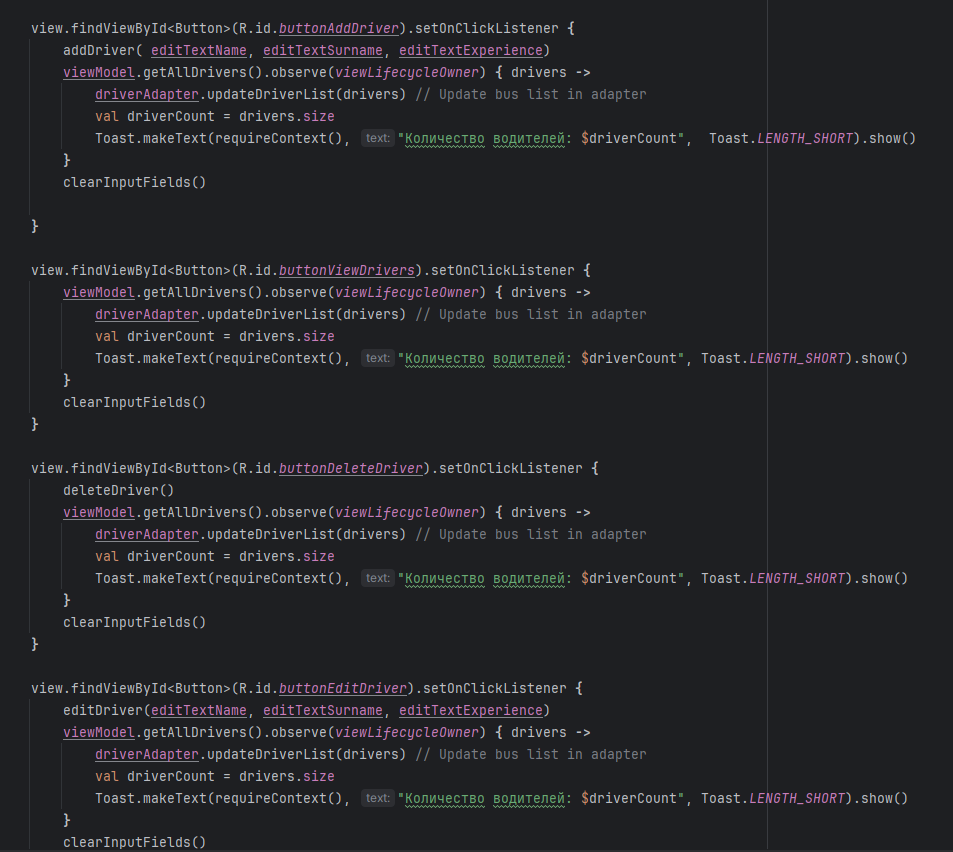
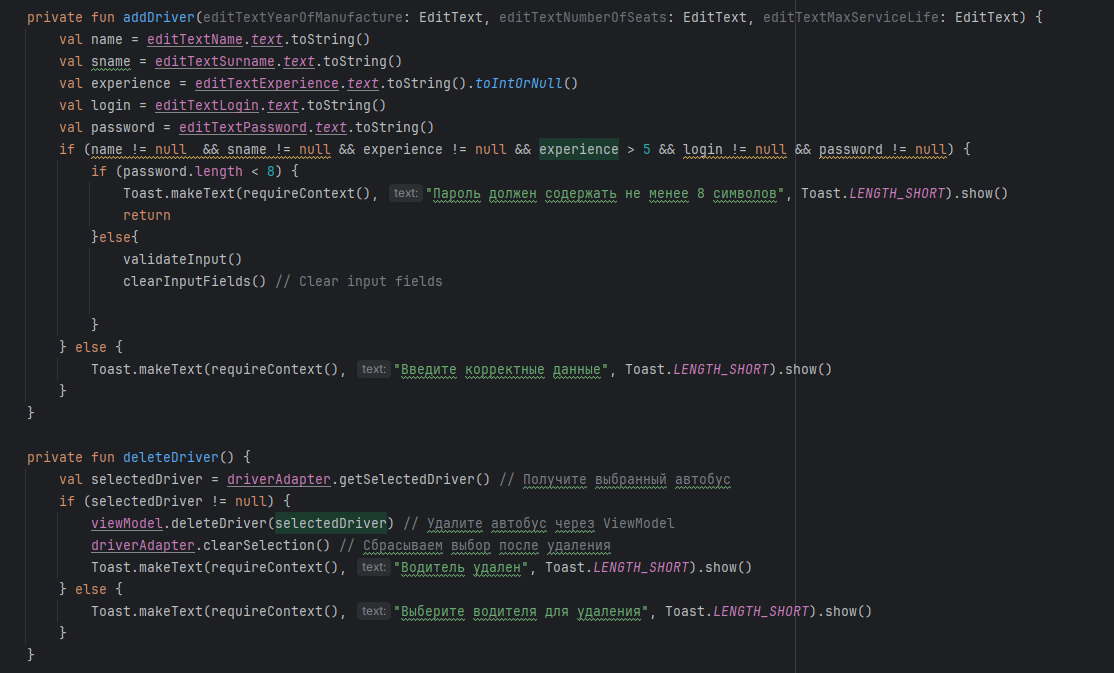
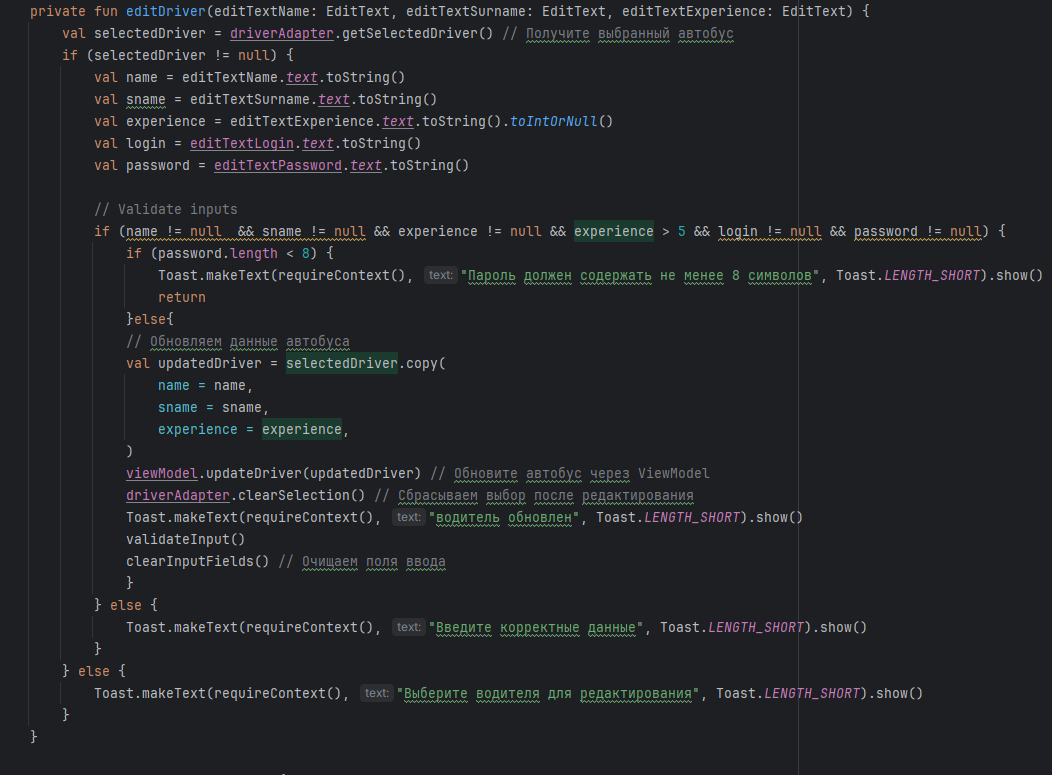
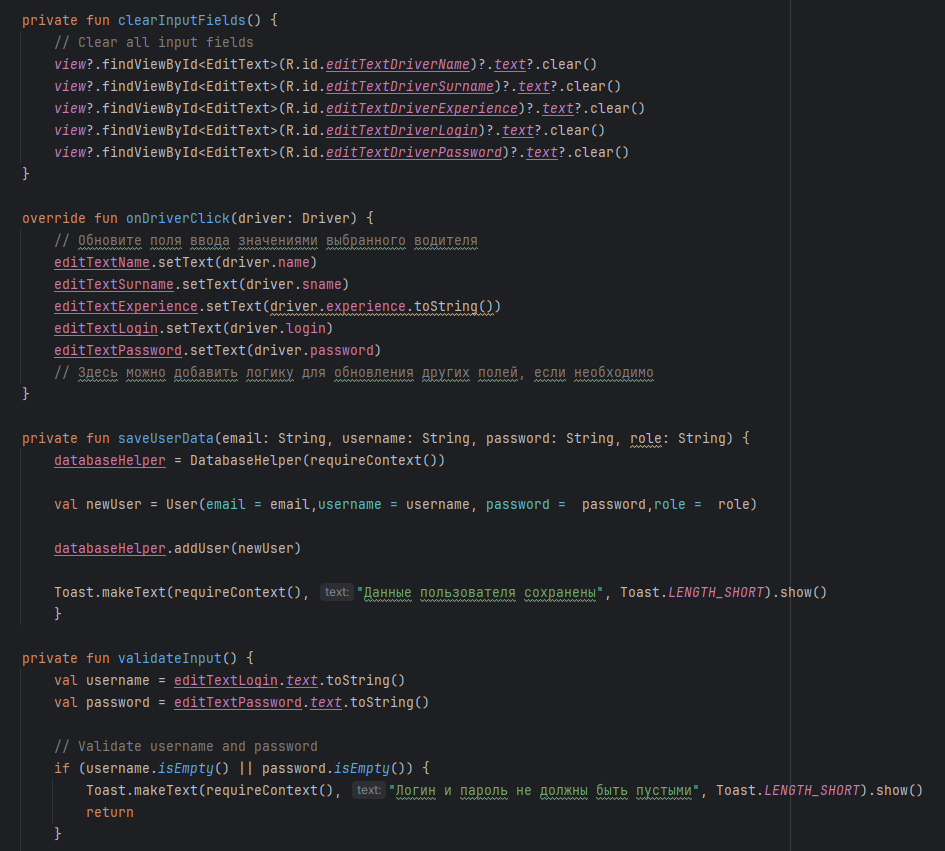
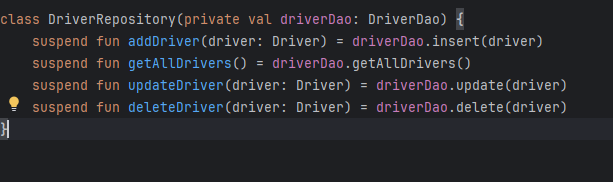
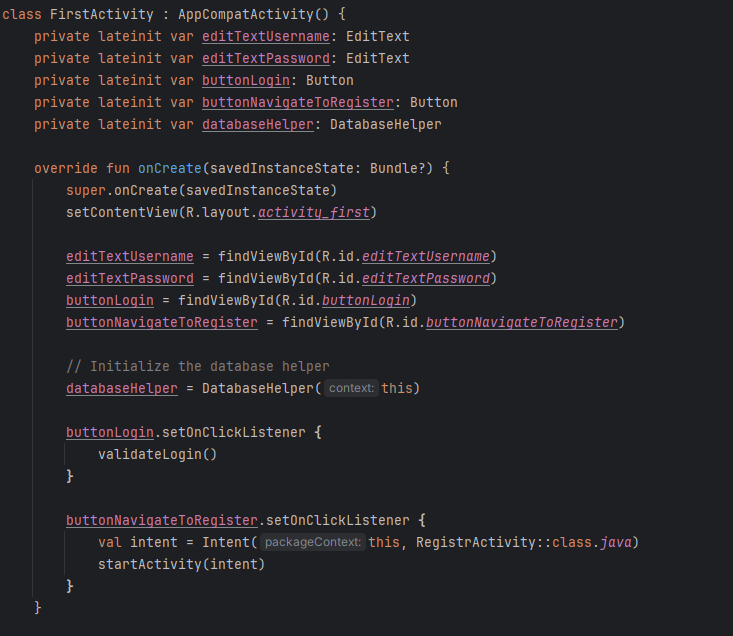
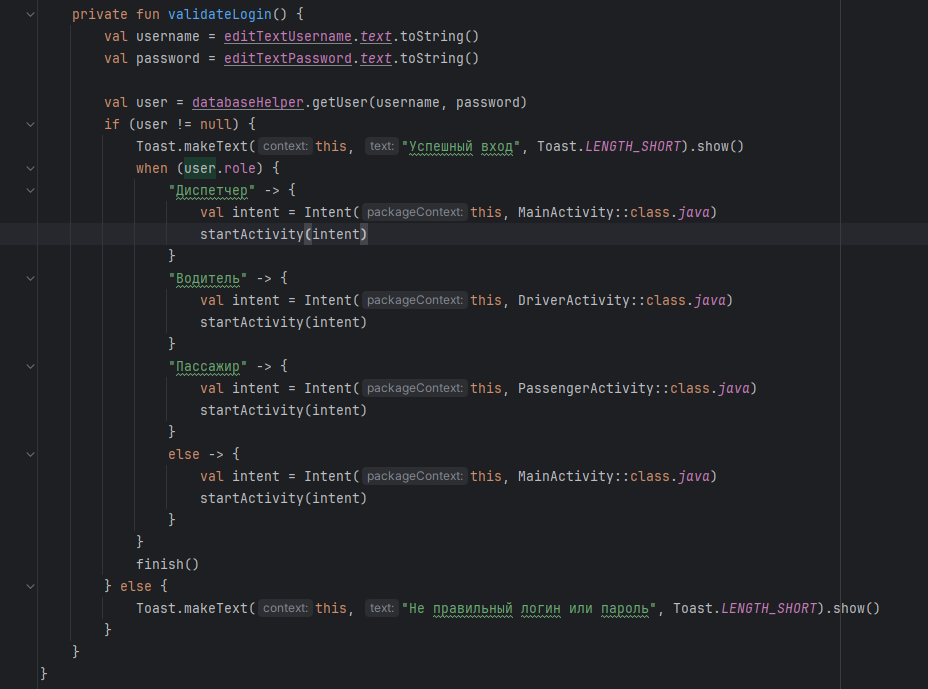
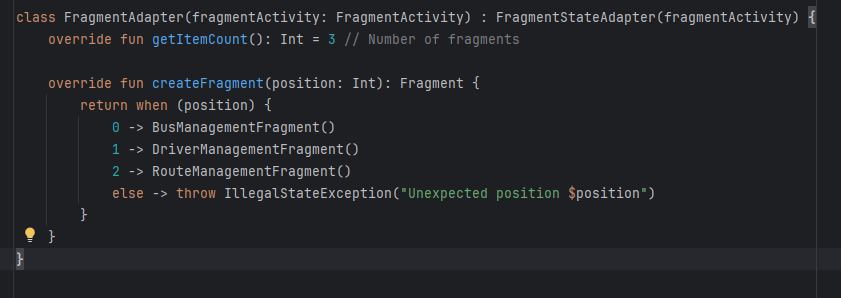
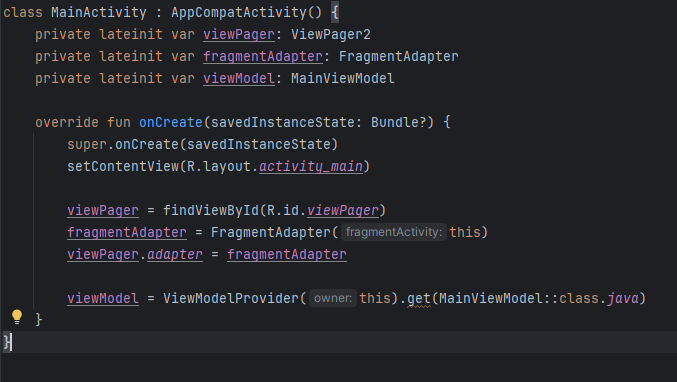
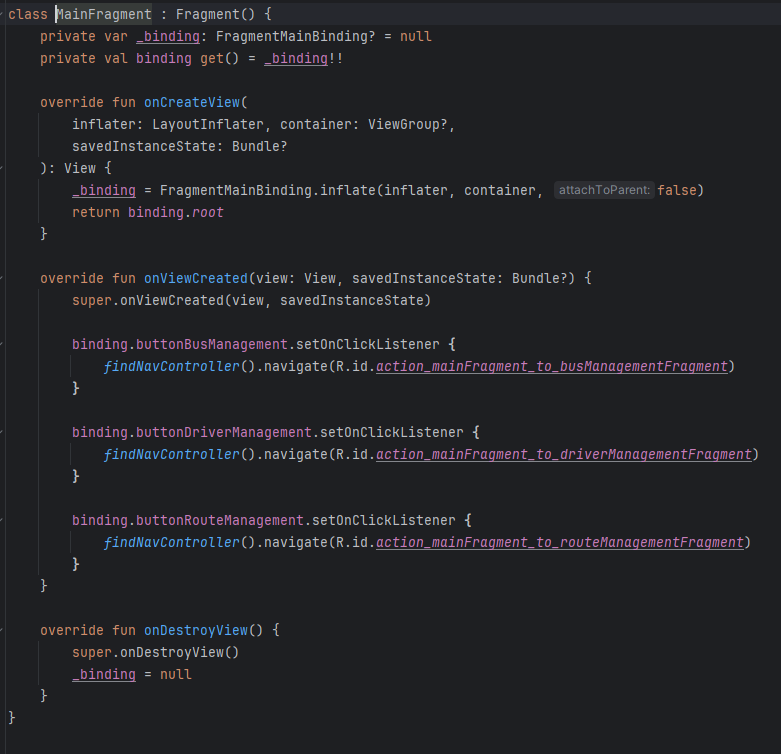
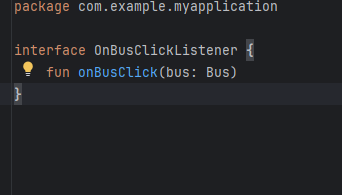
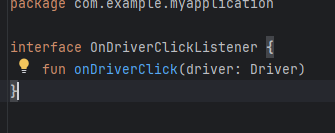
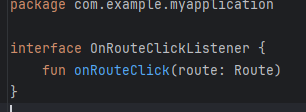
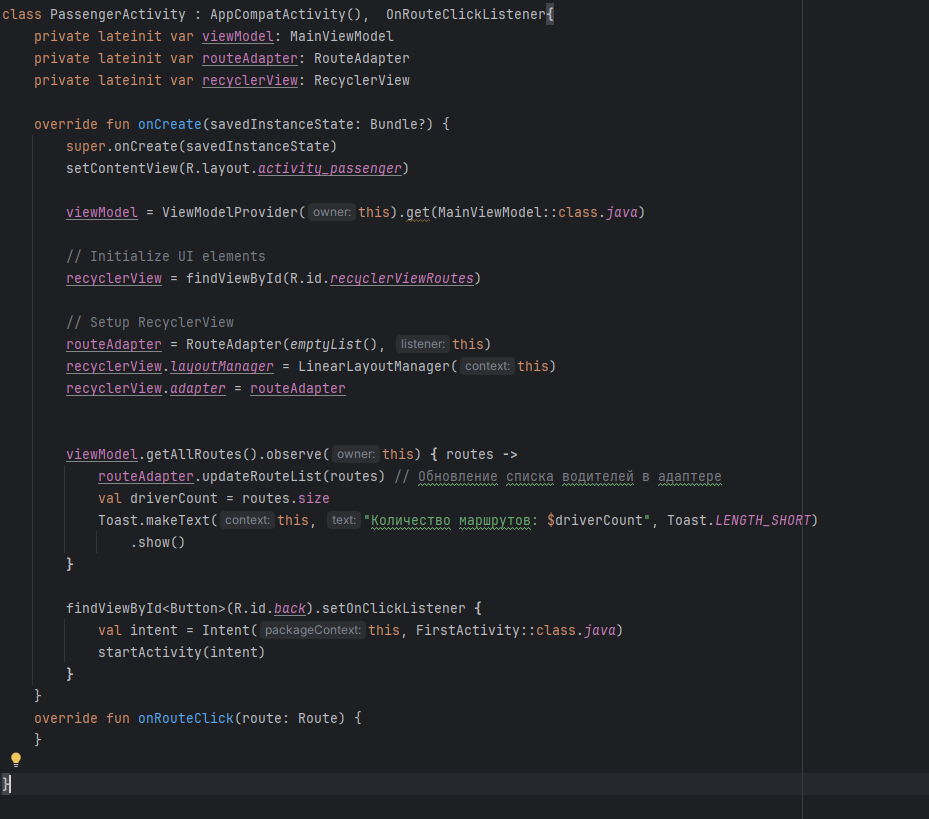
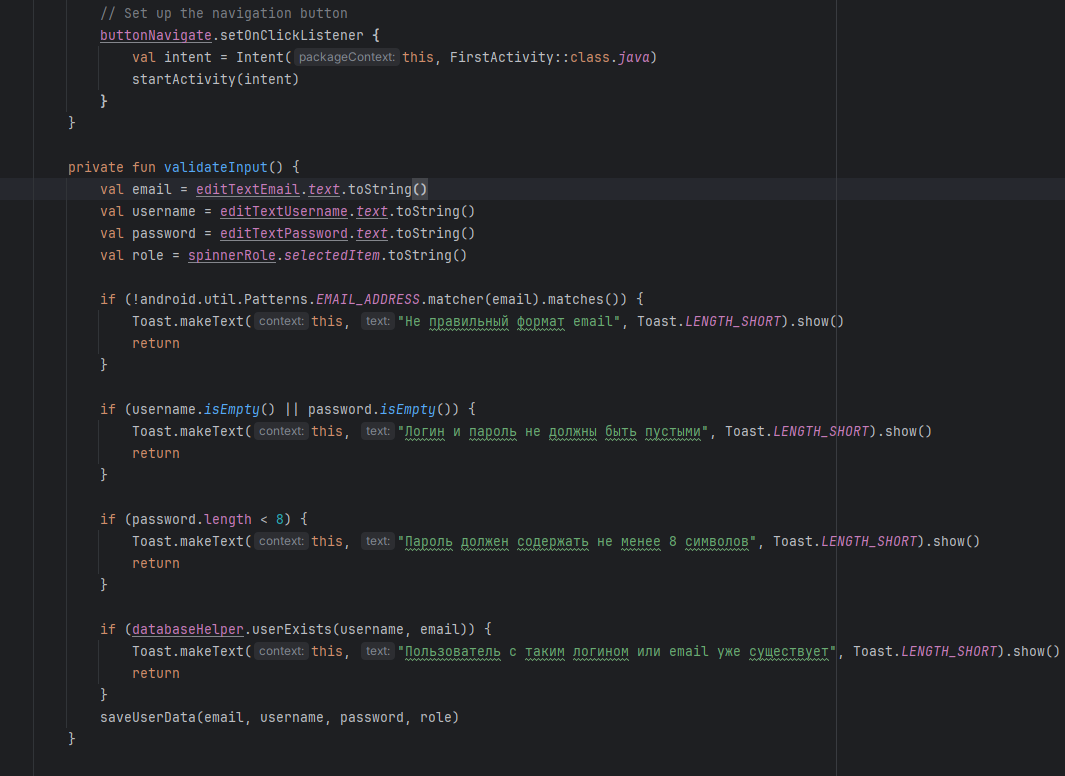
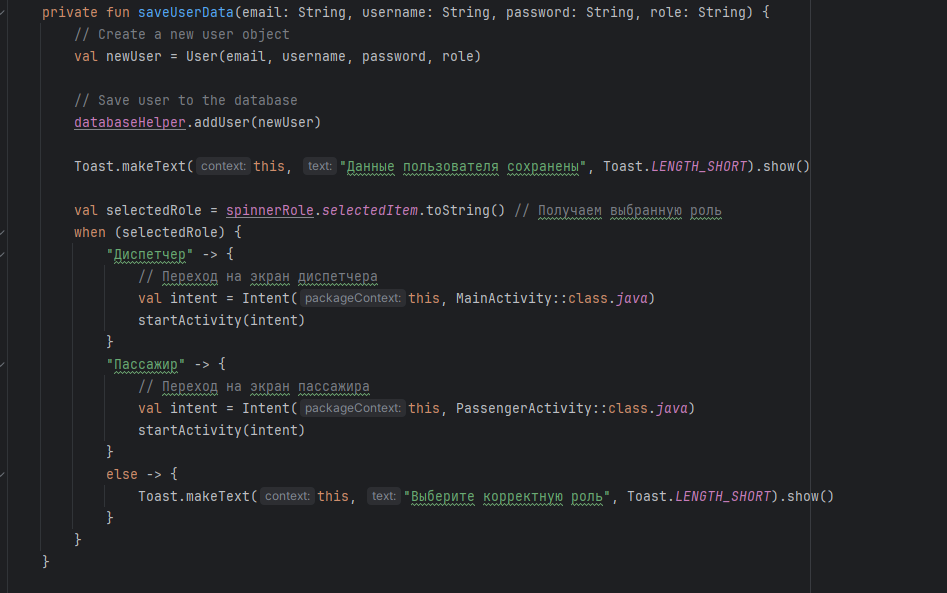
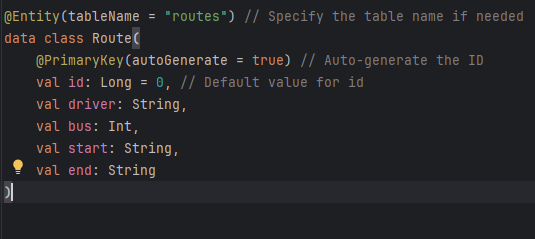
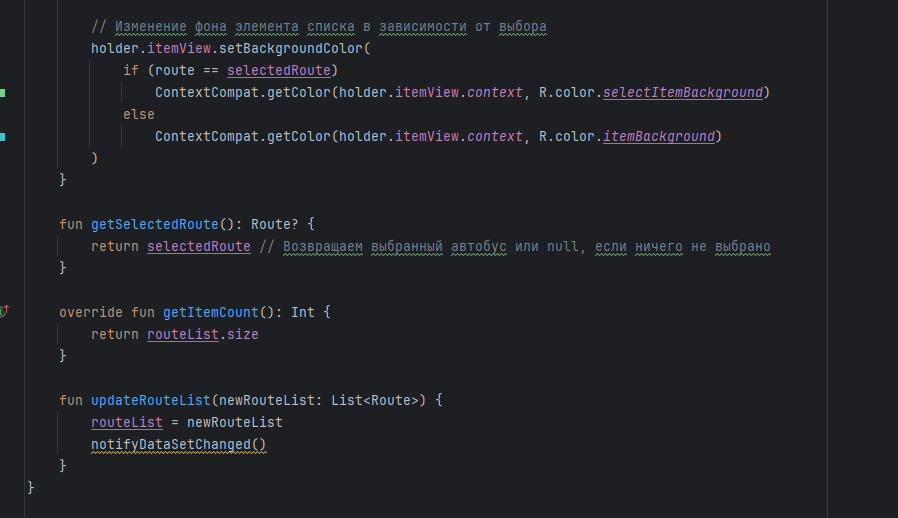
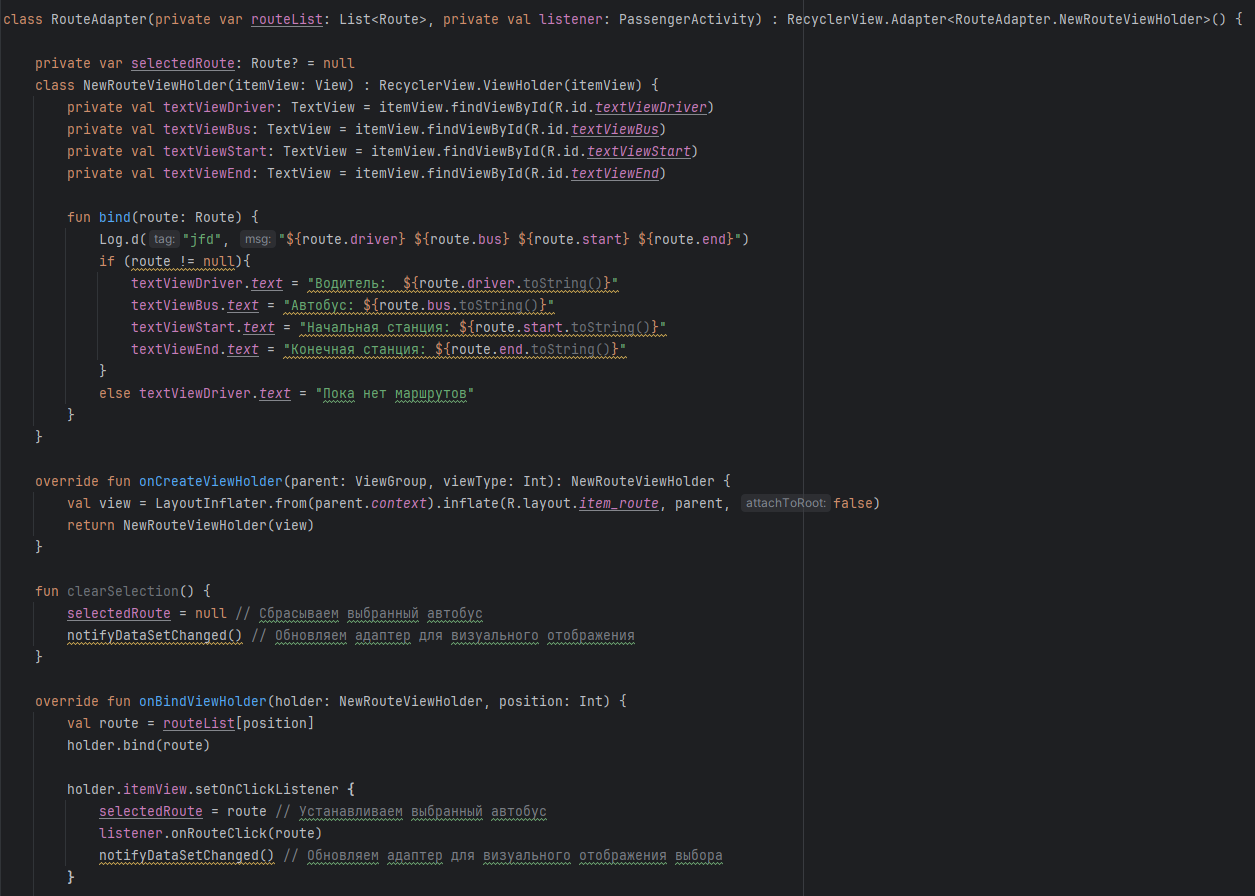
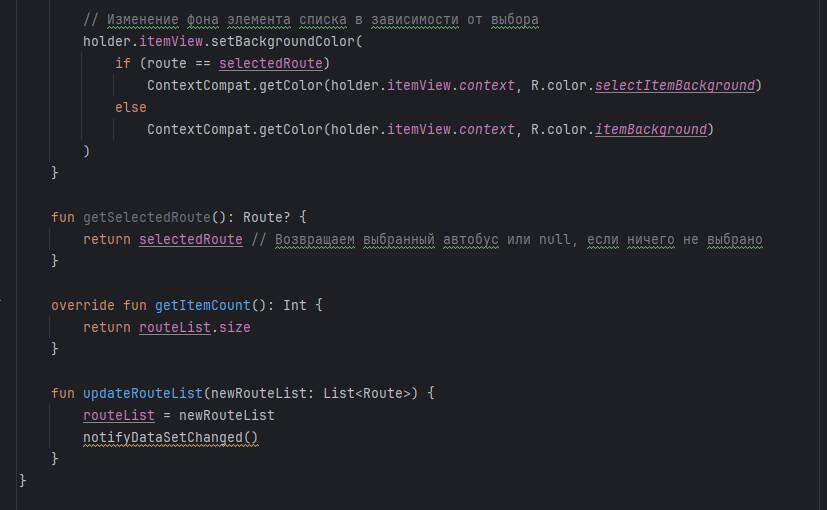
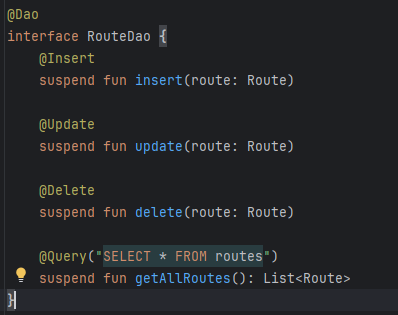
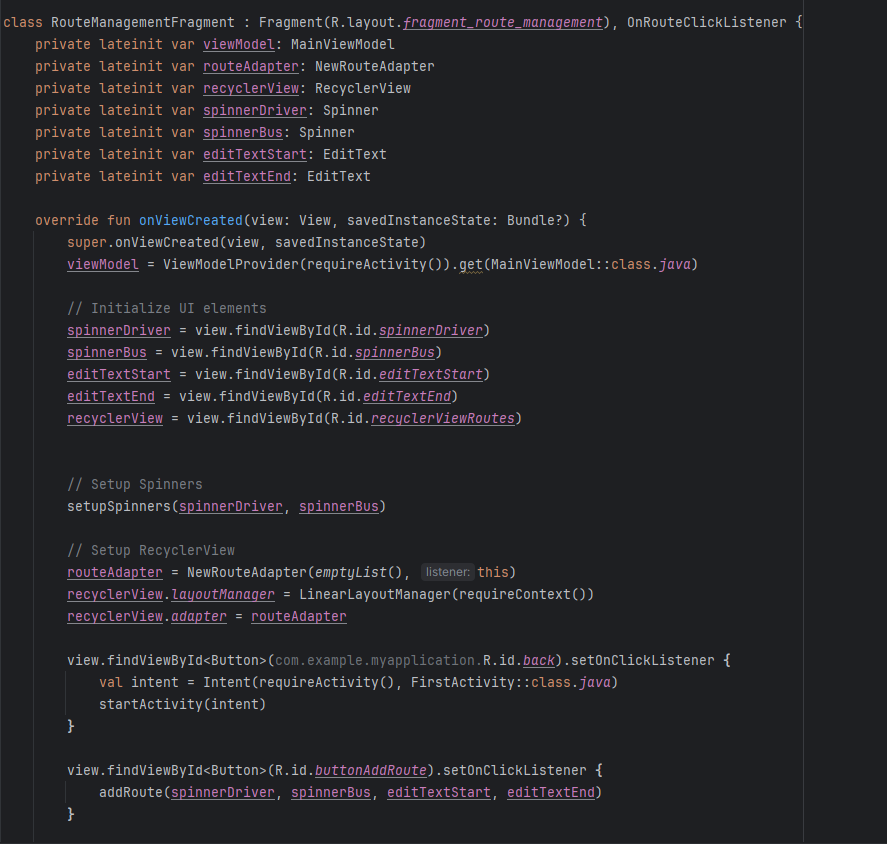
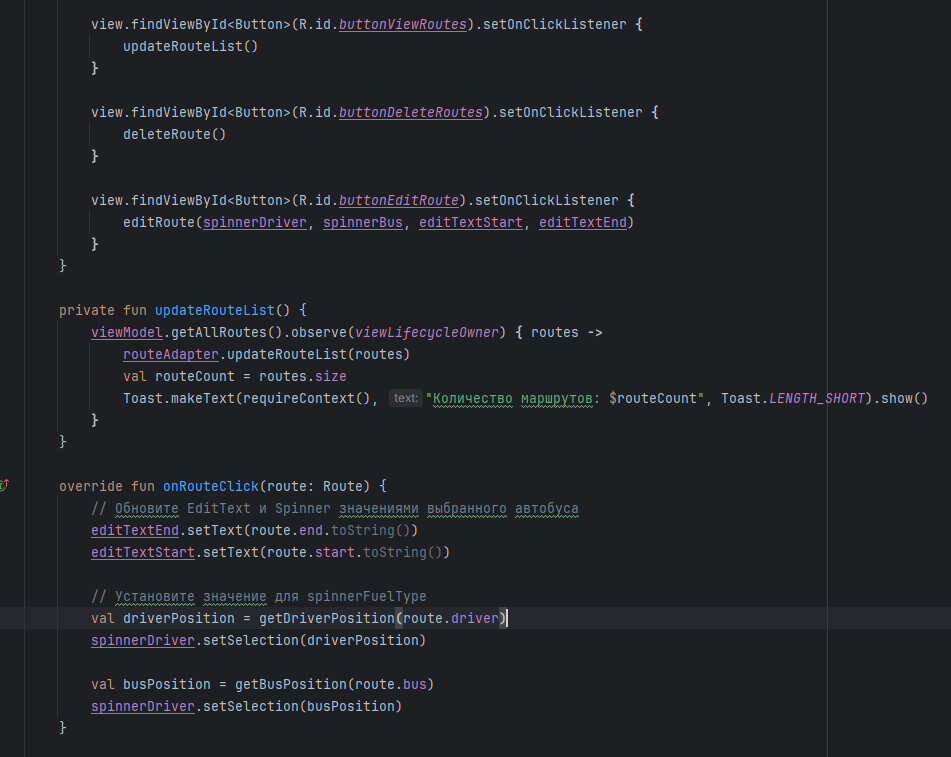
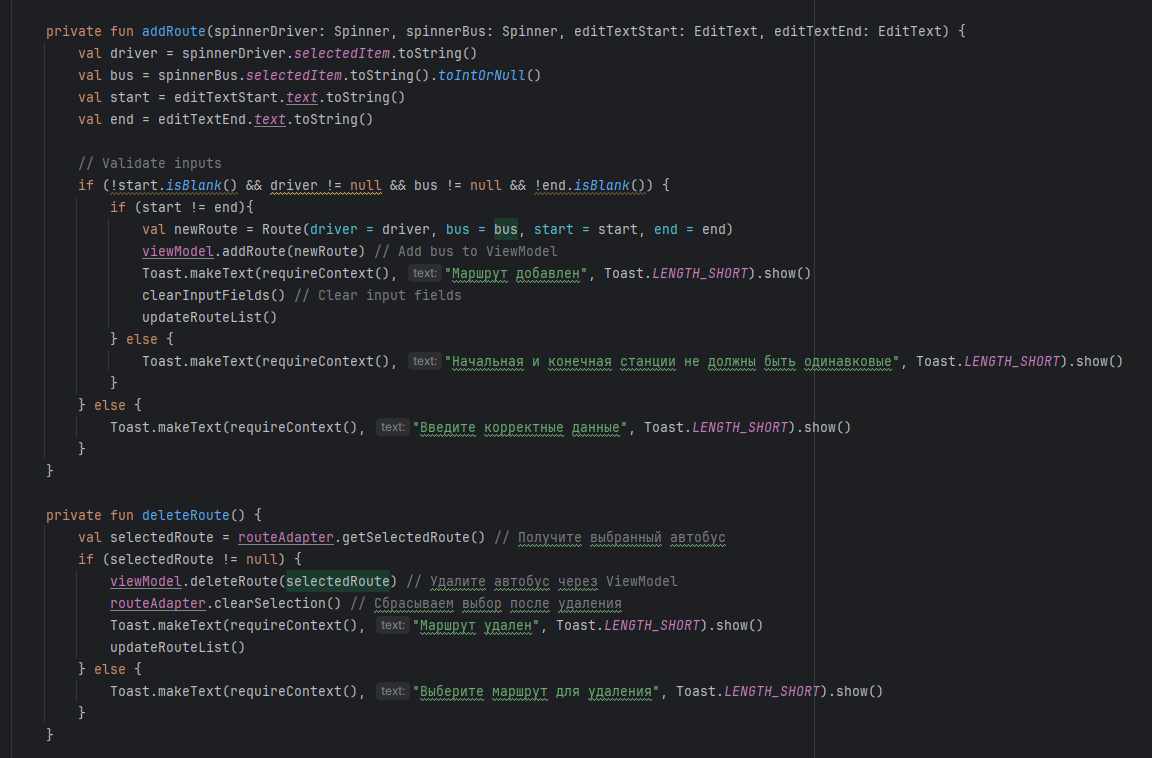
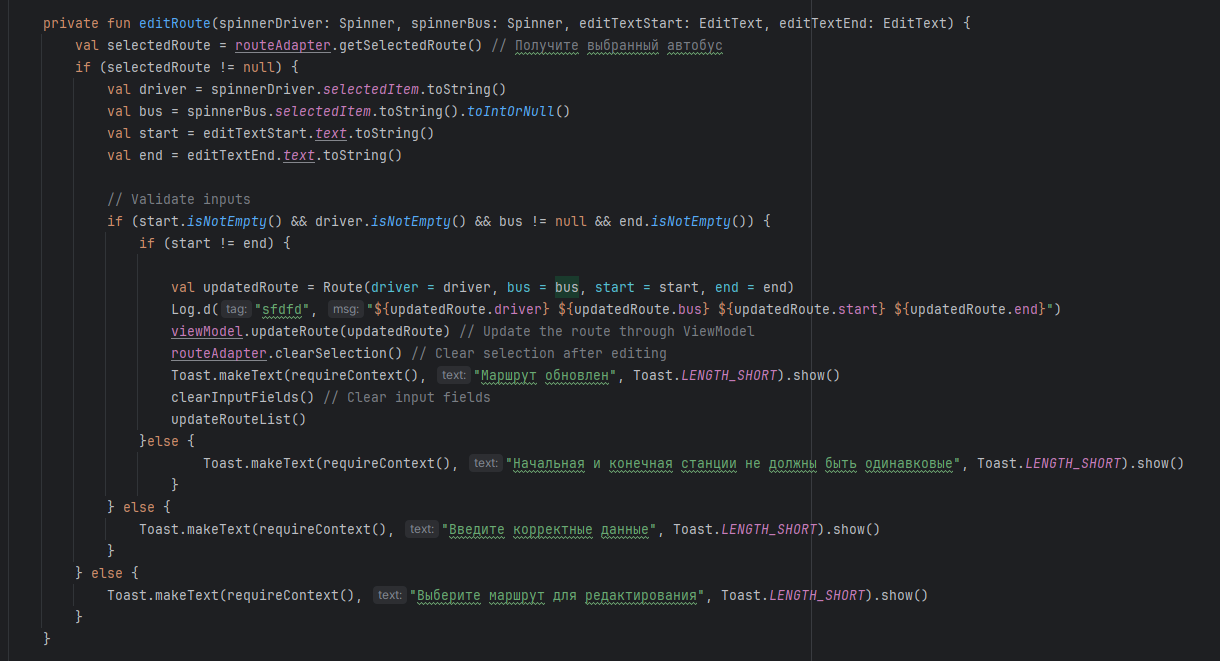
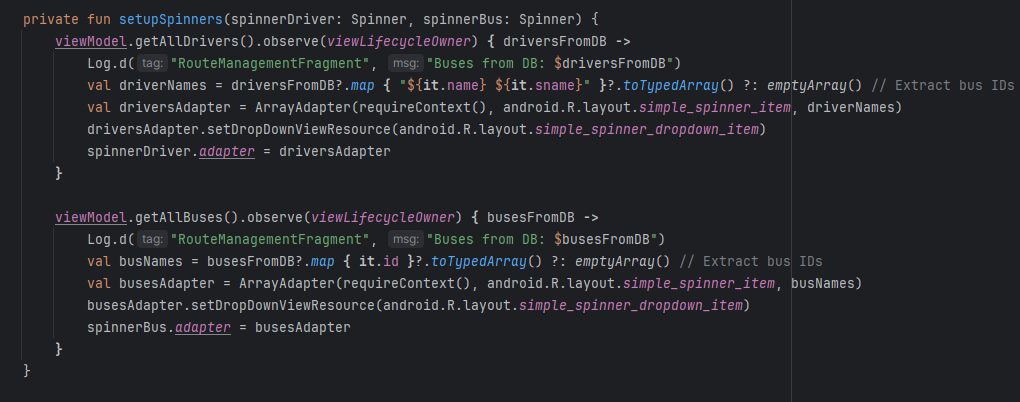
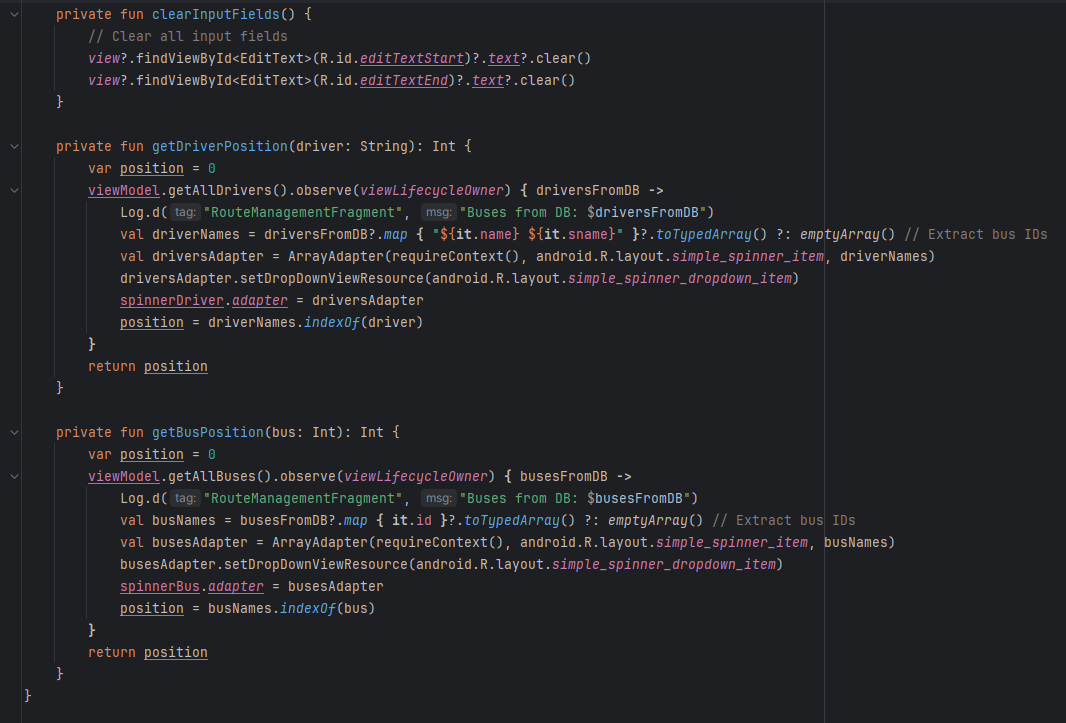
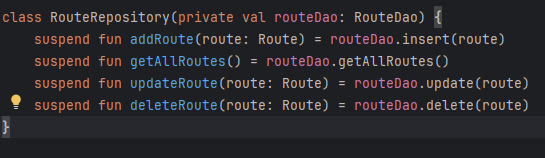
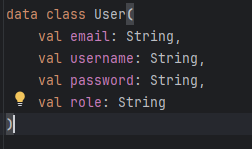
* **Действия:**
  1. Ввести неверный пароль.
  2. Нажать кнопку **"Войти"**.
* **Ожидаемый результат:**
  1. Появляется сообщение: "Не правильный логин или пароль".

**Сценарий 3: Регистрация нового пассажира**

* **Действия:**
  1. Ввести данные:
     + Email: passenger@mail.ru
     + Логин: passenger1
     + Пароль: qwerty123
     + Роль: Пассажир
  2. Нажать кнопку **"Зарегистрироваться"**.
* **Ожидаемый результат:**
  1. Открывается PassengerActivity с возможностью просмотра маршрутов.
  2. Используемые инструменты

Android Studio Ladybug 2024.2.2  
Язык программирование Kotlin  
Название проекта My Application

* 1. Описание пользовательского интерфейса

AppDatabase.kt  
  
  
  
Класс Bus  
  
  
BusAdapter  
  
  
  
  
BusDao  
  
BusManagmentFragment  
  
  
  
  
  
  
BusRepository  
  
Driver  
  
DriverActivity  
  
  
  
DriverAdapter  
  
  
DriverDao  
  
DriverManagmentFragment  
  
  
  
  
  
  
DriverRepository  
  
FirstActivity  
  
  
FragmentStateAdapter.kt  
  
MainActivity  
  
MainFragment  
  
MainViewModel  
  
Интерфейсы OnBus/Driver/RouteClickListener  
  
  
  
PassenegerActivity  
  
RegistrActivity  
  
  
  
Класс Route  
  
RouteAdapter.kt  
  
  
  
  
RouteDao  
  
RouteManagmentFragment  
  
  
  
  
  
  
RouteRepository  
  
Класс User  


* 1. Скриншоты приложения

