

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

РМП задание 5

Выполнил: Лежень Андрей Игоревич

Группа: ПР-31

Преподаватель: Мирошниченко Г.В.

2024

Содержание

[1. **Задание приложение для турфирмы** 3](#_30j0zll)

[1.1](#_1fob9te) Описание задачи 4

[1.2](#_1t3h5sf) Структура проекта 4

[1.3 Описание разработанных функций 4](#_3znysh7)

[1.4 Алгоритм решения 5](#_4d34og8)

[1.5 Используемые библиотеки 6](#_2et92p0)

[1.6 Тестовые случаи 6](#_2s8eyo1)

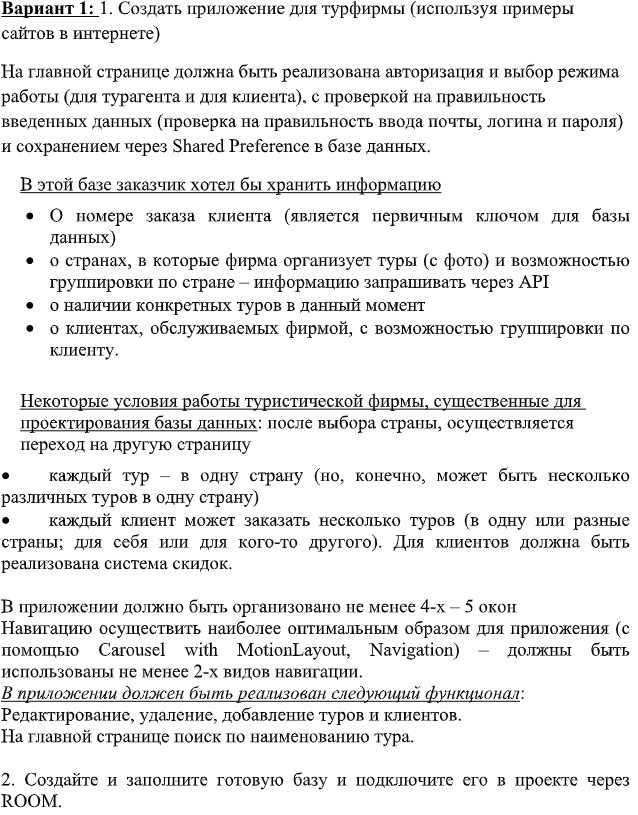
[1.7 Используемые инструменты 6](#_17dp8vu)

[1.8 Описание пользовательского интерфейса 7](#_tyjcwt)

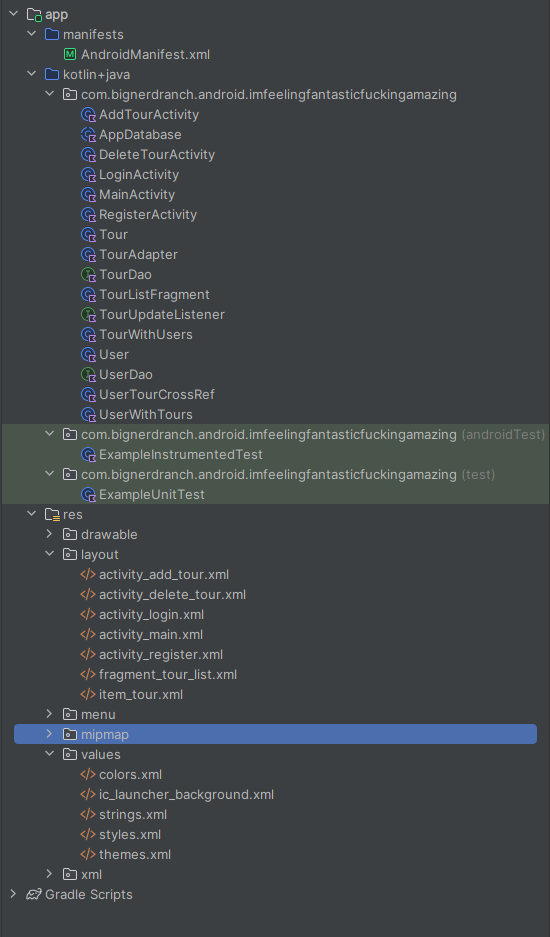
[1.9 Приложение (pr screen экранов) 11](#_3dy6vkm)

# 1. **Задание Неявные интенты.**

## Описание задачи



## Структура проекта



* AddTourActivity.kt - Активность для добавления нового тура пользователем.
* AppDatabase.kt - Конфигурация базы данных Room и получение доступа к DAO для пользователей и туров.
* DeleteTourActivity.kt - Активность для удаления существующего тура из базы данных.
* LoginActivity.kt - Активность входа в приложение, проверяет учетные данные пользователя.
* MainActivity.kt - Главная активность, содержащая навигацию и отображение фрагментов с турами.
* RegisterActivity.kt - Активность регистрации нового пользователя с валидацией данных.
* Tour.kt - Модель данных тура, описывает структуру объекта тура в базе данных.
* TourAdapter.kt - Адаптер для отображения списка туров в RecyclerView.
* TourDao.kt - DAO с методами для взаимодействия с таблицей туров в базе данных.
* TourListFragment.kt - Фрагмент для отображения списка туров (все туры или туры пользователя).
* TourUpdateListener.kt - Интерфейс для отслеживания обновлений списка туров.
* TourWithUsers.kt - Класс для получения тура вместе со списком пользователей, связанных с ним.
* User.kt - Модель данных пользователя, описывает структуру объекта пользователя в базе данных.
* UserDao.kt - DAO с методами для взаимодействия с таблицей пользователей и связями с турами.
* UserTourCrossRef.kt - Класс для установления связи многие-ко-многим между пользователями и турами.
* UserWithTours.kt - Класс для получения пользователя вместе со списком его приобретенных туров.

## Описание разработанных функций

* onCreate(savedInstanceState: Bundle?) - Инициализирует активность, устанавливает интерфейс и обрабатывает добавление нового тура.
* onOptionsItemSelected(item: MenuItem): Boolean - Обрабатывает нажатие кнопки «Назад» в панели действий.
* userDao(): UserDao - Предоставляет доступ к объекту UserDao для операций с пользователями.
* tourDao(): TourDao - Предоставляет доступ к объекту TourDao для операций с турами.
* getDatabase(context: Context): AppDatabase - Возвращает экземпляр базы данных приложения (синглтон).
* onCreate(savedInstanceState: Bundle?) - Инициализирует активность и загружает список туров для удаления.
* loadTours(container: LinearLayout) - Загружает все туры и создает кнопки для их удаления.
* deleteTour(tourId: Int) - Удаляет тур с указанным идентификатором из базы данных.
* onCreate(savedInstanceState: Bundle?) - Инициализирует активность, проверяет, авторизован ли пользователь, и настраивает интерфейс входа.
* performLogin(login: String, password: String) - Выполняет вход пользователя, проверяя логин и пароль.
* onCreate(savedInstanceState: Bundle?) - Инициализирует основную активность, настраивает нижнюю навигацию и элементы интерфейса.
* onTourListUpdated() - Обновляет профиль пользователя при изменении списка туров.
* reloadProfile() - Перезагружает профиль пользователя, заменяя фрагмент.
* onCreate(savedInstanceState: Bundle?) - Инициализирует активность регистрации и настраивает элементы интерфейса.
* validateFields(login: String, email: String, password: String, confirmPassword: String): Boolean - Проверяет корректность введённых данных для регистрации.
* onOptionsItemSelected(item: MenuItem): Boolean - Обрабатывает нажатие кнопки «Назад» в панели действий.
* onCreateViewHolder(parent: ViewGroup, viewType: Int): TourViewHolder - Создаёт и инициализирует ViewHolder для элемента списка туров.
* onBindViewHolder(holder: TourViewHolder, position: Int) - Привязывает данные тура к ViewHolder для отображения.
* getItemCount(): Int - Возвращает количество туров в списке.
* Класс TourViewHolder - Внутренний класс, содержащий ссылки на элементы интерфейса для отображения тура.
* insertTour(tour: Tour) - Вставляет новый тур в базу данных.
* deleteTour(tour: Tour) - Удаляет тур из базы данных.
* getTourWithUsers(tourId: Int): List<TourWithUsers> - Получает тур с связанными пользователями.
* getAllTours(): List<Tour> - Возвращает список всех туров.
* getTourById(tourId: Int): Tour? - Получает тур по его идентификатору.
* getToursByAgent(agentId: Int): List<Tour> - Получает туры, зарегистрированные определённым агентом.
* onAttach(context: Context) - Вызывается при присоединении фрагмента к активности; инициализирует интерфейс обновления туров.
* onCreate(savedInstanceState: Bundle?) - Инициализирует фрагмент и получает аргументы.
* onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? - Создаёт и возвращает представление фрагмента.
* onViewCreated(view: View, savedInstanceState: Bundle?) - Вызывается после создания представления фрагмента; инициализирует RecyclerView и загружает туры.
* loadTours() - Загружает туры из базы данных и отображает их в RecyclerView.
* newInstance(showFiltered: Boolean) - Создаёт новый экземпляр фрагмента с переданными параметрами.
* onTourListUpdated() - Метод интерфейса, вызываемый при обновлении списка туров.
* deleteUserTourCrossRef(userId: Int, tourId: Int) - Удаляет связь между пользователем и туром

## 1.4 Алгоритм решения

## 1.5 Используемые библиотеки



## 1.6 Тестовые случаи

**1. Тесты регистрации пользователя (RegisterActivity.kt)**

Тестовый случай: Регистрация нового пользователя с корректными данными.

Ожидаемый результат: Пользователь успешно зарегистрирован; отображается сообщение "Регистрация успешна"; происходит переход к экрану входа.

Тестовый случай: Попытка регистрации с логином меньше 4 символов.

Ожидаемый результат: Регистрация отклонена; отображается сообщение "Логин должен быть длиннее 3 символов".

Тестовый случай: Попытка регистрации с некорректным email (не содержит "gmail.com", "mail.ru" или символ '@' отсутствует на 4 позиции).

Ожидаемый результат: Регистрация отклонена; отображается сообщение "Некорректный email".

Тестовый случай: Ввод пароля длиной менее 6 символов.

Ожидаемый результат: Регистрация отклонена; отображается сообщение "Пароль должен быть длиннее 5 символов".

Тестовый случай: Введенные пароли не совпадают.

Ожидаемый результат: Регистрация отклонена; отображается сообщение "Пароли не совпадают".

**2. Тесты входа в систему (LoginActivity.kt)**

Тестовый случай: Вход с корректными логином и паролем.

Ожидаемый результат: Вход успешен; отображается сообщение "Вход успешен"; происходит переход к MainActivity.

Тестовый случай: Вход с некорректным паролем.

Ожидаемый результат: Вход отклонен; отображается сообщение "Неверные логин или пароль".

Тестовый случай: Автоматический вход при уже залогиненом пользователе.

Ожидаемый результат: Пользователь перенаправляется сразу на MainActivity без необходимости ввода данных.

**3. Тесты добавления тура (AddTourActivity.kt)**

Тестовый случай: Агент по турам добавляет новый тур с корректно заполненными полями.

Ожидаемый результат: Тур успешно добавлен; отображается сообщение "Тур добавлен успешно"; активность завершается.

Тестовый случай: Попытка добавить тур с пустыми полями или некорректной стоимостью.

Ожидаемый результат: Добавление тура отклонено; отображается сообщение "Пожалуйста, заполните все поля корректно".

**4. Тесты удаления тура (DeleteTourActivity.kt)**

Тестовый случай: Агент по турам удаляет существующий тур.

Ожидаемый результат: Тур удалён из базы данных; отображается сообщение "Тур удалён"; активность завершается.

**5. Тесты покупки тура (TourAdapter.kt, TourListFragment.kt)**

Тестовый случай: Обычный пользователь покупает доступный тур.

Ожидаемый результат: Тур добавлен в список туров пользователя; отображается сообщение "Тур куплен!".

Тестовый случай: Пользователь пытается купить тур, будучи агентом.

Ожидаемый результат: Кнопка "Купить тур" недоступна или не отображается.

Тестовый случай: Пользователь отказывается от ранее купленного тура из своего профиля.

Ожидаемый результат: Тур удалён из списка туров пользователя; отображается сообщение "Вы отказались от тура!"; список туров обновляется.

**6. Тесты навигации и интерфейса (MainActivity.kt)**

Тестовый случай: Пользователь использует нижнюю навигацию для перехода в профиль.

Ожидаемый результат: Отображается список купленных туров пользователя.

Тестовый случай: Пользователь выходит из приложения через пункт меню "Exit".

Ожидаемый результат: Текущий пользователь выходит из системы; возвращение на экран входа.

Тестовый случай: Агент по турам видит дополнительные кнопки для управления турами.

Ожидаемый результат: Кнопки "Добавить тур", "Редактировать тур", "Удалить тур" отображаются для агента.

**7. Тесты базы данных и DAO (AppDatabase.kt, UserDao.kt, TourDao.kt)**

Тестовый случай: Проверка сохранения и получения данных пользователя.

Ожидаемый результат: Данные пользователя корректно сохраняются и извлекаются из базы данных.

Тестовый случай: Проверка связи "многие ко многим" между пользователями и турами.

Ожидаемый результат: Пользователь может иметь несколько туров; тур может быть приобретён несколькими пользователями.

Тестовый случай: Обновление данных при изменении версии базы данных.

Ожидаемый результат: При изменении версии выполняется миграция без потери данных (или данные сбрасываются при использовании fallbackToDestructiveMigration).

**8. Тесты фрагментов и обновления интерфейса (TourListFragment.kt, TourUpdateListener.kt)**

Тестовый случай: Обновление списка туров после покупки или отказа от тура.

Ожидаемый результат: Список туров обновляется автоматически без перезапуска приложения или активности.

Тестовый случай: Отображение всех туров или только туров пользователя в зависимости от переданных аргументов.

Ожидаемый результат: При showFiltered = true отображаются только туры пользователя; иначе — все доступные туры.

**9. Тесты валидации данных (RegisterActivity.kt, AddTourActivity.kt)**

Тестовый случай: Попытка добавить тур с некорректными данными (например, отрицательная стоимость).

Ожидаемый результат: Тур не добавляется; отображается сообщение об ошибке ввода данных.

Тестовый случай: Регистрация пользователя с уже существующим логином.

Ожидаемый результат: Регистрация отклонена; отображается сообщение о том, что логин уже занят.

## 1.7 Используемые инструменты

Язык программирования Kotlin, программа Android Studio Hedgehog 2024.

Приложение проекта Empty Activity

## 1.8 Описание пользовательского интерфейса

При запуске программы загружается экран вида (рис ).

Рис.

## 1.9 Приложение (pr screen экранов)

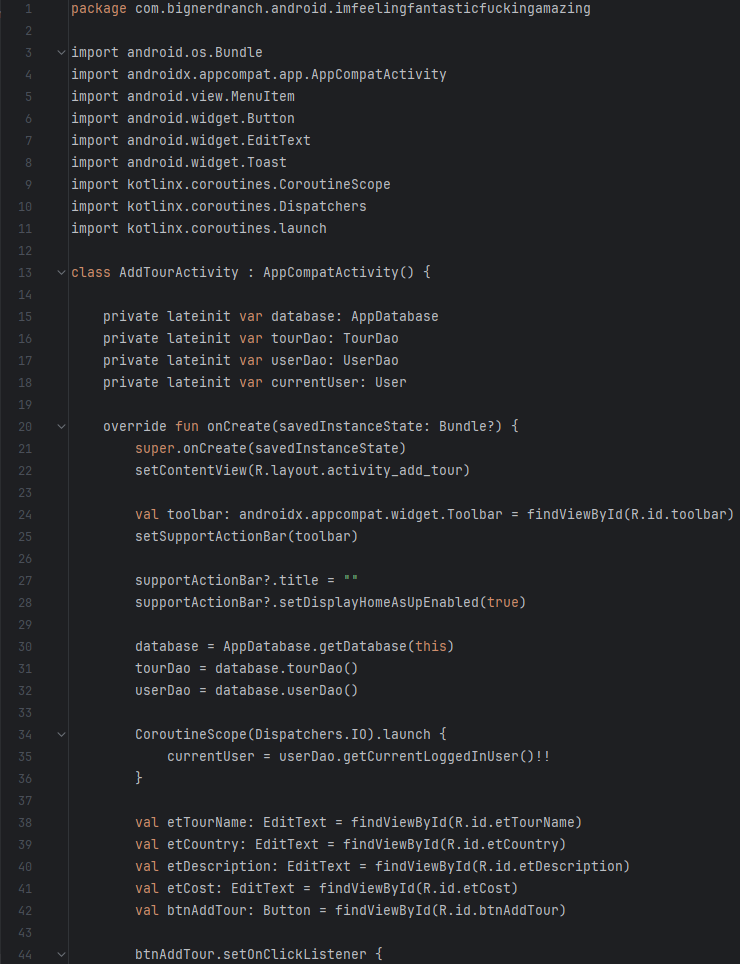
****

Рис.

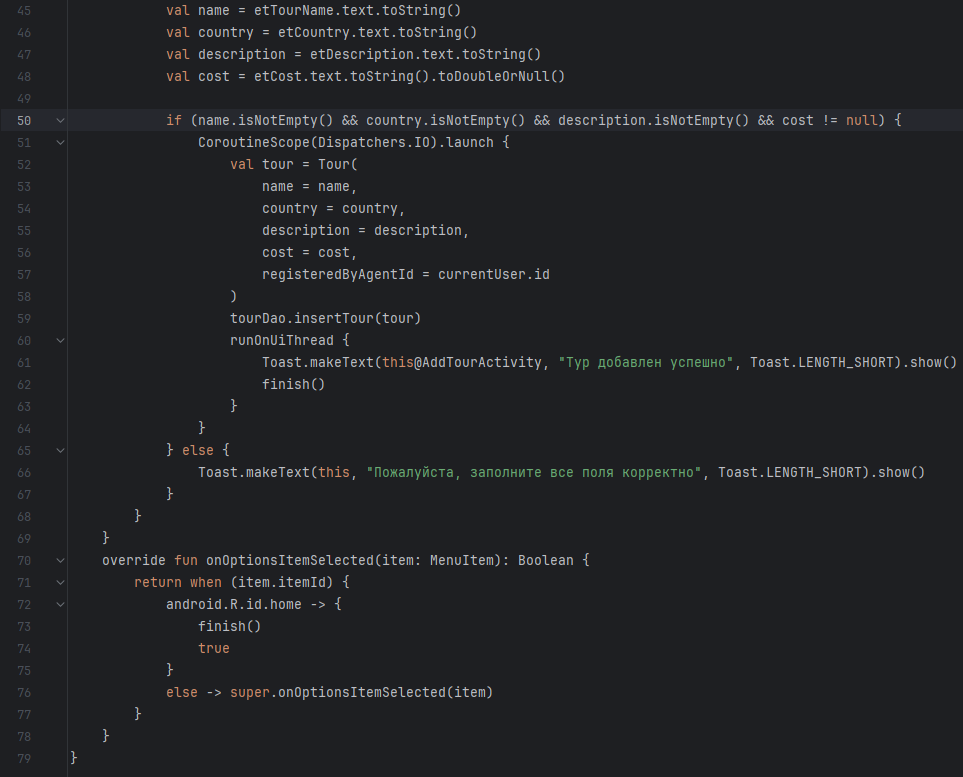


Рис.

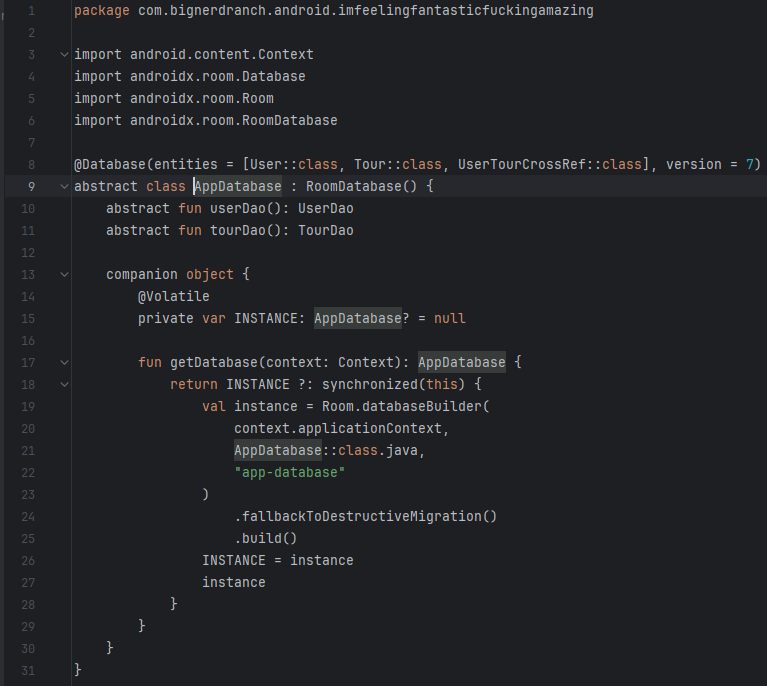
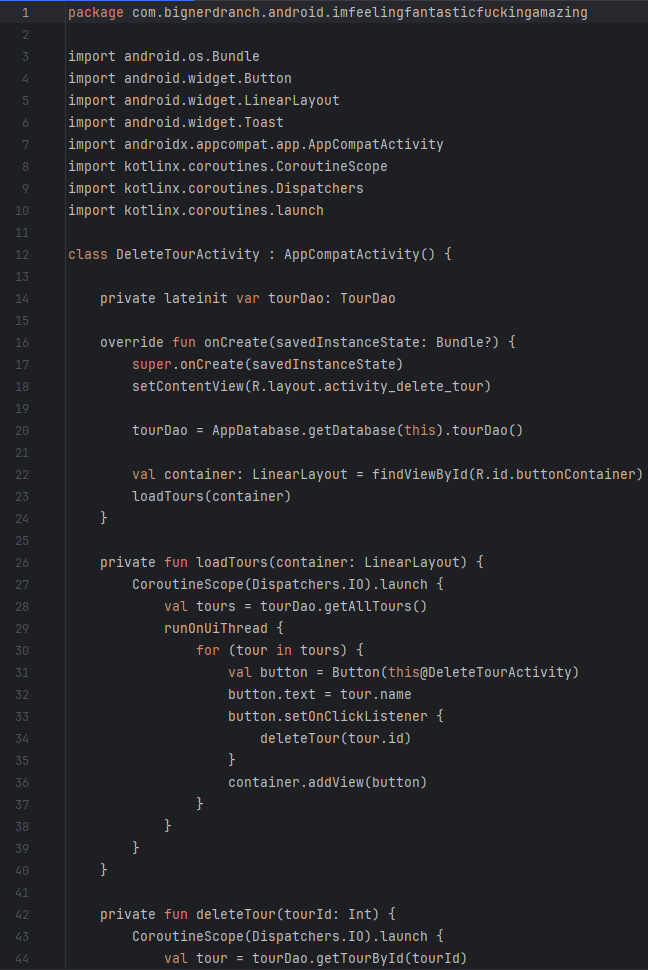
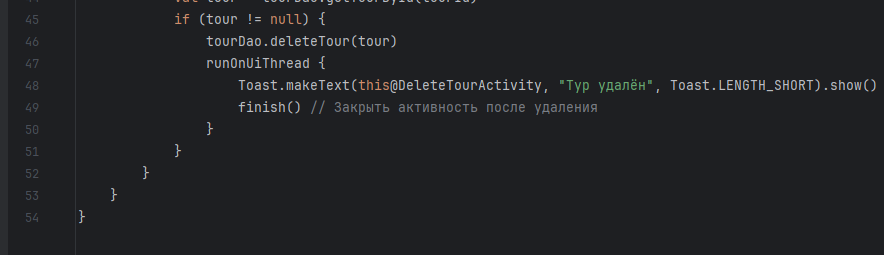


Рис.

  
Рис.

  
Рис.

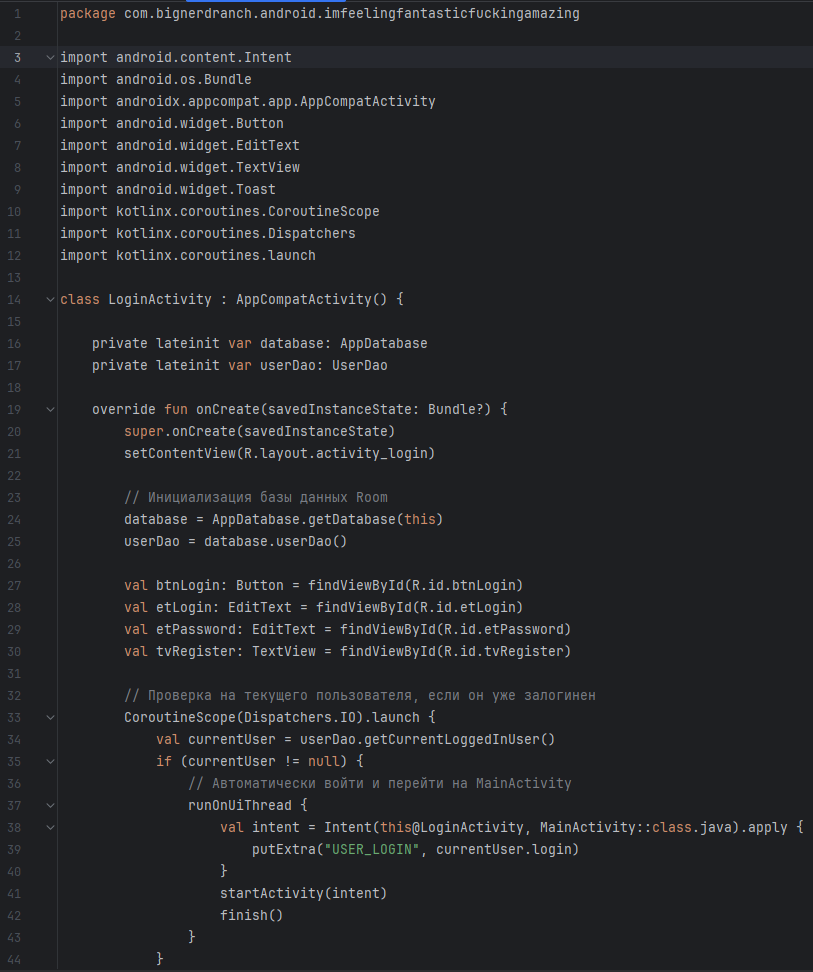


Рис.

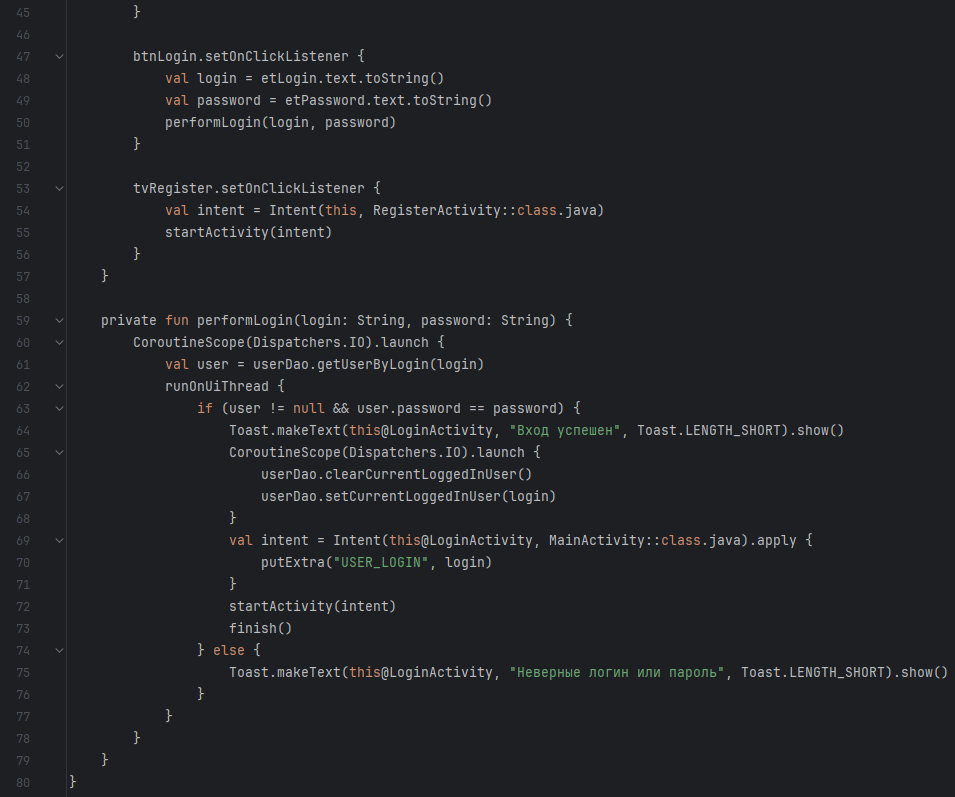


Рис.

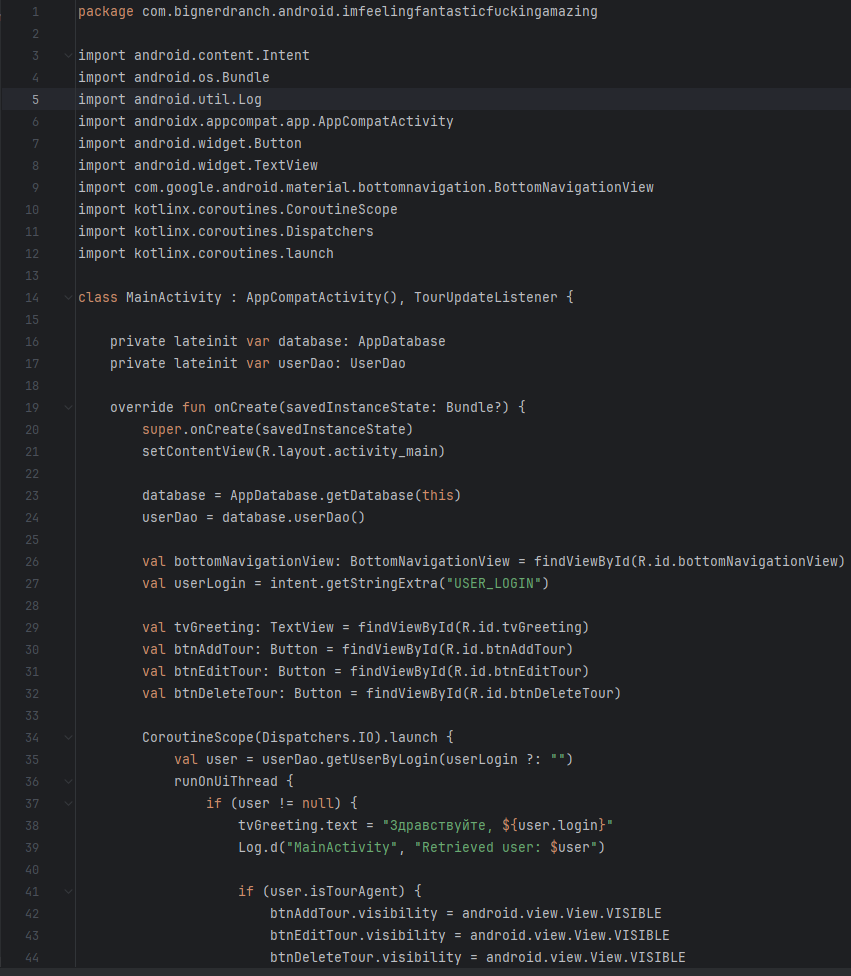


Рис.

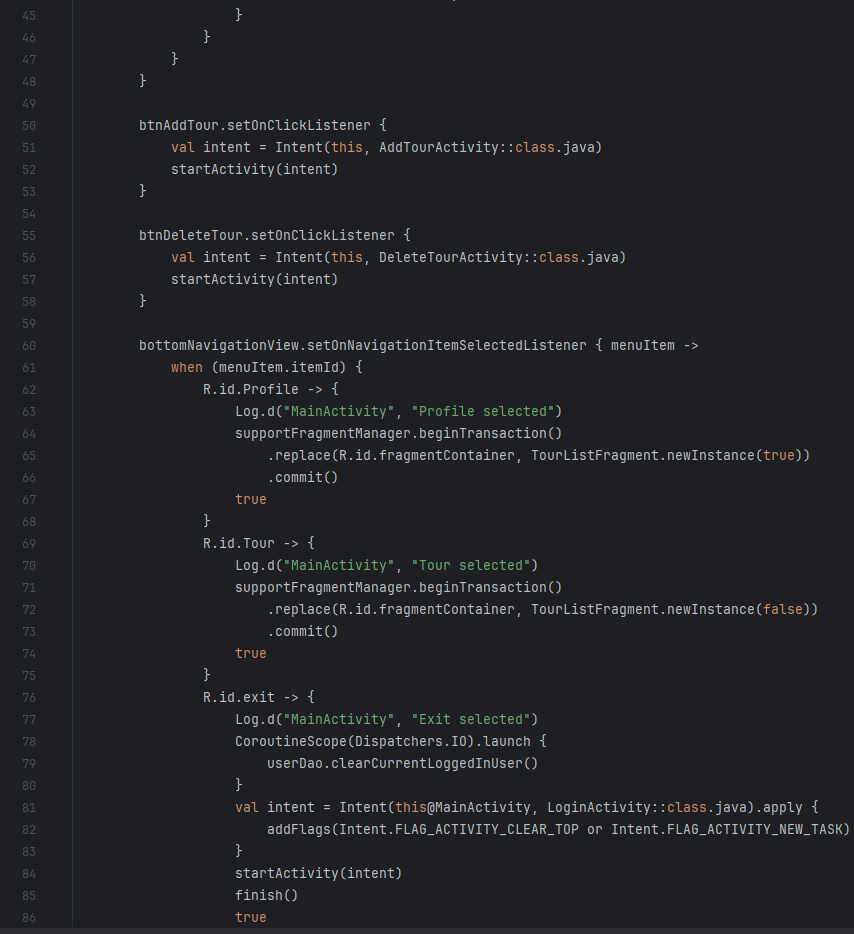


Рис.



Рис.

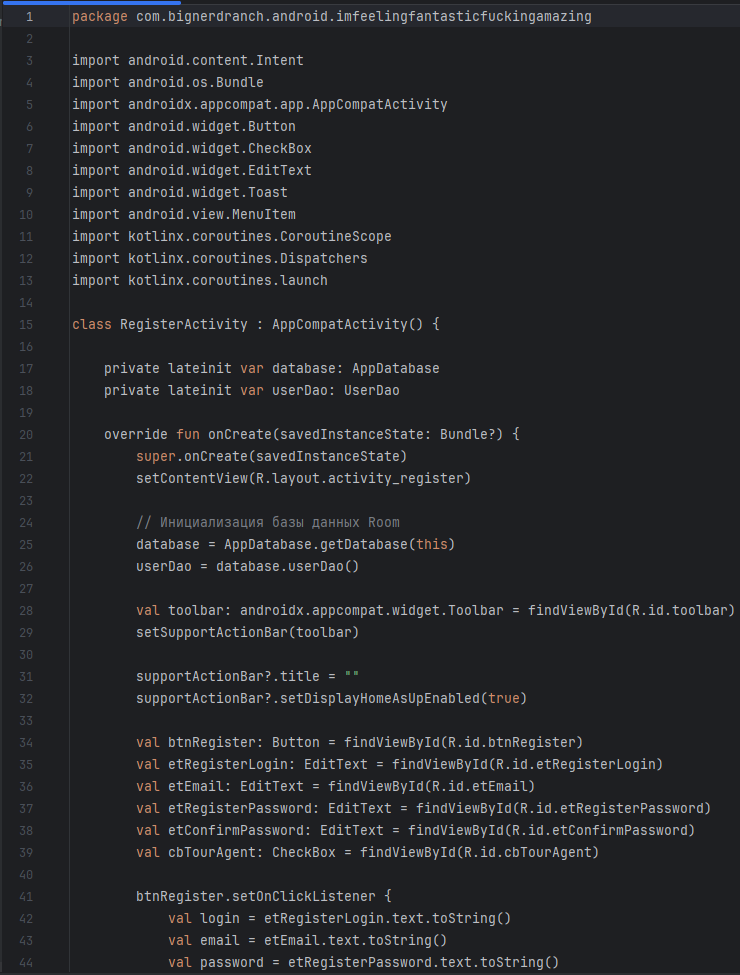


Рис.

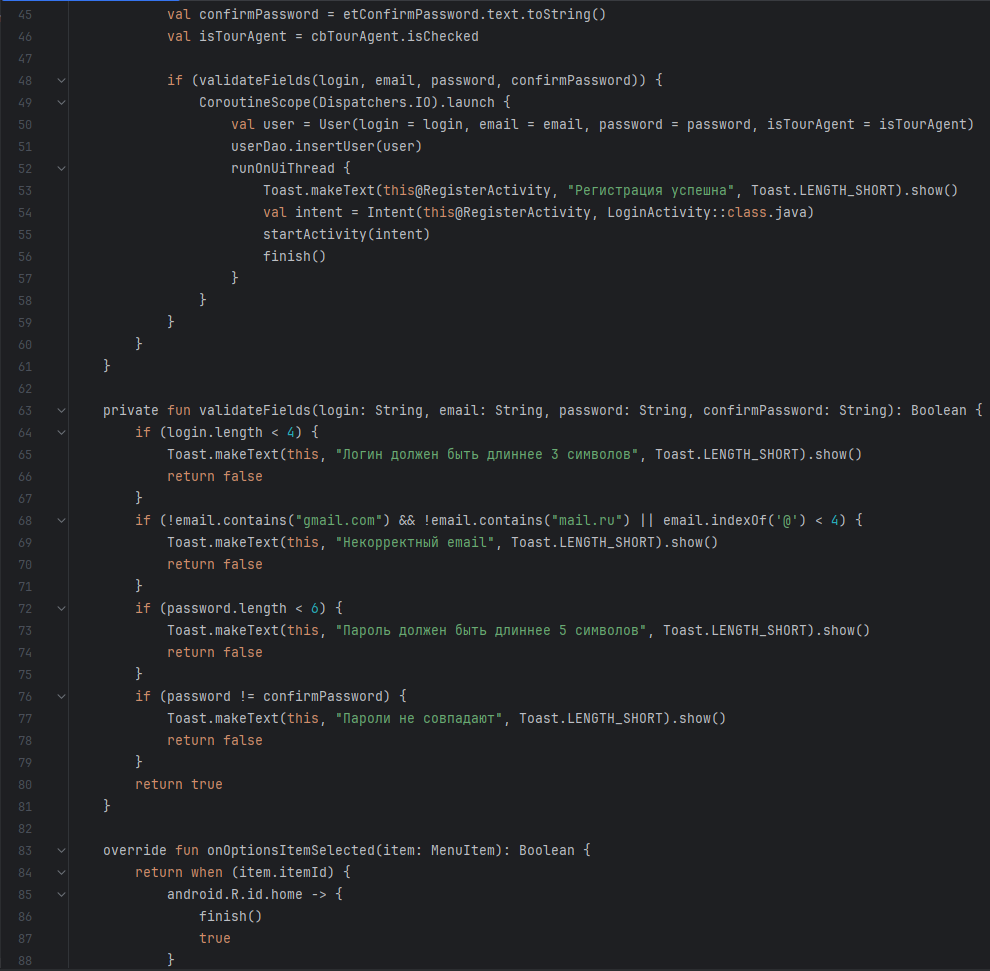


Рис.

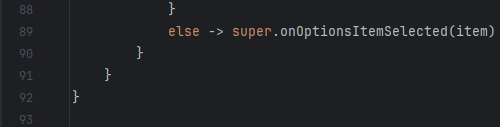


Рис.

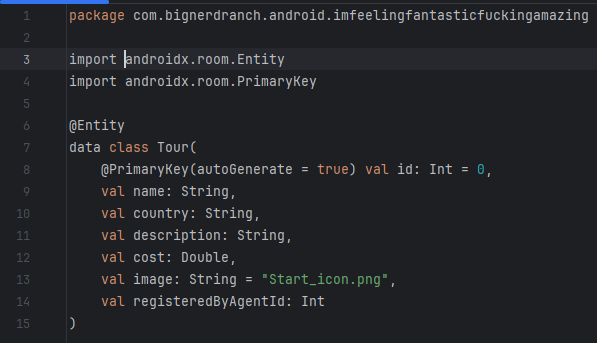


Рис.

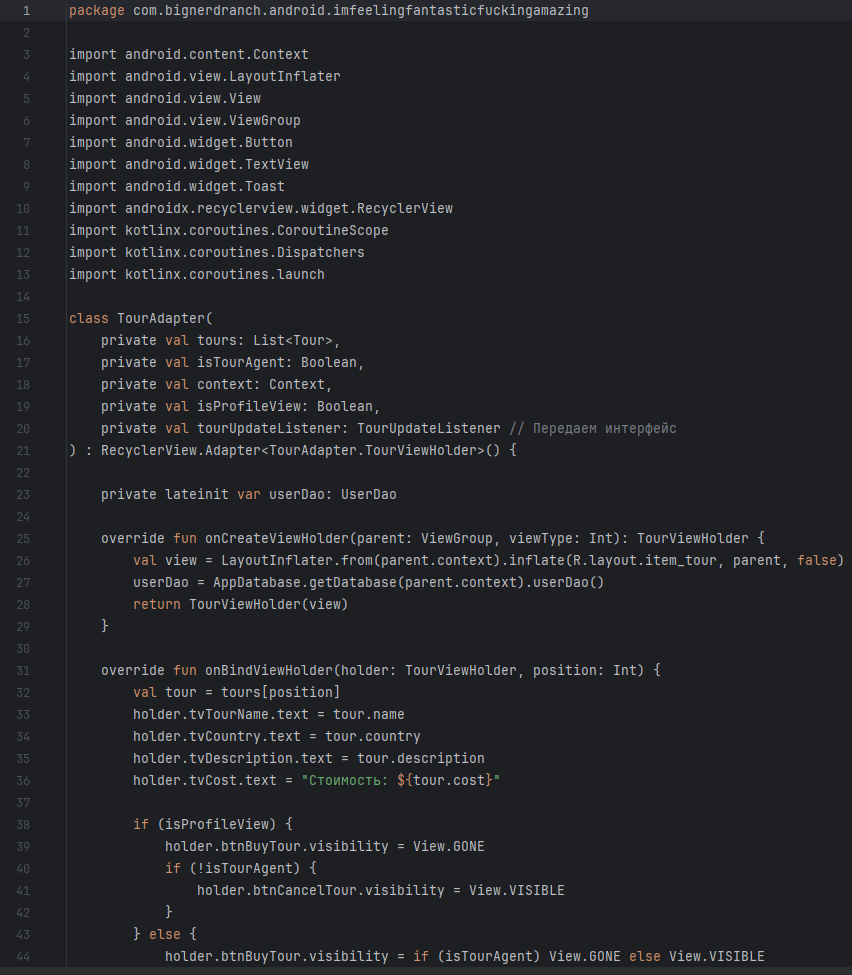


Рис.

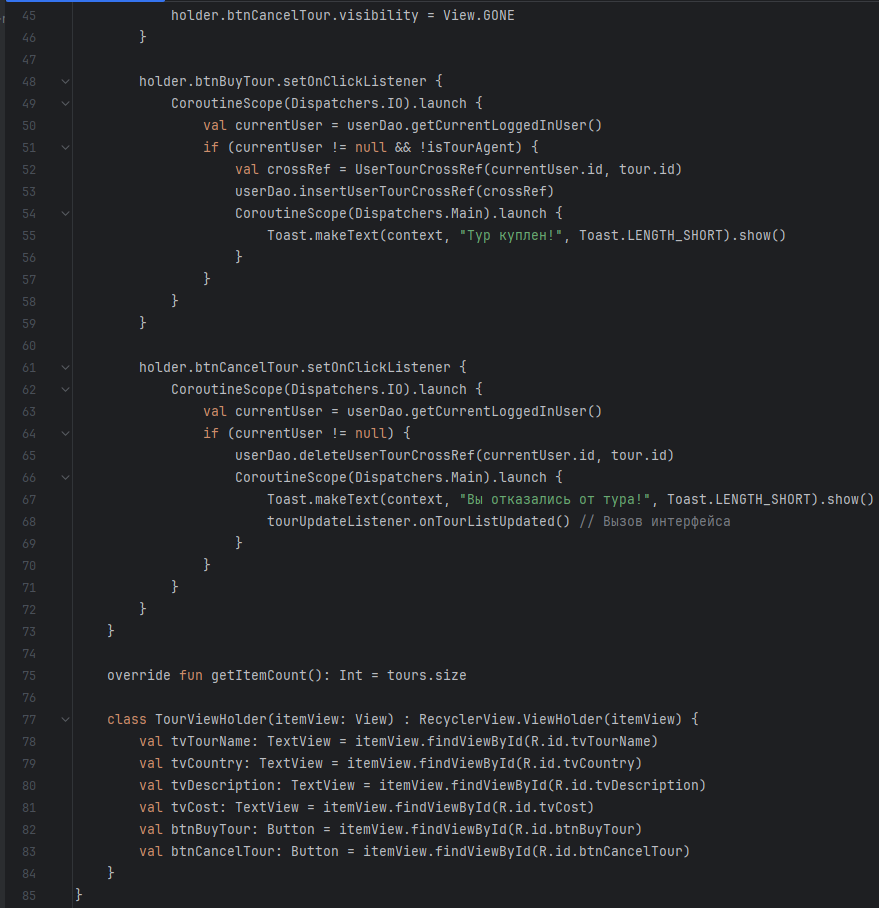


Рис.

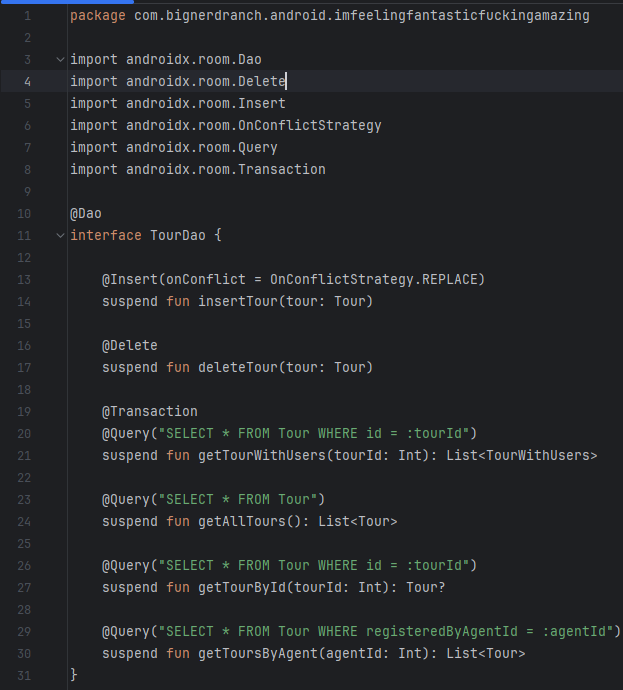


Рис.



Рис.

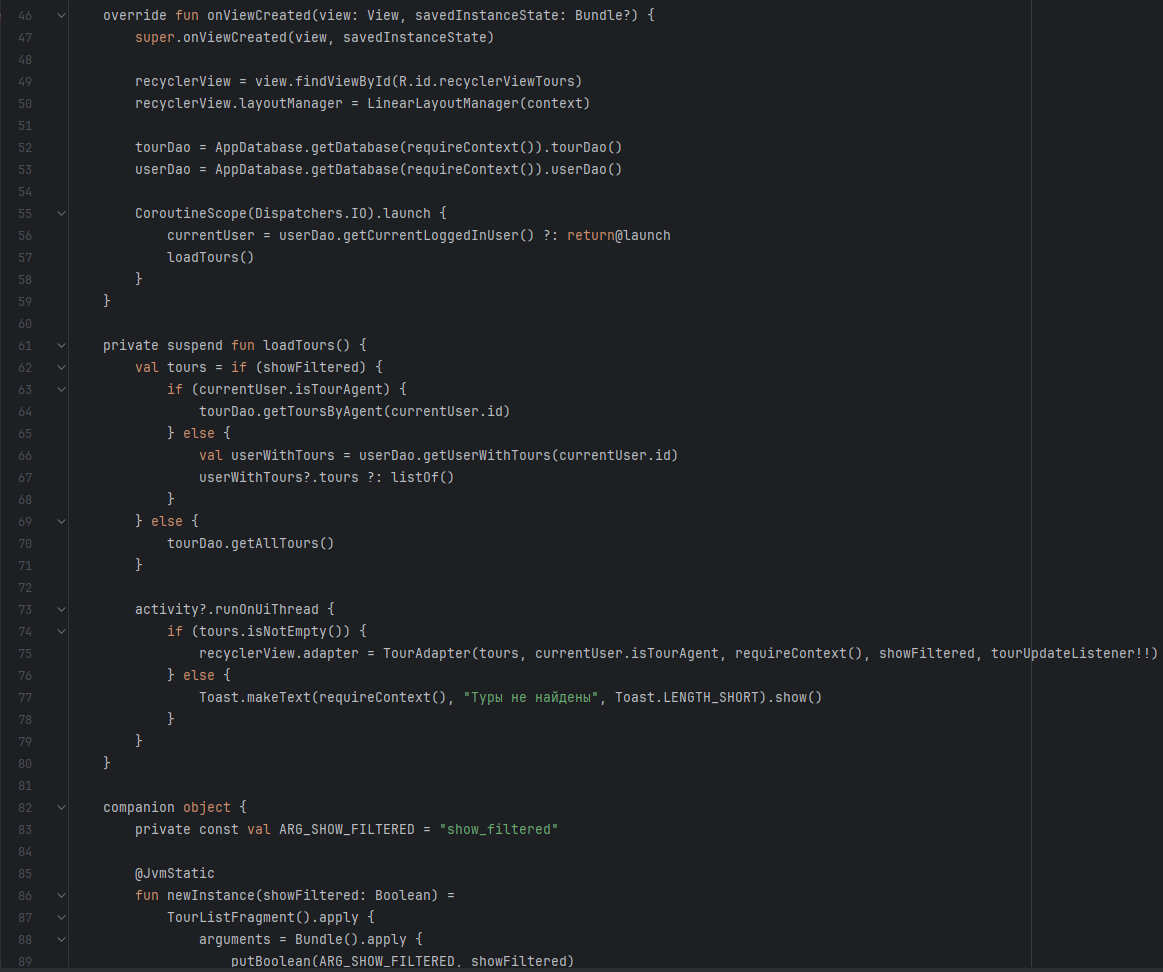


Рис.

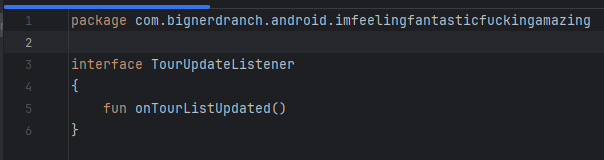


Рис.

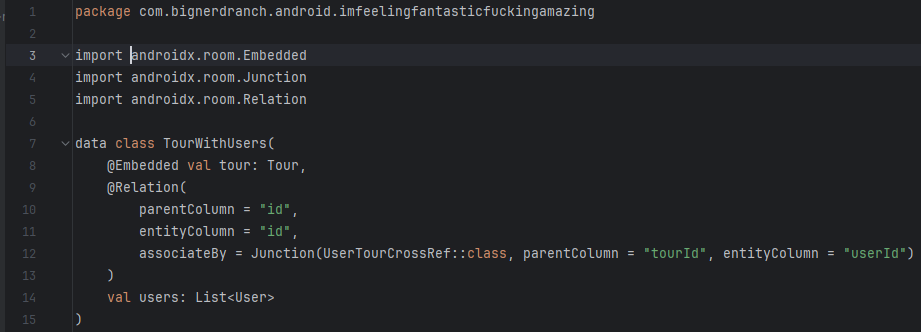


Рис.

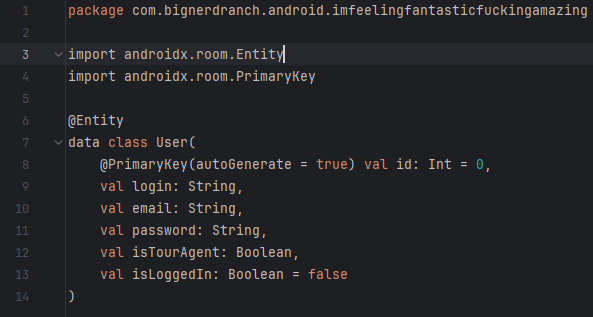


Рис.

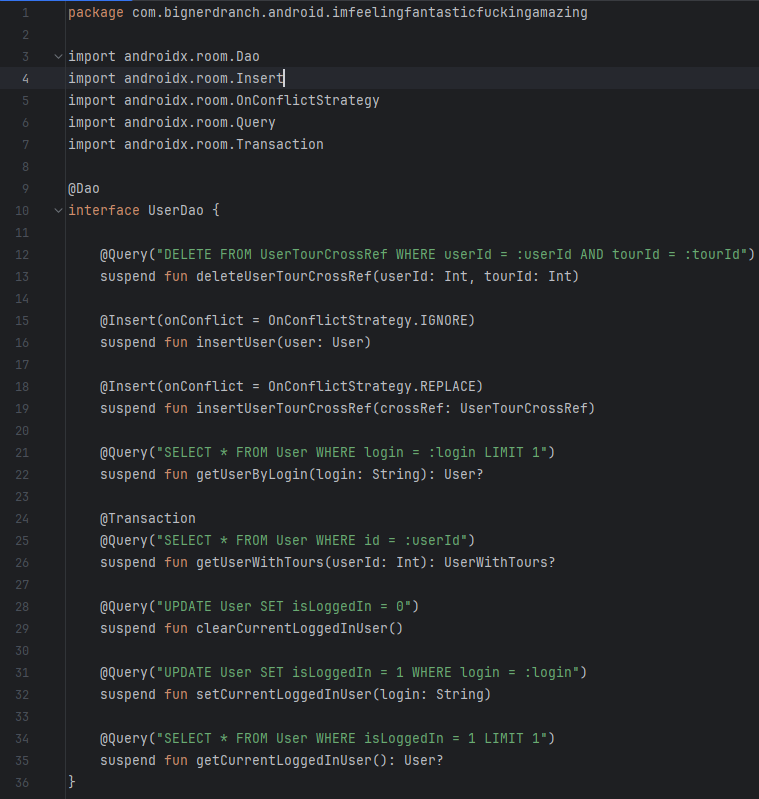


Рис.

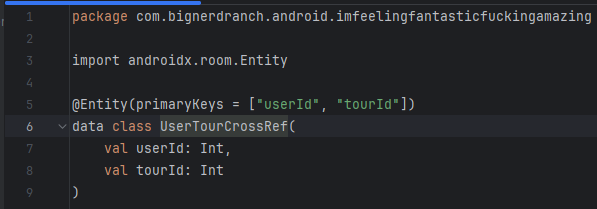


Рис.

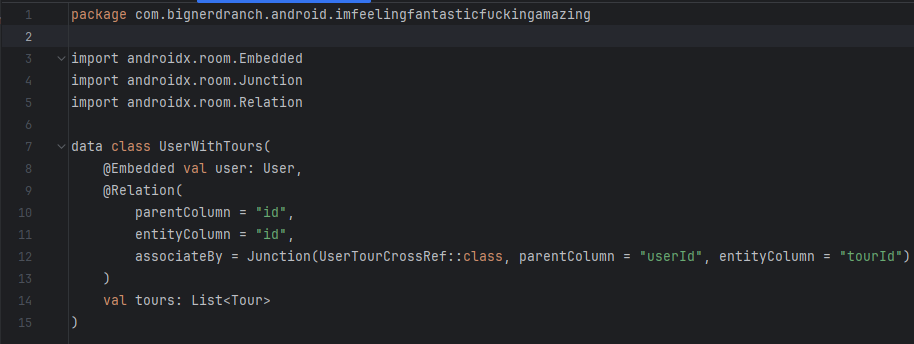


Рис.