**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](#_Toc197807136)

[ГЛАВА 1. АНАЛИЗ СОСТОЯНИЯ ОБЪЕКТА ИССЛЕДОВАНИЯ 2](#_Toc197807137)

[1.1. Описание предметной области мобильного приложения 4](#_Toc197807138)

[1.2. Сравнительный анализ существующих аналогов 5](#_Toc197807139)

[1.3. Обоснование выбора инструментальных средств для разработки мобильного приложения 7](#_Toc197807140)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ 10](#_Toc197807141)

[2.1. Определение функциональных и нефункциональных требований 10](#_Toc197807142)

[2.2. Проектирование пользовательского интерфейса 13](#_Toc197807143)

[2.3. Логическая модель приложения 21](#_Toc197807144)

[2.3.1 Проектирование мобильного приложения 21](#_Toc197807145)

[2.3.2 Проектирование базы данных 28](#_Toc197807146)

[ГЛАВА 3. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 33](#_Toc197807147)

[3.1. Разработка клиентской части мобильного приложения 33](#_Toc197807148)

[3.2. Разработка серверной части мобильного приложения 34](#_Toc197807149)

[3.2.1 Описание основной логики сервера 34](#_Toc197807150)

[3.2.2 Подключение базы данных к серверу 35](#_Toc197807151)

[3.2.3 Подключение сервера к клиентской части мобильного приложения 36](#_Toc197807152)

[3.3. Тестирование разработанного приложения 37](#_Toc197807153)

[ЗАКЛЮЧЕНИЕ 39](#_Toc197807154)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 41](#_Toc197807155)

[ПРИЛОЖЕНИЯ 43](#_Toc197807156)

# **ВВЕДЕНИЕ**

Современные геймеры всё чаще сталкиваются с проблемой управления своей игровой коллекцией и планов по прохождению игр. Большое количество релизов, скидок и подписочных сервисов затрудняет отслеживание того, что уже пройдено, а что ещё только планируется. Бумажные списки и заметки не справляются с этой задачей, а существующие решения часто перегружены или не учитывают индивидуальные потребности игроков. Приложение B4cklog призвано решить эти проблемы, предоставляя удобный и современный инструмент для управления игровым бэклогом.

Актуальность. С ростом популярности цифровых платформ и стремительным увеличением количества игр у пользователей возникает потребность в структурированном подходе к учёту своих игровых активностей. Разработка открытой мобильной системы управления игровым бэклогом поможет геймерам систематизировать свою коллекцию, отслеживать прогресс, находить интересные игры и делиться отзывами с другими.

Цель работы — разработать и протестировать клиент-серверное мобильное приложение игрового бэклога.

Задачи:

1. Описание предметной области мобильного приложения;
2. Сравнительный анализ существующих аналогов мобильного приложения;
3. Обоснование выбора инструментальных средств разработки мобильного приложения;
4. Определение функциональных и нефункциональных требований;
5. Проектирование пользовательского интерфейса мобильного приложения;
6. Логическая модель мобильного приложения (диаграмма в нотации IDEF0, диаграмма классов);
7. Проектирование базы данных мобильного приложения;
8. Разработка клиентской части мобильного приложения;
9. Разработка серверной части мобильного приложения;
10. Тестирование разработанного мобильного приложения.

Объект исследования — процессы учёта и организации информации об играх, интересующих пользователя.

Предмет исследования — клиент-серверное мобильное приложение игрового бэклога.

# **ГЛАВА 1. АНАЛИЗ СОСТОЯНИЯ ОБЪЕКТА ИССЛЕДОВАНИЯ**

## **Описание предметной области мобильного приложения**

Мобильное приложение игровой бэклог (B4cklog) предназначено для систематизации и управления личной коллекцией видеоигр пользователя. Его цель — предоставить удобный инструмент для отслеживания уже пройденных, текущих и запланированных к прохождению игр, а также фиксации впечатлений и прогресса. Предметная область охватывает индивидуальное взаимодействие пользователя с его цифровым игровым опытом, объединяя элементы личной статистики, каталогизации и планирования.

Ранее пользователи часто прибегали к спонтанным методам — вели списки в заметках, Excel-таблицах или просто держали информацию в памяти, что приводило к потере данных, дублированию игр и отсутствию целостной картины игрового бэклога. B4cklog устраняет эти неудобства: каждая игра добавляется в один или несколько списков (например, «Хочу пройти», «Прохожу», «Пройдено»), при этом доступна сопутствующая информация — название, обложка, платформа, статус, дата добавления, а в будущем и пользовательская оценка.

Приложение синхронизируется с сервером, что обеспечивает надёжное хранение данных и возможность доступа к ним с разных устройств. Пользователь всегда может увидеть актуальный список своих игр и быстро отфильтровать их по статусу или платформе. Это особенно удобно в условиях, когда количество игр в библиотеке достигает десятков и сотен.

Не менее важной задачей является поддержка мотивации: визуальное представление прогресса, разделение игр по категориям и текущая активность помогают пользователю не теряться в огромном потоке цифрового контента, а последовательно и осознанно подходить к выбору следующей игры. Кроме того, B4cklog может служить основой для аналитики игрового опыта — пользователь может видеть, сколько игр пройдено за определённый период, на каких платформах он играет чаще всего и какие жанры доминируют в его списках.

В целом, предметная область включает три основных аспекта: учёт и классификация игр, обеспечение доступа к данным в любом месте и в любое время и поддержка пользовательской мотивации через визуальный прогресс и порядок. Это позволяет перевести хаотичный подход к игровому бэклогу на новый уровень, повысить вовлечённость и получить удовольствие не только от самих игр, но и от процесса их отслеживания..

## **1.2.** **Сравнительный анализ существующих аналогов**

Сравнительный анализ служит для всесторонней оценки существующих решений с точки зрения их функциональных возможностей, технической архитектуры, пользовательского опыта и экономических факторов. Он позволяет:

* Выявить сильные и слабые стороны каждого продукта;
* Определить ключевые требования, которым должно соответствовать разрабатываемое приложение;
* Построить обоснованную аргументацию выбора архитектуры и набора технологий на основе объективных данных.

В рамках данного исследования мы отобрали три популярных сервиса для управления игровым бэклогом и оценки времени прохождения игр — Backloggd, HowLongToBeat и IGN Playlist — и сравнили их по критериям, отражающим основные потребности пользователей (описание игр, возможность писать отзывы, доступность API и т.д.) и условия использования (смотреть в таблице 1).

Backloggd — это веб-платформа, ориентированная на ведение списка игр с возможностью оценки, добавления заметок, рецензий и сортировки по различным параметрам. Интерфейс напоминает Letterboxd и ориентирован на сообщество игроков. Платформа активно развивается, но пока не имеет полноценного мобильного приложения. Пользователь может создавать списки вроде "Currently Playing", "Wishlist", "Beaten", "Shelved" и т. д. Доступ к API ограничен, и интеграция с другими сервисами отсутствует. Визуальная часть минималистична, но не всегда удобна с точки зрения UX на мобильных браузерах. Тем не менее, Backloggd отличается высокой социальной активностью и возможностью следить за прогрессом других пользователей.

HowLongToBeat (HLTB) — один из старейших и наиболее авторитетных сервисов для оценки продолжительности игр. Основной функционал сосредоточен вокруг статистики времени прохождения (Main Story, Main+Extra, Completionist), на основе пользовательских оценок. Платформа поддерживает базовое ведение списка пройденных игр и планов, однако не предоставляет гибкой системы тегов или пользовательских категорий. API не является официально открытым, но существует неофициальная документация. Мобильного приложения также нет. Сервис ценен именно как база данных длительности прохождения, но не как персональный трекер прогресса или визуального каталога.

IGN Playlist — относительно новый сервис от медиахолдинга IGN, позволяющий пользователям создавать свои игровые плейлисты и отслеживать прогресс. Присутствует мобильное приложение для актуальных платформ. Сервис встроен в общую экосистему IGN и предлагает тесную интеграцию с рецензиями, новостями и рейтингами. Интерфейс стильный и визуально привлекательный, однако система меток и списков ограничена, отсутствует возможность кастомизации и аналитики. Доступ к данным через API не предусмотрен. IGN Playlist скорее позиционируется как дополнение к редакционному контенту, чем как полноценный трекер.

*Таблица 1 – Сравнительный анализ существующих аналогов*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Backloggd | HowLongToBeat | IGN Playlist | Наш проект |
| Платформы | Web | Web | Web, Android, iOS | Android |
| Описание игр | ограниченное | да | да | да |
| Возможность писать отзывы | да | да | нет | да |
| Открытое REST-API | нет | нет | нет | есть (код открыт, Swagger в разработке) |
| Аутентификация | через соцсети | логин + пароль | через IGN аккаунт | логин + пароль |
| CRUD-операции с играми | только просмотр | только просмотр | только просмотр | Полный набор (создание, чтение, обновление, удаление) |
| Технологический стек | React, Node.js | PHP, JavaScript | React, Node.js | Kotlin + Retrofit, Spring Boot 3 + MySQL |

Все рассмотренные сервисы предоставляют удобные веб-интерфейсы, но не имеют полноценных мобильных приложений и открытого API, что ограничивает возможности интеграции и кастомизации. Наш проект B4cklog выделяется открытым исходным кодом, REST API и ориентирован на мобильные устройства, что позволит пользователям более гибко управлять своим игровым бэклогом и расширять функционал по мере необходимости.

## **1.3.** **Обоснование выбора инструментальных средств для разработки мобильного приложения**

Для клиентской части мобильного приложение были выбраны следующие технологии:

* Kotlin Android Plugin и Android Application Plugin (Gradle KTS). Kotlin выбран за счёт лаконичности синтаксиса, полной совместимости с существующими Java-библиотеками и встроенной поддержки сопрограмм (coroutines), что упрощает асинхронную обработку сетевых и локальных операций. Плагин Gradle KTS даёт типобезопасную конфигурацию сборки и централизованное управление версиями зависимостей.
* ViewBinding. Позволяет безопасно работать с View-элементами без явного вызова findViewById, сокращая шаблонный код и исключая ошибки NullPointerException при обращении к макету.
* ConstraintLayout. Обеспечивает гибкую, но при этом эффективную компоновку экранов, позволяя создавать адаптивные интерфейсы с минимальной глубиной иерархии View для сохранения производительности на устройствах с различной плотностью пикселей.
* AndroidX Core/AppCompat и Material Components v1.11.0. AndroidX Core и AppCompat гарантируют обратную совместимость с устаревшими версиями Android, а Material Components реализуют рекомендации Material Design 3, что позволяет придерживаться современных UX-паттернов и обеспечивает единообразный вид приложения.
* Navigation-Fragment KTX и Navigation-UI KTX. Дают декларативный способ описывать граф навигации в единственной Activity, упрощают передачу аргументов между экранами и автоматически обрабатывают back-стек, что снижает количество «костылей» при реализации переходов.
* Retrofit + OkHttp + Gson. Retrofit — проверенный временем HTTP-клиент с поддержкой аннотаций, а OkHttp обеспечивает надёжное соединение, кеширование и перехват сетевого трафика для логирования. Gson Converter дат быстрые и гибкие средства парсинга JSON и простого текста, что нужно при взаимодействии с разнородным REST-API.
* Kotlinx Serialization (json). Предоставляет нативную сериализацию/десериализацию Kotlin-классов без рефлексии, демонстрирует более высокую производительность по сравнению с Gson и легко расширяется под мультимодульные проекты.
* AndroidX Lifecycle ViewModel KTX + LiveData KTX. Отделяют бизнес-логику от UI, сохраняют состояние при пересоздании Activity/Fragment и упрощают тестирование благодаря чёткому слою управления данными.
* Glide. Оптимизированная библиотека для загрузки, кэширования и отображения изображений, автоматически управляет памятью и поддерживает различные форматы, включая WebP и GIF.

Для серверной части мобильного приложения были выбраны следующие технологии:

* Spring Boot Starter Web, Data JPA и Security. Spring Boot значительно снижает объём конфигурационного кода: встроенные стартеры дают сразу готовый REST-контроллер, слой доступа к данным на JPA/Hibernate и настраиваемую stateless-безопасность через Spring Security.
* Java 21. Последняя LTS-версия JVM с новыми языковыми возможностями (records, sealed classes) и улучшениями производительности, гарантирующая долгосрочную поддержку.
* MySQL Connector/J 8.0.32. Широко распространённый и хорошо документированный драйвер для MySQL, который многие образовательные учреждения уже используют в продакшене.
* Jakarta Servlet API 5.0. Современная спецификация сервлетов, необходимая для корректной работы компонентов веб-контейнера с последними версиями сервлет-движков.
* JUnit 5. Современный фреймворк для модульного и интеграционного тестирования с расширяемой архитектурой и поддержкой параметризованных тестов, упрощающий обеспечение высокого покрытия кода.

# **ГЛАВА 2. ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

## **2.1. Определение функциональных и нефункциональных требований (описание функциональных требований, use case диаграмма, описание нефункциональных требований)**

## Функциональные требования формируют набор конкретных возможностей, которые система должна предоставлять пользователям для выполнения их задач:

* **Регистрация пользователя.** Система должна позволять пользователю (игроку или администратору) создать аккаунт, указав имя, email и пароль.
* **Аутентификация по токену.** После входа пользователь получает токен, связанный с его аккаунтом. Токен сохраняется в отдельной таблице Token и не имеет срока действия. Все защищённые операции требуют валидного токена в заголовке Authorization: Bearer ….
* Ролевое разграничение доступа. Система должна поддерживать как минимум две роли: Пользователь и Администратор. При обращении к защищённым контроллерам:
  + Администратор имеет доступ к управлению данными об играх и платформах.
  + Пользователь может управлять своим бэклогом, просматривать данные о играх и редактировать профиль.
* Управление играми и платформами. Администратор может создавать, изменять и удалять игры и платформы. При этом должны проверяться зависимости: удаление платформы невозможно, если она связана с играми.
* Управление списками бэклога. Пользователь может добавлять игры в один из нескольких списков: «Хочу поиграть», «Играю», «Прошёл», и т.п., а также удалять игры из этих списков. Возможна принадлежность игры к нескольким спискам.
* Просмотр информации об играх. Все пользователи могут просматривать полную информацию об игре, включая обложку, дату выхода, описание и список платформ.
* Редактирование профиля. Пользователь может просматривать и редактировать свои данные (имя, email, аватар).
* Добавление и удаление обложек. При загрузке новой обложки (аватара или обложки игры), предыдущий файл удаляется, а новый становится доступным по публичному URL.

Возможности пользователя::

* Регистрация — создание аккаунта с логином, email и паролем;
* Вход в систему — получение доступа с выдачей токена;
* Просмотр профиля — отображение личной информации, аватара и списков;
* Загрузка фото — обновление аватара, удаление старого изображения;
* Добавление игры в список — выбор одного из доступных списков бэклога;
* Удаление игры из списка — возможность исключить игру;
* Просмотр списков — отображение игр по каждому статусу;
* Поиск и просмотр игр — поиск по названию, переход к деталям.

Возможности администратора:

* Выполнение всех действий пользователя;
* Добавление игры — создание новой записи об игре, указание платформ;
* Редактирование игры — изменение данных существующей игры;
* Удаление игры — возможность удалить игру, если она не прикреплена к пользовательским спискам;
* Управление платформами — добавление и удаление платформ.

Описанные выше возможности каждого типа пользователей отображены на USE CASE диаграмме на рисунке 1.

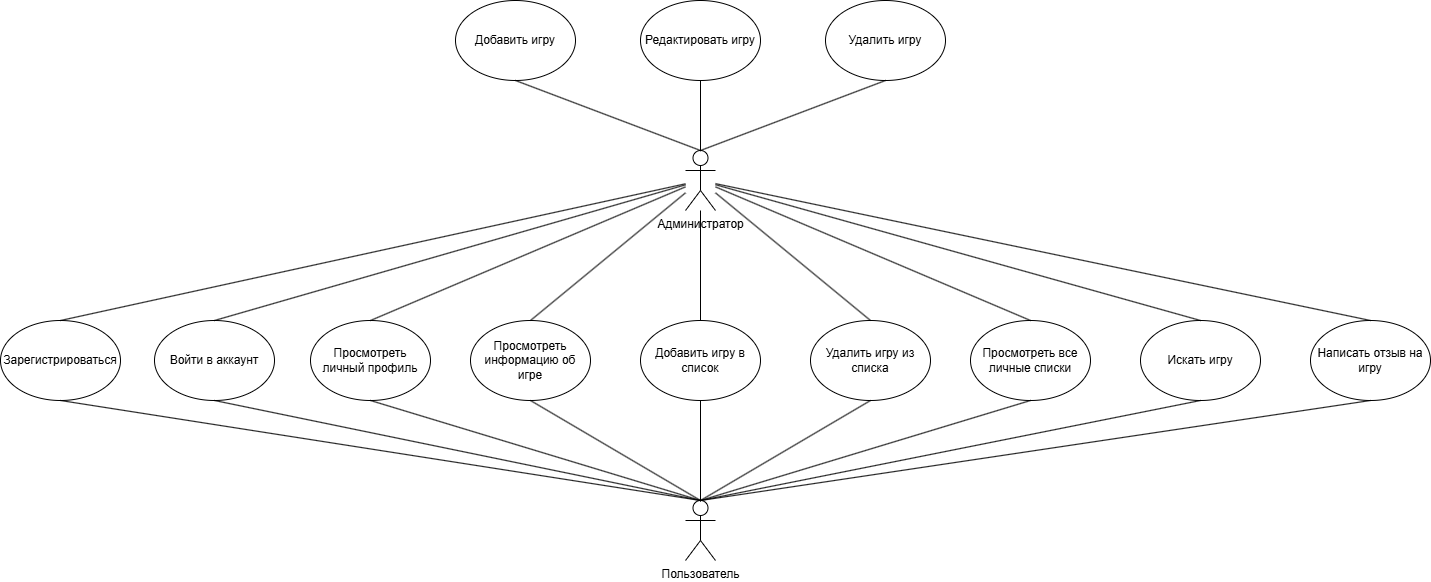


Рисунок 1. Диаграмма USE CASE

Нефункциональные требования задают критерии качества, безопасности, производительности, надёжности, совместимости, эффективности, удобства использования и масштабируемости системы:

* Производительность. Время ответа сервера при нагрузке 100 одновременных пользователей не должно превышать 200 мс в среднем, 95-й персентиль — до 300 мс.
* Надёжность**.** Время безотказной работы (MTBF) должно составлять не менее 1000 часов; время восстановления (MTTR) не более 10 минут.
* Безопасность. Все соединения с сервером должны использовать TLS 1.3; пароли хранятся как BCrypt-хэши; Токен реализован как объект Token, связанный с пользователем; его уникальность обеспечивается на уровне БД.
* Совместимость. Сервер собирается в amazoncorretto:21-alpine, запускается в контейнере на Linux без необходимости изменений. CORS разрешает методы GET, POST, DELETE, PUT с любых источников.
* Масштабируемость. Архитектура stateless позволяет горизонтальное масштабирование серверных экземпляров при общей MySQL-БД.
* Поддерживаемость. Покрытие unit-тестами ≥ 75 % строк (Jacoco); все конфигурации (CORS, uploadDir) настраиваются в application.yml.

## **2.2. Проектирование пользовательского интерфейса**

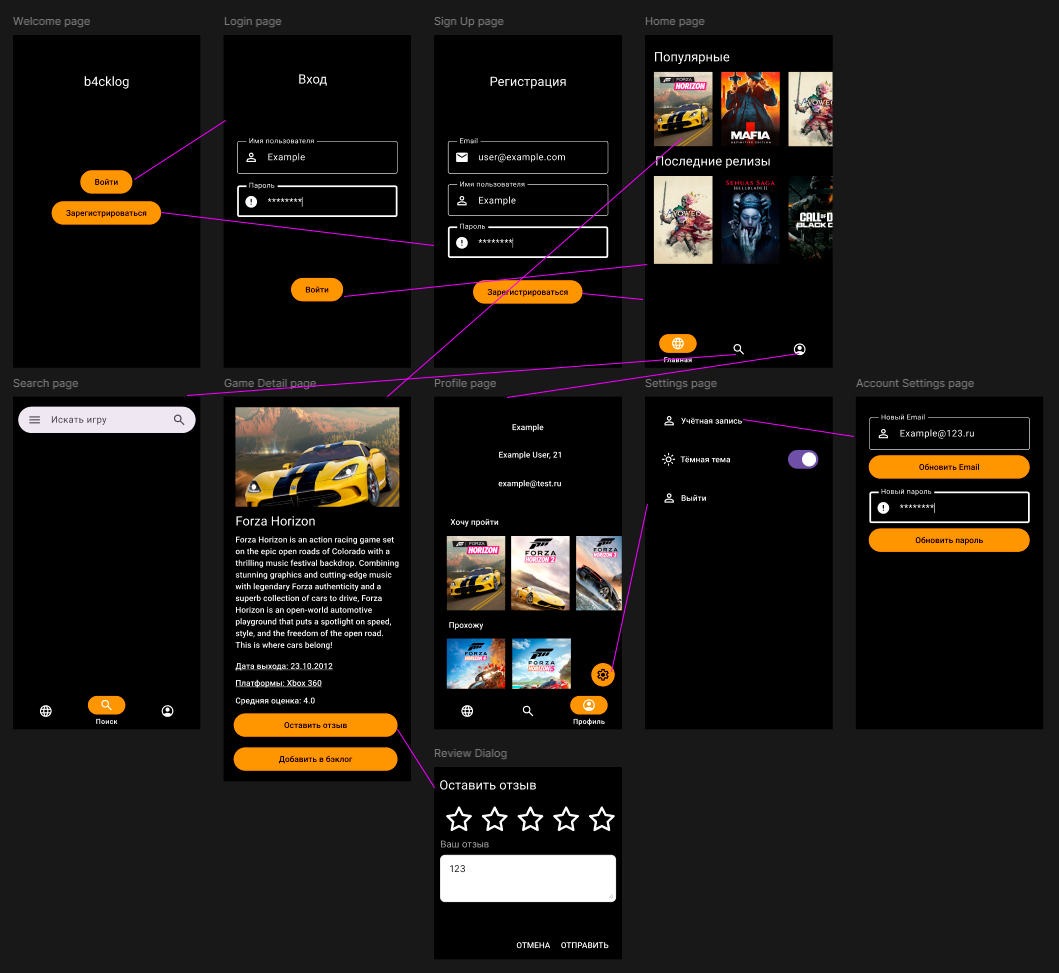


Рисунок 2.1. Межэкранное взаимодействие в Figma

При первом запуске приложения пользователю отображается WelcomeActivity (лист. 38) — экран приветствия, где предлагается выбрать между входом в аккаунт и регистрацией. Однако до отображения интерфейса система выполняет несколько проверок и инициализаций.

Сначала из SharedPreferences читается флаг dark\_mode, на основе которого приложение применяет соответствующую тему (светлую или тёмную) через AppCompatDelegate. Далее WelcomeActivity обращается к AuthPrefs.getToken() — если сохранённый токен авторизации присутствует и не является пустым, пользователь мгновенно перенаправляется в MainActivity, минуя экран приветствия и логина. Это обеспечивает плавный пользовательский опыт для уже авторизованных пользователей.

Если токена нет, пользователь остаётся на приветственном экране (activity\_welcome.xml), где доступны две кнопки: Login и Sign Up. Нажатие на соответствующую кнопку запускает LoginActivity или SignUpActivity через Intent, позволяя пользователю пройти авторизацию или зарегистрироваться.

Если пользователь выбирает вход в систему на экране приветствия, запускается LoginActivity (лист. 35) — экран авторизации. При инициализации активити отображается форма с двумя полями ввода: логином и паролем, а также кнопкой входа.

После нажатия кнопки происходит проверка: если оба поля заполнены, вызывается метод loginUser, отправляющий данные на сервер через ApiClient.authApi.login(...). Запрос выполняется асинхронно с использованием Retrofit.

Если ответ успешен (код 200) и в теле содержится токен, приложение:

1. Сохраняет токен в SharedPreferences через AuthPrefs.saveToken.
2. Показывает уведомление «Успешный вход!».
3. Переходит в MainActivity, которая, в зависимости от роли пользователя, открывает нужный корневой фрагмент.

Если сервер вернул ошибку (например, неверные данные), показывается уведомление «Ошибка входа». В случае сетевой ошибки отображается сообщение с причиной сбоя.

Если пользователь выбирает регистрацию на стартовом экране приложения, запускается SignUpActivity (лист. 37) — экран создания нового аккаунта. При инициализации активити отображается форма, включающая поля для ввода логина, пароля, электронной почты, имени, фамилии и возраста.

После нажатия кнопки регистрации происходит проверка: если все обязательные поля (кроме возраста) заполнены, вызывается метод registerUser. В нём формируется RegisterRequest, отправляемый на сервер с помощью ApiClient.authApi.register(...). Запрос обрабатывается асинхронно с использованием Retrofit.

Если сервер успешно возвращает ответ (код 200) и в теле содержится токен:

1. Токен сохраняется через AuthPrefs.saveToken.
2. Пользователю показывается уведомление «Регистрация прошла успешно».
3. Приложение переходит в MainActivity, где, как и после авторизации, определяется роль пользователя и открывается соответствующий корневой фрагмент.

Если регистрация не удалась (например, пользователь с таким логином уже существует), показывается сообщение «Ошибка регистрации». В случае сетевой ошибки пользователь видит сообщение с причиной сбоя, а в лог записывается текст ошибки.

После успешной авторизации или регистрации, приложение запускает **MainActivity** — основной контейнер для пользовательского интерфейса. При запуске активити из SharedPreferences считывается предпочтение пользователя по тёмной теме, и на его основе устанавливается соответствующий режим отображения (AppCompatDelegate.MODE\_NIGHT\_YES или MODE\_NIGHT\_NO).

Затем инициализируется NavHostFragment, связанный с NavController, который управляет навигацией между основными фрагментами приложения. Нижняя панель навигации (BottomNavigationView) привязывается к NavController через setupWithNavController, обеспечивая корректную навигацию между экранами.

Параллельно с этим отправляется асинхронный запрос к API профиля пользователя (getUserProfile()), чтобы получить текущие данные. При успешном ответе сохраняется userId через SessionManager — это позволяет остальным частям приложения (например, экранам профиля или списка игр) обращаться к ID текущего пользователя без повторного запроса.

В главной зоне пользователя основным контейнером служит MainActivity, использующая NavController для переключения между тремя основными фрагментами: HomeFragment, SearchFragment и ProfileFragment. Каждый из этих экранов отвечает за отображение различной информации и использует свой подход к работе с данными и навигацией.

Главная (HomeFragment):

Основной экран, открывающийся после входа пользователя, — это HomeFragment, отображающий подборки популярных и новых игр. Внутри onCreateView инициализируются два RecyclerView, прокручивающихся горизонтально, каждый со своим адаптером (GameAdapter). Загрузка данных происходит асинхронно через Retrofit (ApiClient.gameApi.getAllGames()), без отображения лоадеров, с простой обработкой ошибок через Toast. Нажатие на игру вызывает переход на GameDetailFragment через Navigation.findNavController(...).navigate(...), где в Bundle передаётся gameId. Переход осуществляется стандартным способом NavController без использования back-stack-манипуляций вручную: кнопка "назад" приводит обратно на домашний экран.

Экран профиля (ProfileFragment) отображает основные данные пользователя (имя, фамилия, возраст, email), а также пять списков игр в формате горизонтальных RecyclerView: Хочу поиграть, Играю, Играл, Закончил, Пройдено на 100%. Каждому списку соответствует отдельный GameAdapter. Данные загружаются из API (ApiClient.profileApi.getUserProfile()) в методе loadProfile(). При успешном ответе данные пользователя и списки игр отображаются на экране. В случае ошибки отображается Toast. Также на экране присутствует FloatingActionButton, по нажатию на который открывается SettingsFragment — переход осуществляется через NavController. Нажатие на любую игру из списков также ведёт на GameDetailFragment с передачей gameId, как и на главной.

Экран поиска (SearchFragment) реализован на основе SearchView, RecyclerView и двух адаптеров: GameAdapter для результатов и SearchHistoryAdapter для истории запросов. История хранится в SharedPreferences и автоматически отображается при фокусе на поле ввода. Поиск выполняется с задержкой (debounce) 2 секунды с помощью Handler и Runnable, чтобы снизить нагрузку при вводе. Если пользователь подтверждает ввод (onQueryTextSubmit), история сохраняется, и запрос отправляется на сервер через Retrofit. Навигация на GameDetailFragment происходит через NavController, с передачей gameId в Bundle. Важно, что при возврате с GameDetailFragment в SearchFragment состояние последнего восстанавливается — запрос и результаты остаются. Это реализовано через сохранение lastQuery в savedInstanceState.

Экран с подробной информацией об игре (GameDetailFragment) отображает название, описание, дату выхода, платформы, обложку, средний рейтинг и пользовательский отзыв на выбранную игру.

Данные загружаются из API:

* Детали игры через ApiClient.gameApi.getGame(gameId)
* Средний рейтинг через ApiClient.reviewApi.getAverageRating(gameId)
* Отзыв текущего пользователя через ApiClient.reviewApi.getUserReview(userId, gameId)
* Профиль пользователя через ApiClient.profileApi.getUserProfile() для определения, в каком из списков пользователя (Хочу поиграть, Играю, Играл, Прошёл, Пройдено на 100%) находится игра.

Если игра есть в одном из списков, показывается текст с названием списка и кнопка удаления игры из всех списков. При нажатии на кнопку удаления выполняется запрос removeGameFromAllLists(userId, gameId) и игра удаляется из списков. Кнопка "Добавить в список" открывает диалог выбора одного из списков, в который пользователь хочет добавить игру. После выбора вызывается запрос addGameToList(userId, gameId, listName) для добавления. Также есть кнопка "Написать отзыв", открывающая диалог для создания/редактирования отзыва. После сохранения отзыв и рейтинг обновляются на экране. В случае ошибок загрузки или отправки данных отображаются соответствующие Toast-сообщения. Навигация внутри фрагмента реализована через NavController, поддерживается перезагрузка экрана для обновления данных.

Экран настроек (SettingsFragment) позволяет пользователю изменить тему приложения и выйти из аккаунта. В интерфейсе присутствует переключатель темы (SwitchCompat), состояние которого сохраняется в SharedPreferences под ключом "dark\_mode". При переключении темы используется AppCompatDelegate.setDefaultNightMode с последующим перезапуском активности для применения изменений. Также есть кнопка перехода в настройки аккаунта, реализованная через Navigation Component. Кнопка "Выйти" удаляет сохранённый токен аутентификации (через AuthPrefs.clearToken) и запускает WelcomeActivity с очисткой стека задач, обеспечивая выход из аккаунта.

Экран настроек аккаунта (AccountSettingsFragment) содержит два поля для редактирования — email и пароль — и две кнопки для их обновления. Поля и кнопки связаны с элементами разметки fragment\_account\_settings.xml. При создании экрана в методе onCreateView() происходит загрузка текущего email пользователя через API (ApiClient.profileApi.getUserProfile()) и установка его в поле email. В случае ошибки загрузки показывается Toast. Для обновления email и пароля используются кнопки updateEmailButton и updatePasswordButton. При нажатии происходит проверка, что поле не пустое, и вызывается соответствующий метод updateEmail(newEmail) или updatePassword(newPassword). Оба метода выполняют сетевой запрос через lifecycleScope.launch и корутины, вызывая методы API для обновления данных. При успешном обновлении показывается Toast с подтверждением, при ошибке — сообщение с кодом ошибки или описание сетевой ошибки (рис. 2.2)

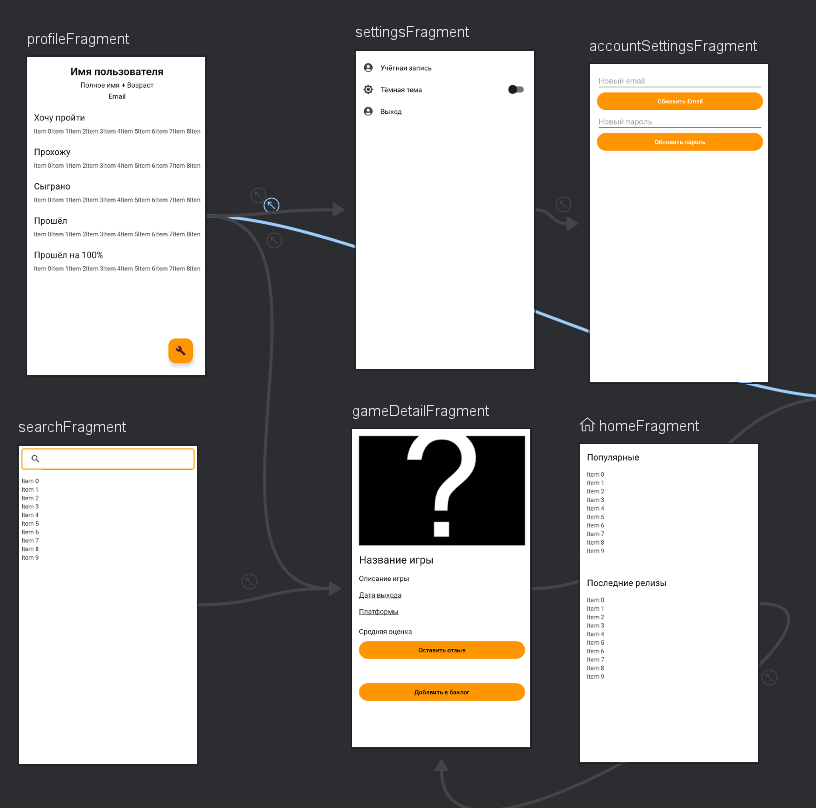


Рисунок 2.2. Экраны пользователя

Администратор может управлять всеми теми же экранами, что и пользователь, но дополнительно в его зоне находится только один фрагмент – EditGameFragment, который позволяет редактировать или добавлять информацию об игре. На него происходит переход либо с фрагмента GameDetailFragment, либо с фрагмента ProfileFragment. На фрагменте указаны поля с описанием игры, которые могут быть заранее заполнены, если администратор редактирует игру вместо её добавления. Кнопка «Сохранить» отправляет запрос на сервер с сохранением информаици об игре. В случае ошибки показывается Toast (рис 2.3)

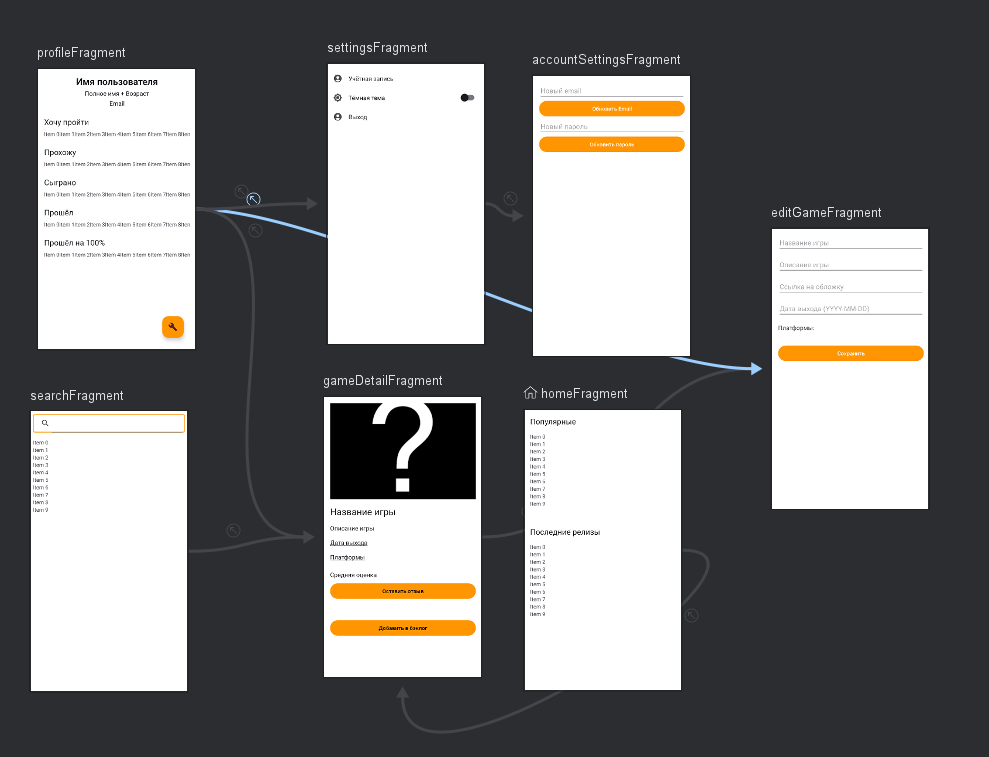


Рисунок 2.3. Экраны администратора

Навигация в приложении построена на двух уровнях: глобальная — через NavHostFragment и статический nav\_graph, обеспечивающий переходы между основными разделами (домашняя страница, поиск, профиль, настройки и подробности игры), и локальная — внутри отдельных фрагментов, например, внутри профиля. Администратор имеет доступ ко всем экранам, включая EditGameFragment для редактирования игр, в то время как обычные пользователи могут просматривать и взаимодействовать со всеми фрагментами, кроме редактирования. Такое разделение навигации позволяет сохранить изолированные back-stack’и, предсказуемое поведение кнопки "назад" и избегать лишних перезапросов или потери состояния при возвратах между разделами.

Таким образом мы полностью выполнили задание по созданию макетов экранов приложения, описали каждый из экранов и указали у каких типов пользователей какие экраны имеются.

## **2.3. Логическая модель приложения**

Логическая модель базы данных для нашего приложения была спроектирована с целью создания четкой и структурированной основы для управления учебными данными. Она позволяет определить ключевые сущности, такие как пользователи, игры, платформы, обзоры и токены, а также их взаимосвязи. Это помогает обеспечить целостность данных, исключить дублирование информации и повысить надёжность хранимых данных. Логическая структура, реализованная в базе, охватывает ключевые аспекты функциональности: добавление и редактирование игр, управление списками бэклога, выставление оценок и комментариев, а также безопасную авторизацию через токены. Логическую модель базы данных, которая разработана для нашего приложения приведена на рисунке 2.

## **2.3.1 Проектирование мобильного приложения**

Методология IDEF0 применяется для моделирования функциональной структуры системы. Она позволяет визуализировать ключевые процессы, определить их входы, выходы, механизмы и управляющие воздействия. С помощью диаграмм IDEF0 можно наглядно отразить, какие функции выполняет система игрового бэклога, как они взаимодействуют между собой и с внешними объектами.

Контекстная диаграмма описывает приложение как единый процесс, взаимодействующий со своими внешними сущностями (рис. 2.4) В качестве входов выступают «Данные пользователя», «Запросы на обновление списка игр», «Запросы на обновление пользовательских списков» и «Оценки и отзывы», а в качестве выходов — «Обновлённый список игр», «Обновлённые списки пользователя» и «Обновлённый список отзывов». Механизм работы задают мобильное приложение и сервер приложения, управление обеспечивается набором правил: правилами обработки запросов, валидации данных, работы с играми, работы с отзывами и политикой безопасности и конфиденциальности.

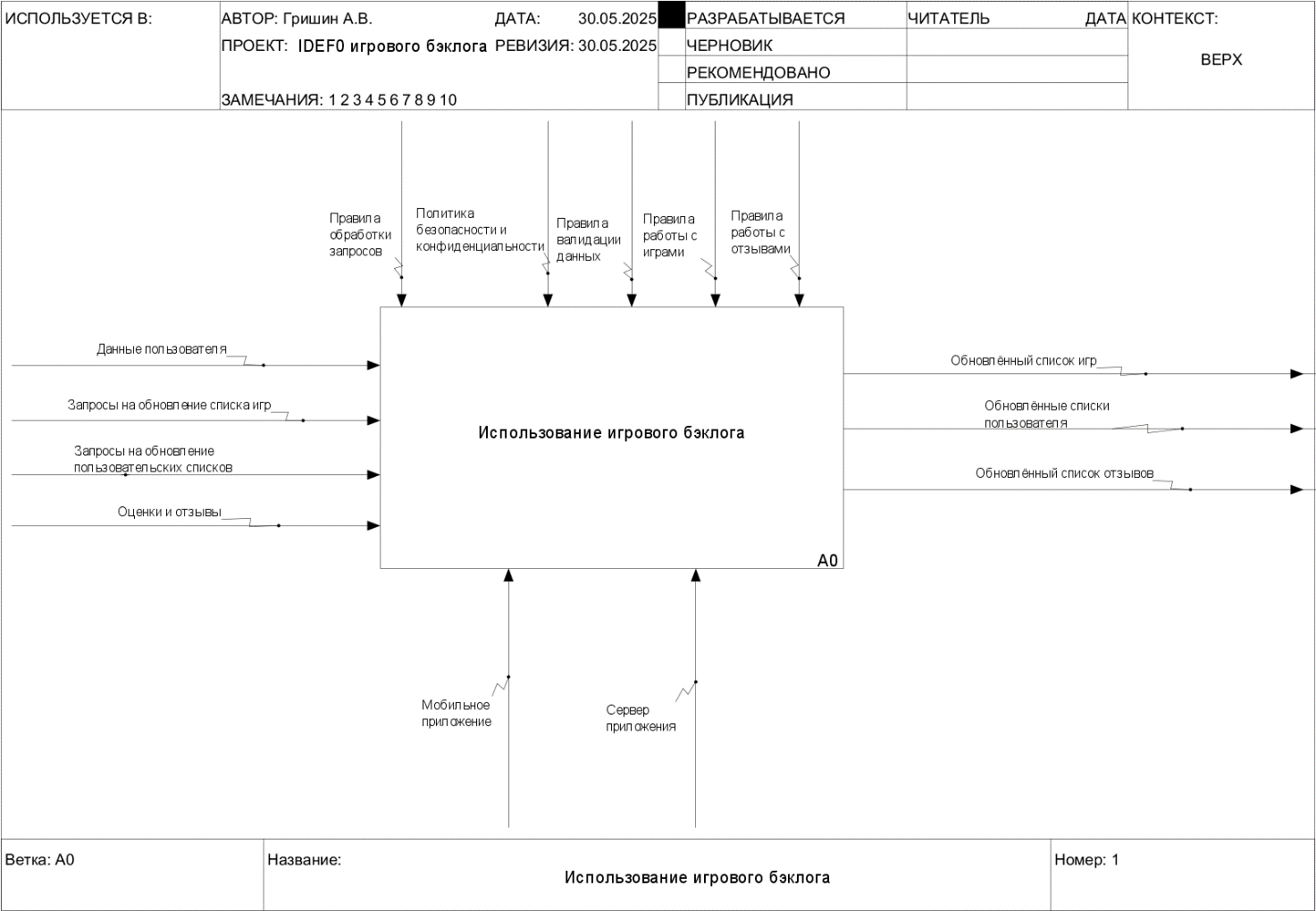


Рисунок 2.4. Контекстный уровень диаграммы IDEF0

Декомпозируем контекст на четыре ключевых подпроцесса: «Регистрация и авторизация», «Управление играми», «Работа с пользовательскими списками» и «Работа с отзывами» (рис. 2.5).

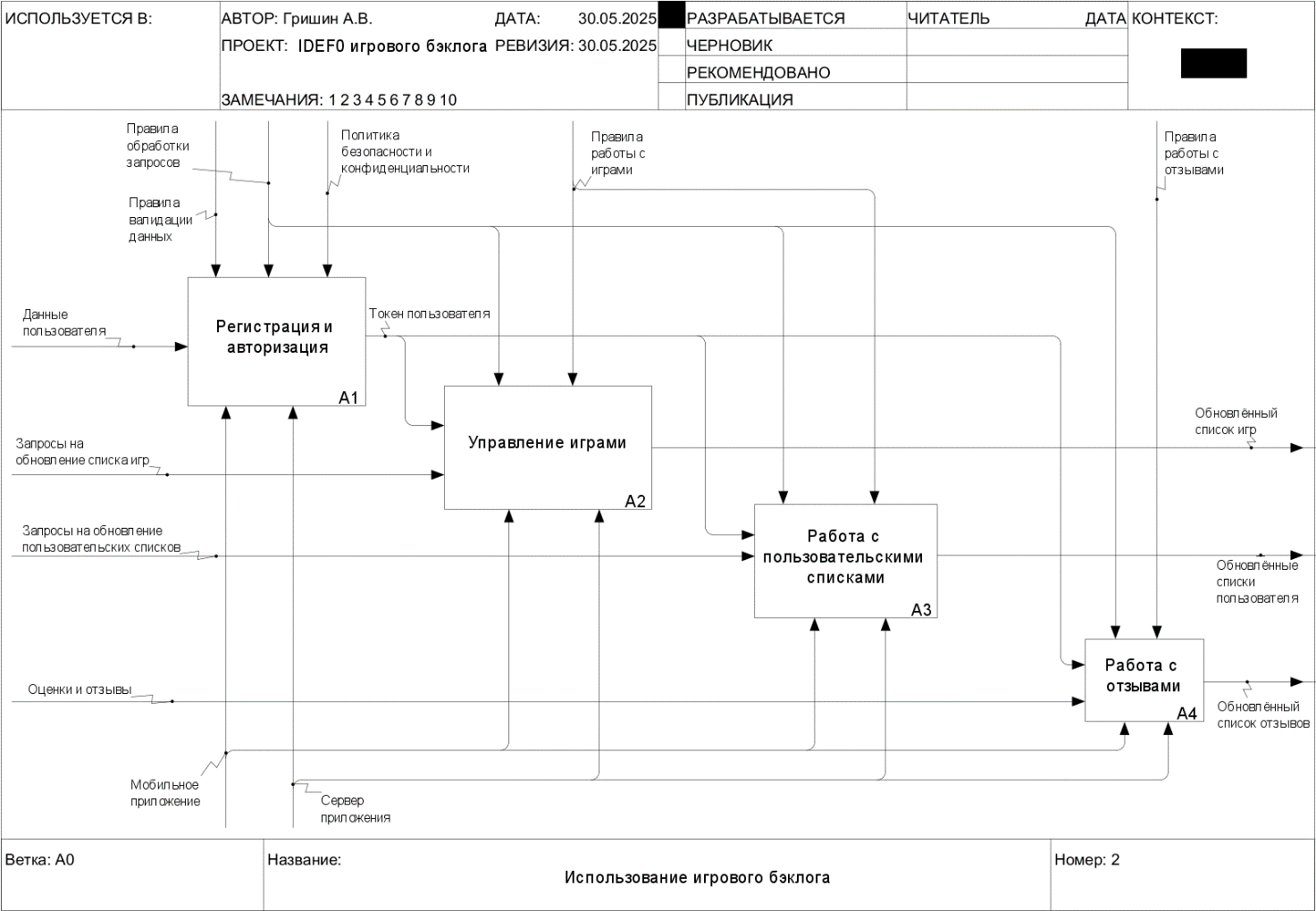


Рисунок 2.5. Декомпозиция контекстного уровня

На уровне A1 представлена декомпозиция блока «Регистрация и авторизация» (рис. 2.6). В него входят три подпроцесса: «Регистрация в приложении», который принимает входные данные и проверяет их на соответствие правилам; «Сохранение данных», обеспечивающее запись валидных записей в базу данных с учётом политики конфиденциальности; и «Авторизация в приложении», отвечающее за проверку учётных данных и выдачу токена.

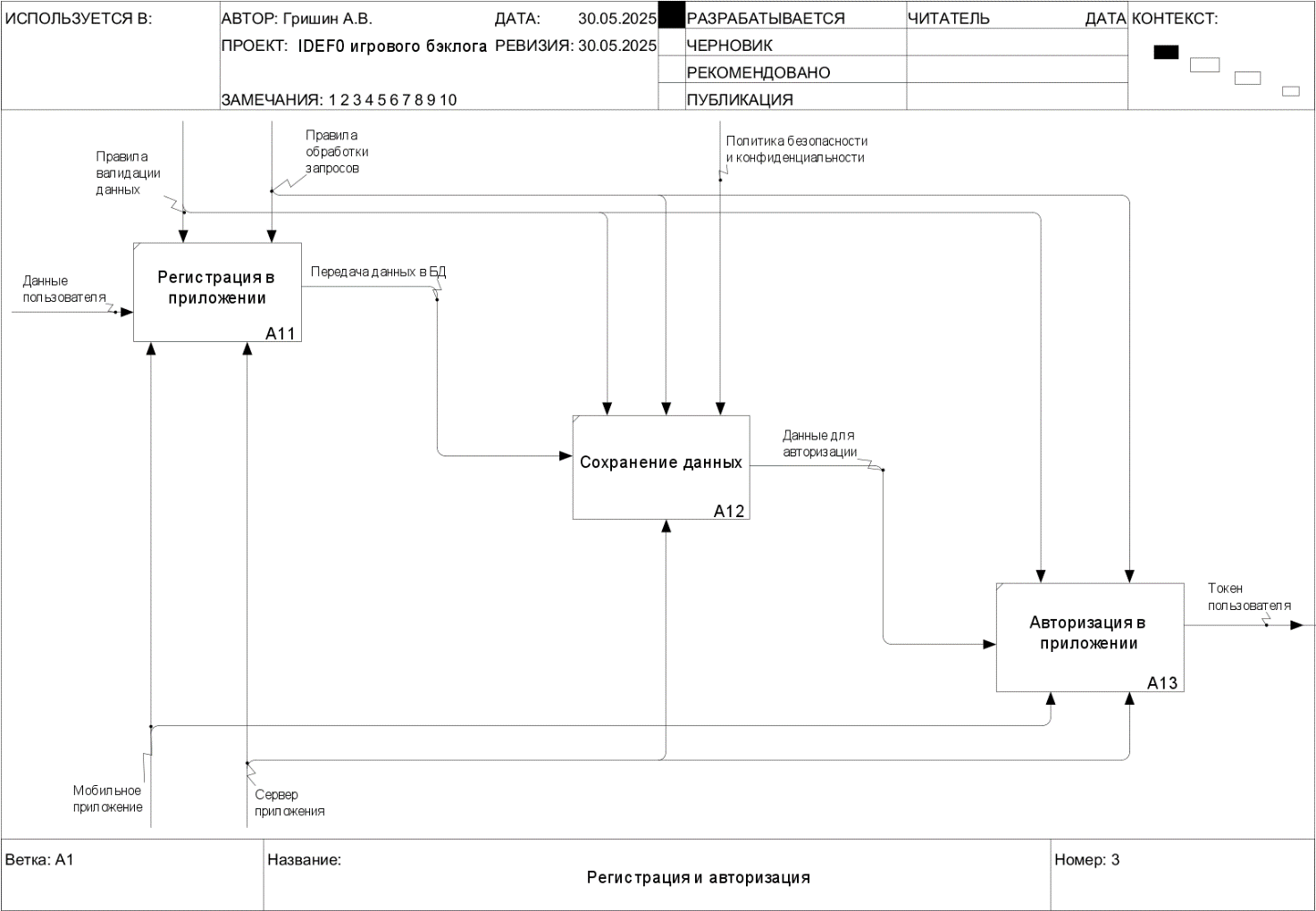


Рисунок 2.6. Декомпозиция процесса “Регистрация и авторизация”

На уровне A2 представлена декомпозиция блока «Управление играми» (рис. 2.7). Он включает три подпроцесса: «Добавление игр» для создания объектов игр; «Удаление игр» для удаления определённых игр; и «Сохранение данных» для фиксации всех изменений в хранилище БД. При каждом из действий используется токен пользователя, т.к. данные операции может выполнять только администратор.

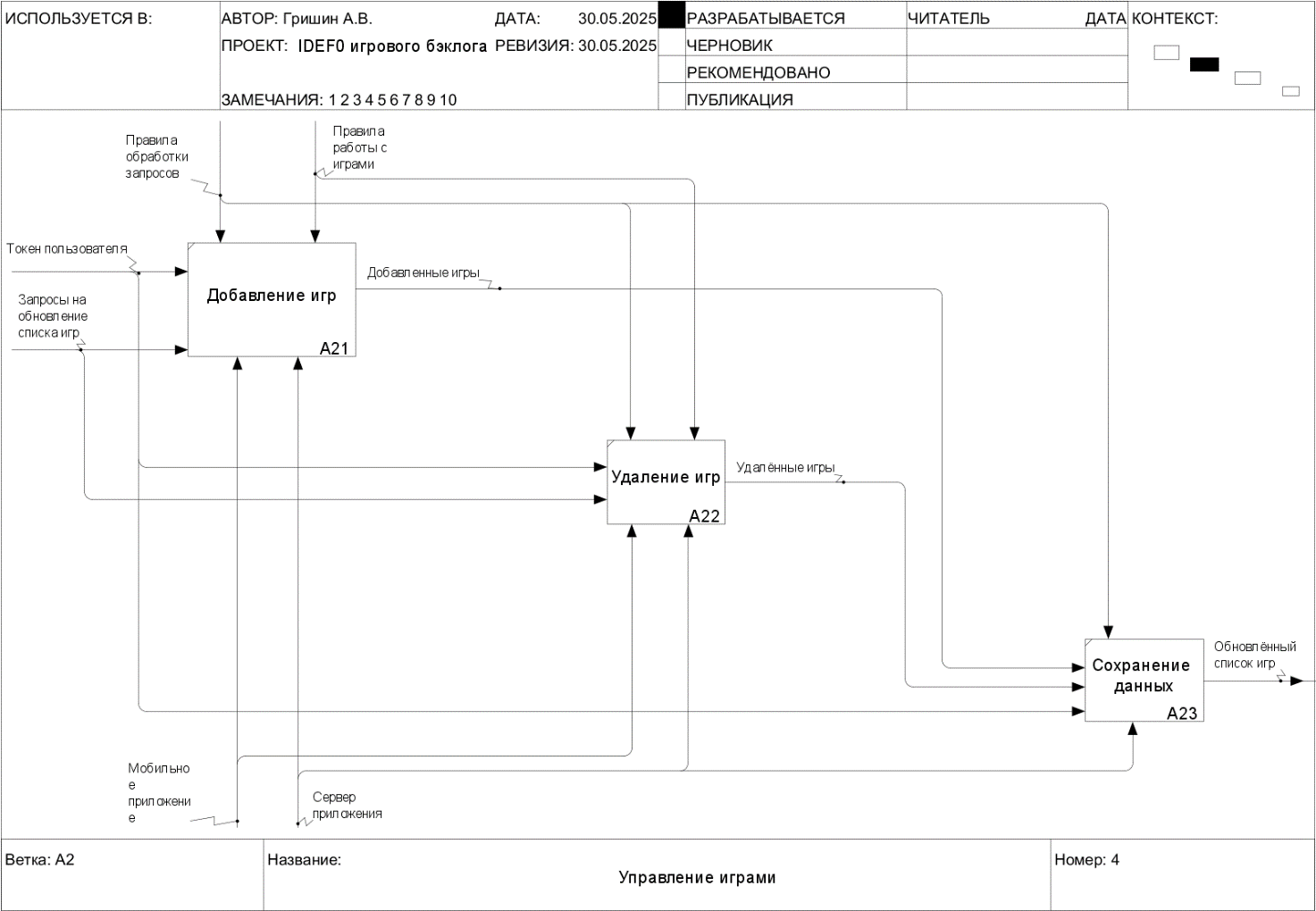


Рисунок 2.7. Декомпозиция процесса “Управление играми”

На уровне A3 представлена декомпозиция блока «Работа с пользовательскими списками» (рис. 2.8). Этот процесс разбит на три подпроцесса: «Получение профиля пользователя» (для получения его списков), «Добавление игры в список» (чтобы игра добавилась в один из списков пользователя), и «Удаление игры из всех списков» (чтобы игра была удалена из всех списков пользователя), где каждый шаг регулируется соответствующими правилами.

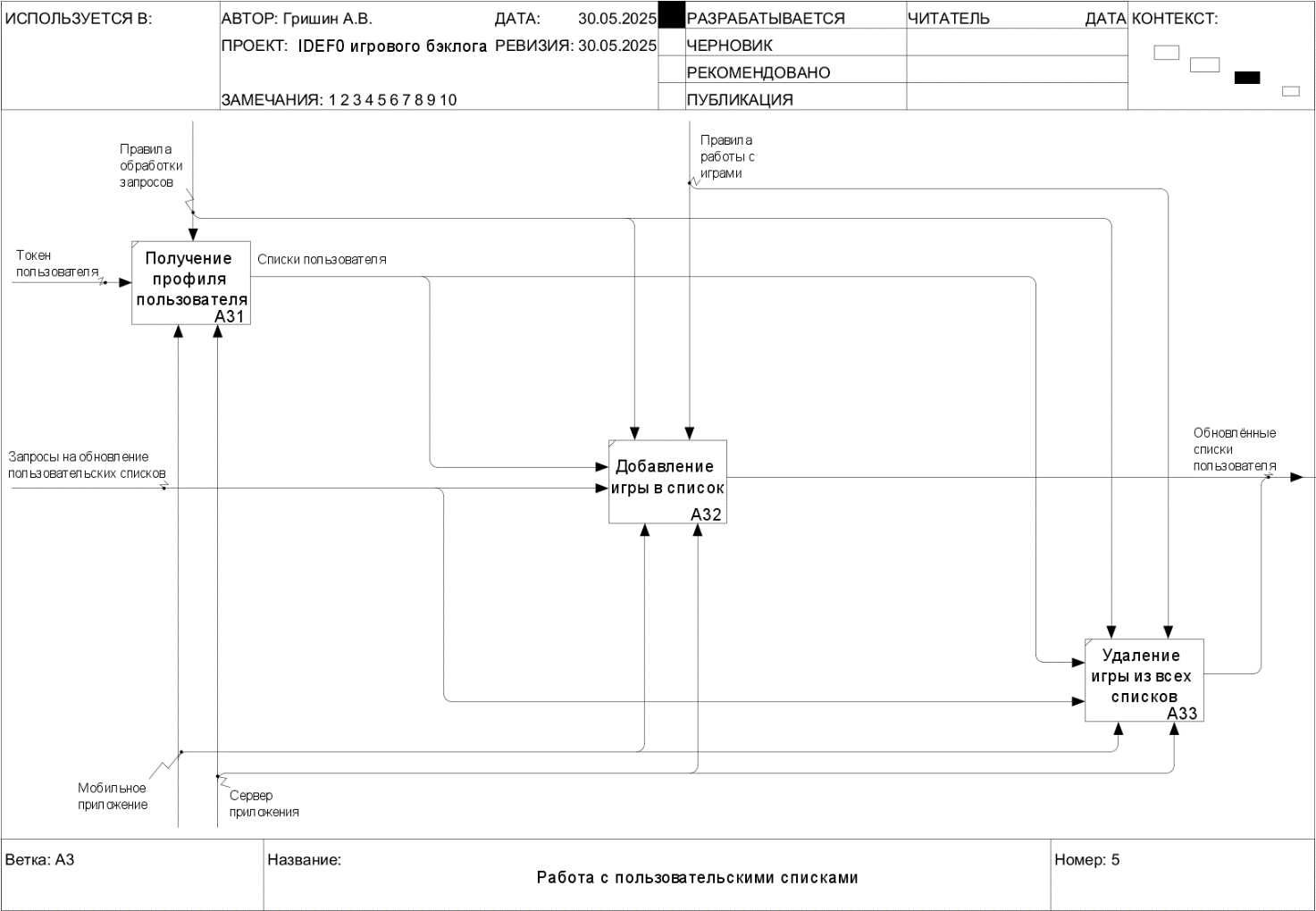


Рисунок 2.8. Декомпозиция процесса “Работа с пользовательскими списками”

На уровне A4 представлена декомпозиция блока «Работа с отзывами» (рис. 2.9). Он состоит из трёх подпроцессов: «Добавление отзыва», «Удаление отзыва», и «Сохранение данных» (в базу данных). На всех шагах используем токен пользователя.

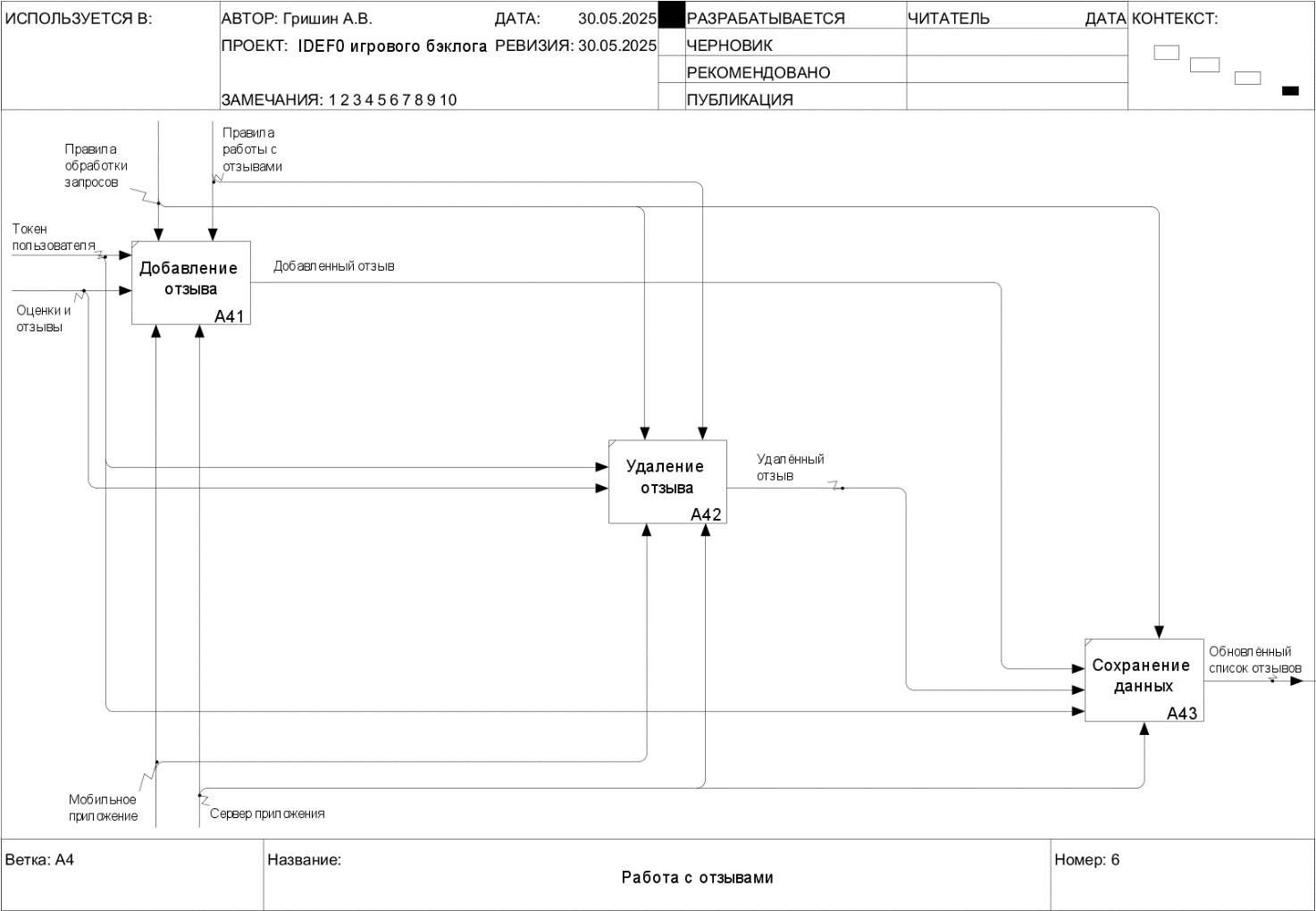


Рисунок 2.9. Декомпозиция процесса “Работа с отзывами”

Диаграмма классов отражает, как сущности взаимодействуют между собой, позволяя понять структуру и логику работы приложения игрового бэклога. Пользователь связан с основными сущностями системы: играми, отзывами и токенами авторизации. Он может добавлять игры в различные списки бэклога, такие как «Хочу поиграть», «Играю» и другие. Каждая игра может принадлежать нескольким платформам, а также иметь отзывы от разных пользователей. Отзывы содержат оценку и комментарий, что позволяет формировать представление о качестве игры. Токены обеспечивают механизм авторизации пользователей.

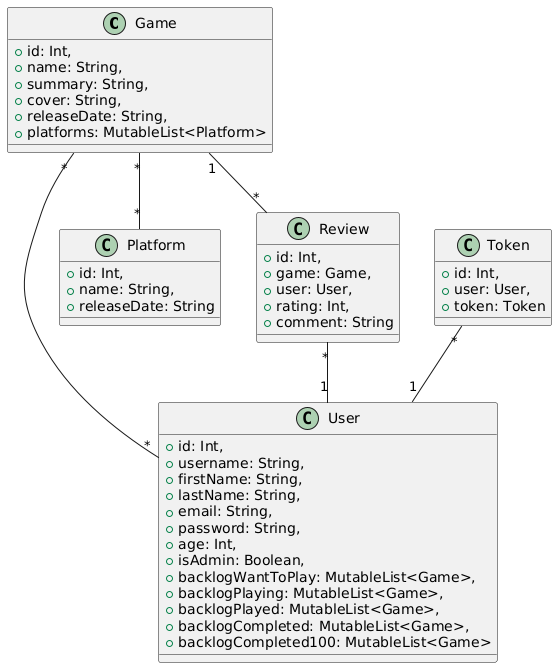


Рисунок 2.10 – Диаграмма классов

Таким образом мы смоделировали диаграмму в нотации IDEF0, в которой отображена функциональная структура системы, а также была успешно создана диаграмма классов.

## **2.3.2 Проектирование базы данных (логическая модель БД, описание каждой таблицы)**

1. user:

Описание: Эта таблица содержит информацию о пользователях системы, включая как обычных пользователей, так и администраторов.

Поля:

* id — Уникальный идентификатор пользователя (первичный ключ).
* username — Уникальное имя пользователя.
* first\_name — Имя пользователя.
* last\_name — Фамилия пользователя.
* email — Уникальная электронная почта пользователя.
* password — Хэшированный пароль.
* age — Возраст пользователя.
* is\_admin — Флаг, указывающий на то, является ли пользователь администратором (boolean).

2. game:

Описание: Таблица содержит информацию об играх, включая описание, дату выхода и обложку.

Поля:

* id — Уникальный идентификатор игры (первичный ключ).
* name — Название игры.
* summary — Подробное текстовое описание игры.
* cover — URL изображения обложки.
* release\_date — Дата выхода игры.

3. platform:

Описание: Таблица содержит информацию о платформах, на которых могут быть доступны игры.

Поля:

* id — Уникальный идентификатор платформы (первичный ключ).
* name — Название платформы.
* release\_date — Дата выхода платформы.

4. review:

Описание: Таблица хранит обзоры пользователей на игры.

Поля:

* id — Уникальный идентификатор обзора (первичный ключ).
* game\_id — Внешний ключ на таблицу Games.
* user\_id — Внешний ключ на таблицу Users.
* rating — Оценка (от 1 до 5).
* comment — Комментарий к обзору (опционально).

5. token:

Описание: Таблица хранит токены авторизации пользователей.

Поля:

* id — Уникальный идентификатор токена (первичный ключ).
* user\_id — Внешний ключ на таблицу Users.
* token — Уникальный токен (например, JWT).

6. game\_platforms:

Описание: Таблица связывает игры с платформами.

Поля:

* game\_id — Внешний ключ на Games, часть составного первичного ключа.
* platform\_id — Внешний ключ на Platforms, часть составного первичного ключа.

7. user\_ backlog\_want\_to\_play:

Описание: Таблица, содержащая список игр, которые пользователь хочет пройти.

Поля:

* user\_id — Внешний ключ на Users, часть составного первичного ключа.
* game\_id — Внешний ключ на Games, часть составного первичного ключа.

8. user\_ backlog\_playing:

Описание: Таблица, содержащая список игр, которые пользователь сейчас проходит.

Поля:

* user\_id — Внешний ключ на Users, часть составного первичного ключа.
* game\_id — Внешний ключ на Games, часть составного первичного ключа.

9. user\_ backlog\_played:

Описание: Таблица, содержащая список игр, в которые пользователь уже сыграл.

Поля:

* user\_id — Внешний ключ на Users, часть составного первичного ключа.
* game\_id — Внешний ключ на Games, часть составного первичного ключа.

10. user\_ backlog\_completed:

Описание: Таблица, содержащая список игр, завершённых пользователем.

Поля:

* user\_id — Внешний ключ на Users, часть составного первичного ключа.
* game\_id — Внешний ключ на Games, часть составного первичного ключа.

11. user\_ backlog\_completed100:

Описание: Таблица, содержащая список игр, завершённых пользователем на 100%.

Поля:

* user\_id — Внешний ключ на Users, часть составного первичного ключа.
* game\_id — Внешний ключ на Games, часть составного первичного ключа.

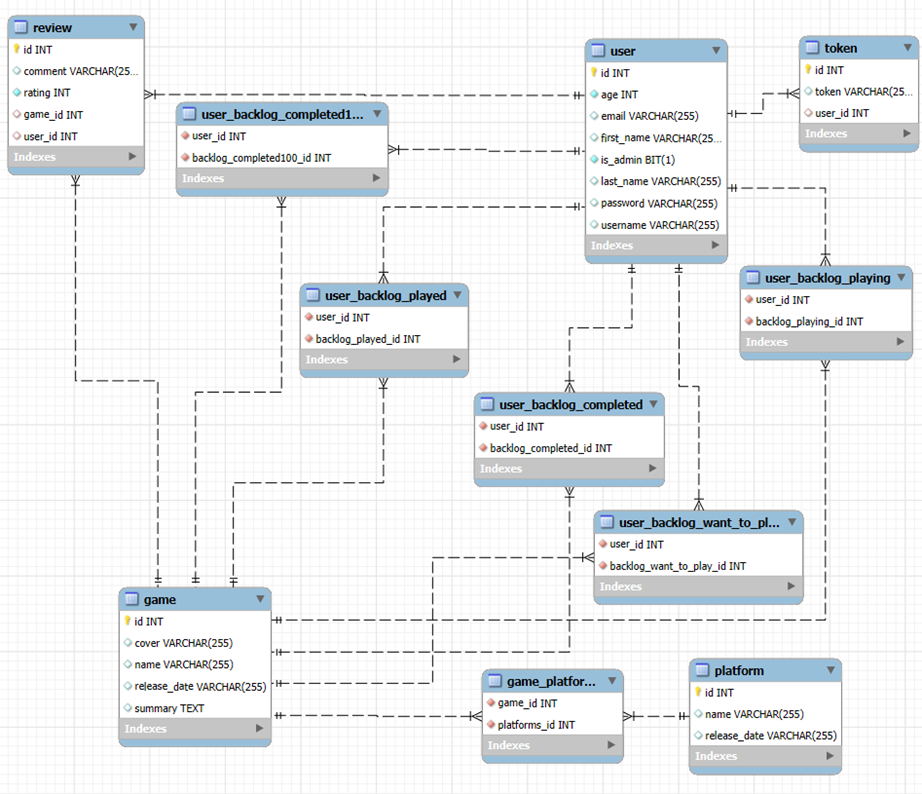


Рисунок 2.11. Логическая модель базы данных

# **ГЛАВА 3. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ**

## **3.1. Разработка клиентской части мобильного приложения (описание основных функций интерфейса)**

## В этом разделе приведён обзор ключевых экранов и технологий, обеспечивающих взаимодействие пользователя с приложением.

## Архитектурный скелет

## activity\_main.xml содержит FragmentContainerView с NavHostFragment: централизованное управление навигацией через Navigation Component.

## Навигационная схема (res/navigation/nav\_graph.xml) определяет переходы между экранами.

## Темы и стили

## Корневой ConstraintLayout использует стиль @style/ Theme.Material3.DayNight.NoActionBar для единого внешнего вида в светлой и тёмной теме.

## Основные экраны (фрагменты)

## HomeFragment: отображает актуальные бэклог-игры.

## SearchFragment: поиск игр по названию.

## GameDetailFragment: отображение информации об игре, с возможностью добавления в бэклог или удаления.

## EditGameFragment: доступен только администраторам, используется для редактирования информации об игре.

## ProfileFragment: отображает списки игр пользователя (например, "Хочу пройти", "Прохожу", "Пройдено").

## SettingsFragment / AccountSettingsFragment: настройки приложения и аккаунта.

## ReviewDialogFragment: диалоговое окно для добавления отзыва на игру.

## Сетевой слой

## Взаимодействие через REST-эндпоинты сервера.

## Интерцептор добавляет заголовок Authorization: Bearer <token>.

## Загрузка изображений

## Используется Glide для подгрузки обложек игр по URL, полученному с сервера.

## **3.2. Разработка серверной части мобильного приложения**

## В этом разделе описана структура и принцип работы серверного приложения на Spring Boot.

## **3.2.1 Описание основной логики сервера**

Ниже изложены ключевые слои и их ответственность:

* Web (REST-контроллеры)
  + **GameController, PlatformController, ReviewController, UserController** — принимают HTTP-запросы, валидируют и преобразуют DTO ↔ сущности.
* Сервисный слой
  + AuthService — инкапсулирует бизнес-логику аутентификации и авторизации: извлечение пользователя по токену, проверка прав администратора по токену.
* Доступ к данным
  + Spring Data JPA репозитории (GameRepository, UserRepository, TokenRepository и др.) обеспечивают CRUD-операции и запросы.
* Безопасность
  + SecurityConfig настраивает PasswordEncoder
  + WebSecurityConfig: отключает CSRF, делает все эндпоинты публичными (permitAll()), так как в самих контроллерах заранее прописаны все ограничения
* Работа с токенами
  + AuthService реализует проверку прав администратора, используя TokenRepository
  + Токены передаются в заголовке Authorization, обрабатываются вручную в контроллерах путём удаления префикса "Bearer ".

## **3.2.2 Подключение базы данных к серверу**

Для хранения и управления данными приложение B4cklog использует реляционную базу данных, интегрированную при помощи Spring Data JPA. Каждая ключевая сущность, такая как пользователь, игра, платформа или список бэклога, представлена классом-моделью, аннотированным как @Entity, что позволяет Hibernate автоматически соотнести её с соответствующей таблицей в базе данных. Связи между сущностями описываются с использованием стандартных аннотаций: «один-к-одному» связывает пользователя с его списками игр, «многие-ко-многим» используется для связи игр с поддерживаемыми платформами.

Конфигурация подключения задаётся через параметры в файле свойств приложения: URL базы, имя пользователя, пароль и настройки Hibernate. Благодаря Spring Boot загрузка драйвера и создание пула соединений происходят автоматически. Механизм автоматического обновления схемы (если он включён) проверяет структуру модели и при необходимости создаёт или изменяет таблицы при старте сервера.

Для выполнения операций с данными используются интерфейсы репозиториев, расширяющие JpaRepository. Они предоставляют готовые методы для сохранения, поиска, обновления и удаления записей без написания SQL-кода вручную. Если требуется выполнение нескольких связанных действий с разными таблицами в рамках одного бизнес-процесса, сервисные методы помечаются как транзакционные, что гарантирует атомарность и целостность данных в случае ошибок.

Таким образом, интеграция базы данных сводится к минимальной конфигурации и декларативному описанию моделей и репозиториев: разработчику достаточно описать сущности и их связи, а Spring Data JPA берет на себя создание запросов и управление соединениями.

## **3.2.3 Подключение сервера к клиентской части мобильного приложения**

В этом пункте описан протокол взаимодействия:

* Формат обмена
  + JSON ↔ DTO (пакет model), сериализация Jackson.
* Аутентификация
  + Клиент отправляет POST /auth/login с LoginRequest{username, password} или POST /auth/register с RegisterRequest{username, email, password, firstName, lastName, age}.
    - Сервер проверяет учётные данные через AuthService, генерирует токен в LoginResponse/RegisterResponse.
    - Клиент сохраняет токен и прикрепляет к HTTP-запросам Authorization: Bearer <token>.
* Эндпоинты
  + /users/\*\*, /games/\*\*, /platforms/\*\*, /auth/\*\*, /reviews/\*\* — организованы под соответствующие задачи.
* Обработка ошибок
  + В случае ошибок бизнес-логики сервисы выбрасывают исключения типа IllegalArgumentException или EntityNotFoundException. Контроллеры перехватывают исключения и возвращают соответствующий статус ответа (400 Bad Request, 404 Not Found и т.д.) с подробным описанием ошибки в теле ответа.

## **3.3. Тестирование разработанного приложения**

В данном разделе мы демонстрируем, как ручные тесты (лист. 32) подтверждают корректность ключевых операций сервера игрового бэклога.

1. registerUserSuccess. Этот тест проверяет, что POST-запрос на регистрацию нового пользователя создаёт учётную запись с валидными полями. При отправке запроса с уникальным username и email сервер возвращает статус 200 и данные созданного пользователя (без пароля). В результате пользователь успешно зарегистрирован, тест пройден.
2. loginReturnsToken. Проверяется, что POST-запрос логина с корректными username и password возвращает JWT токен авторизации. Сервер отвечает статусом 200 и JSON с токеном и базовыми данными пользователя. Токен необходим для авторизации последующих запросов. В результате токен получен, тест успешен.
3. addGameAsAdminSuccess. Тестирует добавление игры с правами администратора. Отправляется POST с данными игры и заголовком Authorization с токеном админа. Если все данные валидны, сервер возвращает созданную игру с ID. В результате игра добавлена и доступна для поиска и получения.
4. getAllGamesReturnsList. Проверяется, что GET-запрос на получение всех игр возвращает массив объектов игр с основными полями. Статус 200, тело ответа не пустое. В результате список игр успешно получен.
5. searchGameReturnsMatches. Тест поиска игры по части названия. При GET-запросе с параметром q=Test сервер возвращает игры, в названии которых есть “Test”. Возвращённый массив содержит хотя бы одну игру. В результате поиск работает корректно.
6. addPlatformAsAdminSuccess. Проверяется добавление платформы с правами администратора. POST с названием платформы и авторизацией создаёт новую платформу. В результате платформа создана и доступна для связывания с играми.
7. getUserProfileAuthorized. Тестирует, что GET-запрос профиля с валидным токеном возвращает информацию о пользователе. Статус 200, в ответе email, username и другие поля. В результате данные профиля доступны.
8. addGameToUserListSuccess. Проверяется добавление игры в пользовательский список (например, backlogWantToPlay). Отправляется POST с userId, gameId и listName, с авторизацией. Сервер подтверждает добавление без ошибок. В результате игра добавлена в нужный список пользователя.
9. removeGameFromAllListsSuccess. Тестирует удаление игры из всех списков пользователя. Отправляется DELETE запрос с userId и gameId, с авторизацией. Сервер подтверждает успешное удаление. В результате игра удалена из всех списков.
10. updateEmailSuccess. Проверяется обновление email пользователя. PATCH-запрос с новым email и авторизацией. Сервер возвращает статус 200, а в базе у пользователя обновлён email. В результате email успешно изменён.

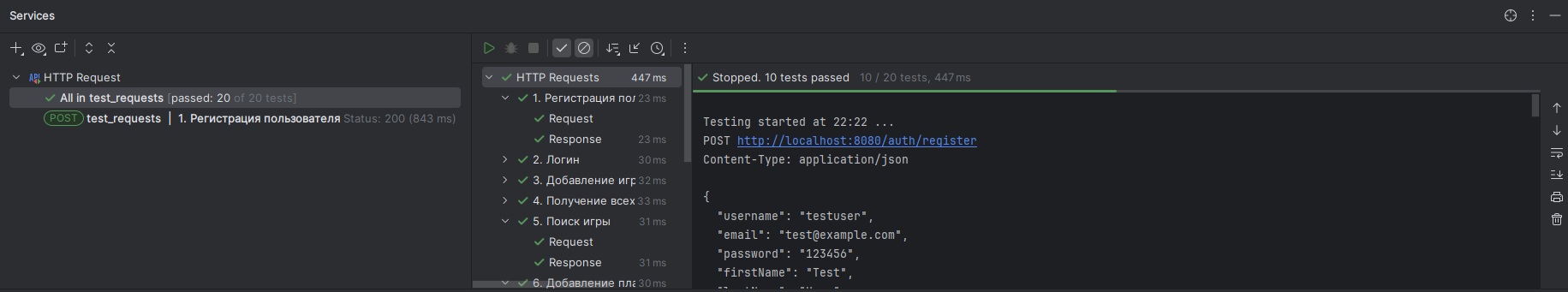


Рисунок 3.1. Успешное выполнение каждого теста

Все тесты (рис. 3.1) проходят успешно, подтверждая корректность базовой работы API для регистрации, аутентификации, работы с играми и пользовательскими списками.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была успешно разработана и протестирована клиент-серверная мобильная система игрового бэклога, предназначенная для ведения, управления и анализа личного игрового бэклога. Основная цель проекта — создание удобного инструмента для каталогизации игр, отслеживания прогресса и написания пользовательских отзывов — была достигнута путём поэтапного выполнения всех поставленных задач: анализа предметной области, сравнительного исследования существующих решений, выбора технологического стека, проектирования логической модели данных и пользовательского интерфейса, разработки серверной и мобильной части, а также проведения тестирования.

В результате сравнительного анализа были выявлены ключевые преимущества нашего приложения по сравнению с конкурентами: наличие открытого REST-API, полная поддержка CRUD-операций как на сервере, так и в клиенте а также расширяемая архитектура, ориентированная на open-source-сообщество и индивидуальную доработку. Выбор стека **Kotlin/Android + Spring Boot/MySQL** обеспечил стабильную производительность и готовность к масштабируемому развертыванию. Логическая модель базы данных, реализованная с использованием Spring Data JPA, позволила минимизировать избыточность и повысить читаемость кода. Навигация в Android-клиенте реализована средствами Jetpack Navigation, что обеспечивает плавные переходы и устойчивость к сбоям.

Практическая реализация продемонстрировала работоспособность всех основных сценариев: регистрация и вход, сохранение токена, отображение профиля и списков игр, добавление и удаление игр из бэклога, публикация отзывов и расчёт средней оценки. Сервер успешно прошёл все тестирования, показав свою устойчивую работоспособность. Разработанная система игрового бэклога доказала свою практическую применимость и может быть использована как основа для дальнейших улучшений, в том числе с интеграцией сторонних игровых API, расширением функций аналитики и реализацией web-версии.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

* + 1. Kotlin Documentation: [Электронный ресурс]. URL: https://kotlinlang.org/docs/home.html (Дата обращения: 01.03.2025).
    2. View Binding (Android Developers): [Электронный ресурс]. URL: https://developer.android.com/topic/libraries/view-binding (Дата обращения:

01.03.2025).

* + 1. ConstraintLayout API Reference (Android Developers): [Электронный ресурс]. URL: https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout (Дата обращения: 01.03.2025).
    2. Material Design Components for Android (Google): [Электронный ресурс]. URL: https://material.io/develop/android (Дата обращения: 01.03.2025).
    3. Navigation Component Guide (Android Developers): [Электронный ресурс]. URL: https://developer.android.com/guide/navigation (Дата обращения: 15.03.2025).
    4. Retrofit — A Type-safe HTTP Client for Android & Java (Square Inc.): [Электронный ресурс]. URL: https://square.github.io/retrofit/ (Дата обращения: 20.03.2025).
    5. OkHttp — Efficient HTTP Client (Square Inc.): [Электронный ресурс]. URL: https://square.github.io/okhttp/ (Дата обращения: 20.03.2025).
    6. Glide — Fast and Efficient Image Loading for Android (BumpTech): [Электронный ресурс]. URL: https://bumptech.github.io/glide/ (Дата обращения: 20.03.2025).
    7. AndroidX Lifecycle Components (Android Developers): [Электронный ресурс]. URL: https://developer.android.com/jetpack/androidx/releases/lifecycle (Дата обращения: 01.04.2025).
    8. kotlinx.serialization Library Documentation (JetBrains): [Электронный ресурс]. URL: https://kotlinlang.org/docs/serialization.html (Дата обращения: 01.04.2025).
    9. Spring Boot Reference Documentation (VMware Spring): [Электронный ресурс]. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ (Дата обращения: 20.04.2025).
    10. Spring Security Reference Documentation (VMware Spring): [Электронный ресурс]. URL: https://docs.spring.io/springsecurity/site/docs/current/reference/html5/ (Дата обращения: 20.04.2025).
    11. Spring Data JPA Reference Documentation (VMware Spring): [Электронный ресурс]. URL: https://docs.spring.io/springdata/jpa/docs/current/reference/html/ (Дата обращения: 20.04.2025).
    12. OpenJDK 21 (Project Page): [Электронный ресурс]. URL: https://openjdk.java.net/projects/jdk/21/ (Дата обращения: 28.04.2025).
    13. MySQL Connector/J Developer Guide (Oracle): [Электронный ресурс]. URL: https://dev.mysql.com/doc/connector-j/8.0/en/ (Дата обращения: 01.05.2025).
    14. Jakarta Servlet 5.0 Specification (Eclipse Foundation): [Электронный ресурс]. URL: https://jakarta.ee/specifications/servlet/5.0/ (Дата обращения: 08.05.2025).
    15. JUnit 5 User Guide (JUnit Team): [Электронный ресурс]. URL: https://junit.org/junit5/docs/current/user-guide/ (Дата обращения: 15.05.2025)

# **ПРИЛОЖЕНИЯ**

Приложение А. Серверная сторона приложения

Приложение Б. Клиентская часть приложение

Приложение В. Файлы разметки экранов приложение

Приложение А

*Листинг 1 – AuthController*

|  |
| --- |
| package com.rarmash.b4cklog\_server.controller  import com.rarmash.b4cklog\_server.dto.LoginRequest  import com.rarmash.b4cklog\_server.dto.LoginResponse  import com.rarmash.b4cklog\_server.dto.RegisterRequest  import com.rarmash.b4cklog\_server.dto.RegisterResponse  import com.rarmash.b4cklog\_server.service.AuthService  import org.springframework.http.ResponseEntity  import org.springframework.web.bind.annotation.\*  import org.springframework.web.server.ResponseStatusException  @RestController  @RequestMapping("/auth")  class AuthController(  private val authService: AuthService  ) {  @PostMapping("/login")  fun login(@RequestBody request: LoginRequest): ResponseEntity<LoginResponse> {  val response = authService.login(request)  return ResponseEntity.ok(response)  }  @PostMapping("/register")  fun register(@RequestBody request: RegisterRequest): ResponseEntity<RegisterResponse> {  val response = authService.register(request)  return ResponseEntity.ok(response)  }  @ExceptionHandler(ResponseStatusException::class)  fun handleResponseStatusException(ex: ResponseStatusException): ResponseEntity<Map<String, String>> {  val body = mapOf("error" to (ex.reason ?: "Error"))  return ResponseEntity.status(ex.statusCode).body(body)  }  } |

*Листинг 2 – GameController*

|  |
| --- |
| package com.rarmash.b4cklog\_server.controller  import com.rarmash.b4cklog\_server.model.game.Game  import com.rarmash.b4cklog\_server.model.game.GameDAO  import com.rarmash.b4cklog\_server.service.AuthService  import org.springframework.web.bind.annotation.\*  import java.util.\*  @RestController  @RequestMapping("/games")  class GameController (  private val gameDAO: GameDAO,  private val authService: AuthService  ) {  @PostMapping("/add")  fun addGame(  @RequestHeader("Authorization") authHeader: String,  @RequestBody game: Game  ): Game {  val token = authHeader.removePrefix("Bearer ").trim()  authService.checkAdmin(token)  return gameDAO.addGame(game)  }  @GetMapping("/get/{id}")  fun getGame(@PathVariable("id") id: Int): Optional<Game> {  return gameDAO.getGame(id = id)  }  @GetMapping("/get/all")  fun getAllGames(): List<Game> {  return gameDAO.getAllGames()  }  @DeleteMapping("/delete/{id}")  fun deleteGame(  @RequestHeader("Authorization") authHeader: String,  @PathVariable("id") id: Int  ) {  val token = authHeader.removePrefix("Bearer ").trim()  authService.checkAdmin(token)  return gameDAO.deleteGame(id = id)  }  @GetMapping("/search")  fun searchGames(@RequestParam("q") query: String): List<Game> {  return gameDAO.searchGamesByName(query)  }  @PostMapping("/update")  fun updateGame(  @RequestHeader("Authorization") authHeader: String,  @RequestBody game: Game  ): Game {  val token = authHeader.removePrefix("Bearer ").trim()  authService.checkAdmin(token)  return gameDAO.updateGame(game)  }  } |

*Листинг 3 – PlatformController*

|  |
| --- |
| package com.rarmash.b4cklog\_server.controller  import com.rarmash.b4cklog\_server.model.platform.Platform  import com.rarmash.b4cklog\_server.model.platform.PlatformDAO  import com.rarmash.b4cklog\_server.service.AuthService  import org.springframework.web.bind.annotation.\*  import java.util.\*  @RestController  @RequestMapping("/platforms")  class PlatformController (  private val platformDAO: PlatformDAO,  private val authService: AuthService  ) {  @PostMapping("/add")  fun addPlatform(  @RequestHeader("Authorization") authHeader: String,  @RequestBody platform: Platform,  ): Platform {  val token = authHeader.removePrefix("Bearer ").trim()  authService.checkAdmin(token)  return platformDAO.addPlatform(platform)  }  @GetMapping("/get/{id}")  fun getPlatform(@PathVariable("id") id: Int): Optional<Platform> {  return platformDAO.getPlatform(id = id)  }  @GetMapping("/get/all")  fun getAllPlatforms(): List<Platform> {  return platformDAO.getAllPlatforms()  }  @DeleteMapping("/delete/{id}")  fun deletePlatform(  @RequestHeader("Authorization") authHeader: String,  @PathVariable("id") id: Int  ) {  val token = authHeader.removePrefix("Bearer ").trim()  authService.checkAdmin(token)  return platformDAO.deletePlatform(id = id)  }  } |

*Листинг 4 – ReviewController*

|  |
| --- |
| package com.rarmash.b4cklog\_server.controller  import com.rarmash.b4cklog\_server.dto.ReviewRequest  import com.rarmash.b4cklog\_server.dto.ReviewResponse  import com.rarmash.b4cklog\_server.model.review.ReviewDAO  import org.springframework.http.HttpStatus  import org.springframework.http.ResponseEntity  import org.springframework.web.bind.annotation.\*  @RestController  @RequestMapping("/reviews")  class ReviewController(private val reviewDAO: ReviewDAO) {  @GetMapping("/user/{userId}/game/{gameId}")  fun getUserReview(  @PathVariable userId: Int,  @PathVariable gameId: Int  ): ResponseEntity<ReviewResponse> {  val review = reviewDAO.getUserReviewForGame(userId, gameId)  ?: return ResponseEntity.notFound().build()  val response = ReviewResponse(  rating = review.rating,  comment = review.comment  )  return ResponseEntity.ok(response)  }  @PostMapping("/add")  fun addOrUpdateReview(@RequestBody request: ReviewRequest): ResponseEntity<String> {  if (request.rating !in 1..10) {  return ResponseEntity.badRequest().body("Рейтинг должен быть от 1 до 10")  }  reviewDAO.addOrUpdateReview(  userId = request.userId,  gameId = request.gameId,  rating = request.rating,  comment = request.comment  )  return ResponseEntity.status(HttpStatus.CREATED).body("Отзыв добавлен/обновлён")  }  @GetMapping("/game/{gameId}/average")  fun getAverageRating(@PathVariable gameId: Int): ResponseEntity<Double> {  val avg = reviewDAO.getAverageRating(gameId)  return ResponseEntity.ok(avg)  }  } |

*Листинг 5 – UserController*

|  |
| --- |
| package com.rarmash.b4cklog\_server.controller  import org.springframework.web.server.ResponseStatusException  import org.springframework.http.HttpStatus  import com.rarmash.b4cklog\_server.model.game.GameDAO  import com.rarmash.b4cklog\_server.model.user.User  import com.rarmash.b4cklog\_server.model.user.UserDAO  import com.rarmash.b4cklog\_server.service.AuthService  import org.springframework.web.bind.annotation.\*  @RestController  @RequestMapping("/users")  class UserController(  private val userDAO: UserDAO,  private val gameDAO: GameDAO,  private val authService: AuthService  ) {  @GetMapping("/profile")  fun getUserProfile(  @RequestHeader("Authorization") authHeader: String  ): User {  val token = authHeader.removePrefix("Bearer ").trim()  val user = authService.getUserByToken(token)  return user  }  @PostMapping("/{userId}/addGameToList")  fun addGameToList(  @PathVariable userId: Int,  @RequestParam gameId: Int,  @RequestParam listName: String  ) {  val game = gameDAO.getGame(gameId).orElseThrow { ResponseStatusException(HttpStatus.NOT\_FOUND, "Игра не найдена") }  userDAO.addGameToList(userId, game, listName)  }  @DeleteMapping("/{userId}/removeGameFromAllLists")  fun removeGameFromAllLists(  @PathVariable userId: Int,  @RequestParam gameId: Int  ) {  val game = gameDAO.getGame(gameId)  .orElseThrow { ResponseStatusException(HttpStatus.NOT\_FOUND, "Игра не найдена") }  userDAO.removeGameFromAllLists(userId, game)  }  @PatchMapping("/updateEmail")  fun updateEmail(  @RequestHeader("Authorization") authHeader: String,  @RequestParam newEmail: String  ) {  val token = authHeader.removePrefix("Bearer ").trim()  val user = authService.getUserByToken(token)  userDAO.updateEmail(user.id, newEmail)  }  @PatchMapping("/updatePassword")  fun updatePassword(  @RequestHeader("Authorization") authHeader: String,  @RequestParam newPassword: String  ) {  val token = authHeader.removePrefix("Bearer ").trim()  val user = authService.getUserByToken(token)  userDAO.updatePassword(user.id, newPassword)  }  } |

*Листинг 6 – LoginRequest*

|  |
| --- |
| package com.rarmash.b4cklog\_server.dto  data class LoginRequest(  val username: String,  val password: String  ) |

*Листинг 7 – LoginResponse*

|  |
| --- |
| package com.rarmash.b4cklog\_server.dto  data class LoginResponse(  val token: String  ) |

*Листинг 8 – RegisterRequest*

|  |
| --- |
| package com.rarmash.b4cklog\_server.dto  data class RegisterRequest(  val username: String,  val email: String,  val password: String,  val firstName: String = "",  val lastName: String = "",  val age: Int = 0  ) |

*Листинг 9 – RegisterResponse*

|  |
| --- |
| package com.rarmash.b4cklog\_server.dto  data class RegisterResponse(  val token: String  ) |

*Листинг 10 – ReviewRequest*

|  |
| --- |
| package com.rarmash.b4cklog\_server.dto  data class ReviewRequest(  val userId: Int,  val gameId: Int,  val rating: Int,  val comment: String? = null  ) |

*Листинг 11 – ReviewResponse*

|  |
| --- |
| package com.rarmash.b4cklog\_server.dto  data class ReviewResponse(  val rating: Int,  val comment: String?  ) |

*Листинг 12 – Game*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.game  import com.rarmash.b4cklog\_server.model.platform.Platform  import jakarta.persistence.\*  @Entity  data class Game(  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  val id: Int = 0,  var name: String,  @Column(columnDefinition = "TEXT")  var summary: String = "",  var cover: String = "",  var releaseDate: String = "",  @ManyToMany  var platforms: MutableList<Platform> = mutableListOf()  ) {  constructor(id: Int) : this(id = id, name = "", summary = "")  } |

*Листинг 13 – GameDao*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.game  import com.rarmash.b4cklog\_server.model.platform.PlatformRepository  import org.springframework.stereotype.Service  @Service  class GameDAO (  private val repository: GameRepository,  private val platformRepository: PlatformRepository  ){  fun addGame(game: Game): Game {  val resolvedPlatforms = game.platforms.mapNotNull { partial ->  platformRepository.findById(partial.id).orElse(null)  }.toMutableList()  val gameToSave = game.copy(platforms = resolvedPlatforms)  return repository.save(gameToSave)  }  fun updateGame(updatedGame: Game): Game {  val existing = repository.findById(updatedGame.id)  .orElseThrow { NoSuchElementException("Игра с id ${updatedGame.id} не найдена") }  existing.name = updatedGame.name  existing.summary = updatedGame.summary  existing.cover = updatedGame.cover  existing.releaseDate = updatedGame.releaseDate  existing.platforms = updatedGame.platforms  return repository.save(existing)  }  fun getGame(id: Int) = repository.findById(id)  fun getAllGames() = repository.findAll().toList()  fun deleteGame(id: Int) = repository.deleteById(id)  fun searchGamesByName(namePart: String) = repository.findByNameContainingIgnoreCase(namePart)  } |

*Листинг 14 – GameRepository*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.game  import org.springframework.data.jpa.repository.Query  import org.springframework.data.repository.CrudRepository  import org.springframework.data.repository.query.Param  import org.springframework.stereotype.Repository  @Repository  interface GameRepository : CrudRepository<Game, Int> {  @Query("SELECT g FROM Game g WHERE LOWER(g.name) LIKE LOWER(CONCAT('%', :namePart, '%'))")  fun findByNameContainingIgnoreCase(@Param("namePart") namePart: String): List<Game>  } |

*Листинг 15 – Platform*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.platform  import jakarta.persistence.Entity  import jakarta.persistence.GeneratedValue  import jakarta.persistence.GenerationType  import jakarta.persistence.Id  @Entity  data class Platform(  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  val id: Int = 0,  var name: String = "",  var releaseDate: String = ""  ) |

*Листинг 16 – PlatformDao*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.platform  import com.rarmash.b4cklog\_server.model.game.Game  import com.rarmash.b4cklog\_server.model.game.GameRepository  import org.springframework.stereotype.Service  @Service  class PlatformDAO (  private val repository: PlatformRepository  ){  fun addPlatform(platform: Platform) = repository.save(platform)  fun getPlatform(id: Int) = repository.findById(id)  fun getAllPlatforms() = repository.findAll().toList()  fun deletePlatform(id: Int) = repository.deleteById(id)  } |

*Листинг 17 – PlatformRepository*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.platform  import org.springframework.data.repository.CrudRepository  import org.springframework.stereotype.Repository  @Repository  interface PlatformRepository : CrudRepository<Platform, Int> |

*Листинг 18 – Review*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.review  import com.rarmash.b4cklog\_server.model.game.Game  import com.rarmash.b4cklog\_server.model.user.User  import jakarta.persistence.\*  @Entity  data class Review(  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  val id: Int = 0,  @ManyToOne  val game: Game,  @ManyToOne  val user: User,  val rating: Int, // от 1 до 5  @Column(columnDefinition = "TEXT")  val comment: String? = null  ) |

*Листинг 19 – ReviewDAO*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.review  import com.rarmash.b4cklog\_server.model.game.Game  import com.rarmash.b4cklog\_server.model.user.User  import org.springframework.stereotype.Service  @Service  class ReviewDAO(private val repository: ReviewRepository) {  fun addReview(review: Review) = repository.save(review)  fun getReviewsForGame(gameId: Int): List<Review> = repository.findAllByGameId(gameId)  fun getAverageRating(gameId: Int): Double {  return repository.findAverageRatingByGameId(gameId) ?: 0.0  }  fun getUserReviewForGame(userId: Int, gameId: Int): Review? {  return repository.findByUserIdAndGameId(userId, gameId)  }  fun addOrUpdateReview(userId: Int, gameId: Int, rating: Int, comment: String?) {  val existing = repository.findByUserIdAndGameId(userId, gameId)  val review = existing?.copy(rating = rating, comment = comment)  ?: Review(user = User(id = userId), game = Game(id = gameId), rating = rating, comment = comment)  repository.save(review)  }  } |

*Листинг 20 – ReviewRepository*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.review  import org.springframework.data.jpa.repository.Query  import org.springframework.data.repository.CrudRepository  import org.springframework.stereotype.Repository  @Repository  interface ReviewRepository : CrudRepository<Review, Int> {  fun findAllByGameId(gameId: Int): List<Review>  @Query("SELECT AVG(r.rating) FROM Review r WHERE r.game.id = :gameId")  fun findAverageRatingByGameId(gameId: Int): Double?  fun findByUserIdAndGameId(userId: Int, gameId: Int): Review?  } |

*Листинг 21 – Token*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.token  import com.rarmash.b4cklog\_server.model.user.User  import jakarta.persistence.\*  @Entity  data class Token (  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  val id: Int = 0,  @ManyToOne  val user: User,  @Column(unique = true)  val token: String  ) |

*Листинг 22 – TokenDAO*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.token  import org.springframework.stereotype.Service  @Service  class TokenDAO(private val repository: TokenRepository) {  fun saveToken(token: Token): Token = repository.save(token)  fun getToken(tokenString: String): Token? = repository.findByToken(tokenString)  fun deleteToken(tokenString: String) = repository.deleteByToken(tokenString)  } |

*Листинг 23 – TokenRepository*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.token  import org.springframework.data.repository.CrudRepository  import org.springframework.stereotype.Repository  @Repository  interface TokenRepository : CrudRepository<Token, Int> {  fun findByToken(token: String): Token?  fun deleteByToken(token: String)  } |

*Листинг 24 – User*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.user  import com.rarmash.b4cklog\_server.model.game.Game  import jakarta.persistence.\*  @Entity  data class User(  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  val id: Int = 0,  @Column(unique = true)  var username: String,  var firstName: String = "",  var lastName: String = "",  @Column(unique = true)  var email: String = "",  var password: String = "",  var age: Int = 0,  var isAdmin: Boolean = false,  // B4cklog lists  @ManyToMany  var backlogWantToPlay: MutableList<Game> = mutableListOf(),  @ManyToMany  var backlogPlaying: MutableList<Game> = mutableListOf(),  @ManyToMany  var backlogPlayed: MutableList<Game> = mutableListOf(),  @ManyToMany  var backlogCompleted: MutableList<Game> = mutableListOf(),  @ManyToMany  var backlogCompleted100: MutableList<Game> = mutableListOf()  ) {  constructor(id: Int) : this(id = id, username = "", password = "", isAdmin = false)  } |

*Листинг 25 – UserDAO*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.user  import com.rarmash.b4cklog\_server.model.game.Game  import com.rarmash.b4cklog\_server.util.HashUtils  import org.springframework.http.HttpStatus  import org.springframework.stereotype.Service  import org.springframework.web.server.ResponseStatusException  @Service  class UserDAO(  private val repository: UserRepository  ) {  fun addUser(user: User) = repository.save(user)  fun getUser(id: Int) = repository.findById(id)  fun getUserByUsername(username: String) = repository.findByUsername(username)  fun getAllUsers() = repository.findAll().toList()  fun deleteUser(id: Int) = repository.deleteById(id)  fun saveUser(user: User) = repository.save(user)  fun addGameToList(userId: Int, game: Game, listName: String) {  val user = getUser(userId).orElseThrow { RuntimeException("User not found") }  // Сначала удаляем игру из всех списков  removeGameFromAllLists(user, game)  // Добавляем игру в нужный список  when (listName) {  "wantToPlay" -> user.backlogWantToPlay.add(game)  "playing" -> user.backlogPlaying.add(game)  "played" -> user.backlogPlayed.add(game)  "completed" -> user.backlogCompleted.add(game)  "completed100" -> user.backlogCompleted100.add(game)  else -> throw ResponseStatusException(HttpStatus.BAD\_REQUEST, "Некорректное имя списка")  }  saveUser(user)  }  private fun removeGameFromAllLists(user: User, game: Game) {  user.backlogWantToPlay.remove(game)  user.backlogPlaying.remove(game)  user.backlogPlayed.remove(game)  user.backlogCompleted.remove(game)  user.backlogCompleted100.remove(game)  }  fun removeGameFromAllLists(userId: Int, game: Game) {  val user = repository.findById(userId)  .orElseThrow { ResponseStatusException(HttpStatus.NOT\_FOUND, "Пользователь не найден") }  user.backlogWantToPlay.remove(game)  user.backlogPlaying.remove(game)  user.backlogPlayed.remove(game)  user.backlogCompleted.remove(game)  user.backlogCompleted100.remove(game)  repository.save(user)  }  fun updateEmail(userId: Int, newEmail: String) {  val user = getUser(userId).orElseThrow { ResponseStatusException(HttpStatus.NOT\_FOUND, "Пользователь не найден") }  if (repository.findByEmail(newEmail) != null) {  throw ResponseStatusException(HttpStatus.CONFLICT, "Этот email уже используется")  }  user.email = newEmail  saveUser(user)  }  fun updatePassword(userId: Int, newPassword: String) {  val user = getUser(userId).orElseThrow { ResponseStatusException(HttpStatus.NOT\_FOUND, "Пользователь не найден") }  user.password = HashUtils.sha256(newPassword)  saveUser(user)  }  } |

*Листинг 26 – UserRepository*

|  |
| --- |
| package com.rarmash.b4cklog\_server.model.user  import org.springframework.data.repository.CrudRepository  import org.springframework.stereotype.Repository  @Repository  interface UserRepository : CrudRepository<User, Int> {  fun findByUsername(username: String): User?  fun findByEmail(email: String): User?  } |

*Листинг 27 – AuthService*

|  |
| --- |
| package com.rarmash.b4cklog\_server.service  import com.rarmash.b4cklog\_server.dto.LoginRequest  import com.rarmash.b4cklog\_server.dto.LoginResponse  import com.rarmash.b4cklog\_server.dto.RegisterRequest  import com.rarmash.b4cklog\_server.dto.RegisterResponse  import com.rarmash.b4cklog\_server.model.token.\*  import com.rarmash.b4cklog\_server.model.user.User  import com.rarmash.b4cklog\_server.model.user.UserRepository  import com.rarmash.b4cklog\_server.util.HashUtils  import org.springframework.http.HttpStatus  import org.springframework.stereotype.Service  import org.springframework.web.server.ResponseStatusException  import java.util.\*  @Service  class AuthService(  private val userRepository: UserRepository,  private val tokenRepository: TokenRepository  ) {  fun login(request: LoginRequest): LoginResponse {  val user = userRepository.findByUsername(request.username)  ?: throw ResponseStatusException(HttpStatus.BAD\_REQUEST, "User not found")  val hashedInput = HashUtils.sha256(request.password)  if (user.password != hashedInput) {  throw ResponseStatusException(HttpStatus.BAD\_REQUEST, "Invalid password")  }  val tokenString = UUID.randomUUID().toString()  val token = Token(user = user, token = tokenString)  tokenRepository.save(token)  return LoginResponse(tokenString)  }  fun register(request: RegisterRequest): RegisterResponse {  if (userRepository.findByUsername(request.username) != null) {  throw ResponseStatusException(HttpStatus.CONFLICT, "Username already taken")  }  if (userRepository.findByEmail(request.email) != null) {  throw ResponseStatusException(HttpStatus.CONFLICT, "Email already registered")  }  val user = User(  username = request.username,  email = request.email,  password = HashUtils.sha256(request.password),  firstName = request.firstName,  lastName = request.lastName,  age = request.age  )  val savedUser = userRepository.save(user)  val tokenString = UUID.randomUUID().toString()  val token = Token(user = savedUser, token = tokenString)  tokenRepository.save(token)  return RegisterResponse(tokenString)  }  fun checkAdmin(tokenString: String): User {  val user = getUserByToken(tokenString)  if (!user.isAdmin) {  throw ResponseStatusException(HttpStatus.FORBIDDEN, "Требуются права администратора")  }  return user  }  fun getUserByToken(tokenString: String): User {  val token = tokenRepository.findByToken(tokenString)  ?: throw ResponseStatusException(HttpStatus.UNAUTHORIZED, "Недействительный токен")  return token.user  }  } |

*Листинг 28 – SecurityConfig*

|  |
| --- |
| package com.rarmash.b4cklog\_server.service  import org.springframework.context.annotation.Bean  import org.springframework.context.annotation.Configuration  import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder  import org.springframework.security.crypto.password.PasswordEncoder  @Configuration  class SecurityConfig {  @Bean  fun passwordEncoder(): PasswordEncoder = BCryptPasswordEncoder()  } |

*Листинг 29 – WebSecurityConfig*

|  |
| --- |
| package com.rarmash.b4cklog\_server.service  import org.springframework.context.annotation.Bean  import org.springframework.context.annotation.Configuration  import org.springframework.security.config.annotation.web.builders.HttpSecurity  import org.springframework.security.web.SecurityFilterChain  @Configuration  class WebSecurityConfig {  @Bean  fun filterChain(http: HttpSecurity): SecurityFilterChain {  http  .csrf { it.disable() }  .authorizeRequests()  .anyRequest().permitAll()  // .authorizeHttpRequests { auth ->  // auth  // .requestMatchers(HttpMethod.POST, "/auth/register", "/auth/login").permitAll()  // .anyRequest().authenticated()  // }  return http.build()  }  } |

*Листинг 30 – HashUtils*

|  |
| --- |
| package com.rarmash.b4cklog\_server.util  import java.security.MessageDigest  object HashUtils {  fun sha256(input: String): String {  val bytes = MessageDigest.getInstance("SHA-256").digest(input.toByteArray())  return bytes.joinToString("") { "%02x".format(it) }  }  } |

*Листинг 31 – B4cklogServerApplication*

|  |
| --- |
| package com.rarmash.b4cklog\_server  import org.springframework.boot.autoconfigure.SpringBootApplication  import org.springframework.boot.runApplication  import io.github.cdimascio.dotenv.dotenv  @SpringBootApplication  class B4cklogServerApplication  fun main(args: Array<String>) {  val dotenv = dotenv()  System.setProperty("DB\_URL", dotenv["DB\_URL"])  System.setProperty("DB\_USERNAME", dotenv["DB\_USERNAME"])  System.setProperty("DB\_PASSWORD", dotenv["DB\_PASSWORD"])  runApplication<B4cklogServerApplication>(\*args)  } |

*Листинг 31 – application.properties*

|  |
| --- |
| spring.application.name=B4cklogServer  spring.datasource.url=${DB\_URL}  spring.datasource.username=${DB\_USERNAME}  spring.datasource.password=${DB\_PASSWORD}  spring.jpa.hibernate.ddl-auto=update  spring.jpa.show-sql=true  spring.jpa.properties.hibernate.format\_sql=true |

*Листинг 32 – test\_requests.http*

|  |
| --- |
| ### 1. Регистрация пользователя  POST http://localhost:8080/auth/register  Content-Type: application/json  {  "username": "testuser",  "email": "test@example.com",  "password": "123456",  "firstName": "Test",  "lastName": "User",  "age": 20  }  ### 2. Логин  POST http://localhost:8080/auth/login  Content-Type: application/json  {  "username": "testuser",  "password": "123456"  }  ### 3. Добавление игры (админ)  @admin\_token = 4d677f49-dee2-46e3-acba-33760fc07e69  POST http://localhost:8080/games/add  Authorization: Bearer {{admin\_token}}  Content-Type: application/json  {  "name": "Test Game",  "summary": "Cool test game",  "cover": "https://example.com/image.png",  "releaseDate": "2025-05-30",  "platforms": []  }  ### 4. Получение всех игр  GET http://localhost:8080/games/get/all  ### 5. Поиск игры  GET http://localhost:8080/games/search?q=Test  ### 6. Добавление платформы  POST http://localhost:8080/platforms/add  Authorization: Bearer {{admin\_token}}  Content-Type: application/json  {  "name": "Steam"  }  ### 7. Получение профиля  @user\_token = 4d677f49-dee2-46e3-acba-33760fc07e69  GET http://localhost:8080/users/profile  Authorization: Bearer {{user\_token}}  ### 8. Добавление игры в список  @userId = 1  POST http://localhost:8080/users/{{userId}}/addGameToList?gameId=1&listName=backlogWantToPlay  Authorization: Bearer {{user\_token}}  ### 9. Удаление игры из всех списков  DELETE http://localhost:8080/users/{{userId}}/removeGameFromAllLists?gameId=1  Authorization: Bearer {{user\_token}}  ### 10. Обновление email  PATCH http://localhost:8080/users/updateEmail?newEmail=new@example.com  Authorization: Bearer {{user\_token}} |

*Листинг 33 – build.gradle.kts*

|  |
| --- |
| plugins {  kotlin("jvm") version "1.9.25"  kotlin("plugin.spring") version "1.9.25"  id("org.springframework.boot") version "3.5.0"  id("io.spring.dependency-management") version "1.1.7"  kotlin("plugin.jpa") version "1.9.25"  }  group = "com.rarmash"  version = "0.0.1-SNAPSHOT"  java {  toolchain {  languageVersion = JavaLanguageVersion.of(21)  }  }  repositories {  mavenCentral()  }  dependencies {  implementation("org.springframework.boot:spring-boot-starter-data-jpa")  implementation("org.springframework.boot:spring-boot-starter-web")  implementation("com.fasterxml.jackson.module:jackson-module-kotlin")  implementation("org.jetbrains.kotlin:kotlin-reflect")  implementation("mysql:mysql-connector-java:8.0.33")  implementation("org.springframework.boot:spring-boot-starter-security")  implementation("io.github.cdimascio:java-dotenv:3.1.7")  testImplementation("org.springframework.boot:spring-boot-starter-test")  testImplementation("org.jetbrains.kotlin:kotlin-test-junit5")  testImplementation("org.mockito.kotlin:mockito-kotlin:4.1.0")  testImplementation("io.mockk:mockk:1.13.4")  testRuntimeOnly("org.junit.platform:junit-platform-launcher")  }  kotlin {  compilerOptions {  freeCompilerArgs.addAll("-Xjsr305=strict")  }  }  allOpen {  annotation("jakarta.persistence.Entity")  annotation("jakarta.persistence.MappedSuperclass")  annotation("jakarta.persistence.Embeddable")  }  tasks.withType<Test> {  useJUnitPlatform()  } |

Приложение Б

*Листинг 34 – AndroidManifest*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools">  <uses-permission android:name="android.permission.INTERNET" />  <application  android:name=".App"  android:allowBackup="true"  android:dataExtractionRules="@xml/data\_extraction\_rules"  android:fullBackupContent="@xml/backup\_rules"  android:icon="@mipmap/ic\_launcher"  android:label="@string/app\_name"  android:roundIcon="@mipmap/ic\_launcher\_round"  android:supportsRtl="true"  android:theme="@style/Theme.B4cklog2"  tools:targetApi="31"  android:usesCleartextTraffic="true">  <activity  android:name=".activities.WelcomeActivity"  android:exported="true">  <intent-filter>  <action android:name="android.intent.action.MAIN" />  <category android:name="android.intent.category.LAUNCHER" />  </intent-filter>  </activity>  <activity  android:name=".activities.LoginActivity"  android:exported="true" />  <activity  android:name=".activities.SignUpActivity"  android:exported="true" />  <activity  android:name=".activities.MainActivity"  android:exported="true" />  </application>  </manifest> |

*Листинг 35 – LoginActivity*

|  |
| --- |
| package com.rarmash.b4cklog.activities  import android.content.Intent  import android.os.Bundle  import android.widget.Button  import android.widget.EditText  import android.widget.Toast  import androidx.appcompat.app.AppCompatActivity  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.network.ApiClient  import com.rarmash.b4cklog.network.AuthResponse  import com.rarmash.b4cklog.network.LoginRequest  import com.rarmash.b4cklog.util.AuthPrefs  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class LoginActivity : AppCompatActivity() {  private lateinit var loginButton: Button  private lateinit var etUsername: EditText  private lateinit var etPassword: EditText  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_login)  loginButton = findViewById(R.id.login\_button)  etUsername = findViewById(R.id.editTextUsername)  etPassword = findViewById(R.id.editTextPassword)  loginButton.setOnClickListener {  val username = etUsername.text.toString()  val password = etPassword.text.toString()  if (username.isNotEmpty() && password.isNotEmpty()) {  loginUser(username, password)  } else {  Toast.makeText(this, "Заполните все поля", Toast.LENGTH\_SHORT).show()  }  }  }  private fun loginUser(username: String, password: String) {  val call = ApiClient.authApi.login(LoginRequest(username, password))  call.enqueue(object: Callback<AuthResponse> {  override fun onResponse(call: Call<AuthResponse>, response: Response<AuthResponse>) {  if (response.isSuccessful && response.body() != null) {  val token = response.body()!!.token  Toast.makeText(this@LoginActivity, "Успешный вход!", Toast.LENGTH\_SHORT).show()  AuthPrefs.saveToken(this@LoginActivity, token)  val intent = Intent(this@LoginActivity, MainActivity::class.java)  startActivity(intent)  } else {  Toast.makeText(this@LoginActivity, "Ошибка входа", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<AuthResponse>, t: Throwable) {  Toast.makeText(this@LoginActivity, "Ошибка сети: ${t.message}", Toast.LENGTH\_SHORT).show()  }  })  }  } |

*Листинг 36 – MainActivity*

|  |
| --- |
| package com.rarmash.b4cklog.activities  import android.os.Bundle  import androidx.appcompat.app.AppCompatActivity  import androidx.appcompat.app.AppCompatDelegate  import androidx.navigation.NavController  import androidx.navigation.fragment.NavHostFragment  import androidx.navigation.ui.setupWithNavController  import com.google.android.material.bottomnavigation.BottomNavigationView  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.models.User  import com.rarmash.b4cklog.network.ApiClient  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  import com.rarmash.b4cklog.util.SessionManager  class MainActivity : AppCompatActivity() {  private lateinit var navController: NavController  override fun onCreate(savedInstanceState: Bundle?) {  val sharedPreferences = getSharedPreferences("settings", MODE\_PRIVATE)  val isDarkMode = sharedPreferences.getBoolean("dark\_mode", false)  AppCompatDelegate.setDefaultNightMode(  if (isDarkMode) AppCompatDelegate.MODE\_NIGHT\_YES  else AppCompatDelegate.MODE\_NIGHT\_NO  )  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  val navHostFragment = supportFragmentManager.findFragmentById(R.id.fragment\_container) as NavHostFragment  navController = navHostFragment.navController  val navBar = findViewById<BottomNavigationView>(R.id.bottomNavigationView)  navBar.setupWithNavController(navController)  ApiClient.profileApi.getUserProfile().enqueue(object : Callback<User> {  override fun onResponse(call: Call<User>, response: Response<User>) {  if (response.isSuccessful) {  val user = response.body()  user?.let {  SessionManager.userId = it.id // Сохраняем  }  }  }  override fun onFailure(call: Call<User>, t: Throwable) {  // Обработка ошибки  }  })  }  } |

*Листинг 37 – SignUpActivity*

|  |
| --- |
| package com.rarmash.b4cklog.activities  import android.content.Intent  import android.os.Bundle  import android.util.Log  import android.widget.Button  import android.widget.EditText  import android.widget.Toast  import androidx.appcompat.app.AppCompatActivity  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.network.ApiClient  import com.rarmash.b4cklog.network.AuthResponse  import com.rarmash.b4cklog.network.RegisterRequest  import com.rarmash.b4cklog.util.AuthPrefs  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class SignUpActivity : AppCompatActivity() {  private lateinit var signupButton: Button  private lateinit var etUsername: EditText  private lateinit var etPassword: EditText  private lateinit var etEmail: EditText  private lateinit var etFirstName: EditText  private lateinit var etLastName: EditText  private lateinit var etAge: EditText  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_signup)  signupButton = findViewById(R.id.signup\_button)  etUsername = findViewById(R.id.editTextUsername)  etPassword = findViewById(R.id.editTextPassword)  etEmail = findViewById(R.id.editTextEmail)  etFirstName = findViewById(R.id.editTextFirstName)  etLastName = findViewById(R.id.editTextLastName)  etAge = findViewById(R.id.editTextAge)  signupButton.setOnClickListener {  val username = etUsername.text.toString()  val password = etPassword.text.toString()  val email = etEmail.text.toString()  val firstName = etFirstName.text.toString()  val lastName = etLastName.text.toString()  val age = etAge.text.toString().toInt()  if (  username.isNotEmpty()  && password.isNotEmpty()  && email.isNotEmpty()  && firstName.isNotEmpty()  && lastName.isNotEmpty()) {  registerUser(username, password, email, firstName, lastName, age)  } else {  Toast.makeText(this, "Заполните все поля", Toast.LENGTH\_SHORT).show()  }  }  }  private fun registerUser(username: String, password: String, email: String, firstName: String, lastName: String, age: Int) {  val call = ApiClient.authApi.register(RegisterRequest(username, email, password, firstName, lastName, age))  call.enqueue(object: Callback<AuthResponse> {  override fun onResponse(call: Call<AuthResponse>, response: Response<AuthResponse>) {  if (response.isSuccessful) {  val token = response.body()!!.token  AuthPrefs.saveToken(this@SignUpActivity, token)  val intent = Intent(this@SignUpActivity, MainActivity::class.java)  Toast.makeText(this@SignUpActivity, "Регистрация прошла успешно", Toast.LENGTH\_SHORT).show()  startActivity(intent)  } else {  //TODO: Добавить больше вариантов ошибок  Toast.makeText(this@SignUpActivity, "Ошибка регистрации", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<AuthResponse>, t: Throwable) {  Toast.makeText(this@SignUpActivity, "Ошибка сети: ${t.message}", Toast.LENGTH\_SHORT).show()  Log.d("SignUpActivity", "ERROR: ${t.message.toString()}")  }  })  }  } |

*Листинг 38 – WelcomeActivity*

|  |
| --- |
| package com.rarmash.b4cklog.activities  import android.content.Intent  import android.os.Bundle  import android.widget.Button  import androidx.appcompat.app.AppCompatActivity  import androidx.appcompat.app.AppCompatDelegate  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.util.AuthPrefs  class WelcomeActivity : AppCompatActivity() {  lateinit var login\_button: Button  lateinit var signup\_button: Button  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  val sharedPreferences = getSharedPreferences("settings", MODE\_PRIVATE)  val isDarkMode = sharedPreferences.getBoolean("dark\_mode", false)  AppCompatDelegate.setDefaultNightMode(  if (isDarkMode) AppCompatDelegate.MODE\_NIGHT\_YES  else AppCompatDelegate.MODE\_NIGHT\_NO  )  val token = AuthPrefs.getToken(this)  if (!token.isNullOrEmpty()) {  val intent = Intent(this, MainActivity::class.java)  startActivity(intent)  finish()  return  }  setContentView(R.layout.activity\_welcome)  login\_button = findViewById(R.id.login\_button)  signup\_button = findViewById(R.id.signup\_button)  login\_button.setOnClickListener{  val intent = Intent(this, LoginActivity::class.java)  startActivity(intent)  }  signup\_button.setOnClickListener{  val intent = Intent(this, SignUpActivity::class.java)  startActivity(intent)  }  }  } |

*Листинг 39 – GameAdapter*

|  |
| --- |
| package com.rarmash.b4cklog.adapters  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.ImageView  import android.widget.TextView  import androidx.recyclerview.widget.RecyclerView  import com.bumptech.glide.Glide  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.models.Game  class GameAdapter(  private var games: List<Game>,  private val onClick: (Game) -> Unit  ) : RecyclerView.Adapter<GameAdapter.GameViewHolder>() {  inner class GameViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  private val coverImage = itemView.findViewById<ImageView>(R.id.gameCover)  private val nameText = itemView.findViewById<TextView>(R.id.gameName)  fun bind(game: Game) {  nameText.text = game.name  Glide.with(itemView.context)  .load(game.cover)  .placeholder(R.drawable.cover\_placeholder)  .error(R.drawable.cover\_placeholder)  .into(coverImage)  itemView.setOnClickListener { onClick(game) }  }  }  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): GameViewHolder {  val view = LayoutInflater.from(parent.context).inflate(R.layout.item\_game, parent, false)  return GameViewHolder(view)  }  override fun onBindViewHolder(holder: GameViewHolder, position: Int) {  holder.bind(games[position])  }  override fun getItemCount(): Int = games.size  fun updateGames(newGames: List<Game>) {  games = newGames  notifyDataSetChanged()  }  } |

*Листинг 40 – SearchHistoryAdapter*

|  |
| --- |
| package com.rarmash.b4cklog.adapters  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.TextView  import androidx.recyclerview.widget.RecyclerView  class SearchHistoryAdapter(  private var items: List<String>,  private val onClick: (String) -> Unit  ) : RecyclerView.Adapter<SearchHistoryAdapter.HistoryViewHolder>() {  inner class HistoryViewHolder(view: View) : RecyclerView.ViewHolder(view) {  val text: TextView = view.findViewById(android.R.id.text1)  init {  view.setOnClickListener {  onClick(items[adapterPosition])  }  }  }  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): HistoryViewHolder {  val view = LayoutInflater.from(parent.context)  .inflate(android.R.layout.simple\_list\_item\_1, parent, false)  return HistoryViewHolder(view)  }  override fun getItemCount(): Int = items.size  override fun onBindViewHolder(holder: HistoryViewHolder, position: Int) {  holder.text.text = items[position]  }  fun update(newItems: List<String>) {  items = newItems  notifyDataSetChanged()  }  } |

*Листинг 41 – AccountSettingsFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import android.widget.EditText  import android.widget.Toast  import androidx.fragment.app.Fragment  import androidx.lifecycle.lifecycleScope  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.network.ApiClient  import kotlinx.coroutines.launch  import com.rarmash.b4cklog.models.User  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class AccountSettingsFragment : Fragment() {  private lateinit var emailEditText: EditText  private lateinit var passwordEditText: EditText  private lateinit var updateEmailButton: Button  private lateinit var updatePasswordButton: Button  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  val view = inflater.inflate(R.layout.fragment\_account\_settings, container, false)  emailEditText = view.findViewById(R.id.editTextEmail)  passwordEditText = view.findViewById(R.id.editTextPassword)  updateEmailButton = view.findViewById(R.id.buttonUpdateEmail)  updatePasswordButton = view.findViewById(R.id.buttonUpdatePassword)  loadUserProfile()  updateEmailButton.setOnClickListener {  val newEmail = emailEditText.text.toString()  if (newEmail.isNotBlank()) {  updateEmail(newEmail)  } else {  Toast.makeText(context, "Введите новый email", Toast.LENGTH\_SHORT).show()  }  }  updatePasswordButton.setOnClickListener {  val newPassword = passwordEditText.text.toString()  if (newPassword.isNotBlank()) {  updatePassword(newPassword)  } else {  Toast.makeText(context, "Введите новый пароль", Toast.LENGTH\_SHORT).show()  }  }  return view  }  private fun loadUserProfile() {  ApiClient.profileApi.getUserProfile().enqueue(object : Callback<User> {  override fun onResponse(call: Call<User>, response: Response<User>) {  if (response.isSuccessful) {  val user = response.body()  emailEditText.setText(user?.email ?: "")  } else {  Toast.makeText(context, "Ошибка загрузки профиля", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<User>, t: Throwable) {  Toast.makeText(context, "Ошибка сети: ${t.localizedMessage}", Toast.LENGTH\_SHORT).show()  }  })  }  private fun updateEmail(newEmail: String) {  lifecycleScope.launch {  try {  val response = ApiClient.profileApi.updateEmail(newEmail)  if (response.isSuccessful) {  Toast.makeText(context, "Email обновлён", Toast.LENGTH\_SHORT).show()  } else {  Toast.makeText(context, "Ошибка обновления: ${response.code()}", Toast.LENGTH\_SHORT).show()  }  } catch (e: Exception) {  Toast.makeText(context, "Сетевая ошибка: ${e.localizedMessage}", Toast.LENGTH\_SHORT).show()  }  }  }  private fun updatePassword(newPassword: String) {  lifecycleScope.launch {  try {  val response = ApiClient.profileApi.updatePassword(newPassword)  if (response.isSuccessful) {  Toast.makeText(context, "Пароль обновлён", Toast.LENGTH\_SHORT).show()  } else {  Toast.makeText(context, "Ошибка обновления: ${response.code()}", Toast.LENGTH\_SHORT).show()  }  } catch (e: Exception) {  Toast.makeText(context, "Сетевая ошибка: ${e.localizedMessage}", Toast.LENGTH\_SHORT).show()  }  }  }  } |

*Листинг 42 – EditGameFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.\*  import androidx.fragment.app.Fragment  import androidx.navigation.fragment.findNavController  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.models.Game  import com.rarmash.b4cklog.models.Platform  import com.rarmash.b4cklog.models.User  import com.rarmash.b4cklog.network.ApiClient  import com.rarmash.b4cklog.util.AuthPrefs  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class EditGameFragment : Fragment() {  private lateinit var nameEditText: EditText  private lateinit var summaryEditText: EditText  private lateinit var coverEditText: EditText  private lateinit var releaseDateEditText: EditText  private lateinit var platformContainer: LinearLayout  private lateinit var saveButton: Button  private var gameId: Int? = null  private var isAdmin = false  private var allPlatforms: List<Platform> = listOf()  private val selectedPlatforms = mutableSetOf<Platform>()  override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {  return inflater.inflate(R.layout.fragment\_edit\_game, container, false)  }  override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  nameEditText = view.findViewById(R.id.etGameName)  summaryEditText = view.findViewById(R.id.etGameSummary)  coverEditText = view.findViewById(R.id.etGameCover)  releaseDateEditText = view.findViewById(R.id.etGameReleaseDate)  platformContainer = view.findViewById(R.id.platforms\_checkbox\_container)  saveButton = view.findViewById(R.id.btnSaveGame)  gameId = arguments?.getInt("gameId")  ApiClient.profileApi.getUserProfile().enqueue(object : Callback<User> {  override fun onResponse(call: Call<User>, response: Response<User>) {  val user = response.body()  isAdmin = user?.isAdmin == true  if (!isAdmin) {  Toast.makeText(context, "Доступ только для администраторов", Toast.LENGTH\_SHORT).show()  findNavController().popBackStack()  return  }  loadPlatforms {  gameId?.let { loadGame(it) }  }  }  override fun onFailure(call: Call<User>, t: Throwable) {  Toast.makeText(context, "Ошибка сети", Toast.LENGTH\_SHORT).show()  }  })  saveButton.setOnClickListener {  if (validateInput()) {  if (gameId != null) {  updateGame()  } else {  createGame()  }  }  }  }  private fun loadPlatforms(onLoaded: () -> Unit) {  ApiClient.platformApi.getAllPlatforms().enqueue(object : Callback<List<Platform>> {  override fun onResponse(call: Call<List<Platform>>, response: Response<List<Platform>>) {  if (response.isSuccessful) {  allPlatforms = response.body() ?: emptyList()  platformContainer.removeAllViews()  for (platform in allPlatforms) {  val checkbox = CheckBox(requireContext()).apply {  text = "${platform.name} (${platform.releaseDate})"  setOnCheckedChangeListener { \_, isChecked ->  if (isChecked) selectedPlatforms.add(platform)  else selectedPlatforms.remove(platform)  }  }  platformContainer.addView(checkbox)  }  onLoaded()  }  }  override fun onFailure(call: Call<List<Platform>>, t: Throwable) {  Toast.makeText(context, "Ошибка загрузки платформ", Toast.LENGTH\_SHORT).show()  }  })  }  private fun loadGame(id: Int) {  ApiClient.gameApi.getGame(id).enqueue(object : Callback<Game> {  override fun onResponse(call: Call<Game>, response: Response<Game>) {  val game = response.body() ?: return  nameEditText.setText(game.name)  summaryEditText.setText(game.summary)  coverEditText.setText(game.cover)  releaseDateEditText.setText(game.releaseDate)  // Отметим выбранные платформы  selectedPlatforms.addAll(game.platforms)  for (i in 0 until platformContainer.childCount) {  val checkbox = platformContainer.getChildAt(i) as CheckBox  val platform = allPlatforms.getOrNull(i)  if (platform != null && game.platforms.any { it.id == platform.id }) {  checkbox.isChecked = true  }  }  }  override fun onFailure(call: Call<Game>, t: Throwable) {  Toast.makeText(context, "Ошибка загрузки игры", Toast.LENGTH\_SHORT).show()  }  })  }  private fun validateInput(): Boolean {  if (nameEditText.text.isBlank() || summaryEditText.text.isBlank() ||  coverEditText.text.isBlank() || releaseDateEditText.text.isBlank() ||  selectedPlatforms.isEmpty()) {  Toast.makeText(context, "Заполните все поля и выберите платформы", Toast.LENGTH\_SHORT).show()  return false  }  return true  }  private fun createGame() {  val token = AuthPrefs.getToken(requireContext()) ?: return  val game = Game(  id = 0,  name = nameEditText.text.toString(),  summary = summaryEditText.text.toString(),  cover = coverEditText.text.toString(),  releaseDate = releaseDateEditText.text.toString(),  platforms = selectedPlatforms.toMutableList()  )  ApiClient.gameApi.addGame(game).enqueue(object : Callback<Game> {  override fun onResponse(call: Call<Game>, response: Response<Game>) {  if (response.isSuccessful) {  Toast.makeText(context, "Игра добавлена", Toast.LENGTH\_SHORT).show()  findNavController().popBackStack()  } else {  Toast.makeText(context, "Ошибка: ${response.code()}", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<Game>, t: Throwable) {  Toast.makeText(context, "Сеть: ${t.localizedMessage}", Toast.LENGTH\_SHORT).show()  }  })  }  private fun updateGame() {  val token = AuthPrefs.getToken(requireContext()) ?: return  val game = Game(  id = gameId!!,  name = nameEditText.text.toString(),  summary = summaryEditText.text.toString(),  cover = coverEditText.text.toString(),  releaseDate = releaseDateEditText.text.toString(),  platforms = selectedPlatforms.toMutableList()  )  ApiClient.gameApi.updateGame(game).enqueue(object : Callback<Game> {  override fun onResponse(call: Call<Game>, response: Response<Game>) {  if (response.isSuccessful) {  Toast.makeText(context, "Игра обновлена", Toast.LENGTH\_SHORT).show()  findNavController().popBackStack()  } else {  Toast.makeText(context, "Ошибка при обновлении", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<Game>, t: Throwable) {  Toast.makeText(context, "Ошибка сети", Toast.LENGTH\_SHORT).show()  }  })  }  } |

*Листинг 43 – GameDetailFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.\*  import androidx.core.os.bundleOf  import androidx.fragment.app.Fragment  import androidx.lifecycle.lifecycleScope  import androidx.navigation.fragment.findNavController  import com.bumptech.glide.Glide  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.models.Game  import com.rarmash.b4cklog.models.ReviewResponse  import com.rarmash.b4cklog.models.User  import com.rarmash.b4cklog.network.ApiClient  import com.rarmash.b4cklog.util.SessionManager  import kotlinx.coroutines.delay  import kotlinx.coroutines.launch  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class GameDetailFragment : Fragment() {  private var currentListName: String? = null  private var currentUserId: Int? = null  private var gameId: Int = -1  override fun onCreateView(  inflater: LayoutInflater,  container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  return inflater.inflate(R.layout.fragment\_game\_detail, container, false)  }  override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)  parentFragmentManager.setFragmentResultListener("review\_added", viewLifecycleOwner) { \_, \_ ->  reloadFragment()  }  gameId = arguments?.getInt("gameId") ?: return  loadGameDetails(gameId)  loadGameRating(gameId)  loadUserReview(gameId)  val addToBacklogButton = view.findViewById<Button>(R.id.add\_to\_backlog\_button)  val removeFromBacklogButton = view.findViewById<ImageButton>(R.id.remove\_from\_backlog\_button)  val listTextView = view.findViewById<TextView>(R.id.game\_list\_info)  addToBacklogButton.setOnClickListener {  showAddToListDialog(gameId)  }  val reviewButton = view.findViewById<Button>(R.id.write\_review\_button)  reviewButton.setOnClickListener {  ReviewDialogFragment(gameId){  viewLifecycleOwner.lifecycleScope.launch {  delay(50)  loadGameRating(gameId)  loadUserReview(gameId)  }  }.show(childFragmentManager, "ReviewDialog")  }  loadCurrentUserAndCheckList(gameId, removeFromBacklogButton, listTextView)  }  private fun loadCurrentUserAndCheckList(gameId: Int, removeButton: ImageButton, listText: TextView) {  ApiClient.profileApi.getUserProfile().enqueue(object : Callback<User> {  override fun onResponse(call: Call<User>, response: Response<User>) {  if (!response.isSuccessful) return  val user = response.body() ?: return  currentUserId = user.id  val editButton = view?.findViewById<Button>(R.id.edit\_game\_button)  if (user.isAdmin) {  editButton?.visibility = View.VISIBLE  editButton?.setOnClickListener {  val bundle = bundleOf("gameId" to gameId)  findNavController().navigate(R.id.action\_gameDetailFragment\_to\_editGameFragment, bundle)  }  }  currentListName = when {  user.backlogWantToPlay.any { it.id == gameId } -> getString(R.string.want\_to\_play)  user.backlogPlaying.any { it.id == gameId } -> getString(R.string.playing)  user.backlogPlayed.any { it.id == gameId } -> getString(R.string.played)  user.backlogCompleted.any { it.id == gameId } -> getString(R.string.completed)  user.backlogCompleted100.any { it.id == gameId } -> getString(R.string.completed\_100)  else -> null  }  currentListName?.let { list ->  listText.text = "В списке: $list"  listText.visibility = View.VISIBLE  removeButton.visibility = View.VISIBLE  removeButton.setOnClickListener {  currentUserId?.let { uid ->  ApiClient.profileApi.removeGameFromAllLists(uid, gameId)  .enqueue(object : Callback<Void> {  override fun onResponse(call: Call<Void>, response: Response<Void>) {  if (response.isSuccessful) {  Toast.makeText(requireContext(), "Удалено из всех списков", Toast.LENGTH\_SHORT).show()  removeButton.visibility = View.GONE  listText.visibility = View.GONE  } else {  Toast.makeText(requireContext(), "Ошибка удаления", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<Void>, t: Throwable) {  Toast.makeText(requireContext(), "Ошибка сети", Toast.LENGTH\_SHORT).show()  }  })  }  }  }  }  override fun onFailure(call: Call<User>, t: Throwable) {  Toast.makeText(context, "Ошибка загрузки профиля", Toast.LENGTH\_SHORT).show()  }  })  }  private fun loadGameDetails(id: Int) {  ApiClient.gameApi.getGame(id).enqueue(object : Callback<Game> {  override fun onResponse(call: Call<Game>, response: Response<Game>) {  if (response.isSuccessful) {  val game = response.body() ?: return  view?.findViewById<TextView>(R.id.game\_name)?.text = game.name  view?.findViewById<TextView>(R.id.game\_summary)?.text = game.summary  view?.findViewById<TextView>(R.id.game\_release\_date)?.text = "Дата выхода: ${game.releaseDate}"  view?.findViewById<TextView>(R.id.game\_platforms)?.text =  "Платформы: ${game.platforms.joinToString(", ") { it.name }}"  val coverView = view?.findViewById<ImageView>(R.id.game\_cover)  Glide.with(requireContext())  .load(game.cover)  .placeholder(R.drawable.cover\_placeholder)  .error(R.drawable.cover\_placeholder)  .into(coverView!!)  }  }  override fun onFailure(call: Call<Game>, t: Throwable) {  Toast.makeText(context, "Ошибка загрузки игры", Toast.LENGTH\_SHORT).show()  }  })  }  private fun loadGameRating(gameId: Int) {  ApiClient.reviewApi.getAverageRating(gameId).enqueue(object : Callback<Double> {  override fun onResponse(call: Call<Double>, response: Response<Double>) {  if (response.isSuccessful) {  val rating = response.body() ?: 0.0  view?.findViewById<TextView>(R.id.game\_average\_rating)?.text =  "Средняя оценка: %.1f".format(rating)  }  }  override fun onFailure(call: Call<Double>, t: Throwable) {  Toast.makeText(context, "Ошибка загрузки рейтинга", Toast.LENGTH\_SHORT).show()  }  })  }  private fun loadUserReview(gameId: Int) {  val userId = SessionManager.userId ?: return  ApiClient.reviewApi.getUserReview(userId, gameId)  .enqueue(object : Callback<ReviewResponse> {  override fun onResponse(call: Call<ReviewResponse>, response: Response<ReviewResponse>) {  if (response.isSuccessful) {  val review = response.body() ?: return  val reviewTextView = view?.findViewById<TextView>(R.id.user\_review\_text)  reviewTextView?.visibility = View.VISIBLE  reviewTextView?.text = "Твоя оценка: ${review.rating}/5\nКомментарий: ${review.comment ?: "нет"}"  }  }  override fun onFailure(call: Call<ReviewResponse>, t: Throwable) {  // Не показываем тост специально — это не критичная информация  }  })  }  private fun showAddToListDialog(gameId: Int) {  val listNames = arrayOf(  getString(R.string.want\_to\_play),  getString(R.string.playing),  getString(R.string.played),  getString(R.string.completed),  getString(R.string.completed\_100),  )  val listKeys = arrayOf(  "wantToPlay",  "playing",  "played",  "completed",  "completed100"  )  androidx.appcompat.app.AlertDialog.Builder(requireContext())  .setTitle("Добавить в список")  .setItems(listNames) { \_, which ->  val listKey = listKeys[which]  addGameToBacklog(gameId, listKey)  }  .setNegativeButton("Отмена", null)  .show()  }  private fun addGameToBacklog(gameId: Int, listName: String) {  ApiClient.profileApi.getUserProfile().enqueue(object : Callback<User> {  override fun onResponse(call: Call<User>, response: Response<User>) {  if (response.isSuccessful) {  val userId = response.body()?.id ?: return  ApiClient.profileApi.addGameToList(userId, gameId, listName)  .enqueue(object : Callback<Void> {  override fun onResponse(call: Call<Void>, response: Response<Void>) {  if (response.isSuccessful) {  Toast.makeText(requireContext(), "Добавлено в список", Toast.LENGTH\_SHORT).show()  view?.post {  val removeFromBacklogButton = view?.findViewById<ImageButton>(R.id.remove\_from\_backlog\_button)  val listTextView = view?.findViewById<TextView>(R.id.game\_list\_info)  if (removeFromBacklogButton != null && listTextView != null) {  loadCurrentUserAndCheckList(gameId, removeFromBacklogButton, listTextView)  }  }  } else {  Toast.makeText(requireContext(), "Ошибка добавления", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<Void>, t: Throwable) {  Toast.makeText(requireContext(), "Ошибка сети", Toast.LENGTH\_SHORT).show()  }  })  }  }  override fun onFailure(call: Call<User>, t: Throwable) {  Toast.makeText(requireContext(), "Не удалось получить профиль", Toast.LENGTH\_SHORT).show()  }  })  }  private fun reloadFragment() {  // Сохраняем аргументы  val args = bundleOf("gameId" to gameId)  // Убираем текущий из стека  findNavController().popBackStack(R.id.gameDetailFragment, true)  // Навигируем заново  findNavController().navigate(R.id.gameDetailFragment, args)  }  } |

*Листинг 44 – HomeFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import android.widget.Toast  import androidx.core.os.bundleOf  import androidx.fragment.app.Fragment  import androidx.navigation.Navigation  import androidx.recyclerview.widget.LinearLayoutManager  import androidx.recyclerview.widget.RecyclerView  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.adapters.GameAdapter  import com.rarmash.b4cklog.models.Game  import com.rarmash.b4cklog.network.ApiClient  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class HomeFragment : Fragment() {  private lateinit var popularAdapter: GameAdapter  private lateinit var latestAdapter: GameAdapter  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_home, container, false)  val popularRecyclerView = view.findViewById<RecyclerView>(R.id.popularGamesRecyclerView)  val latestRecyclerView = view.findViewById<RecyclerView>(R.id.latestReleaseRecyclerView)  popularRecyclerView.layoutManager =  LinearLayoutManager(context, LinearLayoutManager.HORIZONTAL, false)  latestRecyclerView.layoutManager =  LinearLayoutManager(context, LinearLayoutManager.HORIZONTAL, false)  popularAdapter = GameAdapter(emptyList()) { game ->  val bundle = bundleOf("gameId" to game.id)  Navigation.findNavController(requireView()).navigate(R.id.action\_homeFragment\_to\_gameDetailFragment, bundle)  }  latestAdapter = GameAdapter(emptyList()) { game ->  val bundle = bundleOf("gameId" to game.id)  Navigation.findNavController(requireView()).navigate(R.id.action\_homeFragment\_to\_gameDetailFragment, bundle)  }  popularRecyclerView.adapter = popularAdapter  latestRecyclerView.adapter = latestAdapter  // Кнопка полного каталога  // view.findViewById<Button>(R.id.fullCatalogButton).setOnClickListener {  // Navigation.findNavController(view).navigate(R.id.action\_homeFragment\_to\_catalogFragment)  // }  loadGames()  return view  }  private fun loadGames() {  val api = ApiClient.gameApi  api.getAllGames().enqueue(object : Callback<List<Game>> {  override fun onResponse(call: Call<List<Game>>, response: Response<List<Game>>) {  if (response.isSuccessful) {  val games = response.body() ?: emptyList()  val popularGames = games.take(10)  val latestGames = games.sortedByDescending { it.releaseDate }.take(10)  popularAdapter.updateGames(popularGames)  latestAdapter.updateGames(latestGames)  }  }  override fun onFailure(call: Call<List<Game>>, t: Throwable) {  Toast.makeText(context, "Ошибка загрузки игр", Toast.LENGTH\_SHORT).show()  }  })  }  } |

*Листинг 45 – ProfileFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import android.widget.TextView  import android.widget.Toast  import androidx.core.os.bundleOf  import androidx.fragment.app.Fragment  import androidx.navigation.Navigation  import androidx.navigation.fragment.findNavController  import androidx.recyclerview.widget.LinearLayoutManager  import androidx.recyclerview.widget.RecyclerView  import com.google.android.material.floatingactionbutton.FloatingActionButton  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.adapters.GameAdapter  import com.rarmash.b4cklog.models.Game  import com.rarmash.b4cklog.models.User  import com.rarmash.b4cklog.network.ApiClient  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class ProfileFragment : Fragment() {  private lateinit var userName: TextView  private lateinit var userFullNameAge: TextView  private lateinit var userEmail: TextView  private lateinit var recyclerWantToPlay: RecyclerView  private lateinit var recyclerPlaying: RecyclerView  private lateinit var recyclerPlayed: RecyclerView  private lateinit var recyclerCompleted: RecyclerView  private lateinit var recyclerCompleted100: RecyclerView  private lateinit var adapterWantToPlay: GameAdapter  private lateinit var adapterPlaying: GameAdapter  private lateinit var adapterPlayed: GameAdapter  private lateinit var adapterCompleted: GameAdapter  private lateinit var adapterCompleted100: GameAdapter  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_profile, container, false)  userName = view.findViewById(R.id.userName)  userFullNameAge = view.findViewById(R.id.userFullNameAge)  userEmail = view.findViewById(R.id.userEmail)  recyclerWantToPlay = view.findViewById(R.id.recyclerWantToPlay)  recyclerPlaying = view.findViewById(R.id.recyclerPlaying)  recyclerPlayed = view.findViewById(R.id.recyclerPlayed)  recyclerCompleted = view.findViewById(R.id.recyclerCompleted)  recyclerCompleted100 = view.findViewById(R.id.recyclerCompleted100)  // Настройка RecyclerView и адаптеров  adapterWantToPlay = GameAdapter(emptyList(), ::onGameClick)  adapterPlaying = GameAdapter(emptyList(), ::onGameClick)  adapterPlayed = GameAdapter(emptyList(), ::onGameClick)  adapterCompleted = GameAdapter(emptyList(), ::onGameClick)  adapterCompleted100 = GameAdapter(emptyList(), ::onGameClick)  recyclerWantToPlay.layoutManager = LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)  recyclerPlaying.layoutManager = LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)  recyclerPlayed.layoutManager = LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)  recyclerCompleted.layoutManager = LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)  recyclerCompleted100.layoutManager = LinearLayoutManager(requireContext(), LinearLayoutManager.HORIZONTAL, false)  recyclerWantToPlay.adapter = adapterWantToPlay  recyclerPlaying.adapter = adapterPlaying  recyclerPlayed.adapter = adapterPlayed  recyclerCompleted.adapter = adapterCompleted  recyclerCompleted100.adapter = adapterCompleted100  val settingsButton = view.findViewById<FloatingActionButton>(R.id.settingsButton)  settingsButton.setOnClickListener {  Navigation.findNavController(view).navigate(R.id.action\_profileFragment\_to\_settingsFragment)  }  loadProfile()  return view  }  private fun loadProfile() {  ApiClient.profileApi.getUserProfile().enqueue(object : Callback<User> {  override fun onResponse(call: Call<User>, response: Response<User>) {  if (response.isSuccessful && response.body() != null) {  val user = response.body()!!  userName.text = user.username  userFullNameAge.text = "${user.firstName} ${user.lastName}, ${user.age}"  userEmail.text = user.email  val editButton = view?.findViewById<Button>(R.id.edit\_game\_button)  if (user.isAdmin) {  editButton?.visibility = View.VISIBLE  editButton?.setOnClickListener {  findNavController().navigate(R.id.action\_profileFragment\_to\_editGameFragment)  }  }  adapterWantToPlay.updateGames(user.backlogWantToPlay)  adapterPlaying.updateGames(user.backlogPlaying)  adapterPlayed.updateGames(user.backlogPlayed)  adapterCompleted.updateGames(user.backlogCompleted)  adapterCompleted100.updateGames(user.backlogCompleted100)  } else {  Toast.makeText(requireContext(), "Ошибка загрузки профиля", Toast.LENGTH\_SHORT).show()  }  }  override fun onFailure(call: Call<User>, t: Throwable) {  Toast.makeText(requireContext(), "Сеть: ${t.localizedMessage}", Toast.LENGTH\_SHORT).show()  }  })  }  private fun onGameClick(game: Game) {  val bundle = bundleOf("gameId" to game.id)  Navigation.findNavController(requireView()).navigate(R.id.action\_profileFragment\_to\_gameDetailFragment, bundle)  }  } |

*Листинг 46 – ReviewDialogFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.app.AlertDialog  import android.app.Dialog  import android.os.Bundle  import android.view.LayoutInflater  import android.widget.Toast  import androidx.fragment.app.DialogFragment  import com.rarmash.b4cklog.databinding.DialogReviewBinding  import com.rarmash.b4cklog.models.ReviewRequest  import com.rarmash.b4cklog.network.ApiClient  import com.rarmash.b4cklog.util.SessionManager  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class ReviewDialogFragment(  private val gameId: Int,  private val onReviewSubmitted: (() -> Unit)? = null  ) : DialogFragment() {  override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {  val binding = DialogReviewBinding.inflate(LayoutInflater.from(context))  return AlertDialog.Builder(requireContext())  .setTitle("Оставить отзыв")  .setView(binding.root)  .setPositiveButton("Отправить") { \_, \_ ->  val rating = binding.ratingBar.rating.toInt()  val comment = binding.reviewText.text.toString()  val userId = SessionManager.userId  if (userId == null) {  if (isAdded) {  Toast.makeText(requireContext(), "Пользователь не авторизован", Toast.LENGTH\_SHORT).show()  }  return@setPositiveButton  }  val review = ReviewRequest(userId, gameId, rating, comment)  ApiClient.reviewApi.submitReview(review).enqueue(object : Callback<Void> {  override fun onResponse(call: Call<Void>, response: Response<Void>) {  if (!isAdded) return  // всегда показываем тост  Toast.makeText(  requireContext(),  "Отзыв отправлен",  Toast.LENGTH\_LONG  ).show()  // отдаем сигнал старшему фрагменту  parentFragmentManager.setFragmentResult("review\_added", Bundle())  onReviewSubmitted?.invoke()  dismiss()  }  override fun onFailure(call: Call<Void>, t: Throwable) {  if (!isAdded) return  Toast.makeText(requireContext(), "Ошибка сети при отправке: ${t.message}", Toast.LENGTH\_SHORT).show()  }  })  }  .setNegativeButton("Отмена", null)  .create()  }  } |

*Листинг 47 – SearchFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.content.Context  import android.os.Bundle  import android.os.Handler  import android.os.Looper  import android.util.Log  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.view.inputmethod.InputMethodManager  import android.widget.EditText  import android.widget.Toast  import androidx.core.os.bundleOf  import androidx.fragment.app.Fragment  import androidx.navigation.Navigation  import androidx.recyclerview.widget.LinearLayoutManager  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.adapters.GameAdapter  import com.rarmash.b4cklog.adapters.SearchHistoryAdapter  import com.rarmash.b4cklog.databinding.FragmentSearchBinding  import com.rarmash.b4cklog.models.Game  import com.rarmash.b4cklog.network.ApiClient  import org.json.JSONArray  import retrofit2.Call  import retrofit2.Callback  import retrofit2.Response  class SearchFragment : Fragment() {  private lateinit var binding: FragmentSearchBinding  private lateinit var gameAdapter: GameAdapter  private lateinit var historyAdapter: SearchHistoryAdapter  private var lastQuery: String? = null  private val handler = Handler(Looper.getMainLooper())  private var searchRunnable: Runnable? = null  private val SEARCH\_DELAY = 2000L  private val HISTORY\_PREF = "search\_history"  private val MAX\_HISTORY\_SIZE = 10  private var isShowingHistory = true  private var restoringState = false  override fun onCreateView(  inflater: LayoutInflater,  container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  binding = FragmentSearchBinding.inflate(inflater, container, false)  // Инициализация адаптеров  gameAdapter = GameAdapter(emptyList()) { game ->  val bundle = bundleOf("gameId" to game.id)  Navigation.findNavController(requireView()).navigate(R.id.action\_searchFragment\_to\_gameDetailFragment, bundle)  }  historyAdapter = SearchHistoryAdapter(loadHistory()) { selectedQuery ->  binding.searchView.setQuery(selectedQuery, false)  saveToHistory(selectedQuery)  fetchGames(selectedQuery)  hideHistory()  }  binding.recyclerView.layoutManager = LinearLayoutManager(context)  binding.recyclerView.adapter = gameAdapter  binding.searchView.clearFocus()  // Подсказка и цвет текста в поисковой строке  binding.searchView.findViewById<EditText>(androidx.appcompat.R.id.search\_src\_text)  .setHintTextColor(android.graphics.Color.DKGRAY)  binding.searchView.queryHint = "Искать игру"  // Восстановление состояния, если есть сохранённый запрос  savedInstanceState?.let {  restoringState = true  lastQuery = it.getString("lastQuery")  binding.searchView.setQuery(lastQuery, false)  lastQuery?.let { fetchGames(it) }  }  return binding.root  }  override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)  binding.searchView.setOnQueryTextFocusChangeListener { \_, hasFocus ->  if (hasFocus && loadHistory().isNotEmpty() && !restoringState) {  showHistory()  } else {  hideHistory()  }  }  binding.searchView.setOnQueryTextListener(object : androidx.appcompat.widget.SearchView.OnQueryTextListener {  override fun onQueryTextSubmit(query: String?): Boolean {  searchRunnable?.let { handler.removeCallbacks(it) }  query?.let {  saveToHistory(it)  fetchGames(it)  hideHistory()  binding.searchView.clearFocus()  binding.searchView.postDelayed({  hideKeyboard()  }, 100)  }  return true  }  override fun onQueryTextChange(newText: String?): Boolean {  searchRunnable?.let { handler.removeCallbacks(it) }  if (newText.isNullOrEmpty()) {  if (loadHistory().isNotEmpty()) {  showHistory()  }  binding.searchView.clearFocus()  binding.searchView.postDelayed({  hideKeyboard()  }, 100)  return false  }  searchRunnable = Runnable {  saveToHistory(newText)  fetchGames(newText)  hideHistory()  }  handler.postDelayed(searchRunnable!!, SEARCH\_DELAY)  return false  }  })  binding.retryButton.setOnClickListener {  lastQuery?.let { fetchGames(it) }  }  binding.clearHistoryButton.setOnClickListener {  clearHistory()  hideHistory()  }  restoringState = false  }  private fun fetchGames(searchQuery: String) {  if (searchQuery == lastQuery) return  lastQuery = searchQuery  showLoadingState()  ApiClient.gameApi.searchGames(searchQuery).enqueue(object : Callback<List<Game>> {  override fun onResponse(call: Call<List<Game>>, response: Response<List<Game>>) {  Log.d("SearchFragment", "Код ответа: ${response.code()}")  if (response.isSuccessful) {  val games = response.body() ?: emptyList()  gameAdapter.updateGames(games)  binding.progressBar.visibility = View.GONE  updatePlaceholder(games.isEmpty())  } else {  Log.e("SearchFragment", "Ошибка: ${response.code()} ${response.errorBody()?.string() ?: "Нет инфо"}")  showErrorPlaceholder()  }  }  override fun onFailure(call: Call<List<Game>>, t: Throwable) {  binding.progressBar.visibility = View.GONE  Toast.makeText(context, "Ошибка сети: ${t.message}", Toast.LENGTH\_SHORT).show()  showErrorPlaceholder()  }  })  }  private fun showLoadingState() {  binding.recyclerView.visibility = View.GONE  binding.placeholderLayout.visibility = View.GONE  binding.progressBar.visibility = View.VISIBLE  }  private fun updatePlaceholder(isEmpty: Boolean) {  binding.recyclerView.visibility = if (isEmpty) View.GONE else View.VISIBLE  binding.placeholderLayout.visibility = if (isEmpty) View.VISIBLE else View.GONE  binding.placeholderText.text = "Нет результатов"  binding.retryButton.visibility = View.GONE  }  private fun showErrorPlaceholder() {  binding.progressBar.visibility = View.GONE  binding.recyclerView.visibility = View.GONE  binding.placeholderLayout.visibility = View.VISIBLE  binding.placeholderText.text = "Ошибка загрузки данных"  binding.retryButton.visibility = View.VISIBLE  }  private fun hideKeyboard() {  val imm = requireContext().getSystemService(Context.INPUT\_METHOD\_SERVICE) as InputMethodManager  imm.hideSoftInputFromWindow(binding.searchView.windowToken, 0)  }  private fun saveToHistory(query: String) {  val prefs = requireContext().getSharedPreferences("search\_prefs", Context.MODE\_PRIVATE)  val history = loadHistory().toMutableList()  history.remove(query)  history.add(0, query)  val limited = history.take(MAX\_HISTORY\_SIZE)  val jsonArray = JSONArray()  for (item in limited) {  jsonArray.put(item)  }  prefs.edit().putString(HISTORY\_PREF, jsonArray.toString()).apply()  }  private fun loadHistory(): List<String> {  val prefs = requireContext().getSharedPreferences("search\_prefs", Context.MODE\_PRIVATE)  val json = prefs.getString(HISTORY\_PREF, null) ?: return emptyList()  return try {  val jsonArray = JSONArray(json)  List(jsonArray.length()) { i -> jsonArray.getString(i) }  } catch (e: Exception) {  emptyList()  }  }  private fun clearHistory() {  val prefs = requireContext().getSharedPreferences("search\_prefs", Context.MODE\_PRIVATE)  prefs.edit().remove(HISTORY\_PREF).apply()  }  private fun showHistory() {  val history = loadHistory()  if (history.isNotEmpty()) {  isShowingHistory = true  binding.recyclerView.adapter = historyAdapter  historyAdapter.update(history)  binding.recyclerView.visibility = View.VISIBLE  binding.clearHistoryButton.visibility = View.VISIBLE  binding.placeholderLayout.visibility = View.GONE  }  }  private fun hideHistory() {  isShowingHistory = false  binding.recyclerView.adapter = gameAdapter  binding.clearHistoryButton.visibility = View.GONE  }  override fun onSaveInstanceState(outState: Bundle) {  super.onSaveInstanceState(outState)  outState.putString("lastQuery", lastQuery)  }  } |

*Листинг 48 – SettingsFragment*

|  |
| --- |
| package com.rarmash.b4cklog.fragments  import android.content.Intent  import android.content.SharedPreferences  import android.os.Bundle  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import androidx.appcompat.app.AppCompatDelegate  import androidx.appcompat.widget.SwitchCompat  import androidx.fragment.app.Fragment  import androidx.navigation.Navigation  import com.rarmash.b4cklog.R  import com.rarmash.b4cklog.activities.WelcomeActivity  import com.rarmash.b4cklog.util.AuthPrefs  class SettingsFragment : Fragment() {  private lateinit var themeSwitch: SwitchCompat  private lateinit var sharedPreferences: SharedPreferences  override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {  val view = inflater.inflate(R.layout.fragment\_settings, container, false)  sharedPreferences = requireContext().getSharedPreferences("settings", 0)  themeSwitch = view.findViewById(R.id.themeSwitch)  // Установка начального состояния  val isDarkMode = sharedPreferences.getBoolean("dark\_mode", false)  themeSwitch.isChecked = isDarkMode  val accountSettingsButton = view.findViewById<View>(R.id.account)  accountSettingsButton.setOnClickListener {  Navigation.findNavController(view).navigate(R.id.action\_settingsFragment\_to\_accountSettingsFragment)  }  // Обработчик переключения  themeSwitch.setOnCheckedChangeListener { \_, isChecked ->  sharedPreferences.edit().putBoolean("dark\_mode", isChecked).apply()  AppCompatDelegate.setDefaultNightMode(  if (isChecked) AppCompatDelegate.MODE\_NIGHT\_YES  else AppCompatDelegate.MODE\_NIGHT\_NO  )  // Перезапускаем активность  requireActivity().recreate()  }  val logoutButton = view.findViewById<View>(R.id.logout)  logoutButton.setOnClickListener {  // Удаляем токен  AuthPrefs.clearToken(requireContext())  // Запускаем WelcomeActivity  val intent = Intent(requireContext(), WelcomeActivity::class.java)  intent.flags = Intent.FLAG\_ACTIVITY\_NEW\_TASK or Intent.FLAG\_ACTIVITY\_CLEAR\_TASK  startActivity(intent)  }  return view  }  } |

*Листинг 49 – Game*

|  |
| --- |
| package com.rarmash.b4cklog.models  data class Game(  val id: Int,  var name: String,  var summary: String,  var cover: String,  var releaseDate: String,  var platforms: MutableList<Platform>  ) |

*Листинг 50 – Platform*

|  |
| --- |
| package com.rarmash.b4cklog.models  data class Platform (  val id: Int,  var name: String,  var releaseDate: String  )  data class PlatformRequest(  var name: String,  var releaseDate: String  ) |

*Листинг 51 – Review*

|  |
| --- |
| package com.rarmash.b4cklog.models  data class ReviewRequest(  val userId: Int,  val gameId: Int,  val rating: Int,  val comment: String?  )  data class ReviewResponse(  val rating: Int,  val comment: String?  ) |

*Листинг 52 – User*

|  |
| --- |
| package com.rarmash.b4cklog.models  data class User(  val id: Int,  val username: String,  val firstName: String,  val lastName: String,  val email: String,  val age: String,  val isAdmin: Boolean,  val backlogWantToPlay: List<Game>,  val backlogPlaying: List<Game>,  val backlogPlayed: List<Game>,  val backlogCompleted: List<Game>,  val backlogCompleted100: List<Game>  ) |

*Листинг 53 – ApiClient*

|  |
| --- |
| package com.rarmash.b4cklog.network  import android.content.Context  import com.rarmash.b4cklog.util.AuthPrefs  import okhttp3.Interceptor  import okhttp3.OkHttpClient  import okhttp3.logging.HttpLoggingInterceptor  import retrofit2.Retrofit  import retrofit2.converter.gson.GsonConverterFactory  object ApiClient {  private const val BASE\_URL = "http://10.0.2.2:8080/"  lateinit var appContext: Context  fun init(context: Context) {  appContext = context.applicationContext  }  private val logging = HttpLoggingInterceptor().apply {  level = HttpLoggingInterceptor.Level.BODY  }  private val authInterceptor = Interceptor { chain ->  val requestBuilder = chain.request().newBuilder()  val token = AuthPrefs.getToken(appContext)  if (!token.isNullOrEmpty()) {  requestBuilder.addHeader("Authorization", "Bearer $token")  }  chain.proceed(requestBuilder.build())  }  private val client = OkHttpClient.Builder()  .addInterceptor(logging)  .addInterceptor(authInterceptor)  .build()  private val retrofit = Retrofit.Builder()  .baseUrl(BASE\_URL)  .addConverterFactory(GsonConverterFactory.create())  .client(client)  .build()  val authApi: AuthApi = retrofit.create(AuthApi::class.java)  val profileApi: ProfileApi = retrofit.create(ProfileApi::class.java)  val gameApi: GameApi = retrofit.create(GameApi::class.java)  val platformApi: PlatformApi = retrofit.create(PlatformApi::class.java)  val reviewApi: ReviewApi = retrofit.create(ReviewApi::class.java)  } |

*Листинг 54 – AuthApi*

|  |
| --- |
| package com.rarmash.b4cklog.network  import retrofit2.http.Body  import retrofit2.http.POST  import retrofit2.Call  interface AuthApi {  @POST("auth/login")  fun login(@Body request: LoginRequest): Call<AuthResponse>  @POST("auth/register")  fun register(@Body request: RegisterRequest): Call<AuthResponse>  } |

*Листинг 55 – GameApi*

|  |
| --- |
| package com.rarmash.b4cklog.network  import com.rarmash.b4cklog.models.Game  import retrofit2.Call  import retrofit2.http.Body  import retrofit2.http.GET  import retrofit2.http.POST  import retrofit2.http.Path  import retrofit2.http.Query  interface GameApi {  @GET("/games/get/all")  fun getAllGames(): Call<List<Game>>  @GET("/games/get/{id}")  fun getGame(@Path("id") id: Int): Call<Game>  @POST("/games/add")  fun addGame(  @Body game: Game  ): Call<Game>  @POST("/games/update")  fun updateGame(  @Body game: Game  ): Call<Game>  @GET("games/search")  fun searchGames(@Query("q") query: String): Call<List<Game>>  } |

*Листинг 56 – PlatformApi*

|  |
| --- |
| package com.rarmash.b4cklog.network  import com.rarmash.b4cklog.models.Platform  import com.rarmash.b4cklog.models.PlatformRequest  import retrofit2.Call  import retrofit2.http.Body  import retrofit2.http.DELETE  import retrofit2.http.GET  import retrofit2.http.Header  import retrofit2.http.POST  import retrofit2.http.Path  interface PlatformApi {  @GET("/platforms/get/all")  fun getAllPlatforms(): Call<List<Platform>>  @GET("/platforms/get/{id}")  fun getPlatform(@Path("id") id: Int): Call<Platform>  @POST("/platforms/add")  fun addPlatform(  @Body platform: PlatformRequest  ): Call<Platform>  @DELETE("/platforms/delete/{id}")  fun deletePlatform(  @Path("id") id: Int  ): Call<Void>  } |

*Листинг 57 – ProfileApi*

|  |
| --- |
| package com.rarmash.b4cklog.network  import com.rarmash.b4cklog.models.User  import retrofit2.Call  import retrofit2.Response  import retrofit2.http.DELETE  import retrofit2.http.GET  import retrofit2.http.PATCH  import retrofit2.http.POST  import retrofit2.http.Path  import retrofit2.http.Query  interface ProfileApi {  @GET("users/profile")  fun getUserProfile(): Call<User>  @POST("users/{userId}/addGameToList")  fun addGameToList(  @Path("userId") userId: Int,  @Query("gameId") gameId: Int,  @Query("listName") listName: String  ): Call<Void>  @DELETE("/users/{userId}/removeGameFromAllLists")  fun removeGameFromAllLists(  @Path("userId") userId: Int,  @Query("gameId") gameId: Int  ): Call<Void>  @PATCH("users/updateEmail")  suspend fun updateEmail(@Query("newEmail") newEmail: String): Response<Unit>  @PATCH("users/updatePassword")  suspend fun updatePassword(@Query("newPassword") newPassword: String): Response<Unit>  } |

*Листинг 58 – ReviewApi*

|  |
| --- |
| package com.rarmash.b4cklog.network  import com.rarmash.b4cklog.models.ReviewRequest  import com.rarmash.b4cklog.models.ReviewResponse  import retrofit2.Call  import retrofit2.http.Body  import retrofit2.http.GET  import retrofit2.http.POST  import retrofit2.http.Path  interface ReviewApi {  @GET("/reviews/game/{gameId}/average")  fun getAverageRating(@Path("gameId") gameId: Int): Call<Double>  @POST("/reviews/add")  fun submitReview(@Body reviewRequest: ReviewRequest): Call<Void>  @GET("/reviews/user/{userId}/game/{gameId}")  fun getUserReview(  @Path("userId") userId: Int,  @Path("gameId") gameId: Int  ): Call<ReviewResponse>  } |

*Листинг 59 – AuthPrefs*

|  |
| --- |
| package com.rarmash.b4cklog.util  import android.content.Context  object AuthPrefs {  private const val PREFS\_NAME = "auth\_prefs"  private const val KEY\_TOKEN = "token"  fun saveToken(context: Context, token: String) {  val prefs = context.getSharedPreferences(PREFS\_NAME, Context.MODE\_PRIVATE)  prefs.edit()  .putString(KEY\_TOKEN, token)  .apply()  }  fun getToken(context: Context): String? {  val prefs = context.getSharedPreferences(PREFS\_NAME, Context.MODE\_PRIVATE)  return prefs.getString(KEY\_TOKEN, null)  }  fun clearToken(context: Context) {  val prefs = context.getSharedPreferences(PREFS\_NAME, Context.MODE\_PRIVATE)  prefs.edit().remove(KEY\_TOKEN).apply()  }  } |

*Листинг 60 – SessionManager*

|  |
| --- |
| package com.rarmash.b4cklog.util  object SessionManager {  var userId: Int? = null  } |

*Листинг 61 – App*

|  |
| --- |
| package com.rarmash.b4cklog  import android.app.Application  import com.rarmash.b4cklog.network.ApiClient  class App : Application() {  override fun onCreate() {  super.onCreate()  ApiClient.init(this)  }  } |

Приложение В

*Листинг 62 – activity\_login*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayout  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:android="http://schemas.android.com/apk/res/android"  android:id="@+id/main"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:background="?attr/android:colorBackground">  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_gravity="center"  android:orientation="vertical"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintTop\_toTopOf="parent">  <TextView  android:id="@+id/heading"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:gravity="center"  android:text="@string/login\_label"  android:textColor="?attr/colorOnBackground"  android:textStyle="bold"  android:textSize="28sp"/>  <EditText  android:id="@+id/editTextUsername"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="160dp"  android:hint="@string/username"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="text"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/heading" />  <EditText  android:id="@+id/editTextPassword"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:hint="@string/password"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="textPassword"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editTextUsername" />  <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="100sp">  <Button  android:id="@+id/login\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:backgroundTint="@color/orange"  android:gravity="center"  android:text="@string/login"  android:textColor="@color/black"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  </androidx.constraintlayout.widget.ConstraintLayout>  </LinearLayout>  </androidx.constraintlayout.widget.ConstraintLayout> |

*Листинг 63 – activity\_main*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/drawer\_layout"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".activities.MainActivity"  android:background="?attr/android:colorBackground">  <RelativeLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:paddingTop="?attr/actionBarSize"  android:fitsSystemWindows="true">  <androidx.fragment.app.FragmentContainerView  android:id="@+id/fragment\_container"  android:name="androidx.navigation.fragment.NavHostFragment"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_marginBottom="?attr/actionBarSize"  app:navGraph="@navigation/nav\_graph"  app:defaultNavHost="true" />  <View  android:layout\_width="match\_parent"  android:layout\_height="1dp"  android:background="@drawable/divider"  android:layout\_above="@id/bottomNavigationView" />  <com.google.android.material.bottomnavigation.BottomNavigationView  android:id="@+id/bottomNavigationView"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  style="@style/BottomNavigationStyle"  app:itemBackground="@drawable/bottom\_nav\_item\_background"  app:labelVisibilityMode="unlabeled"  app:menu="@menu/bottom\_nav\_menu" />  </RelativeLayout>  </androidx.drawerlayout.widget.DrawerLayout> |

*Листинг 64 – activity\_signup*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayout  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:android="http://schemas.android.com/apk/res/android"  android:id="@+id/main"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:background="?attr/android:colorBackground">  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_gravity="center"  android:orientation="vertical"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintTop\_toTopOf="parent">  <TextView  android:id="@+id/heading"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:gravity="center"  android:text="@string/signup\_label"  android:textColor="?attr/colorOnBackground"  android:textStyle="bold"  android:textSize="28sp"/>  <EditText  android:id="@+id/editTextUsername"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="50dp"  android:hint="@string/username"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="text"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/heading" />  <EditText  android:id="@+id/editTextEmail"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:hint="@string/email"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="textEmailAddress"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editTextUsername" />  <EditText  android:id="@+id/editTextPassword"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:hint="@string/password"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="textPassword"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editTextEmail" />  <EditText  android:id="@+id/editTextFirstName"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:hint="@string/firstName"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="text"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editTextPassword" />  <EditText  android:id="@+id/editTextLastName"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:hint="@string/lastName"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="text"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editTextFirstName" />  <EditText  android:id="@+id/editTextAge"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:hint="@string/age"  android:textColor="?attr/colorOnBackground"  android:textColorHint="?attr/colorOnBackground"  android:inputType="number"  android:paddingStart="16dp"  android:paddingEnd="16dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/editTextLastName" />  <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="100sp">  <Button  android:id="@+id/signup\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:backgroundTint="@color/orange"  android:gravity="center"  android:text="@string/signup"  android:textColor="@color/black"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  </androidx.constraintlayout.widget.ConstraintLayout>  </LinearLayout>  </androidx.constraintlayout.widget.ConstraintLayout> |

*Листинг 65 – activity\_welcome*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:id="@+id/main"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:background="?attr/android:colorBackground">  <TextView  android:id="@+id/heading"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginTop="60sp"  android:gravity="center"  android:text="@string/app\_name"  android:textColor="?attr/colorOnBackground"  android:textStyle="bold"  android:textSize="28sp"/>  <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_gravity="center">  <Button  android:id="@+id/login\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/login"  android:backgroundTint="@color/orange"  android:textColor="@color/black"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent">  </Button>  <Button  android:id="@+id/signup\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/signup"  android:backgroundTint="@color/orange"  android:textColor="@color/black"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/login\_button">  </Button>  </androidx.constraintlayout.widget.ConstraintLayout>  </FrameLayout> |

*Листинг 66 – dialog\_review*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:padding="16dp"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <RatingBar  android:id="@+id/rating\_bar"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:numStars="5"  android:stepSize="1.0" />  <EditText  android:id="@+id/review\_text"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="@string/your\_review"  android:minHeight="100dp"  android:gravity="top"  android:inputType="textMultiLine"  android:layout\_marginTop="12dp"/>  </LinearLayout> |

*Листинг 67 – fragment\_edit\_game*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:padding="16dp"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <!-- Email -->  <EditText  android:id="@+id/editTextEmail"  android:hint="@string/new\_email"  android:inputType="textEmailAddress"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  <Button  android:id="@+id/buttonUpdateEmail"  android:text="@string/update\_email"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  <!-- Пароль -->  <EditText  android:id="@+id/editTextPassword"  android:hint="@string/new\_password"  android:inputType="textPassword"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  <Button  android:id="@+id/buttonUpdatePassword"  android:text="@string/update\_password"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content" />  </LinearLayout> |

*Листинг 68 – fragment\_game\_detail*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <LinearLayout  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:background="?attr/android:colorBackground"  android:padding="16dp">  <ImageView  android:id="@+id/game\_cover"  android:layout\_width="match\_parent"  android:layout\_height="250dp"  android:layout\_marginBottom="16dp"  android:src="@drawable/cover\_placeholder"  android:scaleType="centerCrop" />  <TextView  android:id="@+id/game\_name"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/game\_title"  android:textSize="24sp"  android:textColor="?attr/colorOnBackground" />  <TextView  android:id="@+id/game\_summary"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/game\_summary"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginTop="16dp" />  <TextView  android:id="@+id/game\_release\_date"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/release\_date"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginTop="16dp" />  <TextView  android:id="@+id/game\_platforms"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/plaforms"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginTop="16dp" />  <Button  android:id="@+id/edit\_game\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/edit"  android:visibility="gone"  android:layout\_marginTop="8dp"  />  <TextView  android:id="@+id/game\_average\_rating"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/average\_rating"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginTop="24dp" />  <Button  android:id="@+id/write\_review\_button"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/leave\_review"  android:layout\_marginTop="8dp"  android:backgroundTint="@color/orange"  android:textColor="@color/black"/>  <TextView  android:id="@+id/user\_review\_text"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/review"  android:visibility="gone"  android:paddingTop="8dp"  android:textColor="?attr/colorOnBackground"  android:textSize="16sp" />  <Space  android:layout\_width="match\_parent"  android:layout\_height="0dp"  android:layout\_weight="1" />  <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:padding="16dp">  <TextView  android:id="@+id/game\_list\_info"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/in\_a\_list"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  android:visibility="gone"/>  <ImageButton  android:id="@+id/remove\_from\_backlog\_button"  android:layout\_width="40dp"  android:layout\_height="40dp"  android:src="@drawable/ic\_delete"  android:background="?attr/selectableItemBackgroundBorderless"  android:contentDescription="@string/delete\_from\_backlog"  app:tint="@color/orange"  app:layout\_constraintTop\_toTopOf="@id/game\_list\_info"  app:layout\_constraintBottom\_toBottomOf="@id/game\_list\_info"  app:layout\_constraintEnd\_toEndOf="parent"  android:visibility="gone"/>  </androidx.constraintlayout.widget.ConstraintLayout>  <Button  android:id="@+id/add\_to\_backlog\_button"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:backgroundTint="@color/orange"  android:text="@string/add\_to\_backlog"  android:textColor="@color/black"  android:gravity="center"  android:layout\_marginTop="16dp" />  </LinearLayout>  </ScrollView> |

*Листинг 69 – fragment\_home*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="?attr/android:colorBackground"  android:padding="16dp">  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical">  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/popular\_releases"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="16dp"/>  <!-- RecyclerView для популярных игр -->  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/popularGamesRecyclerView"  android:layout\_width="match\_parent"  android:layout\_height="220dp"  android:overScrollMode="never"  android:layout\_marginBottom="24dp"  android:clipToPadding="false"  android:paddingEnd="8dp"/>  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/latest\_releases"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="16dp"/>  <!-- RecyclerView для последних релизов -->  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/latestReleaseRecyclerView"  android:layout\_width="match\_parent"  android:layout\_height="220dp"  android:overScrollMode="never"  android:layout\_marginBottom="24dp"  android:clipToPadding="false"  android:paddingEnd="8dp"/>  </LinearLayout>  </ScrollView> |

*Листинг 70 – fragment\_profile*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="?attr/android:colorBackground"  android:padding="16dp">  <ScrollView  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical">  <!-- Профиль пользователя -->  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:gravity="center\_horizontal"  android:layout\_marginBottom="24dp">  <!-- <ImageView-->  <!-- android:id="@+id/userAvatar"-->  <!-- android:layout\_width="100dp"-->  <!-- android:layout\_height="100dp"-->  <!-- android:src="@drawable/avatar"-->  <!-- android:scaleType="centerCrop"-->  <!-- android:adjustViewBounds="true"-->  <!-- android:background="@drawable/circle\_background"-->  <!-- android:layout\_marginBottom="8dp"/>-->  <TextView  android:id="@+id/userName"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/username"  android:textSize="24sp"  android:textColor="?attr/colorOnBackground"  android:textStyle="bold"/>  <TextView  android:id="@+id/userFullNameAge"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/full\_name\_age"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginTop="4dp"/>  <TextView  android:id="@+id/userEmail"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/email"  android:textSize="16sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginTop="4dp"/>  </LinearLayout>  <!-- backlogWantToPlay -->  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/want\_to\_play"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="8dp"/>  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerWantToPlay"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="24dp"  android:overScrollMode="never"  android:nestedScrollingEnabled="false"  android:orientation="horizontal"  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  app:layout\_constraintHeight\_min="127dp"  android:layoutDirection="ltr"/>  <!-- backlogPlaying -->  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/playing"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="8dp"/>  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerPlaying"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="24dp"  android:overScrollMode="never"  android:nestedScrollingEnabled="false"  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  android:orientation="horizontal"  app:layout\_constraintHeight\_min="127dp"  android:layoutDirection="ltr"/>  <!-- backlogPlayed -->  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/played"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="8dp"/>  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerPlayed"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="24dp"  android:overScrollMode="never"  android:nestedScrollingEnabled="false"  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  android:orientation="horizontal"  app:layout\_constraintHeight\_min="127dp"  android:layoutDirection="ltr"/>  <!-- backlogCompleted -->  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/completed"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="8dp"/>  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerCompleted"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="24dp"  android:overScrollMode="never"  android:nestedScrollingEnabled="false"  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  android:orientation="horizontal"  app:layout\_constraintHeight\_min="127dp"  android:layoutDirection="ltr"/>  <!-- backlogCompleted100 -->  <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/completed\_100"  android:textSize="20sp"  android:textColor="?attr/colorOnBackground"  android:layout\_marginBottom="8dp"/>  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerCompleted100"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="24dp"  android:overScrollMode="never"  android:nestedScrollingEnabled="false"  app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  android:orientation="horizontal"  app:layout\_constraintHeight\_min="127dp"  android:layoutDirection="ltr"/>  <Button  android:id="@+id/edit\_game\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/add\_game"  android:visibility="gone"  android:layout\_marginTop="8dp"  />  </LinearLayout>  </ScrollView>  <!-- FAB -->  <com.google.android.material.floatingactionbutton.FloatingActionButton  android:id="@+id/settingsButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:src="@drawable/icon\_settings"  android:layout\_gravity="bottom|end"  android:layout\_margin="16dp"  app:backgroundTint="@color/orange"  app:elevation="6dp"  android:tint="@color/black"  app:rippleColor="@color/black"/>  </androidx.coordinatorlayout.widget.CoordinatorLayout> |

*Листинг 71 – fragment\_search*

|  |
| --- |
| <androidx.constraintlayout.widget.ConstraintLayout  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="?attr/android:colorBackground">  <LinearLayout  android:id="@+id/searchLayout"  android:orientation="horizontal"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:padding="8dp">  <androidx.appcompat.widget.SearchView  android:id="@+id/searchView"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:iconifiedByDefault="false"  app:iconifiedByDefault="false"  android:background="@drawable/search\_view\_border" />  </LinearLayout>  <!-- Кнопка очистки истории -->  <Button  android:id="@+id/clearHistoryButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/clear\_history"  android:visibility="gone"  android:layout\_marginBottom="8dp"  android:layout\_marginTop="4dp"  android:backgroundTint="@color/orange"  android:textColor="@color/black"  app:layout\_constraintTop\_toBottomOf="@id/searchLayout"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent" />  <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerView"  android:layout\_width="match\_parent"  android:layout\_height="0dp"  app:layout\_constraintTop\_toBottomOf="@id/clearHistoryButton"  app:layout\_constraintBottom\_toBottomOf="parent"  android:padding="8dp"/>  <LinearLayout  android:id="@+id/placeholderLayout"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:gravity="center"  android:visibility="gone">  <TextView  android:id="@+id/placeholderText"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:textColor="?attr/colorOnBackground"  android:textSize="16sp"  android:text="@string/no\_results" />  <Button  android:id="@+id/retryButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/update"  android:backgroundTint="@color/orange"  android:visibility="gone" />  </LinearLayout>  <!-- Прогрессбар по центру -->  <ProgressBar  android:id="@+id/progressBar"  style="?android:attr/progressBarStyleLarge"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:visibility="gone"  android:indeterminateTint="@color/orange"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent" />  </androidx.constraintlayout.widget.ConstraintLayout> |

*Листинг 72 – fragment\_settings*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:background="?attr/android:colorBackground"  android:padding="16dp">  <androidx.constraintlayout.widget.ConstraintLayout  android:id="@+id/account"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:paddingVertical="12dp"  android:clickable="true"  android:focusable="true"  android:background="?attr/selectableItemBackground"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent">  <ImageView  android:id="@+id/accountIcon"  android:layout\_width="24dp"  android:layout\_height="24dp"  android:src="@drawable/icon\_profile"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"  app:tint="?attr/colorOnBackground" />  <TextView  android:id="@+id/accountText"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:text="@string/account"  android:textColor="?attr/colorOnBackground"  android:textSize="16sp"  android:layout\_marginStart="16dp"  app:layout\_constraintStart\_toEndOf="@id/accountIcon"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"/>  </androidx.constraintlayout.widget.ConstraintLayout>  <androidx.constraintlayout.widget.ConstraintLayout  android:id="@+id/themeSwitchContainer"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:paddingVertical="12dp"  android:background="?attr/selectableItemBackground"  app:layout\_constraintTop\_toBottomOf="@id/account"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent">  <ImageView  android:id="@+id/themeIcon"  android:layout\_width="24dp"  android:layout\_height="24dp"  android:src="@drawable/icon\_theme"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"  app:tint="?attr/colorOnBackground" />  <TextView  android:id="@+id/themeText"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/dark\_mode"  android:textColor="?attr/colorOnBackground"  android:textSize="16sp"  android:layout\_marginStart="16dp"  app:layout\_constraintStart\_toEndOf="@id/themeIcon"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent" />  <androidx.appcompat.widget.SwitchCompat  android:id="@+id/themeSwitch"  android:layout\_width="wrap\_content"  android:layout\_height="6dp"  android:layout\_marginEnd="8dp"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"/>  </androidx.constraintlayout.widget.ConstraintLayout>  <androidx.constraintlayout.widget.ConstraintLayout  android:id="@+id/logout"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:paddingVertical="12dp"  android:clickable="true"  android:focusable="true"  android:background="?attr/selectableItemBackground"  app:layout\_constraintTop\_toBottomOf="@id/themeSwitchContainer"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintEnd\_toEndOf="parent">  <ImageView  android:id="@+id/logoutIcon"  android:layout\_width="24dp"  android:layout\_height="24dp"  android:src="@drawable/icon\_profile"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"  app:tint="?attr/colorOnBackground" />  <TextView  android:id="@+id/logoutText"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:text="@string/exit"  android:textColor="?attr/colorOnBackground"  android:textSize="16sp"  android:layout\_marginStart="16dp"  app:layout\_constraintStart\_toEndOf="@id/logoutIcon"  app:layout\_constraintTop\_toTopOf="parent"  app:layout\_constraintBottom\_toBottomOf="parent"/>  </androidx.constraintlayout.widget.ConstraintLayout>  </androidx.constraintlayout.widget.ConstraintLayout> |

*Листинг 73 – item\_game*

|  |
| --- |
| <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="150dp"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:padding="4dp">  <ImageView  android:id="@+id/gameCover"  android:layout\_width="match\_parent"  android:layout\_height="180dp"  android:scaleType="centerCrop"  android:background="?attr/selectableItemBackground" />  <TextView  android:id="@+id/gameName"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:textSize="12sp"  android:textColor="?attr/colorOnBackground"  android:maxLines="2"  android:ellipsize="end"  android:gravity="center" />  </LinearLayout> |

*Листинг 74 – bottom\_nav\_menu*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="http://schemas.android.com/apk/res/android">  <item  android:id="@+id/homeFragment"  android:icon="@drawable/icon\_home"  android:title="Главная" />  <item  android:id="@+id/searchFragment"  android:icon="@drawable/icon\_search"  android:title="Поиск" />  <item  android:id="@+id/profileFragment"  android:icon="@drawable/icon\_profile"  android:title="Профиль" />  </menu> |

*Листинг 75 – nav\_graph*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <navigation xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/nav\_graph"  app:startDestination="@id/homeFragment">  <fragment  android:id="@+id/homeFragment"  android:name="com.rarmash.b4cklog.fragments.HomeFragment"  android:label="Home Fragment"  tools:layout="@layout/fragment\_home">  <action  android:id="@+id/action\_homeFragment\_to\_gameDetailFragment"  app:destination="@id/gameDetailFragment"  app:popUpTo="@id/homeFragment"  app:popUpToInclusive="false" />  </fragment>  <fragment  android:id="@+id/gameDetailFragment"  android:name="com.rarmash.b4cklog.fragments.GameDetailFragment"  android:label="Game Detail Fragment"  tools:layout="@layout/fragment\_game\_detail">  <argument  android:name="gameId"  app:argType="integer" />  <action  android:id="@+id/action\_gameDetailFragment\_to\_editGameFragment"  app:destination="@id/editGameFragment"  app:popUpTo="@id/gameDetailFragment"  app:popUpToInclusive="false" />  </fragment>  <fragment  android:id="@+id/searchFragment"  android:name="com.rarmash.b4cklog.fragments.SearchFragment"  android:label="SearchFragment"  tools:layout="@layout/fragment\_search" >  <action  android:id="@+id/action\_searchFragment\_to\_gameDetailFragment"  app:destination="@id/gameDetailFragment"  app:popUpTo="@id/searchFragment"  app:popUpToInclusive="false" />  </fragment>  <fragment  android:id="@+id/profileFragment"  android:name="com.rarmash.b4cklog.fragments.ProfileFragment"  android:label="Profile Fragment"  tools:layout="@layout/fragment\_profile">  <action  android:id="@+id/action\_profileFragment\_to\_editGameFragment"  app:destination="@id/editGameFragment"  app:popUpTo="@id/profileFragment"  app:popUpToInclusive="false" />  <action  android:id="@+id/action\_profileFragment\_to\_settingsFragment"  app:destination="@id/settingsFragment"  app:popUpTo="@id/profileFragment"  app:popUpToInclusive="false" />  <action  android:id="@+id/action\_profileFragment\_to\_gameDetailFragment"  app:destination="@id/gameDetailFragment"  app:popUpTo="@id/profileFragment"  app:popUpToInclusive="false" />  </fragment>  <fragment  android:id="@+id/settingsFragment"  android:name="com.rarmash.b4cklog.fragments.SettingsFragment"  android:label="Settings Fragment"  tools:layout="@layout/fragment\_settings">  <action  android:id="@+id/action\_settingsFragment\_to\_accountSettingsFragment"  app:destination="@id/accountSettingsFragment"  app:popUpTo="@id/settingsFragment"  app:popUpToInclusive="false" />  </fragment>  <fragment  android:id="@+id/accountSettingsFragment"  android:name="com.rarmash.b4cklog.fragments.AccountSettingsFragment"  android:label="Account Settings Fragment"  tools:layout="@layout/fragment\_account\_settings">  </fragment>  <fragment  android:id="@+id/editGameFragment"  android:name="com.rarmash.b4cklog.fragments.EditGameFragment"  android:label="Редактировать игру"  tools:layout="@layout/fragment\_edit\_game">  <argument  android:name="gameId"  app:argType="integer"  android:defaultValue="-1" />  </fragment>  </navigation> |

*Листинг 76 – colors*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>  <!-- Общие цвета -->  <color name="black">#FF000000</color>  <color name="white">#FFFFFFFF</color>  <color name="orange">#FF9500</color>  <!-- Цвета для светлой темы -->  <color name="light\_primary">#FF9500</color>  <color name="light\_background">#FFFFFFFF</color>  <!-- Цвета для тёмной темы -->  <color name="dark\_primary">#FF9500</color>  <color name="dark\_background">#FF000000</color>  </resources> |

*Листинг 77 – strings*

|  |
| --- |
| <resources>  <string name="app\_name">b4cklog</string>  <string name="login">Войти</string>  <string name="signup">Зарегистрироваться</string>  <string name="login\_label">Вход</string>  <string name="username">Имя пользователя</string>  <string name="password">Пароль</string>  <string name="email">Email</string>  <string name="firstName">Имя</string>  <string name="lastName">Фамилия</string>  <string name="age">Возраст</string>  <string name="signup\_label">Регистрация</string>  <string name="game\_title">Название игры</string>  <string name="game\_summary">Описание игры</string>  <string name="release\_date"><u>Дата выхода</u></string>  <string name="plaforms"><u>Платформы</u></string>  <string name="add\_to\_backlog">Добавить в бэклог</string>  <string name="want\_to\_play">Хочу пройти</string>  <string name="playing">Прохожу</string>  <string name="played">Сыграно</string>  <string name="completed">Прошёл</string>  <string name="completed\_100">Прошёл на 100%</string>  <string name="wishlist">Желаемое</string>  <string name="full\_name\_age">Полное имя + Возраст</string>  <string name="remove\_from\_backlog">Удалить игру из бэклога</string>  <string name="in\_a\_list">В списке</string>  <string name="new\_password">Новый пароль</string>  <string name="update\_password">Обновить пароль</string>  <string name="new\_email">Новый email</string>  <string name="update\_email">Обновить Email</string>  <string name="latest\_releases">Последние релизы</string>  <string name="popular\_releases">Популярные</string>  <string name="exit">Выход</string>  <string name="dark\_mode">Тёмная тема</string>  <string name="account">Учётная запись</string>  <string name="clear\_history">Очистить историю</string>  <string name="no\_results">Нет результатов</string>  <string name="update">Обновить</string>  <string name="average\_rating">Средняя оценка</string>  <string name="leave\_review">Оставить отзыв</string>  <string name="review">Отзыв</string>  <string name="delete\_from\_backlog">Удалить из списка</string>  <string name="your\_review">Ваш отзыв</string>  <string name="cover\_url">Ссылка на обложку</string>  <string name="release\_date\_with\_format">Дата выхода (YYYY-MM-DD)</string>  <string name="platforms">Платформы:</string>  <string name="save">Сохранить</string>  <string name="platform\_name">Название платформы</string>  <string name="add\_platform">Добавить платформу</string>  <string name="edit">Редактировать</string>  <string name="add\_game">Добавить игру</string>  </resources> |

*Листинг 78 – styles*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>  <style name="BottomNavigationStyle" parent="Widget.MaterialComponents.BottomNavigationView.Colored">  <item name="android:background">@color/black</item>  <item name="itemIconTint">@drawable/bottom\_nav\_icon\_colors</item>  <item name="itemTextColor">@drawable/bottom\_nav\_text\_colors</item>  <item name="itemTextAppearanceActive">@style/BottomNavigationTextActive</item>  <item name="itemTextAppearanceInactive">@style/BottomNavigationTextInactive</item>  </style>  <style name="BottomNavigationTextActive">  <item name="android:textSize">12sp</item>  <item name="android:textColor">@color/black</item>  </style>  <style name="BottomNavigationTextInactive">  <item name="android:textSize">1sp</item>  <item name="android:textColor">@android:color/transparent</item>  </style>  </resources> |

*Листинг 79 – themes*

|  |
| --- |
| <resources xmlns:tools="http://schemas.android.com/tools">  <!-- Базовая светлая тема -->  <style name="Theme.B4cklog2" parent="Theme.Material3.DayNight.NoActionBar">  <!-- Явно задаём светлые цвета -->  <item name="colorPrimary">@color/light\_primary</item>  <item name="colorOnPrimary">@color/white</item>  <item name="android:colorBackground">@color/white</item>  <item name="android:statusBarColor">@color/light\_primary</item>  <item name="android:fitsSystemWindows">true</item>  </style>  </resources> |

*Листинг 80 – night\themes*

|  |
| --- |
| <resources xmlns:tools="http://schemas.android.com/tools">  <!-- Базовая тёмная тема -->  <style name="Theme.B4cklog2" parent="Theme.Material3.DayNight.NoActionBar">  <!-- Явно задаём тёмные цвета -->  <item name="colorPrimary">@color/dark\_primary</item>  <item name="colorOnPrimary">@color/white</item>  <item name="android:colorBackground">@color/black</item>  <item name="android:statusBarColor">@color/black</item>  <item name="android:fitsSystemWindows">true</item>  </style>  </resources> |

*Листинг 81 – network\_security\_config*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <network-security-config>  <domain-config cleartextTrafficPermitted="true">  <domain includeSubdomains="true">10.0.2.2</domain>  </domain-config>  </network-security-config> |