МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра информационных технологий в атомной энергетике

ОТЧЕТ ПО ПРОЕКТУ

по дисциплине «Разработка приложений на языке Котлин»

Студент группы ИКБО-51-23 Епифанов М.О.

_____

(подпись студента)

Руководитель проектной работы Золотухин С.А.

_____

(подпись руководителя)

Работа представлена «____»_____ 2025 г.

Допущен к работе «___» _____ 2025 г.

Москва 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Fantasy Quest Clicker - это мобильная ролевая игра в жанре кликер, разрабатываемая на Kotlin для Android. Проект сочетает простоту кликера с глубиной классических RPG, предлагая игрокам увлекательный геймплей в фэнтезийном мире.

Проблематика

Рынок мобильных игр переполнен однотипными проектами с упрощенным геймплеем без стратегической глубины. Существует потребность в играх, которые сохраняют доступность жанра, но предлагают сложную и разнообразную механику.

Актуальность

Разработка на Kotlin для Android - перспективное направление. Язык обеспечивает безопасность кода, производительность и удобство поддержки. RPG-кликеры популярны благодаря удовлетворению от постепенного прогресса, что соответствует потребностям мобильных пользователей.

Цели и задачи

Цель: Создать увлекательную мобильную игру, сочетающую простоту кликера с глубиной RPG.

Задачи:

- Разработать интуитивный интерфейс для мобильных устройств
- Реализовать сбалансированную систему прокачки персонажа
- Создать разнообразных противников с уникальными тактиками
- Внедрить систему достижений для поддержания интереса
- Обеспечить стабильную работу на различных устройствах
- Реализовать надежное сохранение прогресса

Состав команды

Проект разрабатывается индивидуально. Все этапы разработки, включая проектирование, программирование и тестирование, выполняются самостоятельно.

1. ПЛАНИРОВАНИЕ И ПОДГОТОВКА

1.1 Техническое задание

1.1.1 Общие сведения

Название проекта: Fantasy Quest Clicker

Тип приложения: Мобильная игра для Android

Жанр: RPG-кликер с элементами фэнтези

Целевая платформа: Android 8.0+ (API 26+)

Технологический стек:

- Kotlin + Android Studio
- Jetpack Compose для интерфейса
- Минимальные требования: Android 8.0+

1.1.2 Цели и назначение автоматизированной системы

Разработка мобильной RPG-игры в жанре кликер, которая предоставляет пользователям увлекательный игровой процесс с элементами прокачки персонажа, сражениями с монстрами и выполнением квестов в фэнтезийном мире.

1.1.3 Характеристика объектов автоматизации

- Игровой персонаж - обладает характеристиками (здоровье, атака, защита), навыками и экипировкой
- Система боя - тактические сражения с различными типами противников
- Инвентарь - система хранения и использования предметов
- Магазин - приобретение снаряжения и зелий
- Система достижений - награды за игровой прогресс

1.1.4 Требования к системе

Функциональные требования:

- Создание и кастомизация персонажа
- Интерактивная боевая система с тапами по врагам
- Дерево навыков для прокачки способностей
- Система квестов и достижений
- Локации с различными врагами и боссами
- Сохранение прогресса игры

Нефункциональные требования:

- Адаптивный интерфейс для различных размеров экранов
- Поддержка портретной ориентации
- Оптимизация производительности для мобильных устройств
- Минимальное потребление батареи

1.1.5 Состав и содержание работ

1. Проектирование архитектуры мобильного приложения
2. Разработка пользовательского интерфейса с Jetpack Compose
3. Создание игровой логики и механик
4. Реализация системы сохранения данных
5. Интеграция звуковых эффектов и визуальной обратной связи
6. Тестирование на различных устройствах

1.1.6 Порядок разработки и приемки

Этапы разработки:

- Неделя 1-2: Базовая архитектура и главный экран
- Неделя 3-4: Игровая механика и боевая система
- Неделя 5-6: Система прокачки и инвентарь
- Неделя 7-8: Сохранение данных и полировка

Критерии приемки:

- Стабильная работа на целевых устройствах
- Отсутствие утечек памяти
- Плавная анимация (60 FPS)
- Корректное сохранение прогресса

1.1.7 Требования к подготовке объекта к вводу системы в действие

- Android устройство версии 8.0 или выше
- 100 МБ свободного места в памяти
- Поддержка сенсорного ввода
- Рекомендуется 2 ГБ оперативной памяти

1.1.8 Требования к документированию

- KDoc документация для всех классов и методов
- README с инструкцией по сборке и запуску
- Комментарии для сложной бизнес-логики
- Документация архитектуры приложения

1.1.9 Источники разработки

1. Официальная документация Android Developer
2. Руководства по Material Design
3. Документация Jetpack Compose
4. Примеры игровых приложений на Kotlin

1.2 Диаграмма вариантов использования

1.2.1 Обоснование сценариев использования

Основные сценарии взаимодействия пользователя с приложением:

- Создание и настройка персонажа

- Участие в боях с противниками через тапы
- Изучение и прокачка навыков в древе талантов
- Управление инвентарем и экипировкой
- Покупка предметов в магазине
- Выполнение сюжетных квестов
- Сохранение и загрузка игрового прогресса

1.2.2 Схематическое представление (Use Case Diagram)

Актер "Игрок" взаимодействует с системой через следующие прецеденты:

- Создать персонажа
- Сражаться с врагами
- Прокачивать навыки
- Выполнять квесты
- Сохранять прогресс

Результат создания диаграммы прецедентов представлен на Рисунке 1.

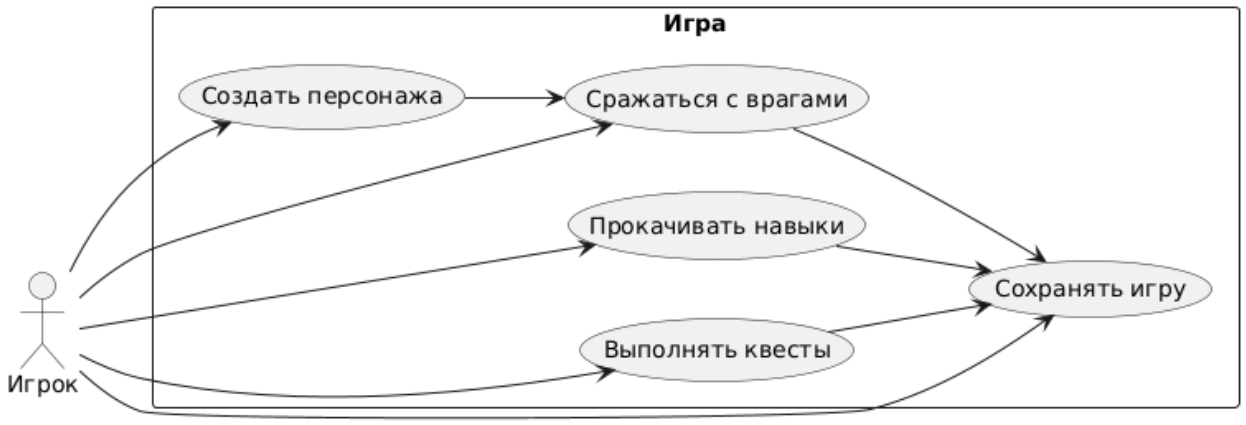

Рисунок 1 - Диаграмма прецедентов мобильной игры Fantasy Quest Clicker

1.3 Создание репозитория

Для контроля версий и управления разработкой создан Git-репозиторий на платформе GitHub. Структура репозитория включает:

- Модуль приложения Android
- Исходный код на Kotlin с использованием современных архитектурных подходов
- Файлы ресурсов (изображения, звуки, строки)
- Конфигурационные файлы Gradle
- Файл README с описанием проекта и инструкциями по сборке

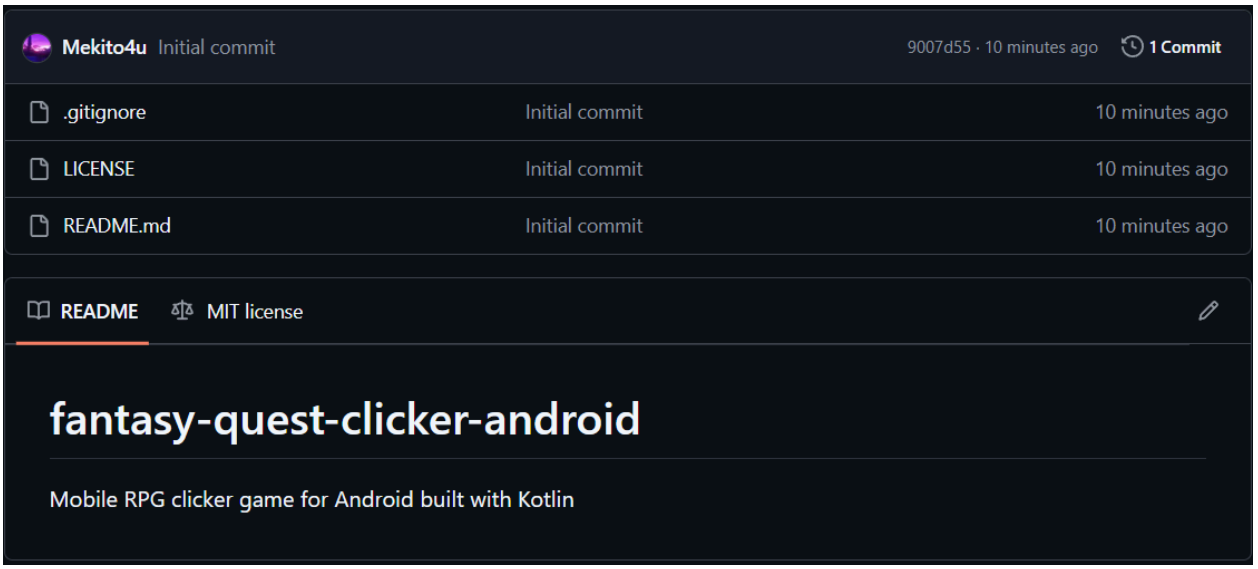

Рисунок 2 - Структура репозитория мобильного приложения

Ссылка на репозиторий: https://github.com/Mekito4u/fantasy-quest-clicker-android

2. ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА

2.1 Макеты экранов

Интерфейс спроектирован по принципу минимализма и фокуса на основном геймплее.

Архитектура экранов:

- Экран создания персонажа - стартовый экран с выбором класса и имени
- Экран боя - центральный игровой экран с врагом и кнопкой атаки
- Экран прокачки - система навыков с визуальным прогрессом
- Экран квестов - список заданий с отслеживанием выполнения

Принципы навигации:

- Быстрый переход между основными разделами
- Экран боя как центральный хаб игры
- Минимальное количество промежуточных экранов
- Интуитивная система возврата к предыдущим экранам

Визуальная иерархия:

- Ключевые действия выделены размером и контрастом
- Важная информация расположена в верхней части экрана
- Основные элементы управления - в нижней зоне доступа

Макет окон игры представлен на Рисунках 3-6.

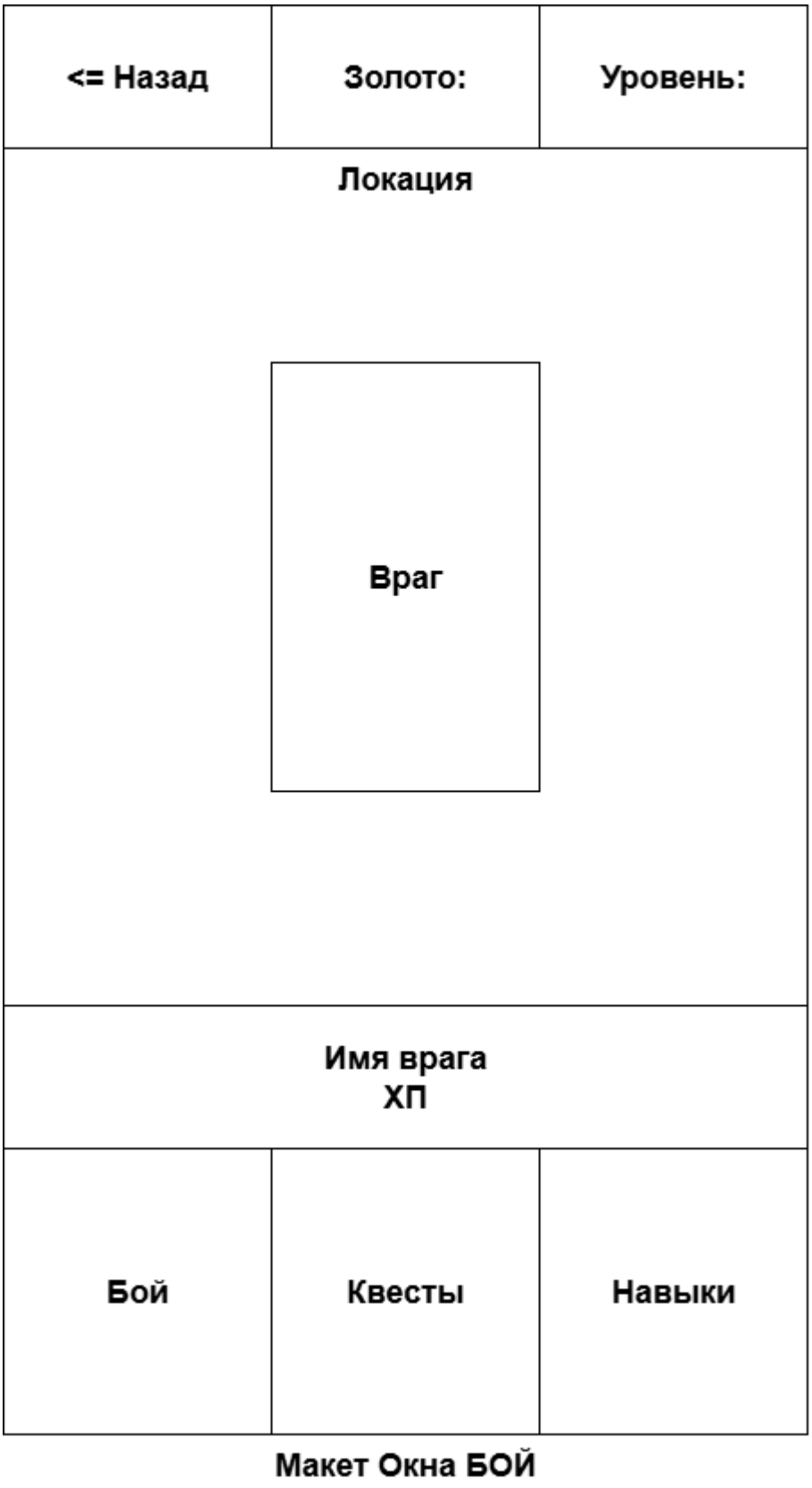

Рисунок 3 — Результат создания макета окна Бой

Рисунок 4 — Результат создания макета окна Создание персонажа

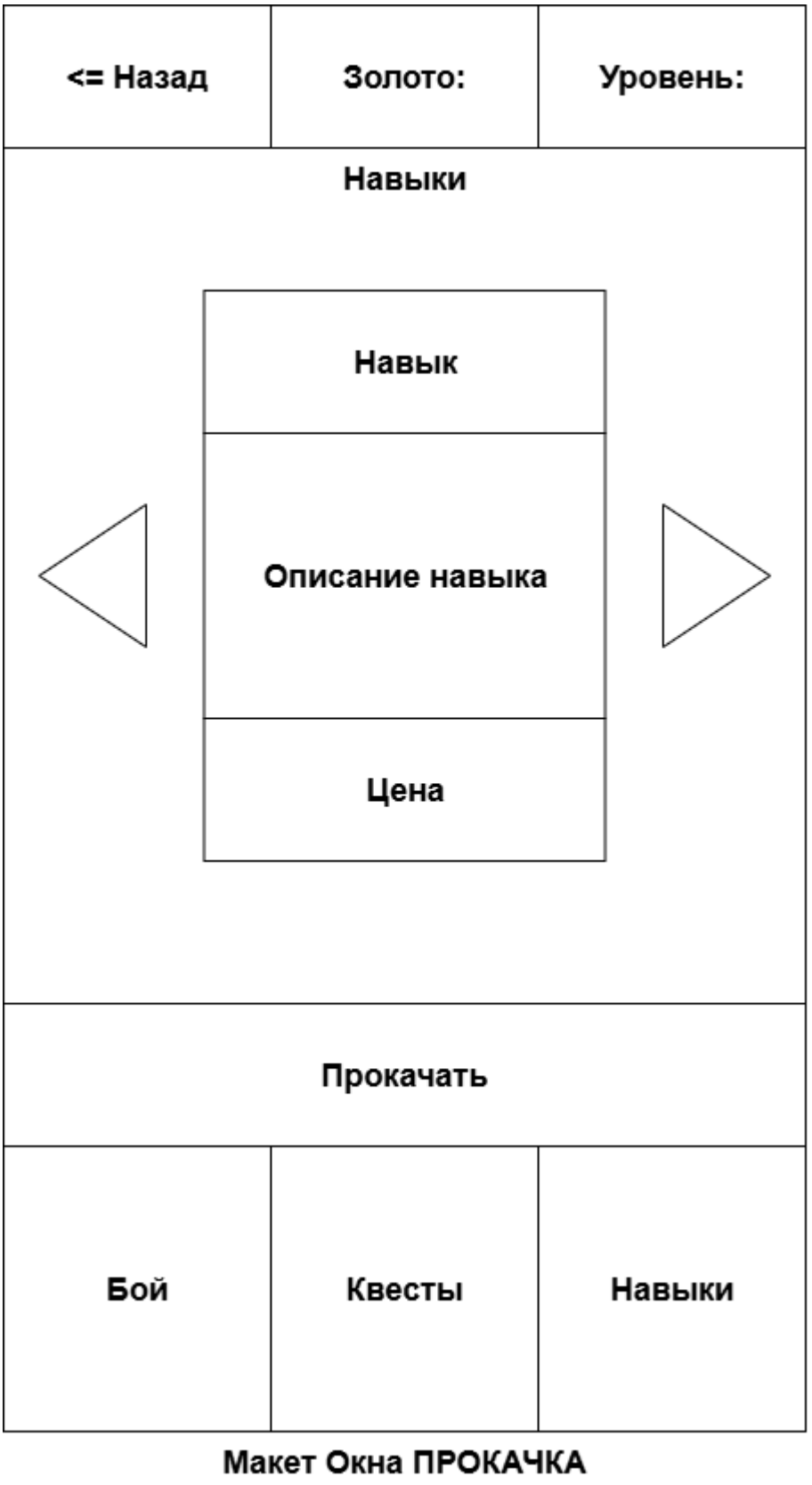

Рисунок 5 — Результат создания макета окна Прокачка

Рисунок 6 — Результат создания макета окна Квесты

3. РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Архитектура приложения

Система сборки и зависимости

Для управления сборкой и зависимостями используется Gradle с Kotlin DSL.

Основные конфигурации:

- Версия Android Gradle Plugin: 8.2.0+
- Минимальная SDK:** 26 (Android 8.0)
- Целевая SDK:** 34 (Android 14)

Ключевые зависимости:

- Jetpack Compose (UI)
- Kotlin Coroutines (асинхронность)
- Lifecycle ViewModel (архитектура)
- Room Database (сохранение прогресса)

```
dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform( dependencyProvider = libs.androidx.compose.bom))
    implementation(libs.androidx.compose.ui)
    implementation(libs.androidx.compose.ui.graphics)
    implementation(libs.androidx.compose.ui.tooling.preview)
    implementation(libs.androidx.compose.material3)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform( dependencyProvider = libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.compose.ui.test.junit4)
    debugImplementation(libs.androidx.compose.ui.tooling)
    debugImplementation(libs.androidx.compose.ui.test.manifest)
}
```

Рисунок 7 - Конфигурация зависимостей в build.gradle.kts

Схематичное изображение архитектуры разрабатываемого проекта представлено на Рисунке 8.

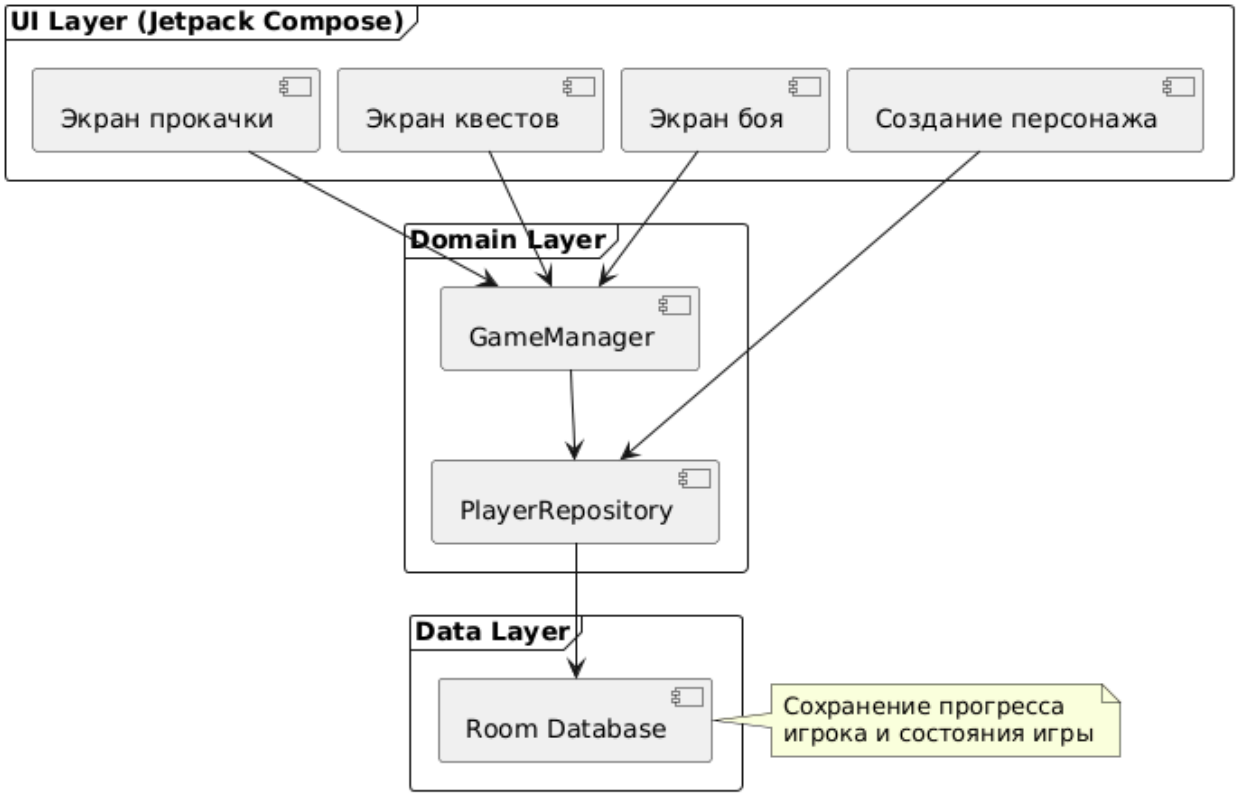

Рисунок 8 — Результат создания схемы архитектуры проекта

3.2 Реализация ключевых компонентов

Заполняется в зависимости от типа проекта каждым участником команды индивидуально. Пункты frontend, backend и т.п – опционально. Создавайте необходимые пункты под свой проект. Листинги кода обязательны.

3.2.1 Frontend

Для реализации клиентской части проекта выбраны технологии/фреймворки…и т.п.

Реализованный функционал карточки товара представлен в Листинге 1.

Листинг 1 – Листинг кода

```
код
```

3.2.2 Backend

Для реализации серверной части проекта выбраны технологии/фреймворки…и т.п.

Реализованный функционал карточки товара представлен в Листинге 2.

Листинг 2 – Листинг кода

```
код
```

3.2.3 База данных (опционально)

Какая СУБД была выбрана, описание структуры БД и т.п.

3.2.4 Особенности реализации отдельных компонентов (опционально)

Особенности….

3.3 Тестирование приложения

Для тестирования приложения были написаны Unit тесты для ключевого функционала ….

Или тестирование проекта производилось с использованием ….Selenium…. и т.п

Результат написания Unit тестов представлен в Листинге …

Результат успешного прохождения тестирования представлен на Рисунке …

3.4 Документирование кода

Использован KDoc для описания классов и функций. Генерация HTML-документации производилась … Пример документирования кода ключевого функционала представлен на Рисунке….

ЗАКЛЮЧЕНИЕ

Разработка интернет-магазина позволила реализовать современное решение для электронной коммерции. Основные задачи проекта выполнены — от проектирования до тестирования и документирования. В будущем возможна интеграция с системами аналитики и добавление новых функций, таких как рекомендации товаров. Разработка проекта также способствовала улучшению навыков программирования на языке Kotlin и развитию командной работы.

Ссылка на GitHub-репозиторий разработанного проекта: ….

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.602—2020 ("Техническое задание на создание автоматизированной системы") является стандартом, который определяет общие требования к разработке автоматизированных систем (АС). Он используется в разных отраслях, включая медицину, и применим для разработки автоматизированных информационных систем (АИС).
2. ГОСТ 34.201—2020. Межгосударственный стандарт. Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем: Приказом Федерального агентства по техническому регулированию и метрологии № 1521-ст от 19 ноября 2021 г.: дата введения 2022-01-01. – М.: Российский институт стандартизации, 2021. – 10 с.
3. Блоштин А.В., Лаптев Д.В. Современные подходы к проектированию клиент-серверных приложений // Программные системы: теория и приложения. 2021. №2. URL: https://elibrary.ru/item.asp?id=46523678 (дата обращения: 26.11.2025)
4. Android Developers — официальный сайт документации по разработке на Kotlin: https://developer.android.com/kotlin (дата обращения: 26.11.2025).
5. Kotlin Documentation — полный справочник по языку Котлин: https://kotlinlang.org/docs (дата обращения: 26.11.2025).
6. Жемеров С., Исакова С. Kotlin в действии. 2-е издание. М.: Питер, 2022. URL: https://www.piter.com/product/kotlin-v-dejstvii (дата обращения: 26.11.2025).

7. Документация JUnit — для тестирования Kotlin-приложений: https://junit.org/junit5 (дата обращения: 26.11.2025).