Техническое Задание

Общая информация:

Проект представляет собой веб-приложение, аналогичное Steam. Цель проекта - создать платформу для приобретения, скачивания и обмена компьютерными играми. Разработка будет вестись с использованием Django для бэкенда, шаблонизатора Django для фронтенда, и отдельного REST API на Django Rest Framework.

I. Архитектура приложения:

- *Бэкенд:*
  - Использование Django для основной логики приложения.
  - Отдельное REST API на Django Rest Framework (DRF) для взаимодействия с фронтендом.
- *База данных:*
  - Интеграция с PostgreSQL для эффективного хранения и управления данными.

II. Аутентификация и авторизация:

- *Регистрация и аутентификация:*
  - Реализация механизмов регистрации и аутентификации пользователя.
- *Токены:*
  - Использование JWT (JSON Web Tokens) для безопасной аутентификации.

III. Модуль управления пользователями:

- *Профиль пользователя:*
  - Возможность редактирования профиля, смены пароля и других персональных настроек.
- *Аватар и изображения:*

- Возможность загрузки аватара и других изображений для профиля.
- *История покупок и скачиваний:*
  - Ведение истории игр, купленных и скачанных пользователями.

IV. Модуль игр:

- *Каталог игр:*
  - Создание каталога игр с разделением на категории и подкатегории.
- *Страница игры:*
  - Отображение информации о каждой игре, включая описание, изображения, рейтинг, отзывы и другие характеристики.
- *Поиск и фильтрация:*
  - Реализация функций поиска и фильтрации для удобного выбора игр.

V. Модуль отзывов:

- *Оставление отзывов:*
  - Возможность пользователей оставлять отзывы с рейтингом и текстовым комментарием.
- *Отзывы на отзывы:*
  - Реализация комментирования отзывов другими пользователями.

VI. Страницы разработчика и пользователя:

- *Страница разработчика:*
  - Отображение информации о разработчике, включая профиль, портфолио игр, контактные данные.
- *Страница пользователя:*
  - Персональная страница пользователя с историей покупок, скачиваний, отзывами и другими персональными данными.

VII. Асинхронные задачи:

- *Рассылка уведомлений:*

- Интеграция Celery для асинхронной обработки и рассылки уведомлений (например, по электронной почте).

VIII. Безопасность:

- *Защита от CSRF и других атак:*
- Реализация механизмов защиты от распространенных видов атак.

IX. Деплой и контейнеризация:

- *Docker контейнер:*
- Упаковка приложения в Docker контейнер для обеспечения легкости развертывания и масштабируемости.

Технологии:

- Django
- Django Rest Framework
- PostgreSQL
- JWT (JSON Web Tokens)
- Celery
- Docker