МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

**ОТЧЕТ**

По лабораторной работе  
**Финальный**

Выполнил: Иванов С. М.

Группа: 6302-010302D

Самара 2025

* 1. Архитектура

Клиент ↔ Сервер ↔ База Данных

* 1. Схема и сущности базы данных

Схема БД:

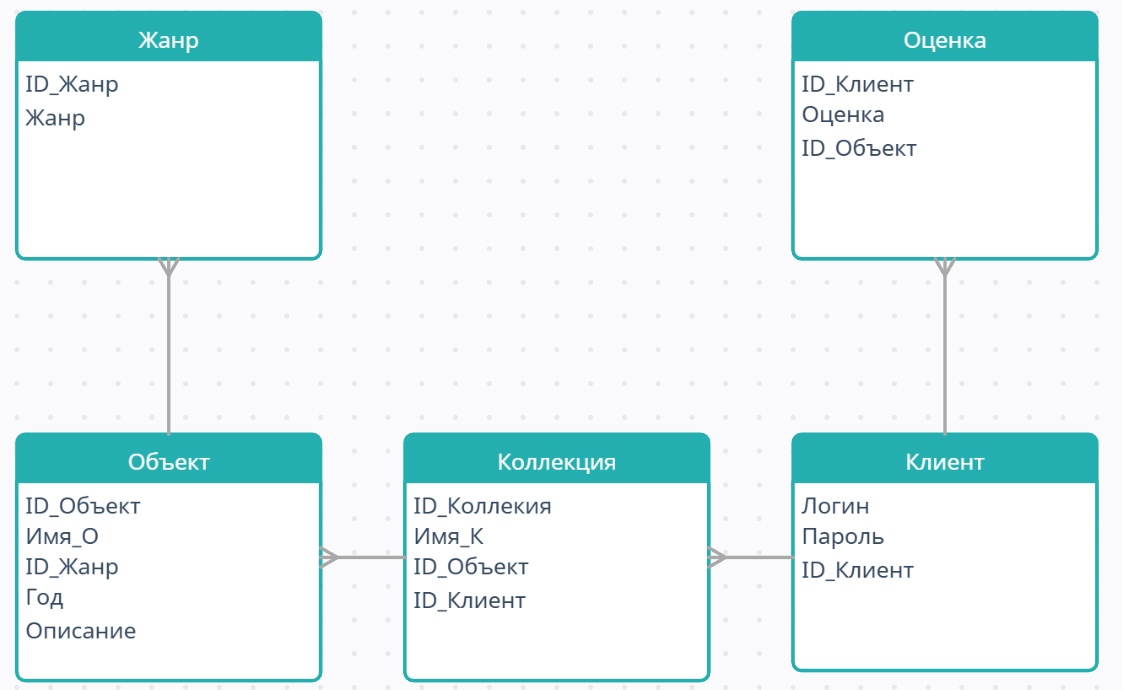


Рисунок 1 — Схема БД

* 1. Возможности пользователя

• Идентифицироваться\Залогиниться

• Просмотр списка объектов

• Просмотр всех объектов одного года

• Просмотр всех объектов одного жанра

• Сортировать список всех объектов по информации об объекте

• Открывать объект отдельно, видеть его индивидуальную информацию

• Создавать личные коллекции

• Добавить объект в личную коллекцию

• Просматривать личные коллекции

• Оценивать объекты

* 1. Стек технологий

Язык программирования: Python

Фреймворк: Flask, SQLAlchemy

Система миграций: Alembic

База данных: PostgreSQL

* 1. Репозиторий

[https://github.com/GreySRV/Project\_R](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2FGreySRV%2FProject_R&utf=1)

* 1. Проверки

В данном отчете приведены тестовые проверки, для экономии места. В репозитории лежит файл test.txt, содержащий в себе вспомогательную для работы с проектом информацию, включая список curl проверок, команд для работы с миграциями, создания объектов через консоль.

* 1. Демонстрация работы БД

Список всех таблиц в базе:

\

Рисунок 2 — Список всех таблиц в базе

Вывод всех строк таблицы objects, как пример. SELECT \* FROM objects;

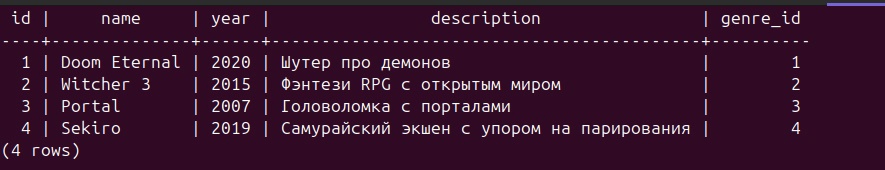


Рисунок 3 — Таблица objects

Добавление нового объекта в таблицу.

INSERT INTO objects (name, year, description, genre\_id)

VALUES ('Half-Life 3', 2025, 'Легендарный шутер, которого все ждали', 1);

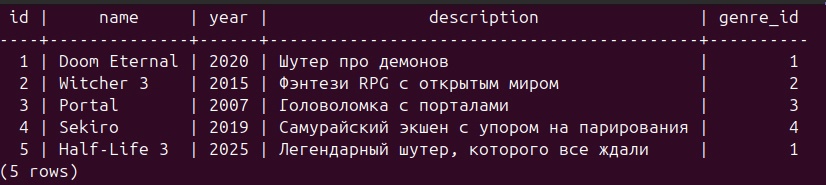


Рисунок 4 — Таблица objects после добавления нового элемента

* 1. Аутентификация

Пользователь вводит свои логин и пароль, после чего возвращается JWT-токен, при помощи которого и происходят запросы на сервер. Токен не видим пользователю и кидается в blacklist после того, как пользователь разлогинится.

* 1. Демонстрация работы API

API описан в файле openapi.yml, находящийся в репозитории. Так же там есть список curl запросов. Проведём демонстрацию через несколько curl-запросов касательно коллекций. Первый – вход и получение токена.

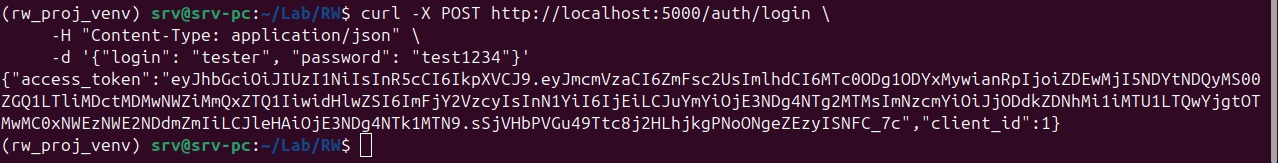


Рисунок 5 — Вход пользователя

Далее получим уже созданные пользователем коллекции. Запрос понимает, какой пользователь обращается к нему, по JWT-токену.

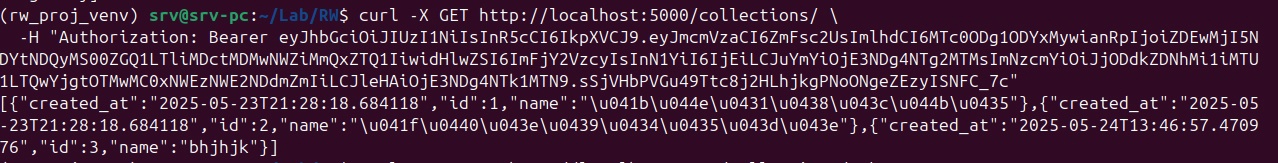


Рисунок 6 — Получение коллекций пользователя

Коллекции 1 и 2 названы кириллицей, из-за чего при выводе в терминал отображаются кодом символов. Это лишь артефакт вывода в терминал и на работоспособность не влияет. Теперь удалим коллекцию 2 и выведем список вновь.

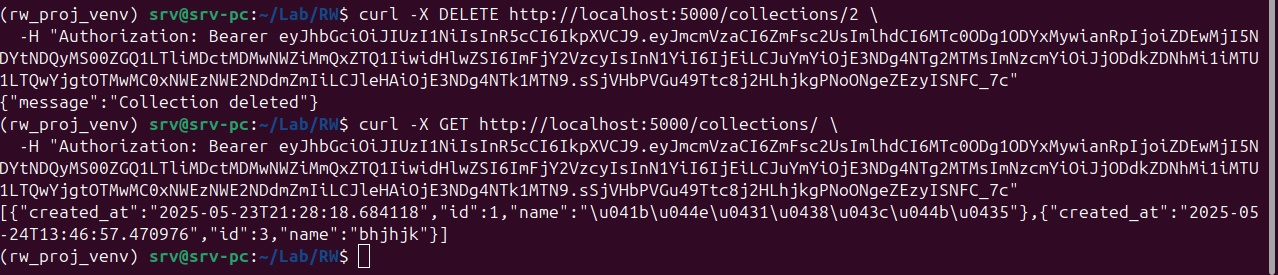


Рисунок 7 — Удаление коллекции

* 1. Интерфейс

Общение между клиентом и сервером происходит при помощи библиотеки Marshmallow. Отрисовкой страниц занимается Blueprint.

Домашняя страница проекта — это общий список объектов.

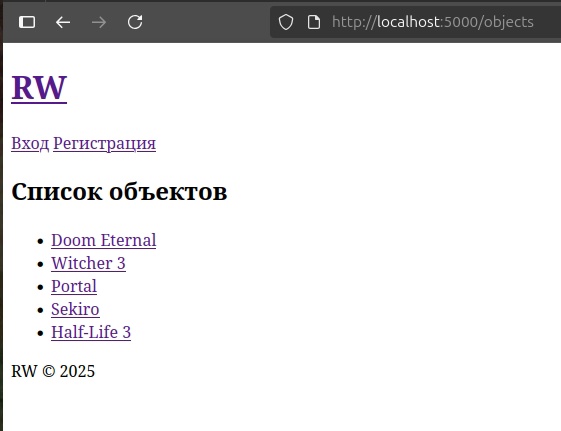


Рисунок 8 — Домашняя страница

Если перемещаться по страницам объектов без регистрации, пользователь будет находиться в гостевом режиме.

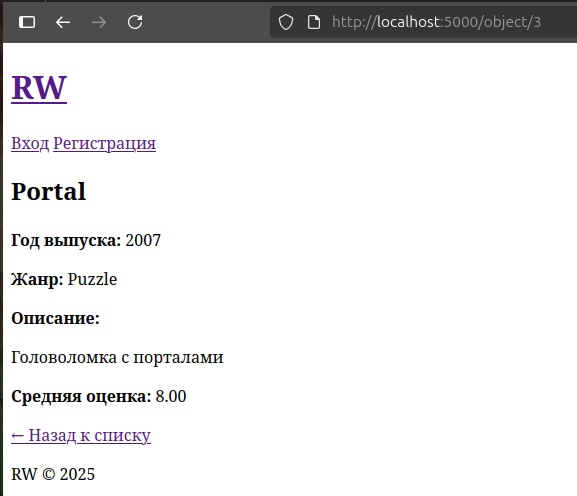


Рисунок 9 — Страница объекта от лица гостя

На странице входа пользователь вводит свои логин и пароль. На странице регистрации логин и пароль вводятся впервые, затем сохраняются, и пользователь сможет войти, используя их на странице входа. Логин проверяется на дубликат – если введённый в окне регистрации логин уже существует, то запись не создаётся и пользователя уведомляют об этом. Так же с обеих страниц можно продолжить посещение сайта в качестве гостя, но возможностей в таком случае будет меньше.

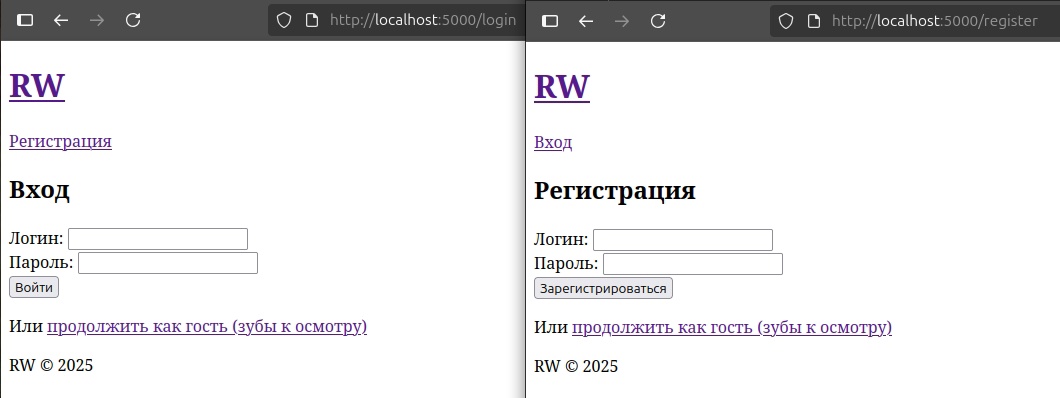


Рисунок 10 — Страницы входа и регистрации

Зарегистрированный пользователь может оценивать объекты и добавлять их в коллекции. Коллекции можно так же создавать прямо на странице объекта. Если объект уже находится в коллекции, она помечается в общем списке и становится не кликабельной, поскольку смысла повторно добавлять один и тот же объект в неё нет.

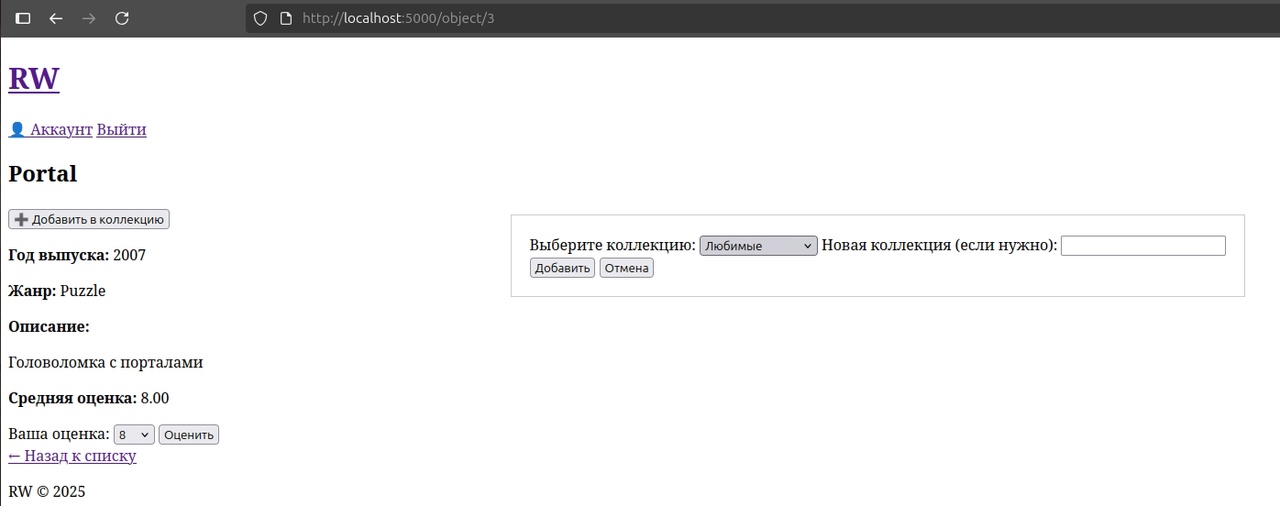


Рисунок 11 — Страница объекта от лица пользователя с активным окном добавления в коллекцию

В верхней части интерфейса зарегистрированного пользователя всегда есть кнопка «Аккаунт», при нажатии на которую пользователь переходит на личную страницу. Здесь можно изучить свои коллекции и сменить пароль в выпадающем меню настроек.

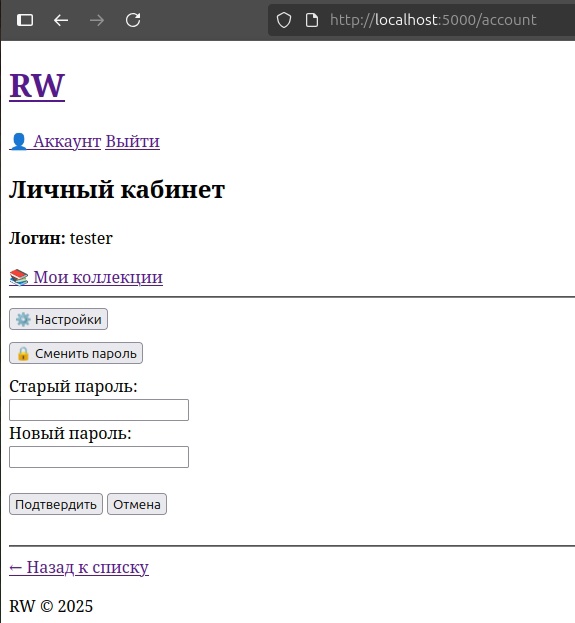


Рисунок 12 — Личный кабинет пользователя с активным меню смены пароля

На странице коллекций отображаются все коллекции, созданные пользователем.

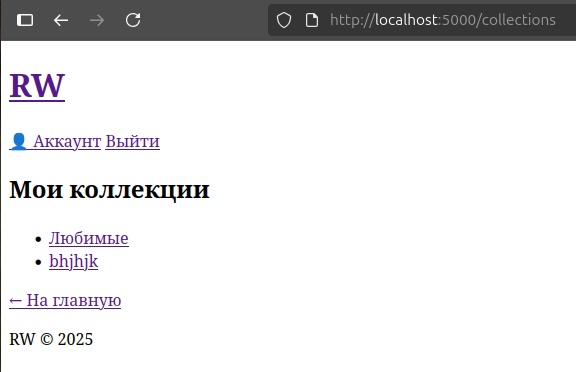


Рисунок 13 — Страница коллекций пользователя

При переходе на коллекцию отображается список добавленных в неё пользователем объектов. Здесь их можно удалить или перейти на их страницу.

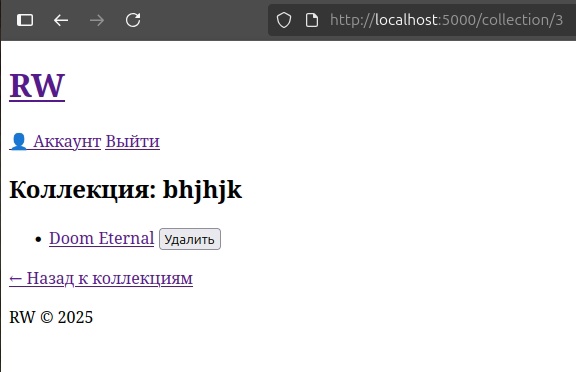


Рисунок 14 — Внутри пользовательской коллекции

* 1. Упаковка в контейнер

В файлах docker-compose.yml и Dockrfile описаны инструкции по созданию, запуску и управлению общим образом, в который входят контейнеры с БД и технической частью проекта. Проект запускается командой docker compose up. Команда docker ps показывает все активные контейнеры.

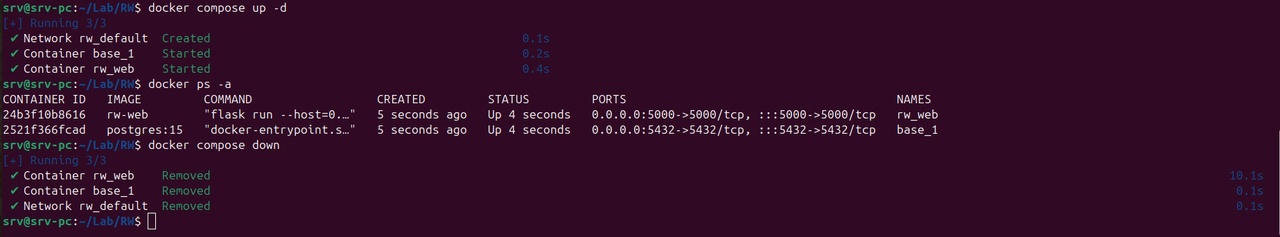


Рисунок 15 — Вывод активных контейнеров

* 1. Финальная архитектура

Пользователь → Flask Frontend → Flask Backend → SQLAlchemy → PostgresSQL DB

Frontend (Jinja2+Flask):

templates/ - HTML-шаблоны

static/ - стили и скрипты

app/routes/frontend.py – обработка переходов по страницам, отображение формы и данных

requests для обращения к REST API на back’е

Flask session для хранения сессий и flash-сообщений

Backend (Flask):

app/routes/ - эндпоинты

app/services/ - логика

app/models/ - модели SQLAlchemy

app/schemas/ - marshmallow-схемы для (де)сериализации

app/extensions.py – инициализация

auth – регистрация, вход, смена пароля, JWT

Возвращает JSON-ответы

База Данных (PostgreSQL):

Развёрнута в Docker-контейнере

Все таблицы определены в моделях SQLAlchemy. Таблицы: Client, Object, Genre, Collection, CollectionObject, Rating

Миграции (Alembic):

migrations/ - хранение миграций

alembic.ini – настройки миграций для работы с БД

Docker (+compose):

Dockerfile – упаковка backend+frontend

docker-compose.yml – управление запуском контейнеров с базой и фронт-беком

.env – хранение конфигураций

Документация:

openapi.yml – описание REST API

test.txt – набор вспомогательных команд и проверок

requirements.txt – список используемых библиотек

Аутентификация:

JWT токены через flask

Поддержка пользователей и гостей

Авторизированные пользователи имеют доступ к личным данным, коллекциям, функциям