**Отчёт по лабораторной работе №5**  
**Тема:** Финализация приложения и упаковка в Docker

**Цель работы**  
Завершить разработку приложения, связать клиентскую и серверную части, создать и запустить Docker-контейнеры для базы данных, backend и frontend, убедиться в их взаимодействии.

**Выполненные задачи**

1. Связаны фронтенд и бэкенд.
2. Устранены ошибки при передаче данных между компонентами.
3. Создан Docker-контейнер для PostgreSQL и сконфигурировано подключение.
4. Создан Dockerfile для backend-сервиса.
5. Создан docker-compose.yml для запуска всех компонентов.
6. Проведено тестирование контейнеров.
7. Подготовлена документация.

**Архитектура приложения**

Общая структура взаимодействия компонентов:

[Пользователь]  
↓  
[Frontend (Jinja2 + Django views)]  
↓  
[Django Backend (REST API)]  
↓  
[PostgreSQL DB (в Docker)]

Описание компонентов:

* **Frontend:** шаблоны Django на Jinja2. Формируют пользовательский интерфейс и передают данные на backend.
* **Backend:** Django REST Framework. Отвечает за обработку API-запросов, авторизацию, сериализацию и бизнес-логику.
* **База данных:** PostgreSQL в контейнере Docker.
* **Docker:** используется для развёртывания backend и базы данных.

**Связывание фронтенда и бэкенда**

* Все шаблоны (например, home.html, content\_detail.html) получают данные от API через requests.
* Авторизация пользователя реализована на основе JWT: токены передаются в HTTP-заголовках.
* Фронтенд-формы регистрации, входа и создания отзывов отправляют данные на API.

Пример запроса из views:

requests.get('http://127.0.0.1:8000/api/contents/<id>/', headers=headers)

**Docker: конфигурация базы данных**

Пример Dockerfile для PostgreSQL (используется официальный образ):

FROM postgres:latest

ENV POSTGRES\_DB=cinema\_db

ENV POSTGRES\_USER=cinema\_user

ENV POSTGRES\_PASSWORD=cinema\_password

EXPOSE 5432

Основной docker-compose.yml:

version: '3.8'

services:

db:

container\_name: o\_cinema

image: postgres:latest

restart: always

environment:

POSTGRES\_DB: cinema\_db

POSTGRES\_USER: cinema\_user

POSTGRES\_PASSWORD: cinema\_password

ports:

- "5438:5432"

volumes:

- cinema\_volume:/var/lib/postgresql/data

web:

build:

context: ../backend\_base/myproject

container\_name: cinema\_web

command: python manage.py runserver 0.0.0.0:8000

volumes:

- ../backend\_base/myproject:/app

ports:

- "8000:8000"

depends\_on:

- db

volumes:

cinema\_volume:

driver: local

driver\_opts:

type: 'none'

o: 'bind'

device: 'D:/GitHub/online\_cinema/database'

Содержимое .dockerignore:

venv/

\_\_pycache\_\_/

\*.pyc

\*.pyo

\*.pyd

db.sqlite3

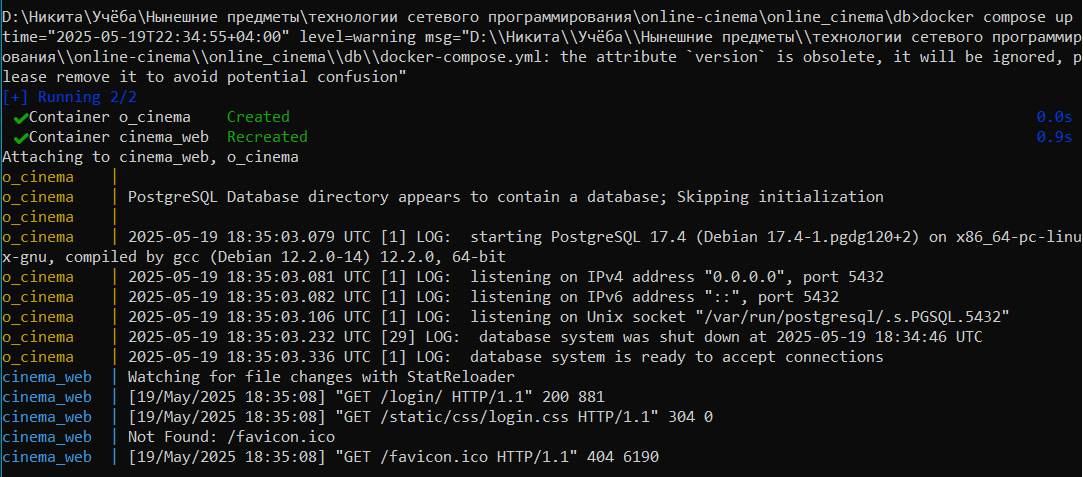
\*.log

**Тестирование работы контейнеров**

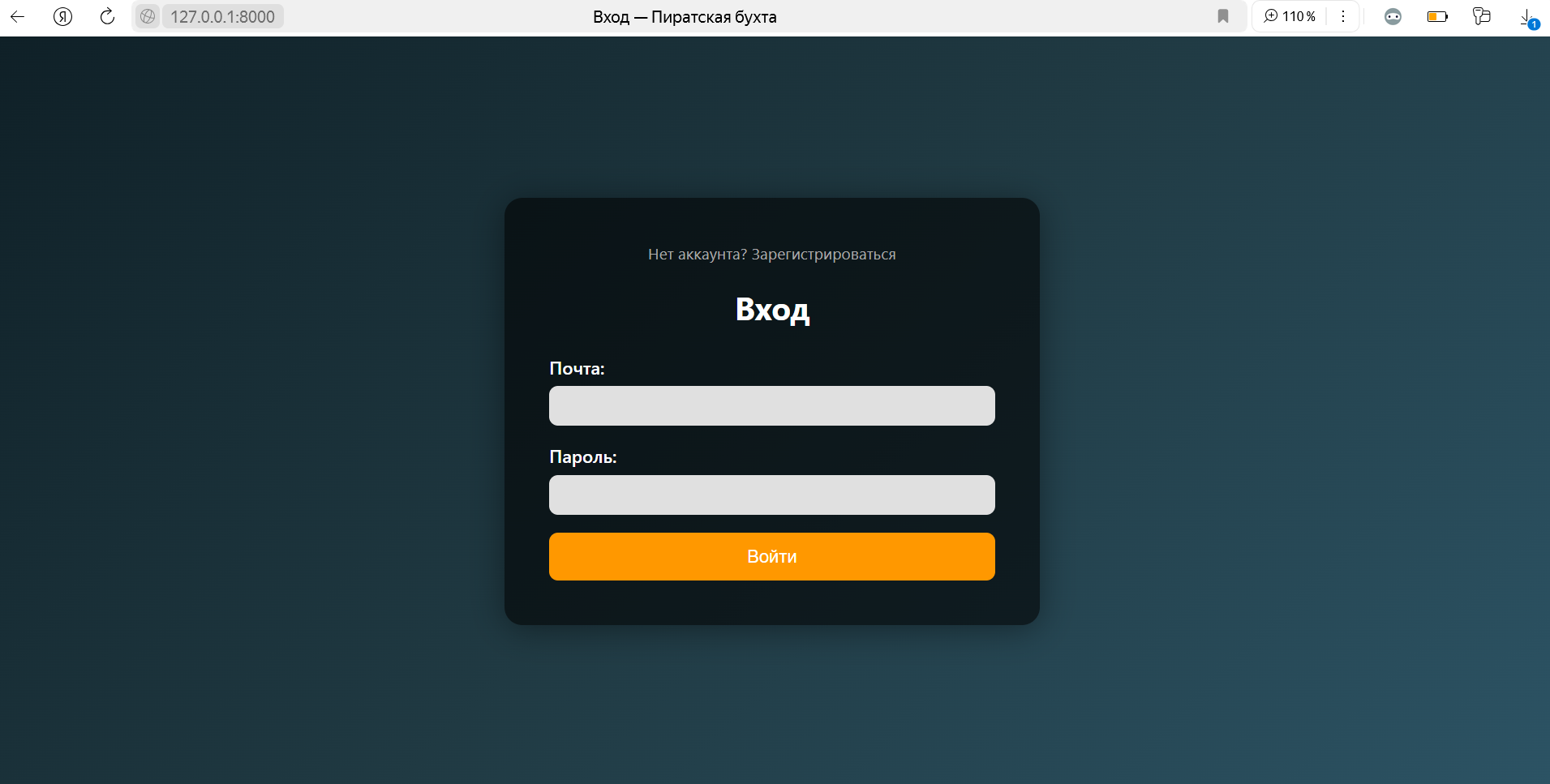
Контейнеры запускаются командой:

docker-compose up

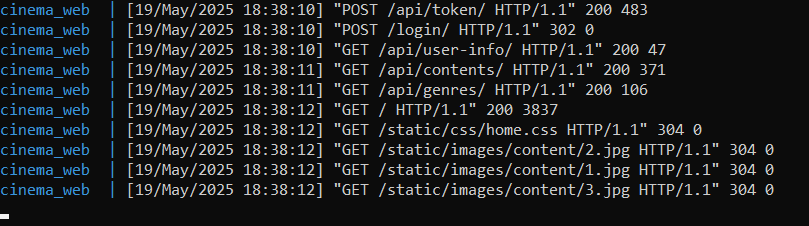
Скриншот с выполненной командой docker ps, показывающий запущенные контейнеры o\_cinema и cinema\_web.



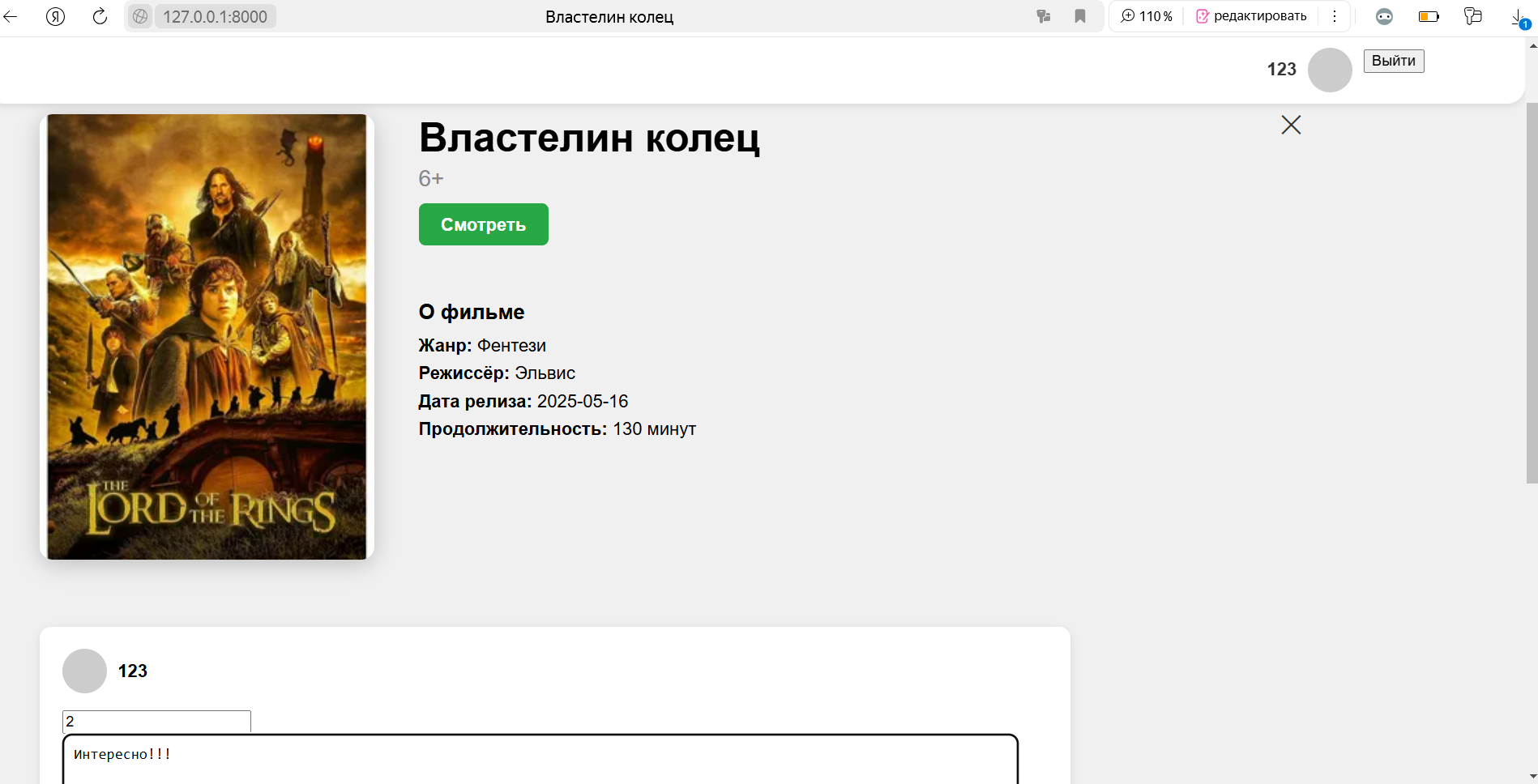
Скриншот открытой главной страницы фронтенда в браузере (по адресу http://localhost:8000/login/).



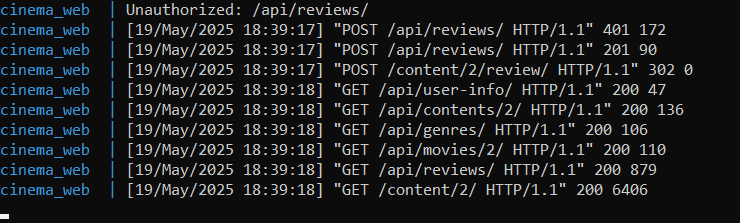
Скриншот ответа API на GET-запрос



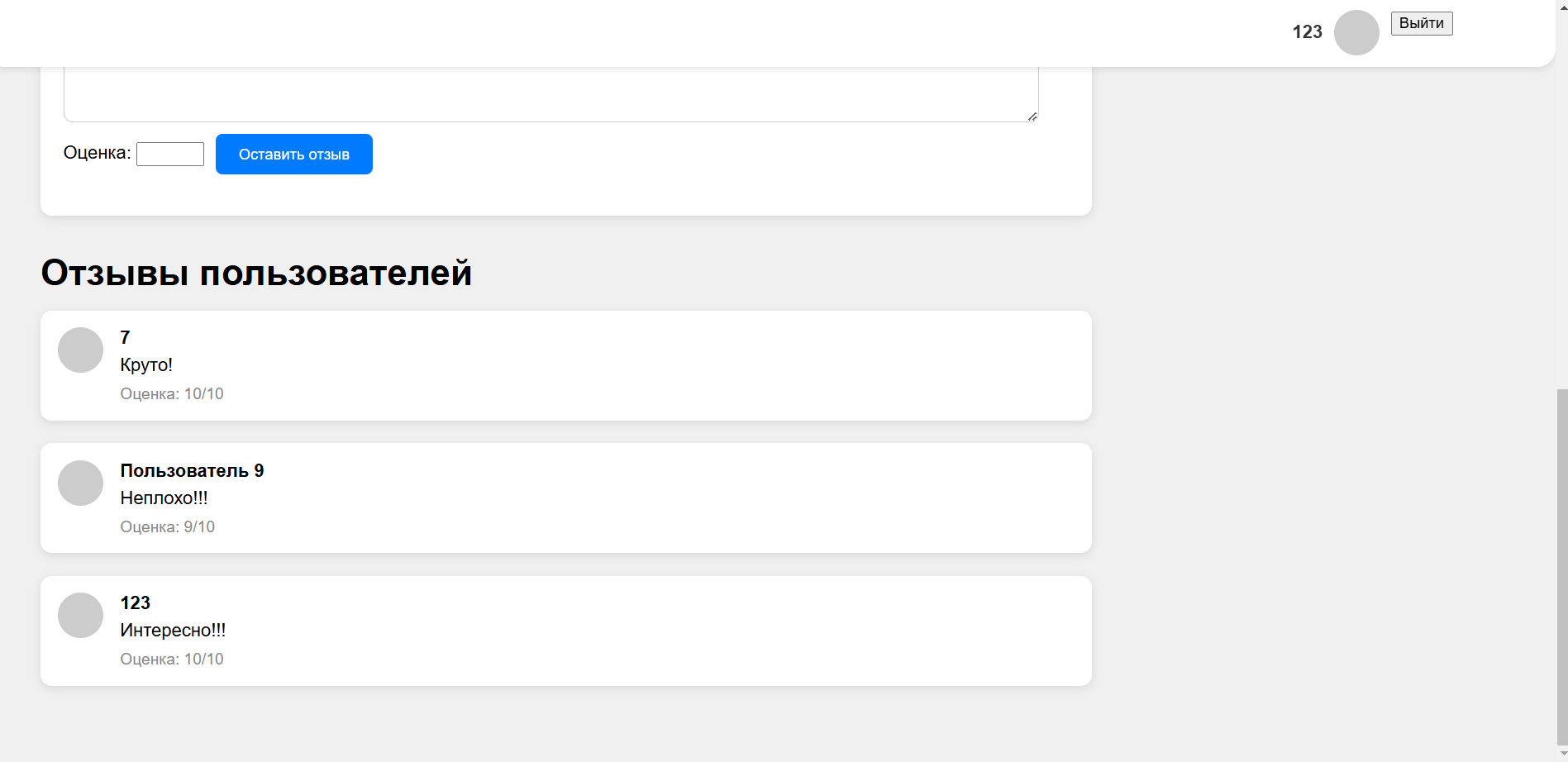
Скриншот работы с отзывами (добавление, просмотр).



Скриншот работы с отзывами (добавление, просмотр).



Скриншот работы с отзывами (добавление, просмотр).



**Описание API**

| **Метод** | **URL** | **Описание** | **Авторизация** |
| --- | --- | --- | --- |
| GET | /api/contents/ | Получение списка контента | Нет |
| GET | /api/contents/{id}/ | Получение конкретного контента | Нет |
| POST | /api/register/ | Регистрация пользователя | Нет |
| POST | /api/token/ | Получение JWT-токена | Нет |
| POST | /api/token/refresh/ | Обновление JWT-токена | Нет |
| POST | /api/reviews/ | Создание отзыва | Да |
| GET | /api/genres/ | Получение жанров | Нет |
| GET | /api/user-info/ | Получение информации о пользователе | Да |

Структура проекта

online\_cinema/

├── db/

│ ├── Dockerfile.txt

│ └── docker-compose.yml

├── backend\_base/

│ └── myproject/

│ ├── cinema/

│ │ ├── views\_api.py

│ │ ├── views\_frontend.py

│ │ ├── models.py

│ │ ├── serializers.py

│ │ └── templates/

│ ├── settings.py

│ ├── urls.py

│ └── manage.py

├── .dockerignore

├── requirements.txt

└── venv/

**Вывод**

В рамках лабораторной работы было:

* Завершено создание функционального приложения;
* Реализовано API и веб-интерфейс;
* Настроены контейнеры Docker для базы данных и backend-сервиса;
* Проверена связность всех компонентов;

Все цели и задачи лабораторной работы выполнены в полном объёме.