**Отчет о выполнении лабораторной работы 1: Разработка базы данных**

**Цель работы**

Развернуть базу данных, разработать ORM-модели и реализовать скрипты для управления данными.

**Выполненные задачи**

**1. Настройка базы данных**

**Развертывание PostgreSQL в Docker**

Для работы с базой данных был использован PostgreSQL, развернутый в контейнере Docker. База данных была развернута с использованием следующей команды:

*docker run --name o\_cinema -e POSTGRES\_DB=cinema\_db -e POSTGRES\_USER=cinema\_user -e POSTGRES\_PASSWORD=cinema\_password -p 5432:5432 -d postgres:latest*

**Подключение к БД через DBeaver**

После развертывания базы данных было выполнено подключение к ней с использованием DBeaver. Параметры подключения:

* **Host:** localhost
* **Port:** 5432
* **Database:** cinema\_db
* **User:** cinema\_user
* **Password:** cinema\_password

**2. Разработка ORM-моделей**

**Определение сущностей и связей**

В рамках проекта были определены основные сущности:

* User (пользователь)
* Film (фильм)
* Review (отзыв)

Связи между сущностями:

* Один пользователь может оставить несколько отзывов (One-to-Many).
* Один фильм может иметь множество отзывов (One-to-Many).

**Реализация моделей с использованием Django ORM**

Для представления данных в базе была использована Django ORM. В файле models.py приложения cinema были определены модели с нужными полями и связями.

**Настройка миграций**

Были выполнены миграции для создания таблиц в базе данных:

*python manage.py makemigrations cinema*

*python manage.py migrate*

**3. Разработка модели пользователя**

**Определение структуры модели**

Была создана модель пользователя (User), включающая следующие поля:

* email (уникальный идентификатор);
* password (хешированный пароль);
* first\_name (имя);
* last\_name (фамилия);

**Настройка хеширования паролей**

Для безопасного хранения паролей было настроено их хеширование с использованием set\_password и проверки check\_password из Django.

**4. Функционал для работы с моделями**

**Написание скриптов для заполнения базы данных**

Был создан Python-скрипт для автоматического заполнения базы данных тестовыми данными. В скрипте использовалась библиотека Faker для генерации случайных пользователей и фильмов.

Пример команды для запуска скрипта:

python manage.py populate\_db

**Реализация API для работы с данными**

Были разработаны методы API для работы с моделями:

* **GET** /films/ — получить список фильмов;
* **POST** /films/ — добавить новый фильм;
* **GET** /films/<id>/ — получить информацию о конкретном фильме;
* **PUT/PATCH** /films/<id>/ — обновить информацию о фильме;
* **DELETE** /films/<id>/ — удалить фильм.

**Выводы**

В рамках лабораторной работы была успешно развернута база данных PostgreSQL в Docker, подключена через DBeaver, реализованы ORM-модели, выполнены миграции и настроены API-методы для работы с данными. Были написаны скрипты для автоматического заполнения базы данных и произведено тестирование API в Postman.

**Список команд, использованных в консоли**

python manage.py makemigrations cinema

python manage.py migrate

python manage.py showmigrations

python manage.py populate\_db

python manage.py runserver

python manage.py createsuperuser