1. **Ссылка на репозиторий (уровень 1):**
2. **Описаний выбранных технологий:**

| **Компонент** | **Технология** | **Описание** |
| --- | --- | --- |
| **Язык программирования** | **Python** | Современный высокоуровневый язык с богатой экосистемой для backend-разработки, обработки данных и машинного обучения. |
| **Веб-фреймворк** | **FastAPI** | Быстрый и асинхронный фреймворк для создания API на базе Python. Позволяет автоматически генерировать документацию Swagger/OpenAPI. Идеально подходит для современных RESTful и ML-приложений. |
| **База данных** | **PostgreSQL** | Надёжная, расширяемая объектно-реляционная СУБД с поддержкой ACID-транзакций. Используется для хранения отзывов и фильмов. Поддерживает сложные типы данных и полнотекстовый поиск. |
| **ORM** | **SQLAlchemy** | Используется как уровень абстракции над SQL. Позволяет управлять моделями и данными Python-объектами. Используем Declarative Mapping через @mapped. |
| **Миграции БД** | **Alembic** | Инструмент управления версиями схемы базы данных. Позволяет создавать, применять и откатывать миграции, сохраняя согласованность базы. Используется в связке с SQLAlchemy. |
| **Валидация данных** | **Pydantic** | Используется для валидации и сериализации входных и выходных данных API. Обеспечивает строгую типизацию, работает на основе аннотаций типов Python. |
| **HTTP-сервер** | **Uvicorn** | ASGI-сервер для запуска FastAPI-приложений. Обеспечивает высокую производительность, поддержку WebSocket и hot-reload в разработке. |

1. **ER-диаграмма и описание таблиц базы данных (названия, столбцы, типы, связи);**

**1. movies – Таблица фильмов**

Таблица хранит основную информацию о фильмах, к которым могут быть оставлены отзывы.

| **Название поля** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | INTEGER | Первичный ключ, уникальный идентификатор фильма. |
| title | VARCHAR | Название фильма. Не может быть пустым. Уникальное значение. |
| created\_at | TIMESTAMP | Дата и время добавления фильма (по умолчанию текущая дата). |

**Ограничения:**

* title должен быть уникальным (unique=True).
* created\_at автоматически устанавливается при создании записи.

**2. reviews – Таблица отзывов**

Таблица хранит отзывы пользователей о фильмах.

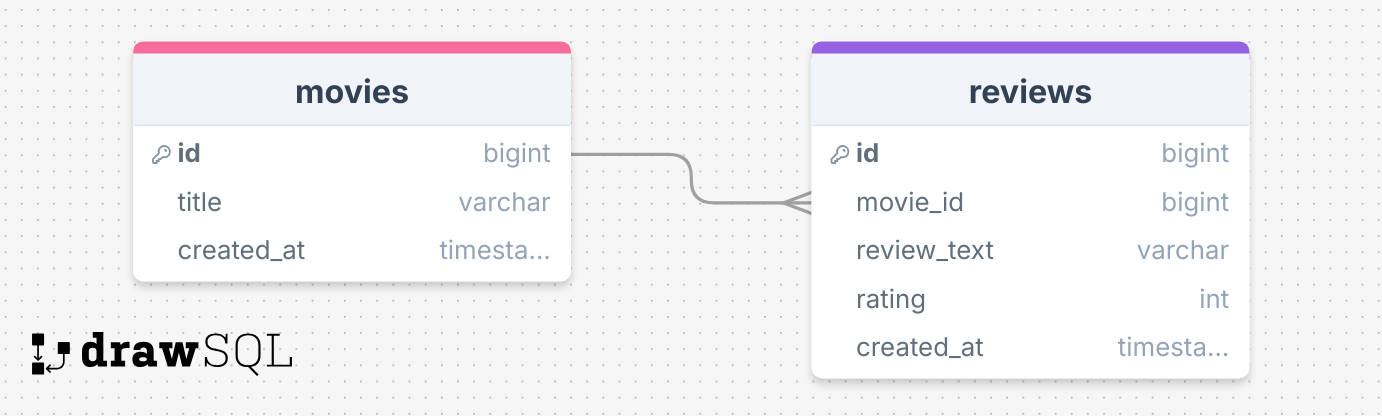
| Название поля | Тип данных | Описание |
| --- | --- | --- |
| id | INTEGER | Первичный ключ, уникальный идентификатор отзыва. |
| movie\_id | INTEGER | Внешний ключ на таблицу movies(id), обозначающий фильм. |
| review\_text | VARCHAR | Текст отзыва. Не может быть пустым. |
| rating | INTEGER | Оценка фильма. Диапазон: от 1 до 10. |
| created\_at | TIMESTAMP | Дата и время создания отзыва (по умолчанию текущая дата). |

**Ограничения:**

* rating валидируется на уровне бизнес-логики (1 ≤ rating ≤ 10).
* movie\_id связан с movies.id, каскадное удаление не включено (по умолчанию поведение RESTRICT).

**Связи:**

* **reviews.movie\_id → movies.id** (связь "многие-к-одному")



1. **Примеры запросов к API (в curl через http://127.0.0.1:8000/docs)**

**POST:**

**curl -X 'POST' \**

**'http://127.0.0.1:8000/reviews' \**

**-H 'accept: application/json' \**

**-H 'Content-Type: application/json' \**

**-d '{**

**"movie\_title": "Я делаю шаг",**

**"review\_text": "Фильм мне очень понравился, интересный сюжет, хорошие актеры.",**

**"rating": 10**

**}'**

GET (все фильмы):

**curl -X 'GET' \**

**'http://127.0.0.1:8000/reviews' \**

**-H 'accept: application/json'**

GET (фильм «Кракен»):

**curl -X 'GET' \**

**'http://127.0.0.1:8000/reviews?movie=%D0%9A%D1%80%D0%B0%D0%BA%D0%B5%D0%BD' \**

**-H 'accept: application/json**

1. **Дополнительно:**

На данном уровне были реализованы миграции БД (через Alembic).

**Миграции позволяют:**

* Создавать и обновлять структуру базы данных безопасно и пошагово;
* Изменять таблицы (добавлять поля, индексы и т.д.) без потери данных;
* Сохранять историю изменений схемы в виде версионированных файлов.

Запуск:

# Применить все миграции (обычно при старте приложения)

alembic upgrade head

Если вы внесли изменения в модели (models.py) и хотите зафиксировать новую миграцию:

# Сгенерировать новую миграцию на основе изменений в models.py

alembic revision --autogenerate -m "Описание изменений"

# Применить миграцию

alembic upgrade head

Также дополнительно в проекте имеется файл index.html и часть кода в main.py, предоставляющие интерактивный фронтэнд. Это очень полезно, чтобы смотреть ответы SQLAlchemy в так называемый «вложенной» структуре (для использования в VSCode нужно установить расширение Live Server: 

И затем в правом нижнем углу VSCode нажать кнопку «Go Live»).