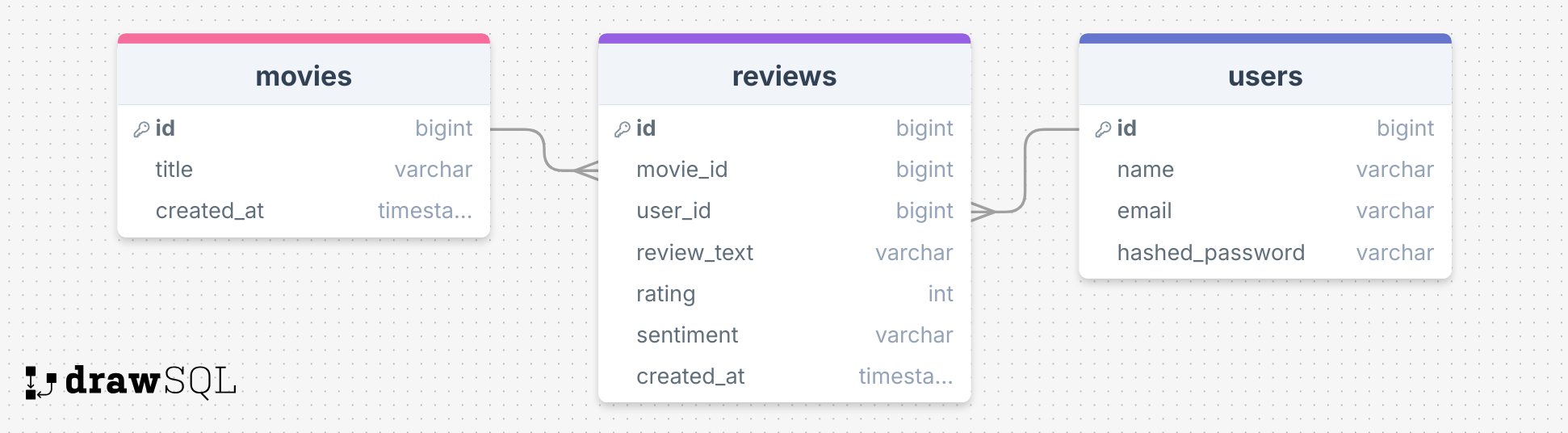
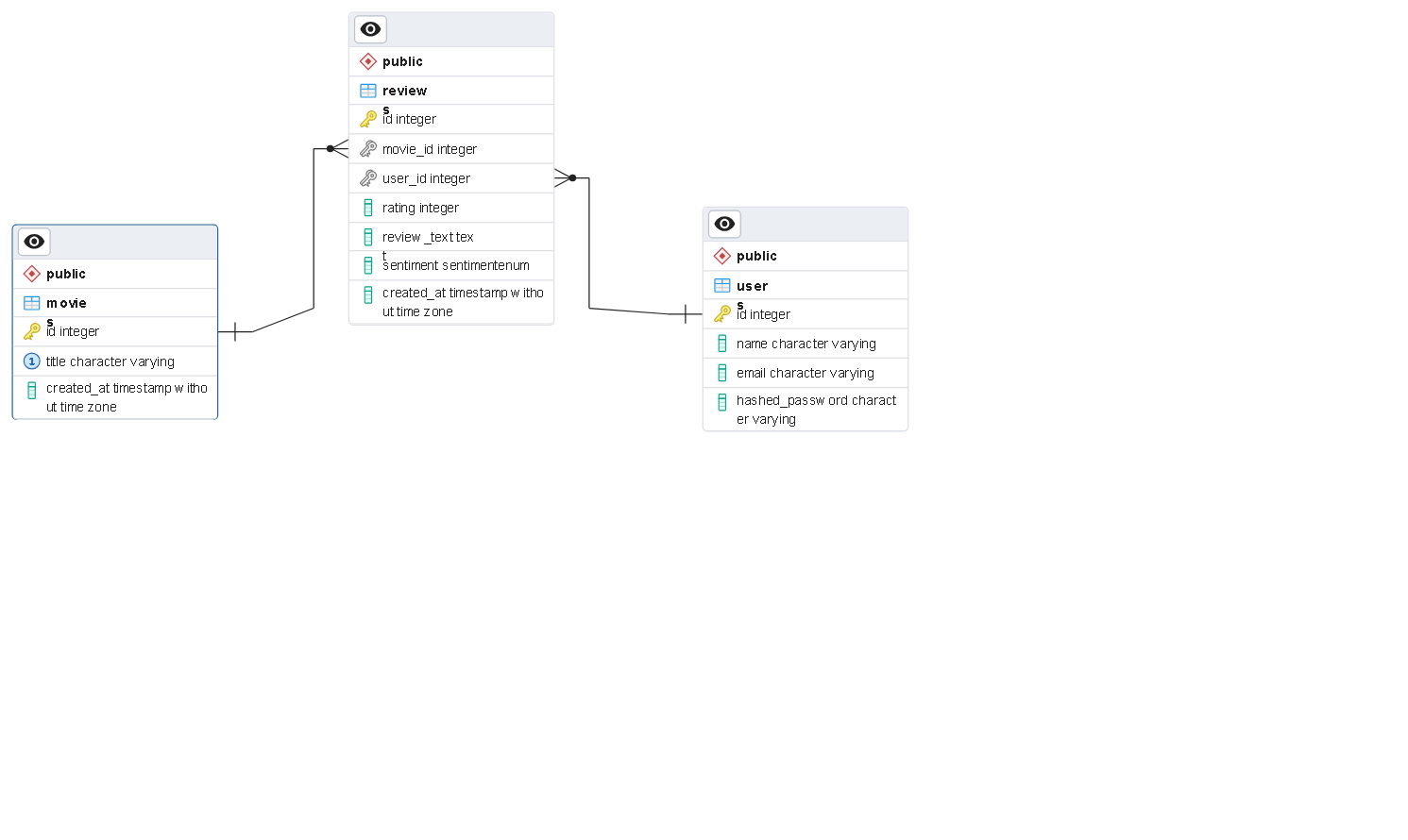
1. **Ссылка на репозиторий (уровень 3):** [ap3814887/movies\_reviews\_project\_lvl3](https://github.com/ap3814887/movies_reviews_project_lvl3)
2. **Обновлённая ER-диаграмма (с таблицей users и связями):**



Либо также можно автоматически в PostgreSQL сгенерировать ERD-диаграмму:

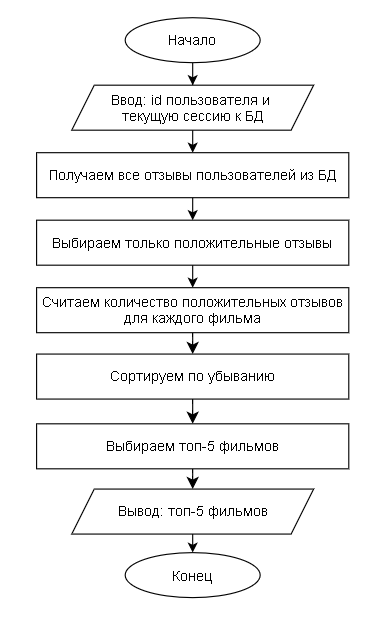


1. **На данном уровне реализовано несколько новых возможностей, рассмотрим их более подробно:**
   1. **@router.get("/recommendations") - возвращает топ-5 популярных фильмов, которые пользователь ещё не смотрел.**

Рекомендации строятся на основе:

* анализа тональности отзывов пользователей;
* частоты упоминания фильмов в позитивных отзывах.

Таким образом, система подбирает наиболее популярные и положительно оценённые фильмы среди всей аудитории, исключая те, что уже просмотрены данным пользователем (модель на основе популярности с фильтрацией просмотренного).



* 1. **@router.get("/semantic-recommendations") - возвращает топ-5 рекомендованных фильмов, используя кастомные эмбединги (Sentence-BERT) для семантического анализа отзывов (хранятся в векторной БД Faiss).**

Для работы данного эндпоинта реализовано 2 модуля semantic\_embeddings.py (это по сути «мозги» по поиску похожих текстов) и semantic\_recommender.py («обёртка», которая применяет мозги, чтобы найти фильмы, похожие на те, что пользователь оценил).

В данных эмбедингах используется модель all-MiniLM-L6-v2 — это компактная версия Sentence-BERT, которая создаёт векторное представление текста так, что смысловые (семантические) близкие фразы будут иметь близкие вектора + косинусное сходство.

**Алгоритм работы данных модулей можно описать следующим образом:**

1. Инициализация модели: загружается предобученная модель Sentence-BERT (all-MiniLM-L6-v2).
2. Векторизация текста (get\_embedding)

* На вход подаётся текст отзыва.
* Модель возвращает эмбеддинг — вектор фиксированной длины (384 числа, numpy.ndarray).
* Вектор нормализуется (делим на длину), чтобы можно было использовать косинусное сходство через скалярное произведение.

1. Построение FAISS-индекса (build\_faiss\_index)

* Собираются векторы всех фильмов, рассчитанные как среднее эмбеддингов всех их отзывов.
* Создаётся IndexFlatIP — структура FAISS для поиска ближайших векторов по косинусному сходству (IP = inner product).
* Все векторы фильмов добавляются в индекс.

1. Формирование вектора предпочтений пользователя

* Для текущего пользователя выбираются все позитивные отзывы.
* Для каждого отзыва считается эмбеддинг, затем берётся их среднее.
* Полученный вектор также нормализуется.

1. Поиск похожих фильмов (get\_top\_k\_similar)

* Вектор предпочтений пользователя подаётся в FAISS-индекс.
* Индекс возвращает k ближайших фильмов по косинусному сходству.
* Из результата исключаются фильмы, которые пользователь уже оценивал.

1. Выдача рекомендаций

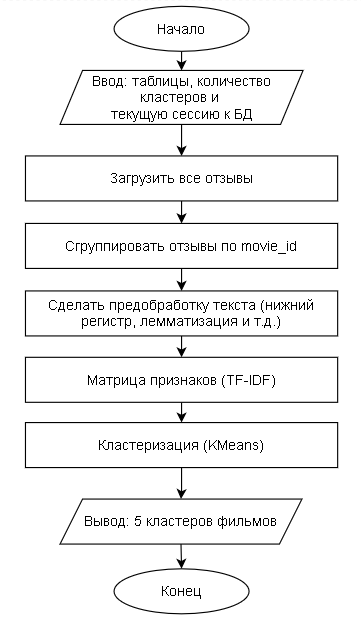
* Возвращается список названий фильмов, упорядоченных по степени схожести с предпочтениями пользователя.
  1. **@router.get("/clustered-movies") – возвращает 5 кластеров фильмов, разделенных по тематики отзывов (число кластеров в будущем можно, например, подбирать автоматически методом «локтя» или «силуэта»).**

Кластеризация помогает выявить тематические группы и предлагать фильмы из того же кластера, что и любимые фильмы пользователя.

**Сам модуль ­– clustering.py**

**Используемая модель: TF-IDF** (способ превратить текст в числовой вектор, который отражает, насколько важно каждое слово) **+ KMeans** (группирует объекты (в нашем случае векторы фильмов) так, чтобы внутри группы были похожие объекты, а между группами — непохожие)

Алгоритм работы модуля представлен ниже:



* 1. **@router.get("/collaborative-recommendations") – возвращает топ-5 фильмов рекомендованных с помощью коллаборативной фильтрации (метод рекомендательных систем, который нужен для того, чтобы рекомендовать пользователю фильмы, которые понравились похожим по вкусам пользователям).**

**Сам скрипт – collaborative\_filtering.py**

**Используемая модель: user-based collaborative filtering с косинусным сходством.**

В данном модуле также предусмотрено: Фильтрация пользователей и фильмов по активности – чтобы рекомендации не портило слишком мало данных. Порог сходства – чтобы учитывать только достаточно похожих пользователей, а не всех подряд.

**Алгоритм:**

