Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский Федеральный Университет имени первого президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Отчет по лабораторной работе №2

«Работа с SQLAlchemy и alembic»

по дисциплине «Разработка приложений»

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_ Федоров А. Д.

Группа РИМ-150950

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьмин Д. И.

Екатеринбург, 2025

Цель работы

Освоить принципы работы с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python, изучить механизмы миграции базы данных.

Задачи

1. Установить необходимые библиотеки.
2. Создать ОРМ для пользователя и адреса.
3. Инициализировать миграции.
4. Наполнить БД данными
5. Проверить запрос связанных данных.
6. Добавить новую таблицу и произвести миграцию.

# Инициализация БД

Установив все необходимые библиотеки создадим ОРМ для пользователя и адреса (Рисунок 1).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 1 – ОРМ для пользователя и адреса

Инициализируем миграции (Рисунок 2).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 - Инициализация миграций

После успешной инициализации в созданном файле alembic.ini изменим строку подключения к БД (Рисунок 3).



Рисунок 3 - Подключение к БД

Далее в созданном файле env.py подставим метаданные для миграций (Рисунок 4).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 - Метаданные для миграций

Создадим первую миграцию с помощью alembic (Рисунок 5).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 - Создание миграции

Применим полученную миграцию к нашей БД (Рисунок 6).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 - Применение миграции

Проверим успешное применение миграции (Рисунок 7).

Изображение выглядит как текст, снимок экрана, Шрифт, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7 - Проверка версии миграции

Из рисунка видно, что к нашей БД применилась созданная миграция.

Заполним БД данными пользователей и адресами (Рисунок 8).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 - Заполнение БД

Проверим заполнение БД (Рисунки 9-10).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 - Таблица пользователей

Изображение выглядит как снимок экрана, текст

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10 - Таблица адресов

# Запрос связанных данных

Успешно инициализировав и заполнив БД выполним запрос необходимых данных (Рисунок 11).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 11 - Запрос данных

Как видно из запроса таблицы успешно связаны.

# Редактирование созданной БД

Успешно проведя тесты нашей БД, создадим новые таблицы, первая будет содержать в себе название продукта и его id, а вторая, которая будет содержать информацию о заказах, в которых будет содержаться id пользователя, его адрес и продукт заказа (Рисунок 12 - 13).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 - Таблица продуктов

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 13 – Таблица заказов

После новой миграции, заполним таблицы данными (Рисунок 14 - 15).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 14 – Добавление продукции

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 15 – Добавление заказов

Заполненные таблицы (Рисунок 16-17).

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 16 - Таблица продуктов

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 17 - Таблица заказов

Проверим связь существующих таблиц (Рисунок 18).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 18 - Проверка запроса

Как видно из рисунка все данные по запросу выводятся корректно.

Ответы на вопросы

Основные подходы маппинга в SQLAlchemy: Классический с использованием специального класса Table и привязкой ORM к таблице через метод mapper. И декларативный, который позволяет определять классы в коде, объединяя определение модели и схемы таблицы в одном классе.

Alemic сохраняет текущую версию миграции в специальной таблице базы данных, называемой alembic\_version.

Три типа связей (один к одному, один ко многим, многие к многим).

Миграция базы данных представляет собой процесс внесения изменений в структуру существующей базы данных таким образом, чтобы поддерживать синхронизацию между состоянием приложения и базой данных.

Отношения "многие-ко-многим" в SQLAlchemy реализуются путем введения дополнительной связующей таблицы, которая служит мостиком между двумя основными таблицами.

Порядок действий:

1) Остановить все работы с БД.

2) Выяснить и визуализировать, где произошла "развилка" (например, используя alembic history --verbose).

3) Написать объединяющую (merge) миграцию через команду alembic merge, чтобы свести разошедшиеся ветви в одну.

4) Проверить корректность результата, протестировать изменения.

5) Зафиксировать и синхронизировать состояние всех сред, чтобы повторно не возник конфликт.

Вывод

В ходе работы была проведена работа инициализации базы данных с установкой нужных библиотек, была создана ОРМ для пользователей и адресов, также была проведена инициализация миграции.

Была добавлена новая таблица с обновлением данных.

Были проведены запросы из БД.