**Министерство науки и высшего образования Российской Федерации**

**ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»**

ОТЧЕТ

Лабораторная работа №2: Работа с SQLAlchemy и alembic

Студент Ярославцев А. А.

**ФИО студента**

Специальность (направление подготовки) 09.04.02 Информационные системы и технологии

Группа РИМ-150950

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.

**Инициализация БД**

Установил необходимые библиотеки:

* sqlalchemy - ORM для работы с БД
* alembic - система миграций базы данных
* psycopg2-binary - драйвер для PostgreSQL
* asyncpg - асинхронный драйвер для PostgreSQL

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Создаю ОРМ для пользователя и адреса**

Импортирую необходимые модули и создаю базовый класс для всех моделей.

Создаем модель пользователя и адреса с полями

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Инициализируем миграции**

Создаем структуру папок и файлов для управления миграциями базы данных

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

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Настройка подключения к БД**

Настроено подключение к PostgreSQL контейнеру с указанием правильных учетных данных и порта

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

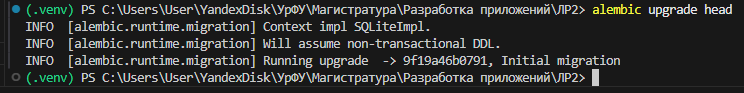
В env.py подставил метаданные для миграции **Изображение выглядит как текст, Шрифт, снимок экрана, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Создал миграцию **Изображение выглядит как текст, программное обеспечение, Шрифт, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.**

Применил к БД последнюю миграцию

****

**После этого наполнил БД данными**

1. Подключаемся к базе: **Изображение выглядит как текст, Шрифт, снимок экрана, линия

   Содержимое, созданное искусственным интеллектом, может быть неверным.**
2. Наполнил БД данными 5 пользователей и их адресами

**Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Запрос связанных данных**

В классе у нас создана связь между таблицами с помощью relationship. Попробуем получить и вывести данные из двух таблиц

****

– указывает ОРМ, которая связана с таблицой в БД

options – указывает дополнительные параметры запроса

**–** указывает на дополнительные данные, которые мы хотим получить в запросе

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Последующие работы с БД и миграции**

В ОРМ добавляем пользователю дополнительное строковое поле description. Добавляем дополнительную таблицу для продукции и заказов. Заказ должен в себе содержать информацию о пользователе, адресе доставки и продукции. Производим миграцию данных и добавляем 5 продукций и 5 заказов в БД.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Ответы на вопросы:**

1. Какие есть подходы маппинга в SQLAlchemy? Когда следует использовать каждый подход?

В SQLAlchemy существует три основных подхода к маппингу:

Classical Mapping (классический маппинг) - отдельное создание таблиц и классов с последующим связыванием через mapper(). Используется редко, в основном для легаси-проектов.

Declarative Mapping (декларативный маппинг) - наиболее популярный подход, когда классы наследуются от Base = declarative\_base(). Мы использовали его в работе, так как он интуитивно понятен и позволяет объединить описание таблицы и модели в одном классе.

Imperative Mapping (императивный маппинг) - гибридный подход, где таблица описывается отдельно, а затем связывается с классом. Полезен при сложных наследованиях или при интеграции с существующими базами данных.

1. Какие есть подходы маппинга в SQLAlchemy? Когда следует использовать каждый подход?

Alembic создает специальную таблицу alembic\_version в базе данных, где хранит хеш последней примененной миграции. При выполнении команды alembic upgrade head система сравнивает хеш в этой таблице с доступными миграциями и применяет только те, которые еще не были выполнены. Это обеспечивает идемпотентность миграций.

1. Какие есть подходы маппинга в SQLAlchemy? Когда следует использовать каждый подход?

В работе реализованы следующие связи:

One-to-Many (один-ко-многим): Пользователь (User) → Адреса (Addresses)

One-to-Many: Пользователь (User) → Заказы (Orders)

One-to-Many: Адрес (Address) → Заказы (Orders)

One-to-Many: Заказ (Order) → Элементы заказа (OrderItems)

One-to-Many: Продукт (Product) → Элементы заказа (OrderItems)

Фактически, связь между Заказами и Продуктами реализована как Many-to-Many через промежуточную таблицу OrderItems.

1. Какие есть подходы маппинга в SQLAlchemy? Когда следует использовать каждый подход?

Миграция базы данных - это процесс управления изменениями схемы БД в контролируемом и систематическом порядке. В нашей работе мы использовали миграции для:

Добавления новых таблиц (products, orders, order\_items)

Добавления поля description к users

Управления версиями схемы БД

Важность миграций:

Обеспечивают согласованность схемы БД между разными средами (dev/stage/prod)

Позволяют откатывать изменения при необходимости

Автоматизируют процесс обновления БД

Сохраняют историю изменений схемы

1. Как обрабатываются отношения многие-ко-многим в SQLAlchemy?

В нашей работе отношение Many-to-Many между Заказами и Продуктами реализовано через промежуточную таблицу OrderItems с дополнительными атрибутами (quantity, price). Это классический паттерн "ассоциативного объекта".

Альтернативный подход - использование relationship с параметром secondary, который указывает на простую промежуточную таблицу без дополнительных полей.

1. Каков порядок действий при возникновении конфликта версий в Alembic?

При конфликте версий в Alembic:

Анализ текущего состояния: alembic current - показывает текущую версию

Просмотр истории: alembic history - отображает цепочку миграций

Откат конфликтующих миграций: alembic downgrade -1

Синхронизация с актуальной версией: alembic upgrade head

При необходимости - ручное редактирование миграций в папке versions

Перегенерация миграции при серьезных конфликтах

**Вывод:**

В ходе выполнения лабораторной работы я освоил принципы работы с ORM SQLAlchemy и системой миграций Alembic для управления реляционными базами данных. На практике изучил механизмы создания связей между таблицами "один-ко-многим" и организацию сложных запросов к связанным данным. Полученные навыки позволяют эффективно проектировать и поддерживать структуру базы данных в Python-приложениях, что является важным аспектом в разработке информационных систем.