Министерство науки и высшего образования Российской Федерации

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

Институт радиоэлектроники и информационных технологий - РТФ

Лабораторная работа №2

**Работа с SQLAlchemy и alembic**

по дисциплине «Разработка приложений»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Исаков А.А.

Группа РИМ-150950

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

(подпись)

Екатеринбург 2025

# Исходные данные

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:

* Изучить и освоить работу с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python.
* Создать ORM-модели для пользователей, адресов, продукции и заказов с использованием различных типов связей между таблицами.
* Настроить и применить миграции базы данных при помощи Alembic для корректного обновления структуры БД.
* Наполнить базу данных начальными данными (5 пользователей, адресов, товаров и заказов) и реализовать запросы к связанным таблицам.
* Изучить работу с миграциями, управлением версиями и многократно-связанными таблицами для поддержки целостности данных и удобного развития приложения.

# Ход работы

## Инициализация БД

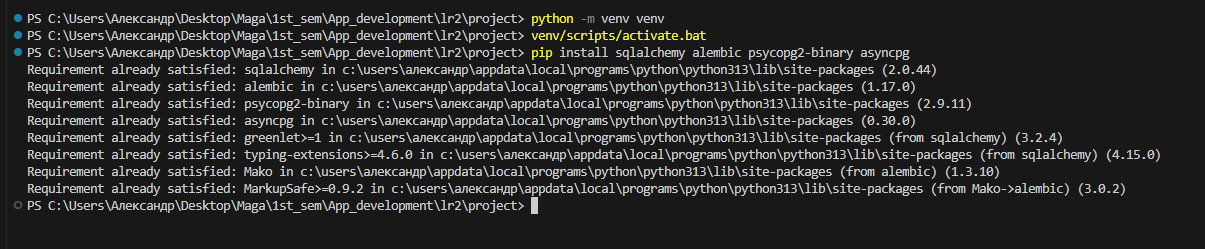
Было создано виртуальное окружение, в нем установлены необходимые библиотеки.  


Рисунок 1 – установка библиотек в виртуальном окружении

Создан файл models.py с моделями ORM.

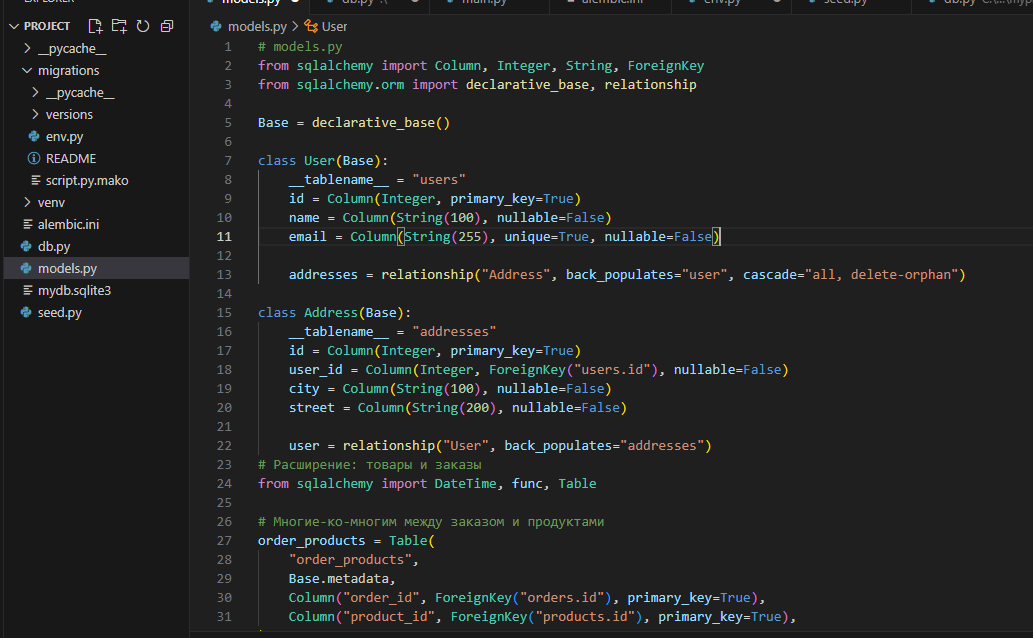


Рисунок 2 – часть файла models.py

После этого был создан файл db.py как точка подключения к БД и «фабрике» сессий.

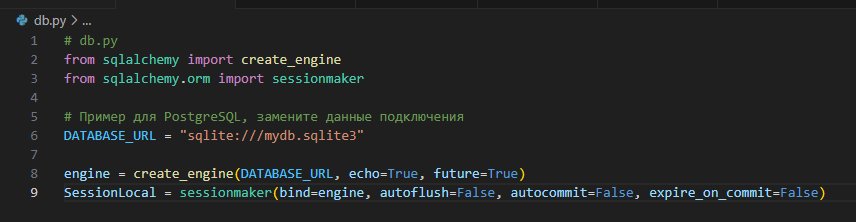


Рисунок 3 – файл db.py

Была инициирована миграция. В результате в директории программы добавился файл alembic.ini и папка migrations.

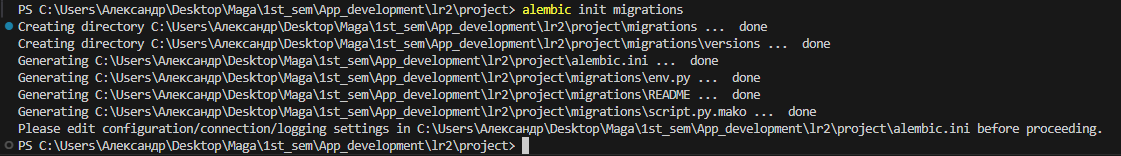


Рисунок 4 - инициация миграции

В env.py были подставлены метаданные для миграции (Base.metadata – каталог ORM-моделей). Создана миграция и применена к БД.

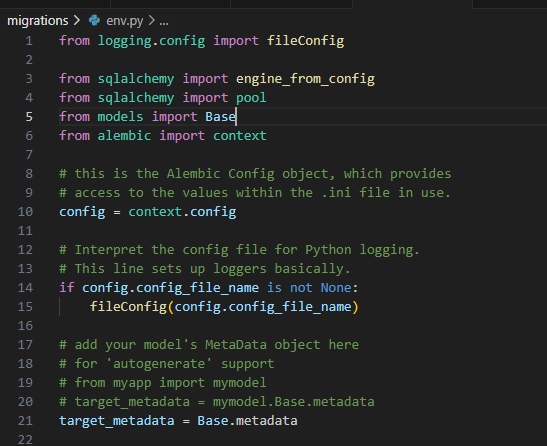


Рисунок 5 - редактирование env.py

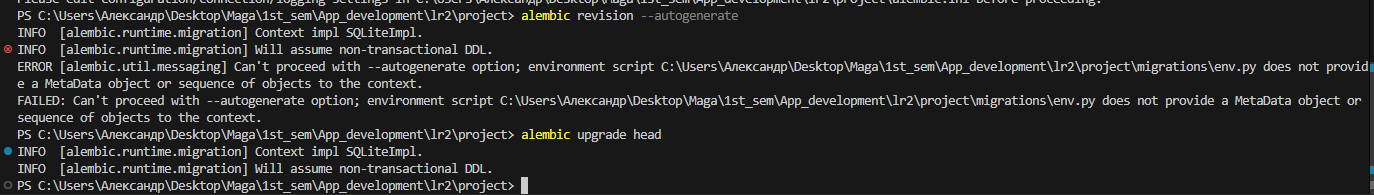


Рисунок 6 - создание миграции и применение миграции к БД

## Наполнение БД данными

Создан файл seed.py. В нем задана функция для заполнения БД пятью пользователями и адресами.

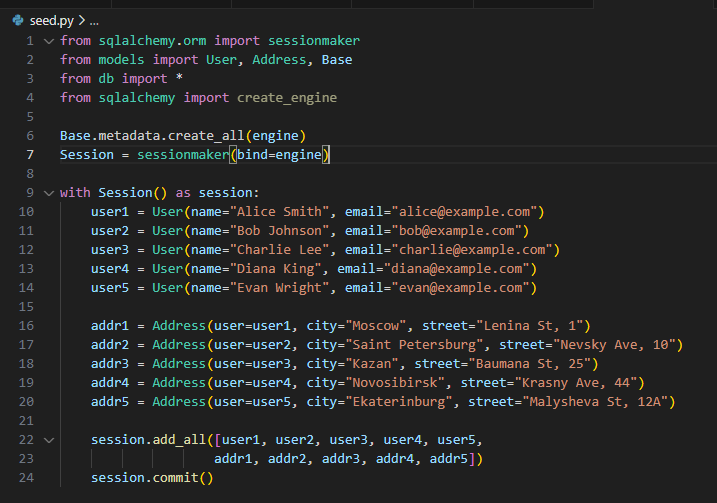


Рисунок 7 - состав файла seed.py

## Проверка записи данных в БД

После записи были сделаны выводы полей базы данных.

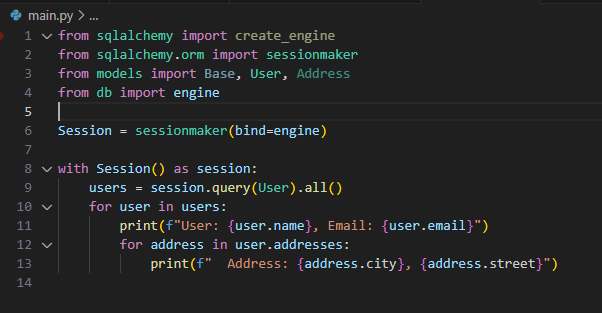


Рисунок 8 - содержимое файла main.py

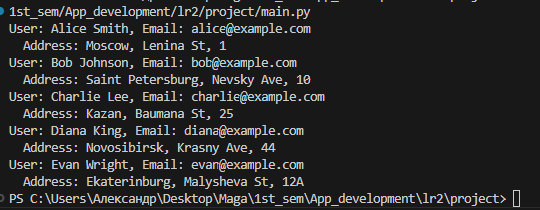


Рисунок 9 - запрос данных из БД о пользователях

## Добавление пользователям атрибута Description и таблицу для продукции и заказов

Было добавлено поле description в таблицу пользователей в файле models.py.

Созданы новая таблица для заказов и продукции.

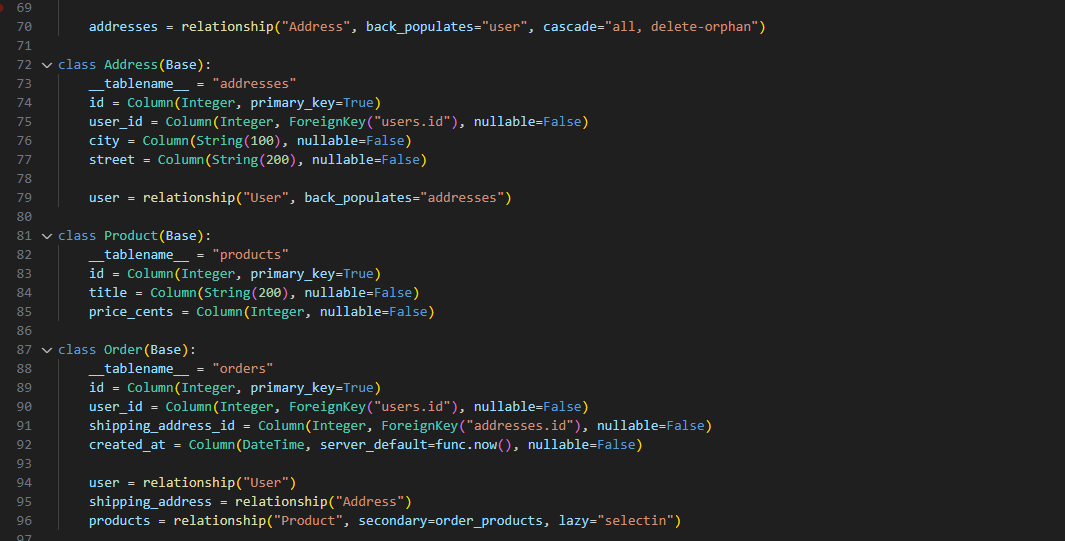


Рисунок 10 – измененный файл models.py

Также добавлено заполнение таблицы с продукцией и поля description у пользователей.

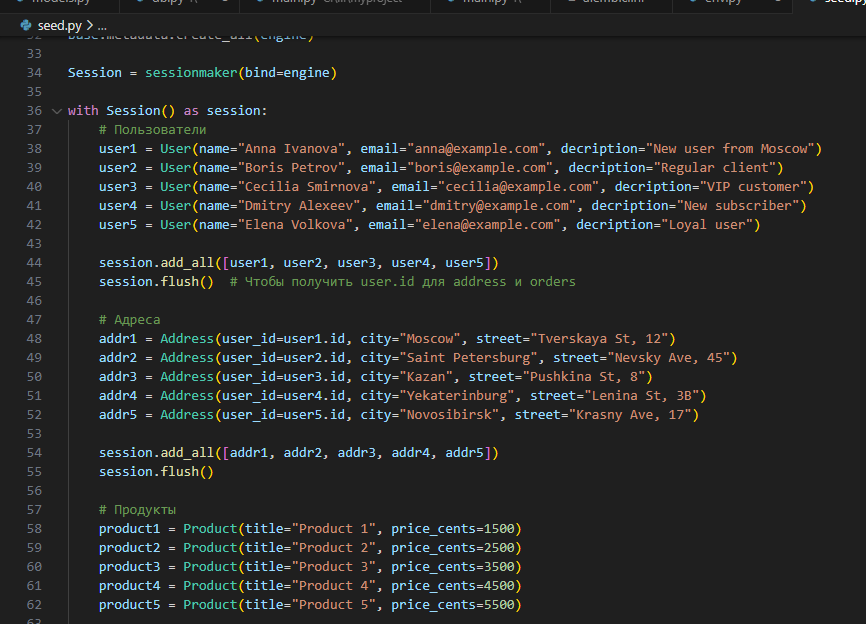


Рисунок 11 - заполнение таблиц (seed.py)

Была создана миграция и применены изменения.

Вывод объединенного содержимого по всем таблицам представлен на рисунках ниже.

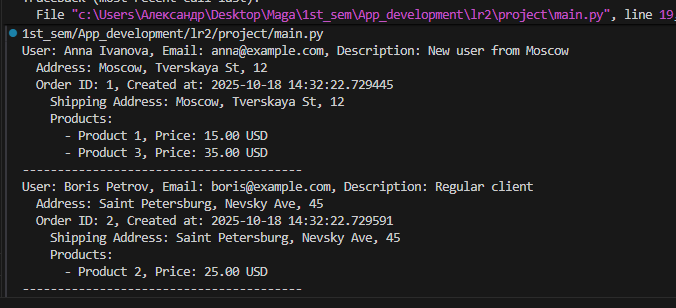


Рисунок 12 - вывод объединенных таблиц (1/3)

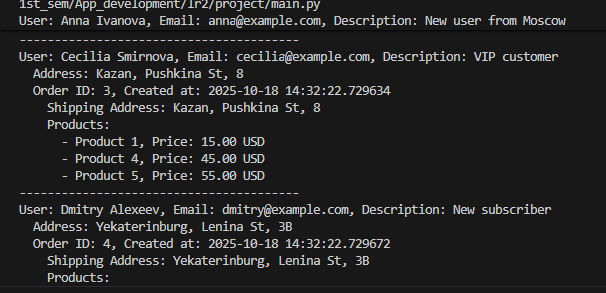


Рисунок 13 - вывод объединенных таблиц (2/3)

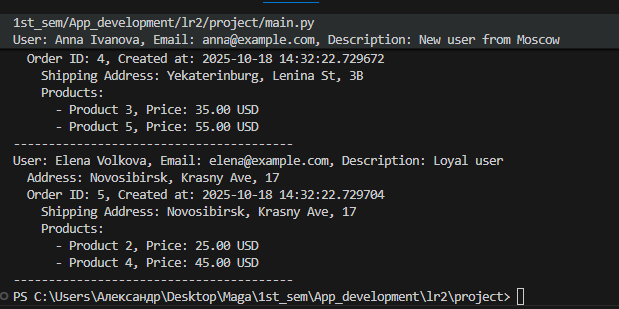


Рисунок 14 - вывод объединенных таблиц (3/3)

Содержимое файла main.py представлено на рисунке 15.

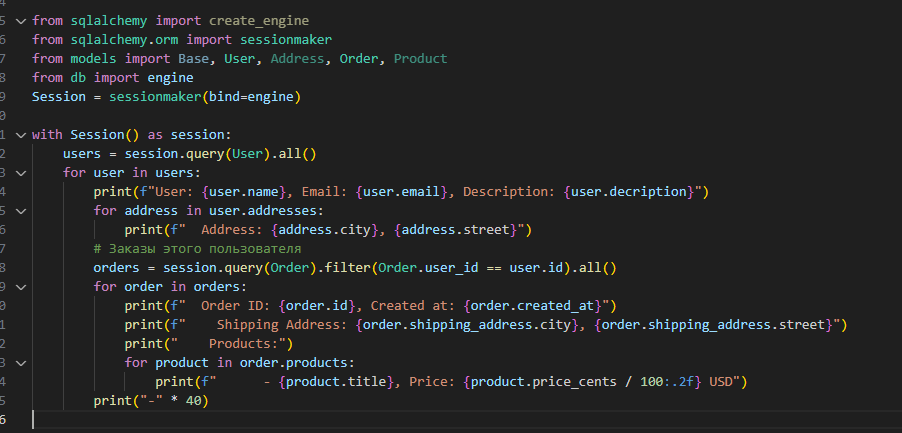


Рисунок 15 - файл main.py

# Ответы на вопросы

1. Декларативный, классический, автоматический.

Декларативный маппинг рекомендуют для новых проектов благодаря простоте и читаемости. Классический — для сложных и нестандартных случаев. Automap применяют при работе с уже существующими базами.

1. С помощью таблицы alembic\_version, которая обновляется при каждой миграции.
2. Один-ко-многим: между User и Address — один пользователь может иметь несколько адресов.

Многие-ко-многим: между Order и Product через промежуточную таблицу order\_products, где один заказ может содержать несколько продуктов, и один продукт может входить в несколько заказов.

1. Миграция базы данных — это процесс переноса данных и структуры из одной системы управления базами данных (СУБД) в другую, а также переезд из одного местоположения (например, с локальных серверов) в другое (например, в облако). Цель состоит в том, чтобы сохранить целостность и доступность данных при обновлении систем, переходе в облако или модернизации инфраструктуры.
2. Многие-ко-многим реализуются через вспомогательную таблицу (association table) без собственного класса, содержащую внешние ключи на обе связанные таблицы
3. -Определить, какие миграции конфликтуют (разные направления в истории).

-Разрешить конфликт вручную, например, создать "слияние" (merge) миграций с помощью Alembic (alembic merge) или исправить последовательность миграций.

-Применить согласованную последовательность миграций на всех окружениях.

-Проверить базу данных и откорректировать при необходимости.

# Вывод

В ходе лабораторной работы была изучена и освоена работа с библиотеками SQLAlchemy и Alembic для создания реляционных баз данных на Python и управления их структурой с помощью миграций.

Были успешно созданы ORM-модели для пользователей, адресов, продукции и заказов с соответствующими типами связей (один-ко-многим, многие-ко-многим). Выполнена настройка и применение миграций, что позволило корректно обновлять структуру базы данных.

Также реализован процесс наполнения базы данными, включая создание пользователей, адресов, товаров и заказов с корректной связью между таблицами.