# **Лабораторная работа №2: Работа с SQLAlchemy и alembic**

РИМ-150950

Жунёв Андрей

**Цель работы**

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

**Ход работы:**

**Инициализируем проект через UV**

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Устанавливаем необходимые библиотеки

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

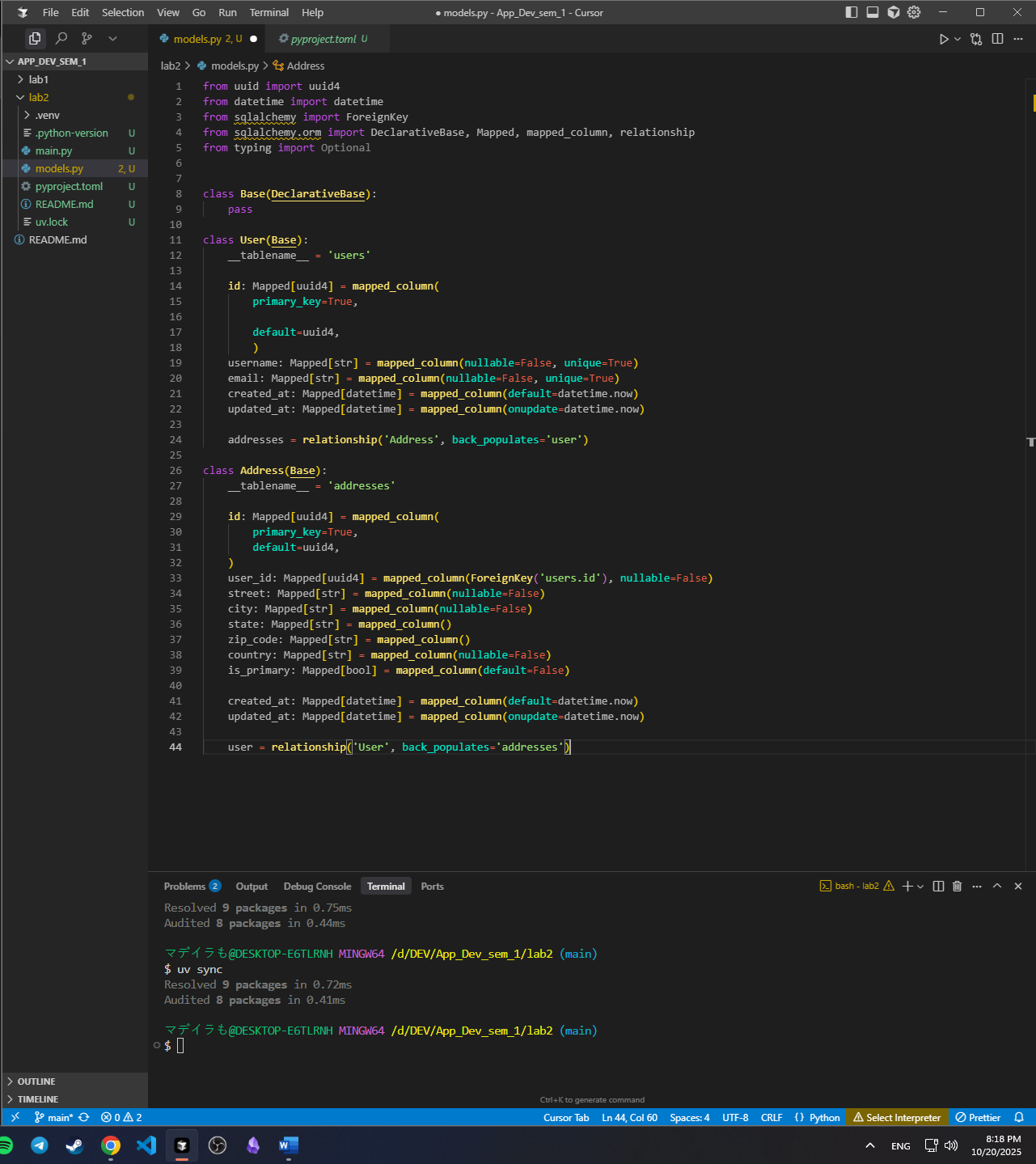
SQLAlchemy – основная библиотека дял работы с базами данных через ORM

Alembic – инструмент для управления миграциями базы данных

psycorpg2-binary – драйвер для подключения к PostgreSQL(синхронный)

asyncpg – драйвер для асинхронной работы с PostgreSQL

Создание ОРМ для пользователя и адреса



.env файл содержит следующие переменные:  
POSTGRES\_USER=admin

POSTGRES\_PASSWORD=admin

POSTGRES\_DB=lab\_db2

PGADMIN\_DEFAULT\_EMAIL=admin@lab2.com

PGADMIN\_DEFAULT\_PASSWORD=admin

БД развернута. Таблицы заранее не создаются (сначала создал заранее, добавив скрипт инициализации, не понял задание – затем переделал)

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Настройка миграций с Alembic**

Инициализировал миграции, заменил строку подключения к БД в alembic.ini

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Теперь настраиваем env.py. Сначала импортируем модели, затем указываем метаданные для автогенерации миграций.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Создаем миграцию

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Применяем к БД последнюю миграцию

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Как видим, создались таблицы с верными колонками

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

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

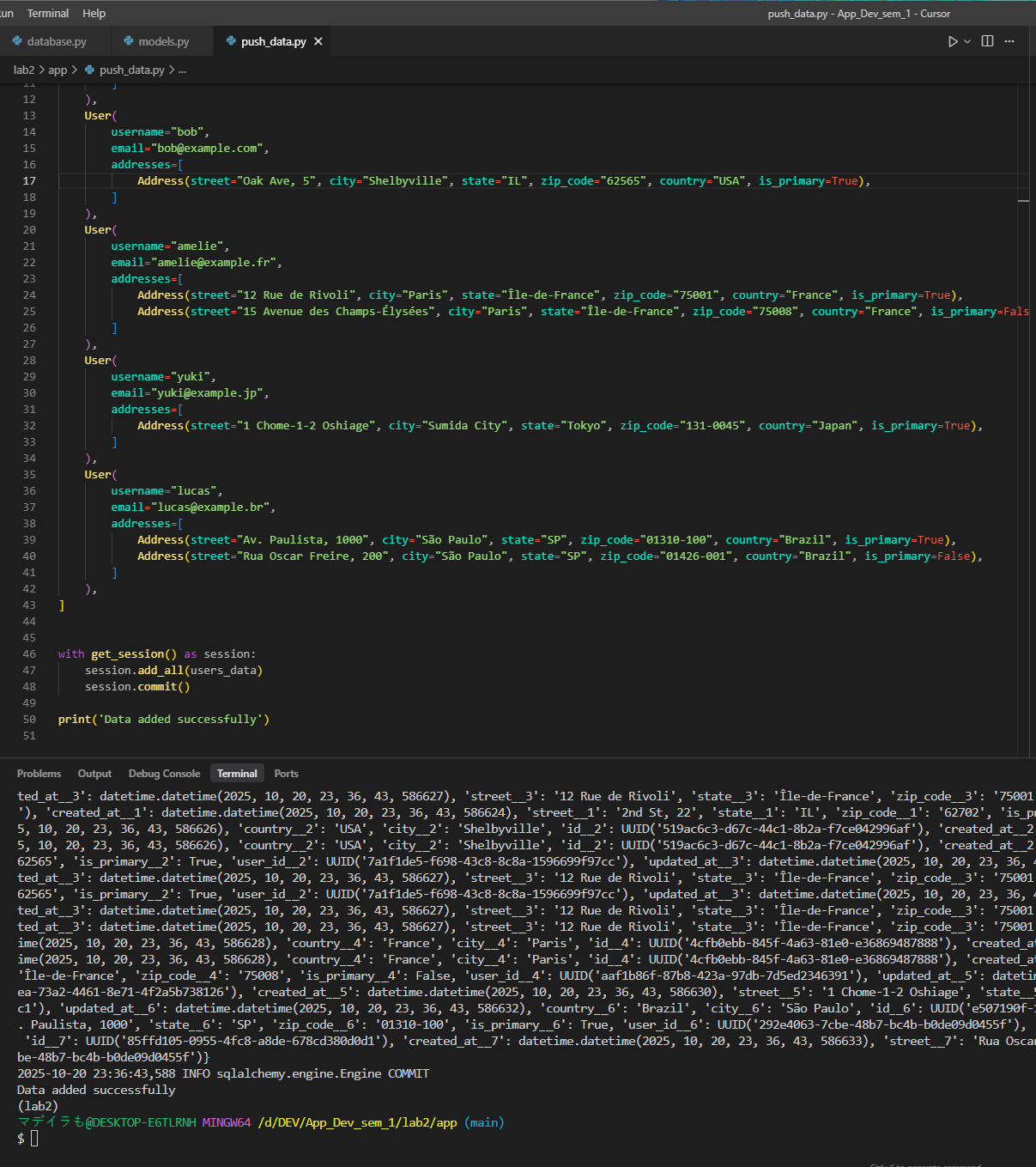
Содержимое, созданное искусственным интеллектом, может быть неверным.

НАПОЛНЕНИЕ БАЗЫ ДАННЫМИ

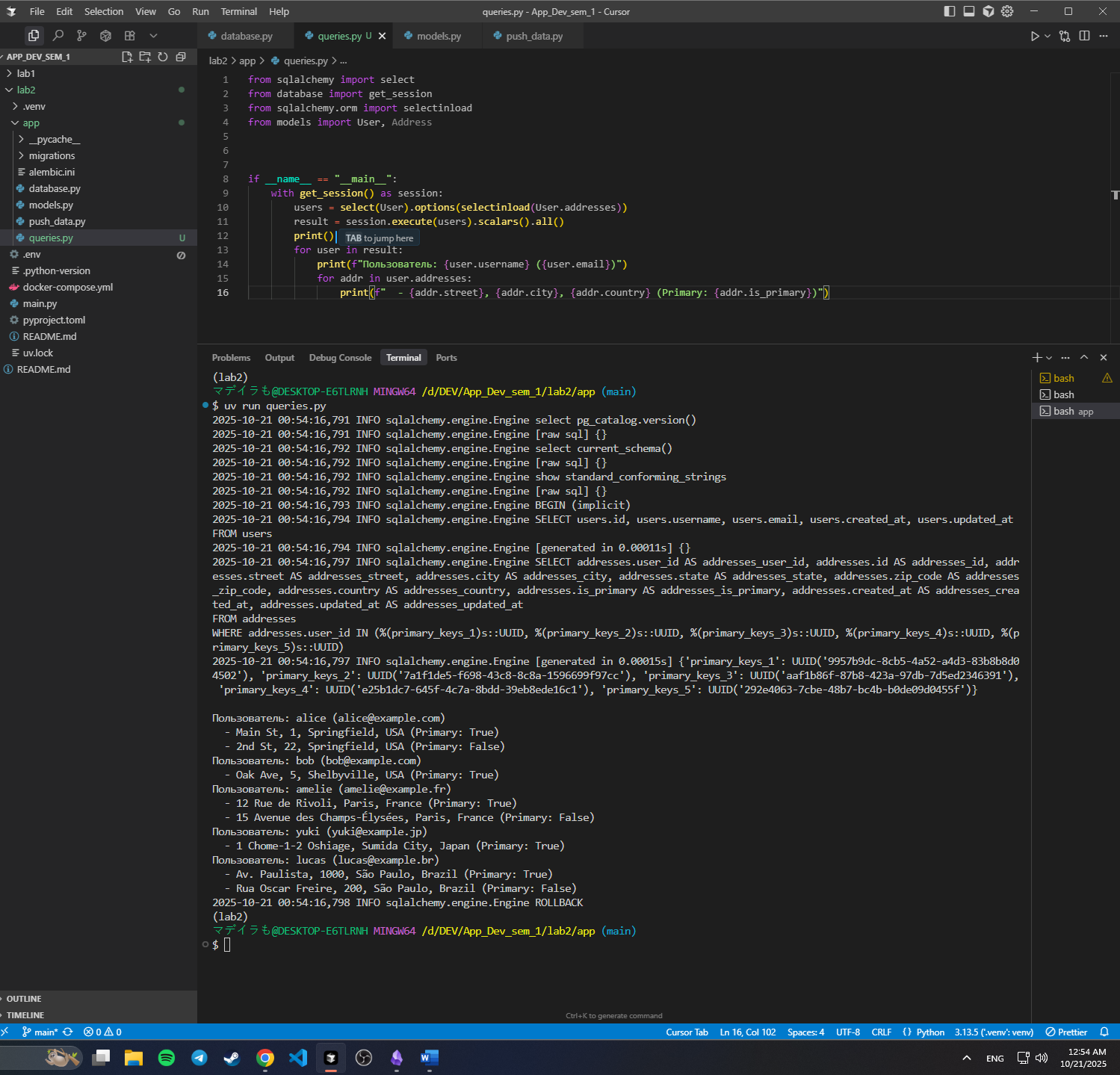
Создадим фабрику для наполнения и добавим 5 пользователей

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.



## Запрос связанных данных



## Последующие работы с БД и миграции

Добавил классы Product и Order, добавил поле description для класса User, добавил недостающие поля связей

Изображение выглядит как текст, снимок экрана, программное обеспечение

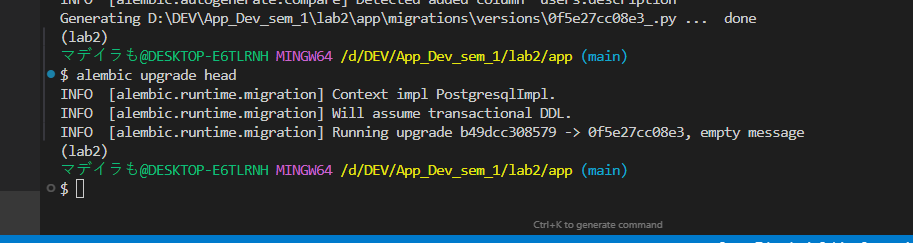
Содержимое, созданное искусственным интеллектом, может быть неверным.

Создание новой миграции

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Применяем миграцию

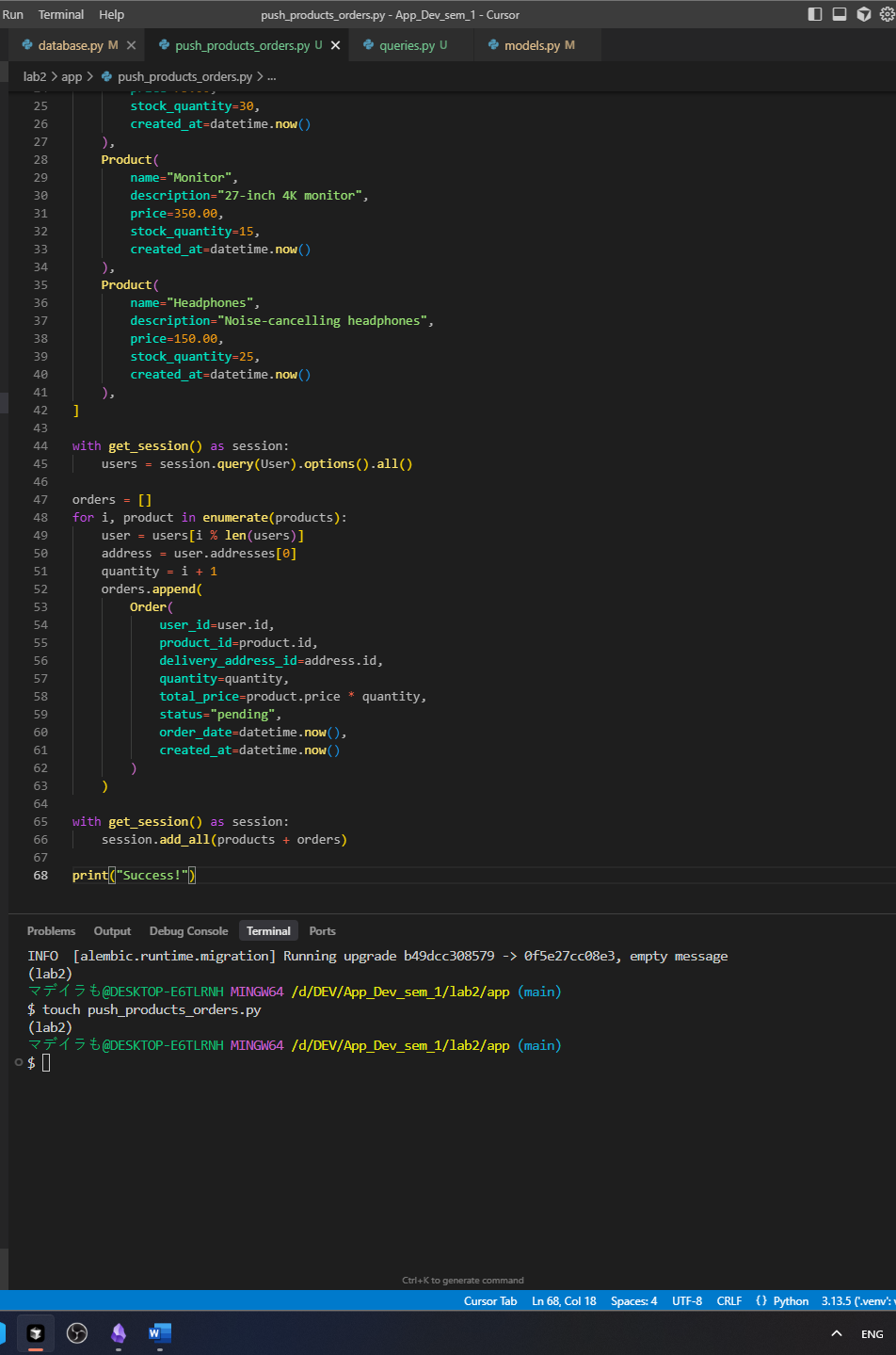


Видим, что миграция прошла успешно

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Создал скрипт для заполнения новых таблиц данными



Доработал get\_session() обернув в декоратор контекстного менеджера

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Успешно добавлены 5 заказов и 5 продуктов

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Ответы на вопросы:

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

Ответ: В SQLAlchemy есть два основных подхода к маппингу:

Декларативный описывает структуру таблиц и связей прямо в теле класса, строится на наследовании от базового класса, удобен для новых проектов и быстрого прототипирования.

Императивный (классический) использует функцию mapper() для привязки классов к отдельным объектам Table, подходит для сложных или импортируемых схем.

Для новых проектов чаще всего рекомендуют декларативный подход – он наглядней и выглядит лаконичней (большинство современных библиотек рекомендует его). Классический стиль же, когда требуется сложная и тонкая настройка маппинга.

1. **Как Alembic отслеживает текущую версию базы данных?**

Ответ: Alembic сохраняет информацию о текущей схеме в служебной таблице alembic\_version. При применении миграций Alembic сравнивает метаданные моделей (target\_metadata) с фактической структурой базы и генерирует скрипты миграций. Каждая миграция получает уникальный идентификатор (revision ID). Таблица alembic\_version содержит этот ID последней успешно выполненной миграции. При выполнении alembic upgrade Alembic находит неприменённые миграции по цепочке down\_revision–revision, выполняет их последовательно и обновляет запись в alembic\_version.

1. **Какие типы связей между таблицами вы реализовали в данной работе?**

Ответ: Один ко многим (пользователь, адрес или пользователь заказ – многие к одному обратная связь этой), Один к одному (между заказом и адресом доставки, реализована через внешний ключ)

1. **Что такое миграция базы данных и почему она важна?**

Ответ: миграция базы данных - это скрипт, который описывает изменения в структуре таблиц (создание, удаление, изменение столбцов, связей и индексов). Миграции гарантируют, что изменения схемы версионируются и могут быть автоматически применены или откатаны.

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

Ответ: отношение «многие ко многим» реализуется через вспомогательную (association) таблицу, содержащую два внешних ключа. В SQLAlchemy её можно определить либо как отдельную ORM-модель, либо как экземпляр Table (я мог так реализовать связь между заказами и продуктами, но не стал)

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

Ответ: Для диагностики миграций Alembic необходимо проверить текущий revision\_id в таблице alembic\_version и сравнить его с миграциями в папке migrations/versions. В случае необходимости следует выполнить откат до нужного состояния базы с помощью команды alembic downgrade <revision\_id> или до базового состояния через alembic downgrade base. После этого стоит проверить и при необходимости исправить зависимости между миграциями, объединив или переименовав конфликтующие файлы. При значительных изменениях может потребоваться удалить проблемные миграции, сгенерировать новые с помощью alembic revision --autogenerate и применить их повторно командой alembic upgrade head. Важно убедиться, что таблица alembic\_version обновилась, и структура базы данных соответствует моделям. Если в проекте появились несколько веток миграций, их следует объединить с помощью alembic merge heads, создав новую миграцию для разрешения конфликта.