План урока

- Напишем большой запрос на языке SQL и перенесем их на язык SQLAlchemy
- Поймем, как конвертировать модели Алхимии в Pydantic схемы

Погружение в SQLAlchemy и SQL

В этом уроке мы продолжим работу с Алхимией и научимся писать сложные запросы. Я буду честен с вами и скажу, что в реальных проектах, когда код должен быть написан уже вчера, зачастую прибегают к использованию сырых SQL запросов (они создаются так: `query = text("SELECT * FROM hotels")`). И я говорю лишь о тех запросах, которые не используют параметры, передаваемые клиентом, так как такие запросы подвержены SQL-инъекциям!

Запрещено делать так (подвержено SQL-инъекции):

```
from sqlalchemy import text
from XXX import async_session_maker
from YYY import router

@router.get("/users/{user_id}")
def get_user(user_id: int):
    query = text(f"SELECT * FROM users WHERE id = {user_id}")  # <--
запрещено вставлять пользовательские данные в запрос
    async with async_session_maker as session():
        result = await session.execute(query)
        return result.one()
```

Никогда не вставляйте в сырой SQL код пользовательские параметры, пишите запросы с вставкой пользовательских параметров через SQLAlchemy ORM.

Несмотря на вышесказанное, SQLAlchemy все-таки используется, когда руки доходят до рефакторинга кода или на его написание выделяется достаточное количество времени. SQLAlchemy в связке с современными IDE может ускорять разработку:

- Вы можете в кратчайшие сроки сменить PostgreSQL на MySQL или ClickHouse или любую другую СУБД, для которой есть нужная библиотека-коннектор
- Алхимия указывает на ошибки при написании кода
- Вы никогда не ошибетесь в названии стоблца или талбицы при использовании ORM