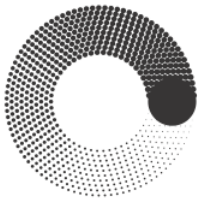
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 9

Дисциплина: Backend-разработка (Часть 2 - Python)

Тема: Работа с базой данных через SQLAlchemy

Выполнил(а):
студент(ка) группы: 221-3711

Фамилия И.О. До Дык Зунг
(Фамилия И.О.)

Проверил:

_____
(Фамилия И.О., степень, звание)

Москва

2024

Отчет: Работа с базой данных через SQLAlchemy и интеграция с FastAPI

Цель:

Целью данного задания является освоение основных принципов работы с базой данных с использованием библиотеки **SQLAlchemy**. Задание охватывает подключение к базе данных, создание таблиц, выполнение операций CRUD (создание, чтение, обновление, удаление) и интеграцию с веб-приложением на **FastAPI**.

Часть 1: Подключение к базе данных и создание таблиц

Выбор базы данных:

Для выполнения задания выбрана **SQLite** как простая реляционная база данных для разработки и тестирования. В реальной среде можно использовать другие базы данных, такие как **PostgreSQL** или **MySQL**, в зависимости от требований проекта.

Установка зависимостей:

Для работы с **SQLAlchemy** и выбранной базой данных необходимо установить следующие библиотеки:

pip install sqlalchemy sqlite

Создание модели данных:

1. **Таблица Users**:
   - id: Целое число, первичный ключ, автоинкремент.
   - username: Строка, уникальное значение.
   - email: Строка, уникальное значение.
   - password: Строка.
2. **Таблица Posts**:
   - id: Целое число, первичный ключ, автоинкремент.
   - title: Строка.
   - content: Текст.
   - user_id: Целое число, внешний ключ, ссылающийся на поле id таблицы Users.

Пример кода для создания таблиц:

```
from sqlalchemy import create_engine, Column, Integer, String, Text, ForeignKey
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker

# Создаем базовый класс для всех моделей
Base = declarative_base()

# Модель Users
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True, autoincrement=True)
    username = Column(String, unique=True)
    email = Column(String, unique=True)
    password = Column(String)
```

```
    # Связь с таблицей Posts
    posts = relationship('Post', backref='author')

# Модель Posts
class Post(Base):
    __tablename__ = 'posts'

    id = Column(Integer, primary_key=True, autoincrement=True)
    title = Column(String)
    content = Column(Text)
    user_id = Column(Integer, ForeignKey('users.id'))

# Подключение к базе данных SQLite
DATABASE_URL = "sqlite:///./test.db"
engine = create_engine(DATABASE_URL,
connect_args={"check_same_thread": False})

# Создание таблиц
Base.metadata.create_all(bind=engine)
```

Этот код создает две таблицы — **Users** и **Posts** — и устанавливает связь между ними. Внешний ключ user_id в таблице **Posts** ссылается на поле id в таблице **Users**.

Часть 2: Взаимодействие с базой данных

Добавление данных:

Для добавления данных в таблицы **Users** и **Posts**, создадим сессию и вставим несколько записей.

```
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
session = SessionLocal()

# Добавление пользователей
user1 = User(username="user1", email="user1@example.com",
password="password123")
user2 = User(username="user2", email="user2@example.com",
password="password456")

session.add(user1)
session.add(user2)

# Добавление постов
post1 = Post(title="Post 1", content="Content of Post 1", user_id=user1.id)
post2 = Post(title="Post 2", content="Content of Post 2", user_id=user2.id)

session.add(post1)
```

```
session.add(post2)

# Сохраняем данные в базе
session.commit()
```

Извлечение данных:

1. **Извлечение всех пользователей**:

```
users = session.query(User).all()
for user in users:
    print(user.username, user.email)
```

2. **Извлечение всех постов с информацией о пользователях**:

```
posts = session.query(Post).join(User).all()
for post in posts:
    print(post.title, post.content, post.author.username)
```

3. **Извлечение постов конкретного пользователя**:

```
user_posts = session.query(Post).filter(Post.user_id == user1.id).all()
for post in user_posts:
    print(post.title, post.content)
```

Обновление данных:

1. **Обновление email одного из пользователей**:

```
user_to_update = session.query(User).filter(User.username == "user1").first()
user_to_update.email = "new_email@example.com"
session.commit()
```

2. **Обновление содержимого поста**:

```
post_to_update = session.query(Post).filter(Post.title == "Post 1").first()
post_to_update.content = "Updated content for Post 1"
session.commit()
```

Удаление данных:

1. **Удаление поста**:

```
post_to_delete = session.query(Post).filter(Post.title == "Post 2").first()
session.delete(post_to_delete)
session.commit()
```

2. **Удаление пользователя и всех его постов**:

```
user_to_delete = session.query(User).filter(User.username == "user1").first()
session.delete(user_to_delete)
session.commit()
```

Часть 3: Интеграция с веб-приложением на FastAPI

Установка зависимостей для FastAPI:

pip install fastapi uvicorn

Создание приложения FastAPI:

1. **Создание экземпляра приложения FastAPI**:

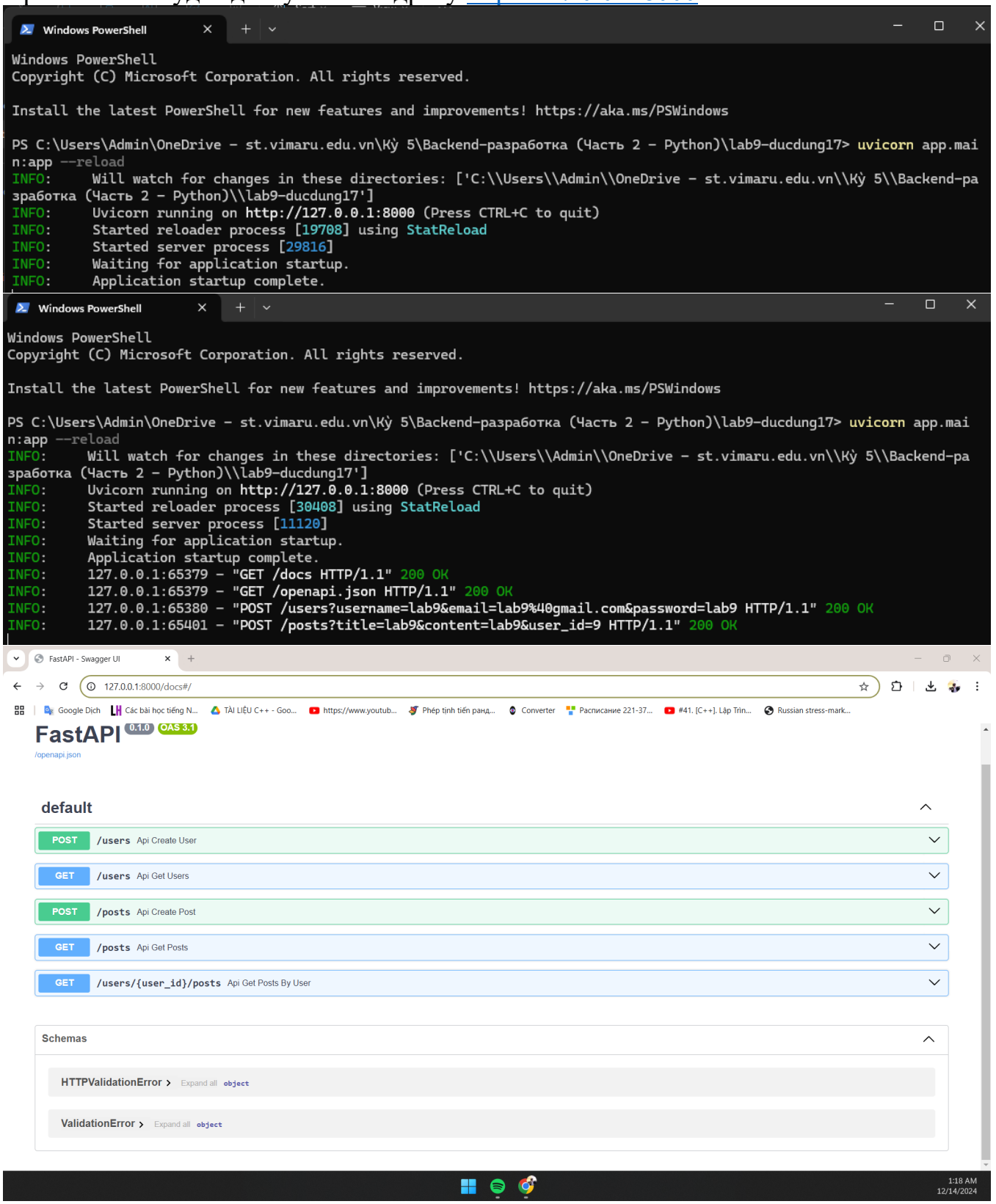
```
from fastapi import FastAPI, Depends
from sqlalchemy.orm import Session

app = FastAPI()

# Получение сессии базы данных
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# Маршрут для получения списка пользователей
@app.get("/users")
def get_users(db: Session = Depends(get_db)):
    users = db.query(User).all()
    return users

# Маршрут для создания нового пользователя
@app.post("/users")
def create_user(username: str, email: str, password: str, db: Session =
Depends(get_db)):
    user = User(username=username, email=email, password=password)
    db.add(user)
    db.commit()
    return {"message": "User created successfully!"}

# Маршрут для создания нового поста
@app.post("/posts")
def create_post(title: str, content: str, user_id: int, db: Session = Depends(get_db)):
    post = Post(title=title, content=content, user_id=user_id)
    db.add(post)
    db.commit()
    return {"message": "Post created successfully!"}
```

Запуск приложения:

Для запуска FastAPI-приложения используем команду:

uvicorn main:app --reload

Приложение будет доступно по адресу http://127.0.0.1:8000.



Заключение:

В результате выполнения задания была продемонстрирована интеграция библиотеки **SQLAlchemy** с веб-приложением на **FastAPI**. Мы научились подключаться к базе данных, создавать таблицы, выполнять CRUD-операции и интегрировать это в веб-приложение для реализации функционала управления пользователями и постами.