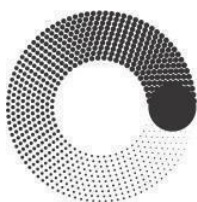


**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ**

**Факультет информационных технологий  
Кафедра Информатики и информационных технологий**

**направление подготовки 09.03.02 «Информационные системы и  
технологии»,**

**профиль «Программное обеспечение игровой компьютерной  
индустрии»**

**ЛАБАТОРНАЯ РАБОТА № 9**

**Дисциплина: BackEnd-разработка**

**Выполнил:** студент группы 221-3710

Бахвалов А.А

**Дата, подпись** \_\_\_\_\_  
20.09.2024\_ \_\_\_\_\_

(Дата)

(Подпись)

**Проверила:** \_\_\_\_\_

**(Оценка)**

**Замечания:**

**Дата, подпись** \_\_\_\_\_  
\_\_\_\_\_

(Дата)

(Подпись)

## Листинг

Чтобы запустить сервер в консоли напишите `uvicorn main:app --reload` предварительно указав путь с проектом через `cd`  
Чтобы выйти `ctrl+c`

### Main.py

```
from fastapi import FastAPI, Depends, HTTPException
```

```
from sqlalchemy.orm import Session
```

```
from . import models, schemas, crud
```

```
from .database import engine, Base, get_db
```

```
app = FastAPI()
```

```
Base.metadata.create_all(bind=engine)
```

```
@app.post("/users/", response_model=schemas.User)
```

```
def create_user(user: schemas.UserCreate, db: Session = Depends(get_db)):
```

```
    return crud.create_user(db=db, user=user)
```

```
@app.get("/users/", response_model=list[schemas.User])
```

```
def read_users(db: Session = Depends(get_db)):
```

```
    return crud.get_users(db=db)
```

```
@app.put("/users/{user_id}/")
```

```
def update_user(user_id: int, email: str, db: Session = Depends(get_db)):
```

```
    updated_user = crud.update_user_email(db, user_id, email)
```

```
    if not updated_user:
```

```
        raise HTTPException(status_code=404, detail="User not found")
```

```
    return updated_user
```

```
@app.delete("/users/{user_id}/")
```

```
def delete_user(user_id: int, db: Session = Depends(get_db)):
```

```
    crud.delete_user(db, user_id)
```

```
    return {"message": "User deleted"}
```

```
    email: str
```

```
    password: str
```

```
class RegisterRequest(BaseModel):
```

```
    username: str
```

```
    email: str
```

```
    password: str
```

## schemas.py

```
from pydantic import BaseModel
from typing import List, Optional
```

```
class PostBase(BaseModel):
    title: str
    content: str
```

```
class PostCreate(PostBase):
    user_id: int
```

```
class Post(PostBase):
    id: int
    user_id: int
```

```
class Config:
    orm_mode = True
```

```
class UserBase(BaseModel):
    username: str
    email: str
```

```
class UserCreate(UserBase):
    password: str
```

```
class User(UserBase):
    id: int
    posts: List[Post] = []
```

```
class Config:
    orm_mode = True
```

## models.py

```
from sqlalchemy import Column, Integer, String, Text, ForeignKey
from sqlalchemy.orm import relationship
from .database import Base
```

```
class User(Base):
    __tablename__ = "users"
    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True, nullable=False)
    email = Column(String, unique=True, index=True, nullable=False)
    password = Column(String, nullable=False)
```

```
posts = relationship("Post", back_populates="user")
```

```
class Post(Base):
    __tablename__ = "posts"
    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, nullable=False)
    content = Column(Text, nullable=False)
    user_id = Column(Integer, ForeignKey("users.id"), nullable=False)
    user = relationship("User", back_populates="posts")
```

## database.py

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
```

```
DATABASE_URL = "sqlite:///./test.db"
```

```
engine = create_engine(DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()
```

```
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

## crud.py

```
from sqlalchemy.orm import Session
from . import models, schemas
```

```
def create_user(db: Session, user: schemas.UserCreate):
    db_user = models.User(username=user.username, email=user.email, password=user.password)
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return db_user
```

```
def get_users(db: Session):
    return db.query(models.User).all()
```

```
def update_user_email(db: Session, user_id: int, new_email: str):
```

```

user = db.query(models.User).filter(models.User.id == user_id).first()
if user:
    user.email = new_email
    db.commit()
return user

```

```

def delete_user(db: Session, user_id: int):
    user = db.query(models.User).filter(models.User.id == user_id).first()
    if user:
        db.delete(user)
        db.commit()
    return user

```

```

PS C:\Users\Worci\Desktop\BackEnd\9> uvicorn app:main --reload
INFO: Will watch for changes in these directories: ['C:\Users\Worci\Desktop\BackEnd\9']
INFO: Will watch for changes in these directories: ['C:\Users\Worci\Desktop\BackEnd\9']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [22368] using StatReload
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [22368] using StatReload
C:\Users\Worci\AppData\Local\Programs\Python\Python312\Lib\site-packages\pydantic\_internal\_config.py:345: UserWarning: Valid config keys have changed in V2:
INFO: Started reloader process [22368] using StatReload
C:\Users\Worci\AppData\Local\Programs\Python\Python312\Lib\site-packages\pydantic\_internal\_config.py:345: UserWarning: Valid config keys have changed in V2:
C:\Users\Worci\AppData\Local\Programs\Python\Python312\Lib\site-packages\pydantic\_internal\_config.py:345: UserWarning: Valid config keys have changed in V2:
* 'orm_mode' has been renamed to 'from_attributes'
  warnings.warn(message, UserWarning)
INFO: Started server process [11464]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:65222 - "GET /docs HTTP/1.1" 200 OK
INFO: Application startup complete.
INFO: 127.0.0.1:65222 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:65222 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:65222 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:65223 - "POST /users/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65235 - "GET /users/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65236 - "GET /users/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65245 - "PUT /users/1/?email=albebra2240mail.ru HTTP/1.1" 200 OK
INFO: 127.0.0.1:65247 - "GET /users/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65249 - "DELETE /users/1/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65249 - "DELETE /users/1/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65249 - "GET /users/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65250 - "POST /users/ HTTP/1.1" 200 OK
INFO: 127.0.0.1:65251 - "GET /users/ HTTP/1.1" 200 OK
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [11464]
INFO: Stopping reloader process [22368]
PS C:\Users\Worci\Desktop\BackEnd\9>

```

C: > Users > Norci > Desktop > BackEnd > 9 > test.db

Filter 2 tables...		Rows: 1
TABLES		
posts	id	username
users	email	password
	1	abebra
	2	

Как и требовалось можно удалять редактировать добавлять пользователей в базу данных