sql

📚 FastAPI + SQLAlchemy (with SQLite) Full CRUD App Tutorial

This course teaches you how to:

✅ Set up a FastAPI app
✅ Use SQLAlchemy ORM with SQLite
✅ Build full CRUD endpoints
✅ Structure your project professionally
✅ Understand each part with clear explanations

🗂️ Project Structure

```
fastapi_sqlalchemy_example/
├── main.py          # FastAPI app with routes
├── db/database.py     # SQLAlchemy & SQLite setup
├── db/dep.py.            # get_db
├── models/users.py       # ORM model classes
├── schemas/users.py      # Pydantic schemas
├── crud/users.py         # DB logic (CRUD operations)
```

1️⃣ `database.py` — Database Setup

```python
# database.py
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

SQLALCHEMY_DATABASE_URL = "sqlite:///./test.db"

engine = create_engine(
    SQLALCHEMY_DATABASE_URL, connect_args={"check_same_thread": False}
)
```

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

Base = declarative_base()
```

🧠 Teaching Notes:

- `engine` connects SQLAlchemy to SQLite.
- `SessionLocal()` creates DB sessions per request.
- `Base` is the parent for all ORM models.

2️⃣ `models.py` — ORM Models

```
# models.py
from sqlalchemy import Column, Integer, String
from database import Base

class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, index=True)
    email = Column(String, unique=True, index=True)
```

🧠 Teaching Notes:

- Each model maps to a table.
- Use `__tablename__` to define the table name.
- Use types (`Integer` , `String`) to define columns.

3️⃣ `schemas.py` — Pydantic Data Schemas

```
# schemas.py
from pydantic import BaseModel

class UserCreate(BaseModel):
    name: str
    email: str

class UserUpdate(BaseModel):
```

```
    name: str | None = None
    email: str | None = None

class User(BaseModel):
    id: int
    name: str
    email: str

    class Config:
        orm_mode = True
```

🧠 Teaching Notes:

- `UserCreate` : input data for POST/PUT.
- `User` : output data for responses.
- `orm_mode = True` : enables reading from SQLAlchemy models.

4️⃣ `crud.py` — CRUD Logic

```
# crud.py
from sqlalchemy.orm import Session
from fastapi import HTTPException
import models, schemas

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

def get_user(db: Session, user_id: int):
    return db.query(models.User).filter(models.User.id == user_id).first()

def get_users(db: Session, skip: int = 0, limit: int = 10):
    return db.query(models.User).offset(skip).limit(limit).all()

def update_user(db: Session, user_id: int, user_update:
schemas.UserCreate):
    user = get_user(db, user_id)
    if not user:
        raise HTTPException(status_code=404, detail="User not found")
    user.name = user_update.name
    user.email = user_update.email
    db.commit()
```

```
    db.refresh(user)
    return user

def delete_user(db: Session, user_id: int):
    user = get_user(db, user_id)
    if not user:
        raise HTTPException(status_code=404, detail="User not found")
    db.delete(user)
    db.commit()
    return {"ok": True, "message": f"User {user_id} deleted"}
```

🧠 Teaching Notes:

- CRUD is isolated for clean logic and testability.
- You use SQLAlchemy ORM querying: `.query().filter().first()`.

Routes

```
# Routes

from fastapi import APIRouter, status, Depends
from schema.users import UserCreate, UserUpdate
from db.dep import get_db
from sqlalchemy.orm import Session
from crud.users import (
create_user,
get_user,
get_users,
update_user,
user_delete
)
from uuid import UUID

router = APIRouter()

@router.post("/create")
def create(user: UserCreate, db: Session = Depends(get_db)):
resp = create_user(db=db, user_create=user)
return resp

@router.get("/get/{user_id}")
def user_get(user_id: UUID, db: Session = Depends(get_db)):
resp = get_user(user_id=user_id, db=db)
return resp
```

```python
@router.get("/getAll")
def users_get_all(db: Session = Depends(get_db)):
resp = get_users(db=db)
return resp

@router.patch("/update/{user_id}")
def user_update(
user_id: UUID,
user_update: UserUpdate,
db: Session = Depends(get_db)
):

resp = update_user(db=db, user_id=user_id, user_update=user_update)
return resp

@router.delete("/delete")
def delete_user(user_id: UUID, db: Session = Depends(get_db)):
resp = user_delete(user_id=user_id, db=db)
return resp
```

5️⃣ `main.py` — FastAPI App & Routes

```python
# main.py
from fastapi import FastAPI, Depends, HTTPException, Path
from sqlalchemy.orm import Session

import models, schemas, crud
from database import SessionLocal, engine

# Create tables on startup
models.Base.metadata.create_all(bind=engine)

app = FastAPI()
```

🧠 Teaching Notes:

- `Depends(get_db)` injects a DB session.
- Routes call `crud` for DB operations.
- HTTP verbs match their actions (POST, GET, PUT, DELETE).