FastAPI\_Postgres

Table of Content

[1 REPO 2](#_Toc152352420)

[2 Environment: 2](#_Toc152352421)

[3 Create app 2](#_Toc152352422)

[4 Start the app 8](#_Toc152352423)

|  |
| --- |
| REPO <https://github.com/ecrvmal/Trading_app>  web:  <https://www.educative.io/answers/how-to-use-postgresql-database-in-fastapi> |
| Environment: pipenv install fastapi fastapi-sqlalchemy pydantic alembic psycopg2 uvicorn python-dotenv |
| Create appCreate proj/models.py from sqlalchemy import Column, DateTime, ForeignKey, Integer, String, Float from sqlalchemy.ext.declarative import declarative\_base from sqlalchemy.orm import relationship from sqlalchemy.sql import func  Base = declarative\_base()  class Book(Base):  \_\_tablename\_\_ = 'book'  id = Column(Integer, primary\_key=True, index=True)  title = Column(String)  rating = Column(Float)  time\_created = Column(DateTime(timezone=True), server\_default=func.now())  time\_updated = Column(DateTime(timezone=True), onupdate=func.now())  author\_id = Column(Integer, ForeignKey('author.id'))   author = relationship('Author')   class Author(Base):  \_\_tablename\_\_ = 'author'  id = Column(Integer, primary\_key=True)  name = Column(String)  age = Column(Integer)  time\_created = Column(DateTime(timezone=True), server\_default=func.now())  time\_updated = Column(DateTime(timezone=True), onupdate=func.now()) |
| Edit proj/schema.py # build a schema using pydantic from pydantic import BaseModel   class Book(BaseModel):  title: str  rating: int  author\_id: int   class Config:  orm\_mode = True   class Author(BaseModel):  name: str  age: int   class Config:  orm\_mode = True |
| Edit prj/main.py import uvicorn from fastapi import FastAPI from fastapi\_sqlalchemy import DBSessionMiddleware, db  from schema import Book as SchemaBook from schema import Author as SchemaAuthor  from schema import Book from schema import Author  from models import Book as ModelBook from models import Author as ModelAuthor  import os from dotenv import load\_dotenv  load\_dotenv('.env')  app = FastAPI()  # to avoid csrftokenError app.add\_middleware(DBSessionMiddleware, db\_url=os.environ['DATABASE\_URL'])   @app.get("/") async def root():  return {"message": "hello world"}   @app.post('/book/', response\_model=SchemaBook) async def book(book: SchemaBook):  db\_book = ModelBook(title=book.title, rating=book.rating, author\_id=book.author\_id)  db.session.add(db\_book)  db.session.commit()  return db\_book   @app.get('/book/') async def book():  book = db.session.query(ModelBook).all()  return book   @app.post('/author/', response\_model=SchemaAuthor) async def author(author: SchemaAuthor):  db\_author = ModelAuthor(name=author.name, age=author.age)  db.session.add(db\_author)  db.session.commit()  return db\_author   @app.get('/author/') async def author():  author = db.session.query(ModelAuthor).all()  return author   # To run locally if \_\_name\_\_ == '\_\_main\_\_':  uvicorn.run(app, host='0.0.0.0', port=8000) |
| Edit proj/.env DATABASE\_URI = 'postgresql://postgres:<password>@localhost/<name\_of\_the\_datbase>'  db\_name db\_passw /   prj\_user prj\_user postgres\_fastapi1 |
| Create DB via PGAdmin Databases > create > postgres\_fastapi1 |
| Init alembic Command  (venv) PS D:\GB\pythonProject\FastAPI\_Postgre> alembic init alembic Creating directory 'D:\\GB\\pythonProject\\FastAPI\_Postgre\\alembic' ... done Creating directory 'D:\\GB\\pythonProject\\FastAPI\_Postgre\\alembic\\versions' ... done Generating D:\GB\pythonProject\FastAPI\_Postgre\alembic.ini ... done Generating D:\GB\pythonProject\FastAPI\_Postgre\alembic\env.py ... done Generating D:\GB\pythonProject\FastAPI\_Postgre\alembic\README ... done Generating D:\GB\pythonProject\FastAPI\_Postgre\alembic\script.py.mako ... done Please edit configuration/connection/logging settings in 'D:\\GB\\pythonProject\\FastAPI\_Postgre\\alembic.ini' before proceeding. (venv) PS D:\GB\pythonProject\FastAPI\_Postgre>  Folder alembic created |
| Edit prj/alembic/env.py from logging.config import fileConfig  from sqlalchemy import engine\_from\_config from sqlalchemy import pool  from alembic import context  # ---- user code starts ----- import os, sys from dotenv import load\_dotenv  BASE\_DIR= os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_))) load\_dotenv(os.path.join(BASE\_DIR, '.env')) sys.path.append(BASE\_DIR) # ---- user code ends -----    # this is the Alembic Config object, which provides # access to the values within the .ini file in use. config = context.config  # --- user code starts ----- # Making a connection config.set\_main\_option('sqlalchemy.url', os.environ['DATABASE\_URL']) # ---- user code ends ----   # Interpret the config file for Python logging. # This line sets up loggers basically. if config.config\_file\_name is not None:  fileConfig(config.config\_file\_name)  # add your model's MetaData object here # for 'autogenerate' support import models target\_metadata = models.Base.metadata # target\_metadata = None  ...... |
| Enable migration alembic revision --autogenerate -m "New Migration"  alembic upgrade head  (venv) PS D:\GB\pythonProject\FastAPI\_Postgre> alembic revision --autogenerate -m "New Migration" INFO [alembic.runtime.migration] Context impl PostgresqlImpl. INFO [alembic.runtime.migration] Will assume transactional DDL. INFO [alembic.autogenerate.compare] Detected added table 'author' INFO [alembic.autogenerate.compare] Detected added table 'book' INFO [alembic.autogenerate.compare] Detected added index 'ix\_book\_id' on '['id']' Generating D:\GB\pythonProject\FastAPI\_Postgre\alembic\versions\a6994256e4a2\_new\_migration.py ... done (venv) PS D:\GB\pythonProject\FastAPI\_Postgre>  (venv) PS D:\GB\pythonProject\FastAPI\_Postgre> alembic upgrade head INFO [alembic.runtime.migration] Context impl PostgresqlImpl. INFO [alembic.runtime.migration] Will assume transactional DDL. (venv) PS D:\GB\pythonProject\FastAPI\_Postgre> |
| Start the app uvicorn main:app --reload  Then go  <http://127.0.0.1:8000/>    , and to use the app go  <http://127.0.0.1:8000/docs> |