Unit selection

ساختار پروژه در پایتون به این صورت هست که یک دایرکتوری به نام sql\_app میسازیم

و فایل هایی به نام:  
main.py

Crud.py

Database.py

Schemas.py

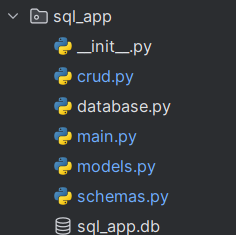
Models.py

و در نهایت برای اینکه پایتون بفهمد این یک دایرکتوری است و حاوی پکیج یک فایل به نام

\_\_init\_\_.py

میسازیم

بعد از این اون پوشه تبدیل به یک دایرکتوری میشود و ظاهرش هم تغییر میکند:



درفایل main.py محتوای اصلی قرار میگیره و تمامی فایل ها در مین ایمپورت میشن:

from fastapi import Depends, FastAPI, HTTPException  
from sqlalchemy.orm import Session  
import crud  
import models  
import schemas  
from .database import SessionLocal, engine  
  
models.Base.metadata.create\_all(bind=engine)  
  
app = FastAPI()  
  
  
# Dependency  
def get\_db():  
 db = SessionLocal()  
 try:  
 yield db  
 finally:  
 db.close()  
  
  
@app.post("/CreateCou/", response\_model=schemas.Course)  
def create\_course(course: schemas.Course, db: Session = Depends(get\_db)):  
 db\_course = crud.get\_Course(db=db, course=course)  
 if db\_course:  
 raise HTTPException(status\_code=400, detail="Course already exists")  
 return crud.create\_Course(db=db, course=course)  
  
@app.get("/GetCou/{course\_id}", response\_model=schemas.Course)  
def read\_Course(course\_id: int, db: Session = Depends(get\_db)):  
 db\_course = crud.get\_Course(db, course\_id=course\_id)  
 if db\_course is None:  
 raise HTTPException(status\_code=404, detail="Course not found")  
 return db\_course  
@app.delete("/DeleteCou/{course\_id}")  
def delete\_course(course\_id: int, db: Session = Depends(get\_db)):  
 db\_course = crud.get\_Course(db, course\_id=course\_id)  
 if db\_course is None:  
 raise HTTPException(status\_code=404, detail="Course not found")  
 crud.delete\_Course(db, course\_id=course\_id)  
 return {"message": "Course deleted successfully"}  
  
@app.patch("/UpdateCou/{course\_id}", response\_model=schemas.Course)  
def update\_course(course\_id: int, course: schemas.Course, db: Session = Depends(get\_db)):  
 db\_course = crud.get\_Course(db, course\_id=course\_id)  
 if db\_course is None:  
 raise HTTPException(status\_code=404, detail="Course not found")  
 return crud.patch\_Course(db, course\_id=course\_id, course=course(

برای ران کردن برنامه هم از همین فایل استفاده میشه

```

fastapi dev main.py

```

و یا   
```

Uvicorn main:app –reload

```

Crud.py:

در این فایل اتصالات پایگاه داده رو قرار میدیم و به کمک توابع خواندن و ایجاد و حذف و آپدیت اعمال رو روی داده ها انجام میدیم

def get\_Course(db: Session, cid: int):  
 return db.query(models.Course).filter(models.Course.cid == cid).first()  
  
  
def create\_Course(db: Session, course: schemas.Course):  
 db\_course = models.Course(cid=course.cid,  
 cname=course.cname,  
 department=course.department,  
 credit=course.credit)  
 db.add(db\_course)  
 db.commit()  
 db.refresh(db\_course)  
 return "درس با موفقیت ساخته شد."  
  
def delete\_Course(db: Session):  
 db.query(models.Course).filter(models.Course)  
 db.delete(db.query(models.Course).filter(models.Course))  
 return "با موفقیت حذف شد."  
  
def patch\_Course(db: Session, course: schemas.Course):  
 db\_course = models.Course(cid=course.cid, cname=course.cname, department=course.department, credit=course.credit)  
 db.query(models.Course).filter(models)  
 db.delete(db.query(models.Course).filter(models))  
 db.add(db\_course)  
 db.commit()  
 db.refresh(db\_course)  
 return "درس با موفقیت آپدیت شد."

Database.py:

در این فایل تعیین میکنیم که از چه دیتابیسی استفاده میکنیم

from sqlalchemy import create\_engine  
from sqlalchemy.ext.declarative import declarative\_base  
from sqlalchemy.orm import sessionmaker  
  
# SQLALCHEMY\_DATABASE\_URL = "sqlite:///./sql\_app.db"  
SQLALCHEMY\_DATABASE\_URL = "mysql+pymysql://root:M13821382r@127.0.0.1:3306/select\_unit"  
  
engine = create\_engine(  
 SQLALCHEMY\_DATABASE\_URL, connect\_args={"check\_same\_thread": False}  
)  
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
  
Base = declarative\_base()

Schemas.py:

در این فایل کلاس هارو میسازیم و میتوان صحت سنجی رو هم در همینجا قرار داد و میشود هم یک فایل جدا به نام datavalidation.py ساخت و بعد اینطرف ایمپورت کرد  
اما خب من در همین فایل صحت سنجی هارو هم انجام دادم

from pydantic import BaseModel, validator  
import re  
from fastapi import HTTPException, status  
from .database import SessionLocal  
from ..sql\_app import crud  
  
  
  
db = SessionLocal()  
  
  
  
class Course(BaseModel):  
 cid: str  
 cname: str  
 department: str  
 credit: int #تعداد واحد درس  
  
 @validator("cid")  
 def CI(CID: str):  
  
 len300 = len(CID)  
  
 regexcid = re.compile("^[0-9]\*$")  
 findcid = regexcid.match(CID)  
  
 if len300 != 5:  
 raise HTTPException(detail="شماره درس باید 5 رقم باشد.", status\_code=status.HTTP\_400\_BAD\_REQUEST)  
 if not findcid:  
 raise HTTPException(detail="لطفا برای شماره درس فقط از اعداد استفاده نمایید.",  
 status\_code=status.HTTP\_400\_BAD\_REQUEST)  
 else:  
 return "شماره درس ثبت شد."  
  
 @validator("cname")  
 def CN(CName):  
 CNamee = str(CName)  
 regexN100 = re.compile("^[آ-ی 1-4]\*$")  
  
 findeN100 = regexN100.match(CNamee)  
  
 if len(CNamee) > 20:  
 raise HTTPException(detail="حروف نام درس نباید بیش از 20 کاراکتر باشد.",  
 status\_code=status.HTTP\_400\_BAD\_REQUEST)  
  
 if len(CNamee) < 3:  
 raise HTTPException(detail="نام درس باید حداقل 3 حرف داشته باشد.", status\_code=status.HTTP\_400\_BAD\_REQUEST)  
  
 if not findeN100:  
 raise HTTPException(detail="نام درس وارد شده نامعتبر است، لطفا نام را به حروف فارسی وارد کنید.",  
 status\_code=status.HTTP\_400\_BAD\_REQUEST)  
  
 else:  
 return ("نام درس ثبت شد.")  
  
 @validator("department")  
 def CD(CDepartment: str):  
 allDepartment1 = ["فنی و مهندسی", "علوم پایه",  
 "شیمی", "اقتصاد", 'ادبیات', "دامپزشکی"]  
  
 if CDepartment not in allDepartment1:  
 raise HTTPException(detail="دانشکده وارد شده نادرست است", status\_code=status.HTTP\_404\_NOT\_FOUND)  
 else:  
 return "دانشکده وارد شده ثبت شد."  
  
 @validator("credit")  
 def CR(credit):  
 Creditt = str(credit)  
 regexcr = re.compile("^[1-4]$")  
 findcr = regexcr.match(Creditt)  
  
 if not findcr:  
 raise HTTPException(detail="لطفا برای واحد درس فقط از اعداد یک تا چهار استفاده نمایید.",  
 status\_code=status.HTTP\_400\_BAD\_REQUEST)  
 else:  
 return "واحد درس ثبت شد."

Models.py:

در این فایل تمامی جداول و همچنین جداول رابطه ای رو مسازیم:

from sqlalchemy import Column, ForeignKey, Integer, String, Table  
from sqlalchemy.orm import relationship  
from database import Base  
  
student\_course = Table(  
 'students\_courses',  
 Base.metadata,  
 Column('student\_id', String, ForeignKey('students.Sid')),  
 Column('course\_id', String, ForeignKey('courses.cid'))  
)  
  
course\_teacher = Table(  
 'courses\_teachers',  
 Base.metadata,  
 Column('course\_id', String, ForeignKey('courses.cid')),  
 Column('teacher\_id', String, ForeignKey('teachers.Tid'))  
)  
  
student\_teacher = Table(  
 'student\_teacher',  
 Base.metadata,  
 Column('student\_id', Integer, ForeignKey('students.Sid')),  
 Column('teacher\_id', Integer, ForeignKey('teachers.Tid'))  
)  
  
class Course(Base):  
  
 \_\_tablename\_\_ = "courses"  
  
 cid = Column(Integer, primary\_key=True)  
 cname = Column(String)  
 department = Column(String)  
 credit = Column(Integer)  
 students = relationship("Student", secondary=student\_course, back\_populates="courses")  
 teachers = relationship("Teacher", secondary=course\_teacher, back\_populates="courses")