پروژه پایانترم برنامه نویسی پیشرفته

در این پروژه ما به فایلهای database.py,main.py,crud.py,schemas.py,model.py, و فایل\_\_init\_\_.py در مسیر folder<sql\_app نیاز داریم پس آنها را ایجاد میکنیم –init--.py یک فایل خالی است و صرفا به این دلیل به وجود می آید که باعث **پکیج** شدن فایلهای یک دایرکتوری میشود.

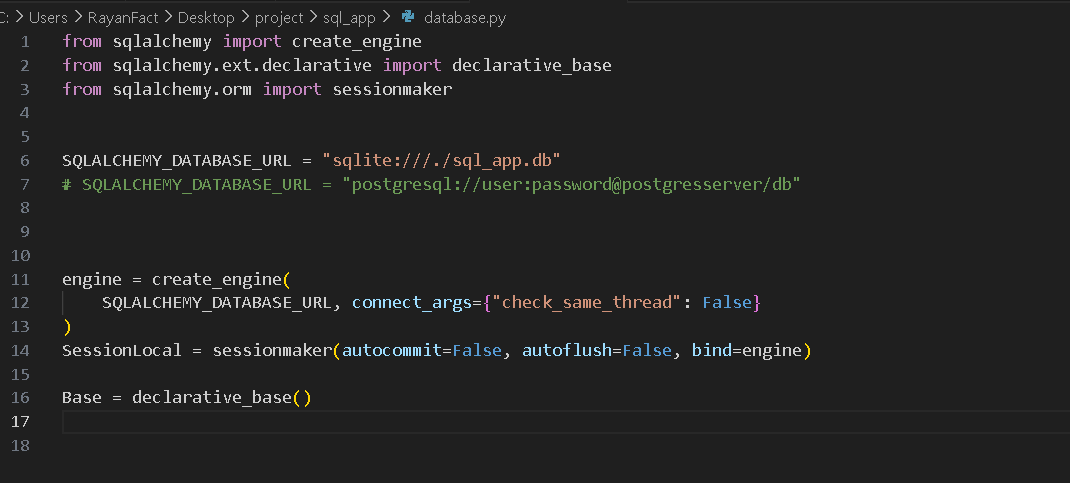
درابتدا ما نیاز داریم به sqlslchemy پس با دستور

**pip install sqlalchemy**

آن را دانلود و نصب میکنیم .

حالا شروع به ساختن پروژه میکنیم.

از فایل database.py شروع میکنیم.دراین فایل ما به دیتابیس متصل میشویم.

در ابتدا موارد لازم را از sqlalchemy وارد میکنیم.

سپس برای sqlalchemy یک url از دیتابیس مورد استفاده (دراینجا sqlite) میسازیم.بدین شکل که با استفاده از sqlite و در همین دایرکتوری فایل sql\_app.db

را میسازیم. و در متغییر SQLALCHEMY\_DATABASE\_URL میریزیم.

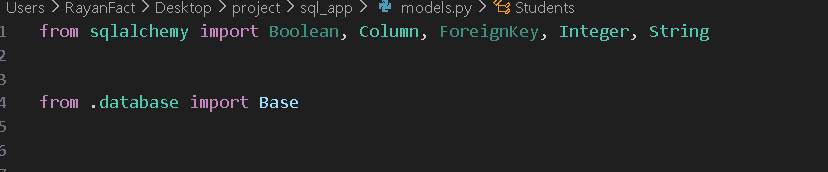
در گام بعد با استفاده از تابع create\_engine یک engine ایجاد میکنیم.engine نحوه اتصال را نشان میدهد. توجه داشته باشید که آپشن connect\_args فقط برای دیتابیس sqlite و به دلیل ضعیف بودن آن در کار با tread ها استفاده میشود.که باعث میشود از tread قبلی استفاده نکند و از tread های جدید استفاده کند.

حالا با متغییر SessionLocal یک اتصال به دیتابیس میسازیم.این کلاس با تابع sessionmaker ساخته میشود.

و درنهایت بااستفاده از declarative\_base() یک Base کلاس میسازیم.(از این کلاس در فایلهای بعدی برای ساختن جداول و مدل های دیتابیس استفاده میکنیم.)

حالا با ساختن فایل models با دیتابیس تعامل میکنیم. در این فایل ستون های هر جدول ساخته میشود.

در ابتدا موارد مورد نیاز برای ساختن جدول را از sqlalchemy ایمپورت میکنیم. و همینطور کلاس Base که در فایل قبل ساختیم را وارد میکنیم تا با ارثبری از ان جداول را بسازیم.



حالا جداول را ایجاد میکنیم. برای هر جدول کلاسی میسازیم که ا زBase ارثبری میکند.

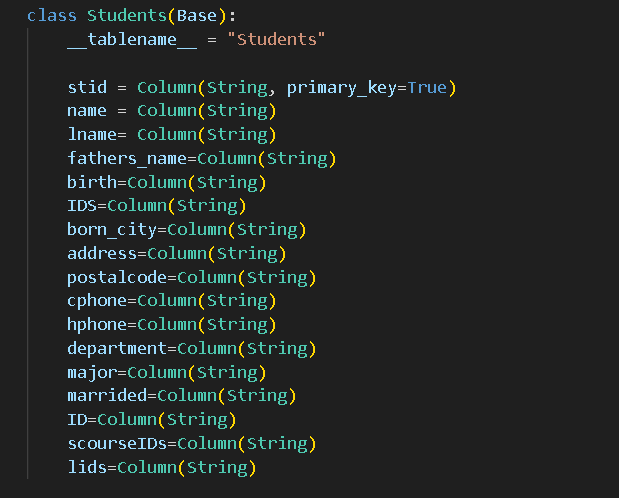
هر جدول باید یک نام داشته باشد بدین شکل که یک مقداررشته ای به عنوان نام جدول به –tablename— میدهیم.

هر فیلد ازکلاس یک ستون جدول است.این موضوع را با Column نشان میدهیم.

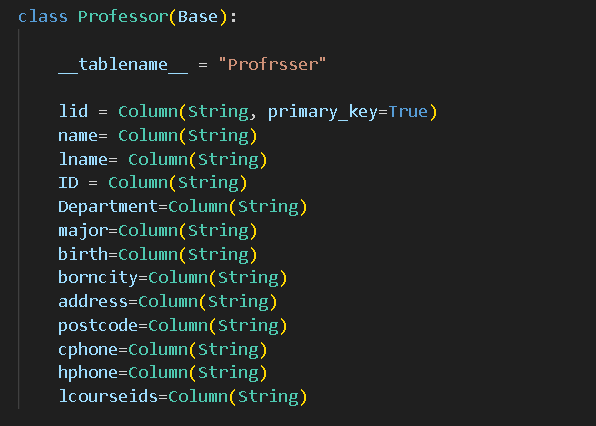
نوع داده هر ستون را باید مشخص کنیم.( String,Integer و...)

برای هر جدول یک کلید اصلیprimery\_key)) وجود دارد. این مقدار هیچوقت تکراری نمیشود و با استفاده از آن میتوانیم اطلاعات و داده های یک کاربر(یا هر چیز دیگر)از جدول کاربرها را به دست آوریم.

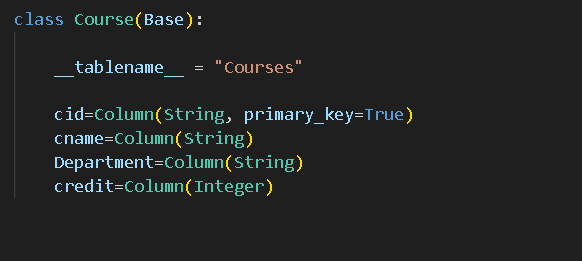
جدول دانشجو:



جدول استاد:



جدول درس:

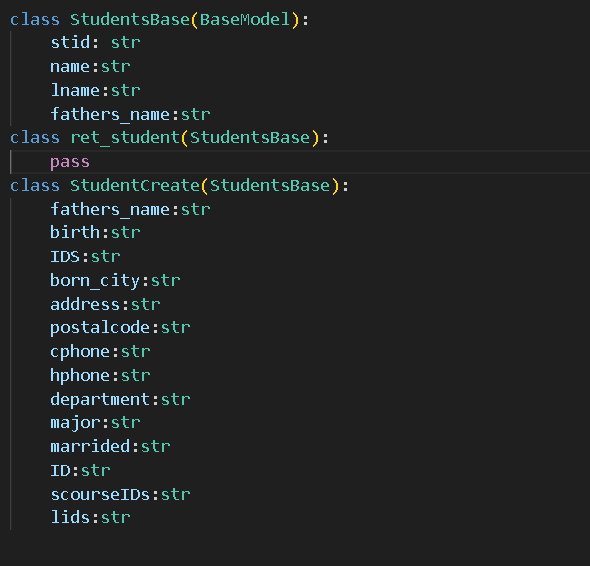


حالا فایل schemas.py پروژه را میسازیم. این فایل pydantic models است که نوع داده ی ورودی هر ستون از جدول را مشخص میکند.

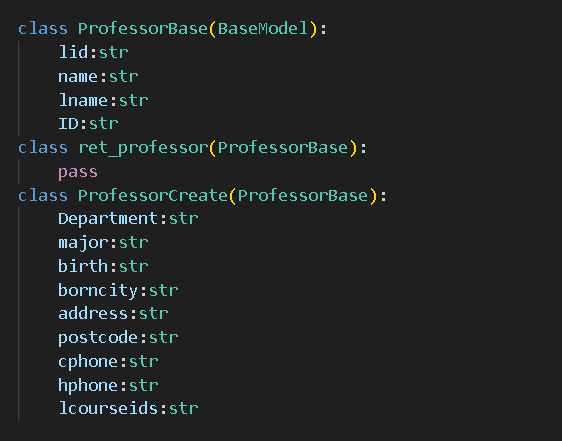
وقتی از سمت کاربر اطلاعات دریافت میشود اعتباری سنجی نوع داده به وسیله ی pydantic انجام میشود.

**برای محدود کردن خروجی هایی که به کاربر نشان داده میشود، مجبوریم کد را در چند کلاس بنویسیم.**

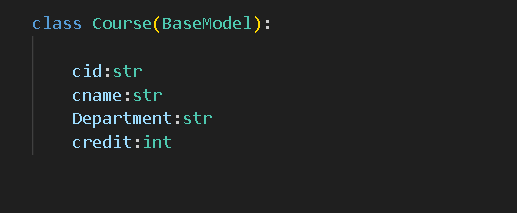
جدول دانشجو:



جدول استاد:

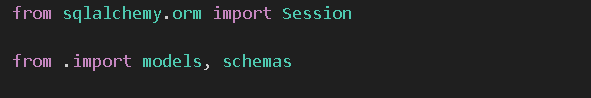


جدول درس:



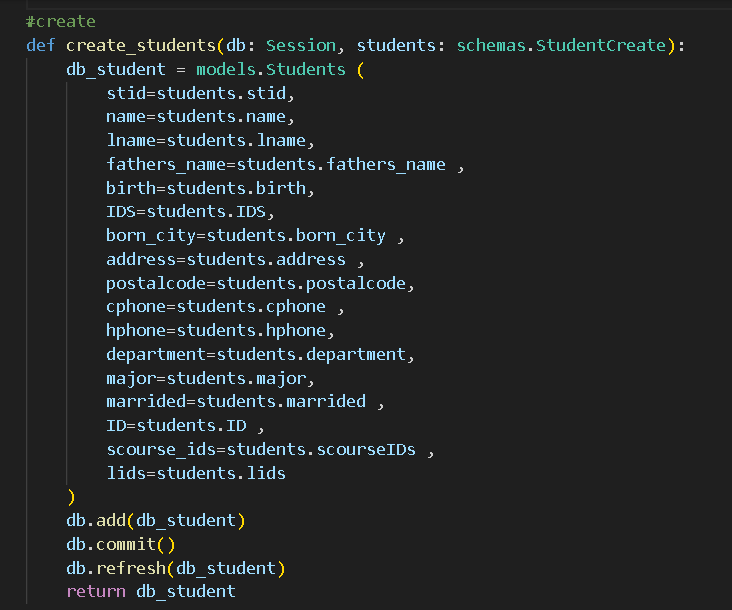
حالا فایل crud پروژه را میسازیم.این فایل شامل عملیات های create,read,update,delete است.

در ابتدا ایمپورت های لازم را انجام میدهیم.



حال فایلها را میسازیم.

با فایل create شروع میکنیم.



Session که در بالا ایمپورت کردیم در اینجا به کار میرود. هر بار که عملیات انجام میشود یک Session جدید استفاده میشود.همانطور که قبلا گفتیم ,Session ها ارتباط با دیتابیس را میسر میکنند. با قرار دادن session در db ما به وسیله ی db با دیتابیس ارتباط میگیریم.

از فایل schemas کلاس مربوط به هر جدول را در یک متغییرمشخص (در اینجا student) میریزیم.

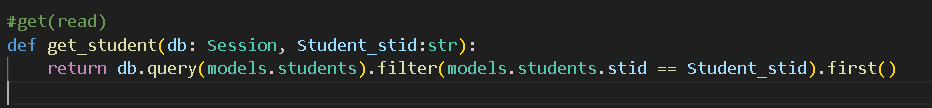
حالا میخواهیم یک دانشجوی جدید ایجاد کنیم این دانشجو از روی مدل های sqlalchemy(فایل models ساخته میشود.) پس فیلد های لازم در آن را پر میکنیم. ايلاعات از سمت کاربر می اید پس باید روی متغییری که برای schemas ساختیم(همان متغییرstudents داخل پرانتز) اطلاعات را دریافت کنیم.

سپس روی دیتابیس اطلاعات را add (اضافه)

Commit (ثبت) و در نهایت رفرش میکنیم.

در آخر نتیجه را با return db\_student میتوانیم ببینیم.

سپس کد get

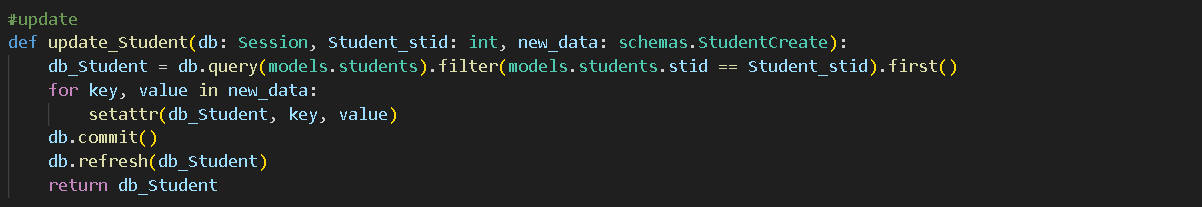


در خط اول session را در db قرار میدهیم تا از طریق آن با دیتابیس ارتباط برقرار کنیم.

سپس یک id از دانشجو میگیریم.

به دیتابیس query میزنیم و از فایل models کلاس students را میگیریم بر اساس stid دانشجو فیلتر میکنیم. و اگر چند دانشجو با این کد وجود داشت اولی را برمیگرداند.

حالا کد آپدیت



در خط اول ابتدا session را در db میریزیم.داده های جدید را با new\_data و بر اساس فایل schemas دریافت میکنیم. یک id از دانشجو دریافت میکنیم.

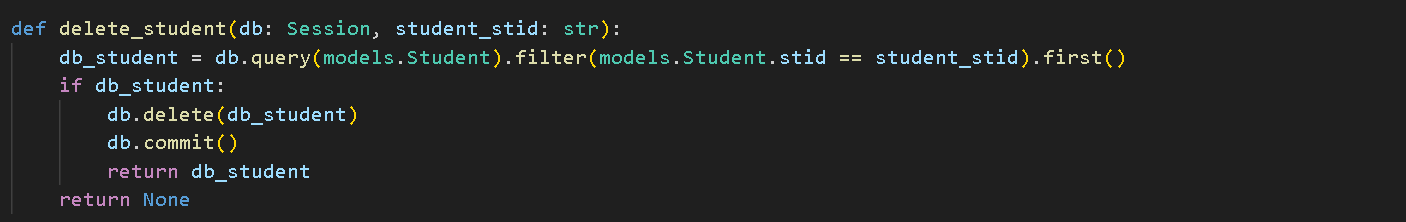
یک متغییر تعریف میکنیم و از models ها stid دانشجو را در آن قرار میدهیم.

برای هر کلید و value در دیتای جدید محتوا را بروزرسانی میکنیم.

اطلاعات را اضافه ، ثبت و رفرش میکنیم.

db\_stduent را ریترن میکنیم.

و در نهایت کد حذف



در خط اول Session را در db میریزیم تا ازطریق آن با دیتا بیس ارتباط برقرار کنیم.

سپس یک متغییر تعریف میکنیم. به دیتابیس درخواست زده و از فایل models ، stid را دریافت میکنیم و در متغییر میگزاریم.

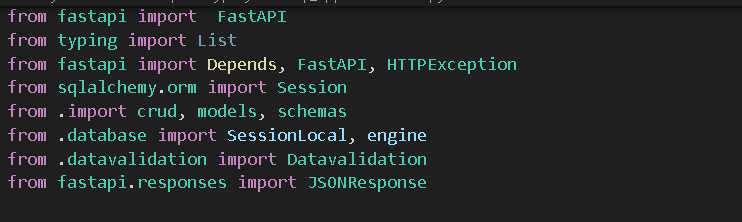
اگر این دانشجو وجود داشت(یعنی del\_student دارای مقدار بود) روی دیتابیس آن را حذف میکنیم .سپس اطلاعات را ثبت میکنیم.

**برای دو جدول دیگر یعنی professor و course هم به همین شکل کد crud را مینویسیم.**

حالا باید فایل main پروژه را بنویسیم.در این فایل از همه ی

بخش هایی که ساختیم،استفاده میکنیم. در این فایل برای هر عملیات یک url نوشته میشود.

فایل main :



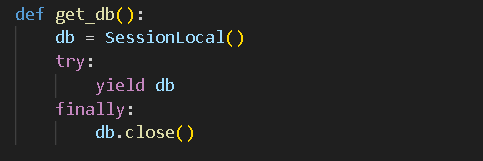
در ابتدا ایمپورت های لازم را انجام میدهیم.



ما میخواهیم جداول را از روی فایلی models بسازیم. پس کلاس Base ،models را استفاده میکنیم. میخواهیم همه ی جداولی که از Base ارثبری کردند را بسازیم.پس باید از metadata.create\_all() استفاده کنیم.

ساخت جداول به bind نیاز دارد.bind باید همان engine باشد.engine همان سرور دیتابیس است که در فایل database ساختیم.

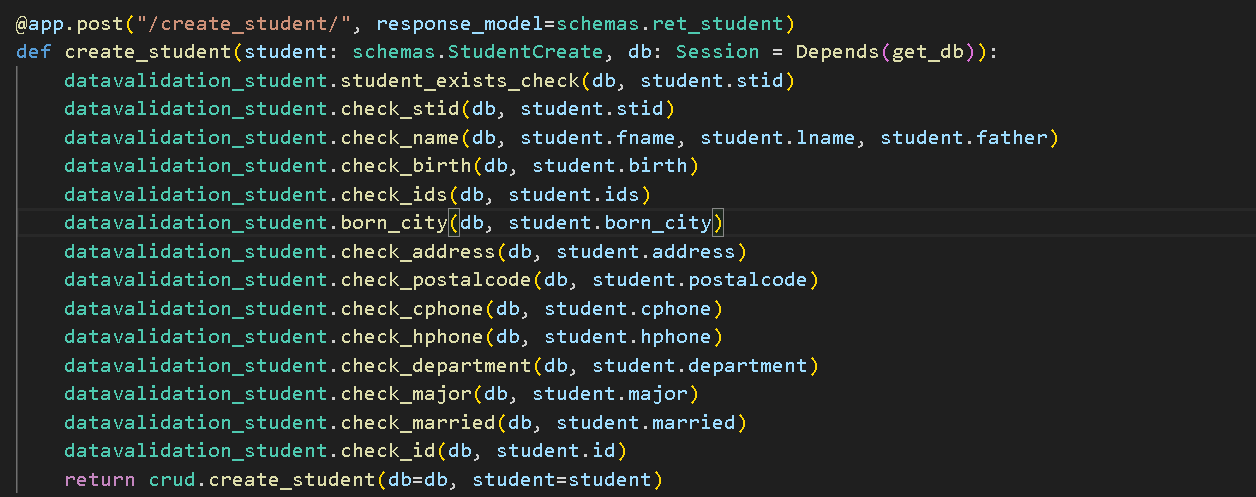
انجام دادن این کار به این معنی است که جداول را در سرور دیتابیس میسازیم.



برای اینکه در request های مختلف session ها باهم تداخل پیدا نکنند،برای هر درخواست یک session متفاوت را بکار میبریم.

برای اینکار باید یک تابع بنویسیم.یک متغییر میسازیم و sessionlocal() که در فایل قبل ساختیم ر در آن قرار میدهیم. تا یک session بگیرد. سپس در بدنه ی تابع db را برگرداند. و پس از استفاده اتصال را ببندد.

حالا برای ایجاد هر کاربر باید اطلاعات آن را بگیریم. در ابتدا url مربوط به گرفتن اطلاعات را در آدرس دلخواه میسازیم. از آنجا که در نمایش اطلاعات به کاربر محدودیت وجود دارد پس از response\_model استفاده میکنیم و اطلاعاتی که باید به کاربر نشان داده شود را تعیین میکنیم.(در اینجا محتوای کلاس ret\_student )

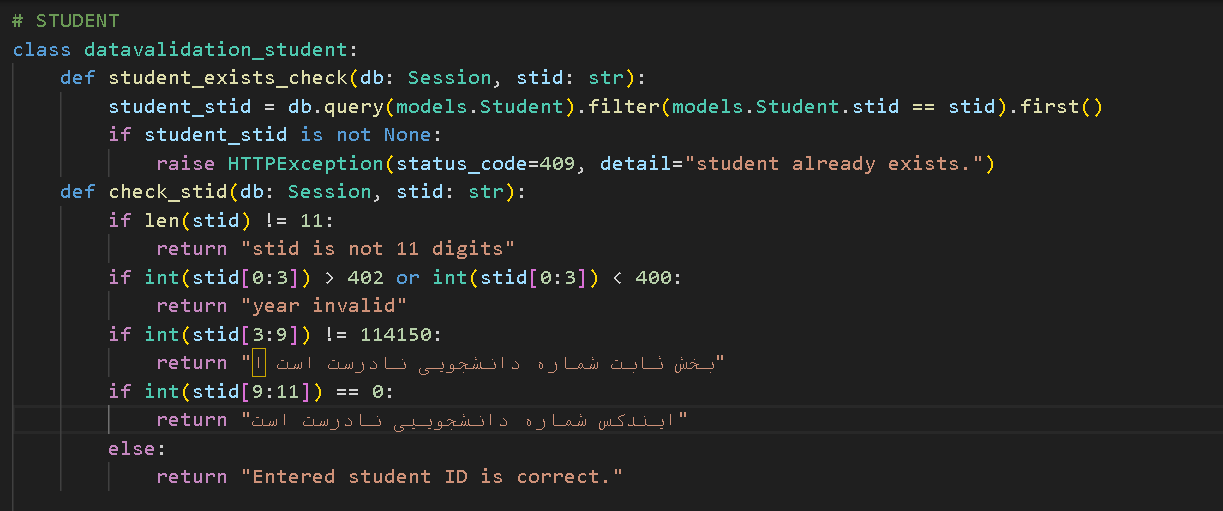


برای ایجاد ،آپدیت یا هر گونه عملیات دیگر باید به دیتابیس query بزنیم. برای این کار روی db که قبلا تعریف کردیم query میزنیم. این db ها را از تابع get\_db که در بالا تعریف کردیم،میگیریم.

داده ها روی schemas گرفته میشوند، پس یک متغییرتعریف میکنیم و کلاس های لازم از schemas در آن قرار میدهیم.

قبل از انجام عملیات باید اعتبار سنجی های لازم را انجام شود.این اعتبارسنجی ها در فایل datavalidation انجام میشود.

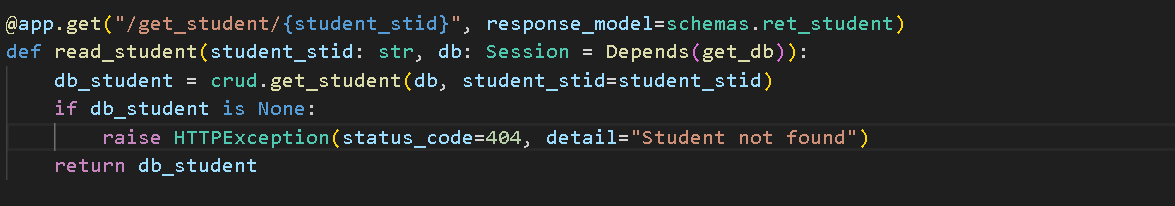
نمونه ای از اعتبارسنجی های فایل datavalidation :



همه ی اعتبارسنجی ها به همین شکل انجام میشود.

در آخرین خط کد create فایل crud را return میکنیم.

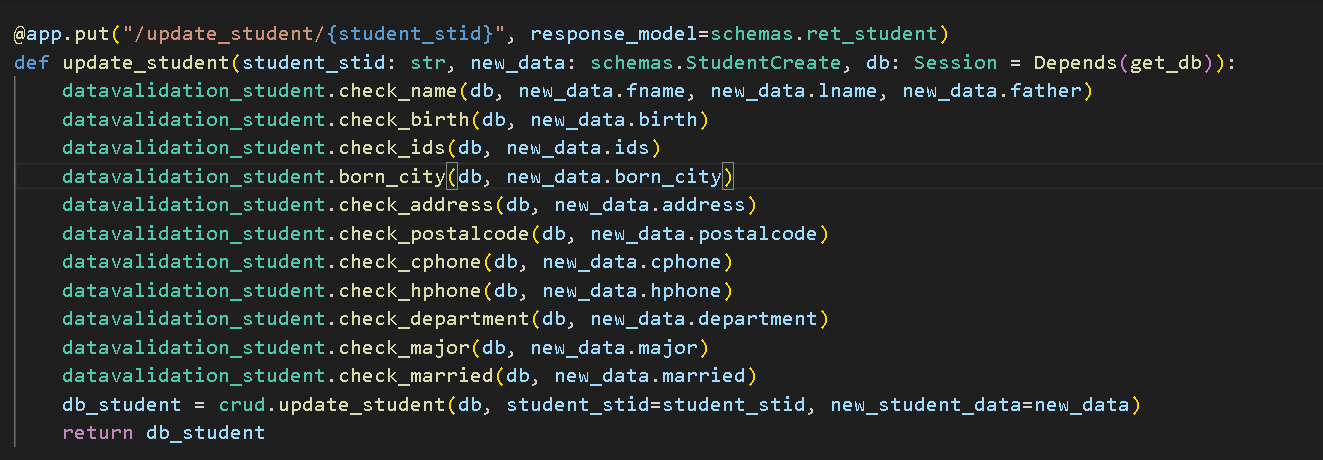
کد read :



Student\_stid را به عنوان query پارامتر میگیریم و اطلاعات و دیتا های مربوط به آن را برمیگردانیم. با response\_model خروجی را به کلاس ret\_student در schemasمحدود میکنیم تا فقط 4فیلد اول را برگرداند.

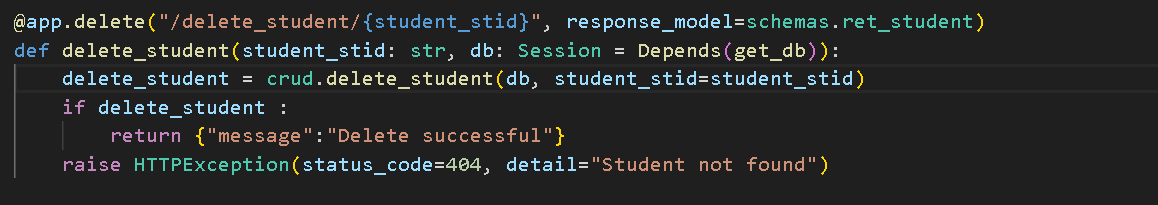
اگر کاربری با این کد وجود نداشت raise میکنیم.

کد آپدیت:



کد آپدیت و create شبیه به هم است.با این تفاوت که schemas.studentcreate را در new\_data میریزیم. دیتا های جدید را دریافت کنیم. اعتبار سنجی های لازم را روی آنها انجام میدهیم. و در نهایت کد آپدیت فایل create را با مقادیر لازم در یک متغییر ریخته آن را ریترن میکنیم.

کد delete

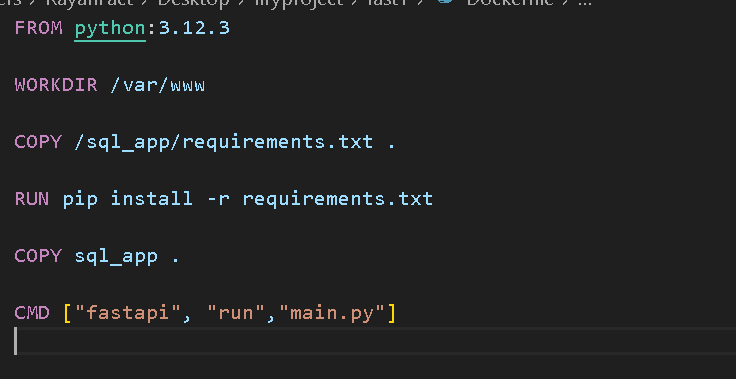


Student\_stid را به عنوان query پارامتر دریافت میکنیم.به مانند کد get برای آن یک db میگیریم و عملیات را با آن انجام میدهیم .یک متغییر تعریف میکنیم و عملیات delete فایل crud را در انجام میدهیم .اگر عملیات انجام شود یک پیغام موفقت آمیز نشان میدهیم

در غیر اینصورت خطای پیدا نشدن کاریر را raise میکنیم.

**این چهار عملیات را برای professor و course هم انجام میدهیم.**

حالا باید پروژه را داکراز کنیم.



در خط اول میگوییم که از پایتون 3:12:3 استفاده میکنیم.

سپس یک مسیر تعیین میکنیم.

محتوای فایل requirments.txt را از فولدر sql\_app در همین دایرکتوری کپی میکنیم.

فایل requirments.txt فایلی است حاوی پکیج ها و نیازمندی های لازم که با زدن کامند

pipreqs . در مسیر فولدر sql\_app ، ساخته شده و در همین فولدر ذخیره میشود.

RuN pip install -r requirements.txt

را ران میکنیم.

سپس فولدر sql\_app را کپی میکنیم.

و در نهایت با cmd اجرا میکنیم. "run" به این دلیل گذاشته میشود که بتوانیم از بیرون به برنامه دسترسی داشته باشیم.

**حالا با دستورات زیر کانتینر را میسازیم.**

در مسیر PARENT فولدر sql\_app کامند های زیر را وارد میکنیم:



این دستور ایمیج را میسازد (برای ساختن آن از فایل داکر دایرکتوری فعلی استفاده میکند با **.** بهآن میفهمانیم که از داکرفایل همین دایرکتوری استفاده کند. ) نام کانتینر---> fastapi

حالا با استفاده از ایمیج کانتینر را ران میکنیم.



این دستور کانتینر را روی پورت 8000 اجرا میکند.

پروژه داکرایز شد.

**حالا باید پروژه را روی حساب گیت هاب خود قرار دهیم. تا در دسترس عموم قرار گیرد**.

برای این منظور یک ریپازیتوری در گیت هاب خود میسازیم.

آدرس آن را کپی میکنیم .و با این دستور در مسیر parent فولدر sql\_app :

git clone ripasitoryaddres

با نام ریپازیتوری ساخته شده یک فولدرساخته میشود .فولدر sql\_app و داکر فایل را در آن قرار میدهیم.سپس دستور های زیر را به ترتیب در مسیر دایرکتوری ساخته شده وارد میکنیم:

git add .

git commit -m "message"

git push

کامند اول اضافه ، کامند دوم ثبت میکنند و کامند سوم آن را به ریپازیتوری وارد میکند.