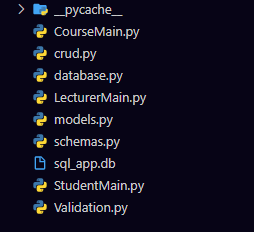
ساختار پروژه ی پایانی من به این صورت انجام شده :



فایل های CourseMain ، LecturerMain ،StudentMain برای ساختن API های سه بخش درس ، استاد و دانشجو می باشد.

فایل database برای ایجاد دیتابیس ما و استفاده از ORM (SQLAlchemy) ساخته می شود.

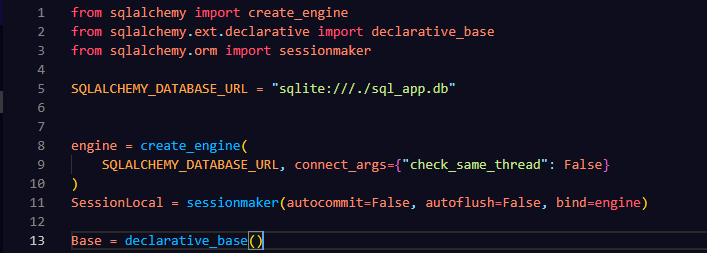
فایل Crud برای انجام عملیات های Create ,Read, Update,Delete پروژه ما می باشد.

فایل models برای ساختن مدل های SQLAlchemy از کلاس Base ما می باشد.

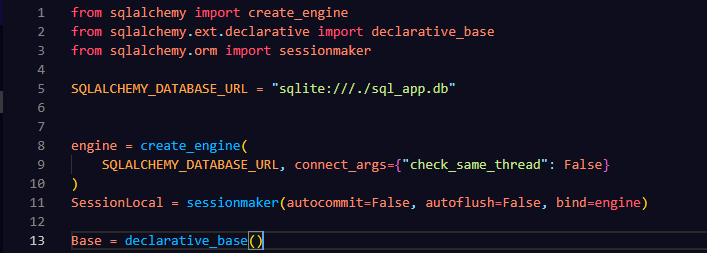
فایل Schemas به این دلیل ایجاد می شود که با استفاده از Pydantic شکل ورودی داده های ما را کنترل می کند.

فایل Validation هم برای صحت سنجی اطلاعاتی که کاربر وارد می کند ساخته شده است.

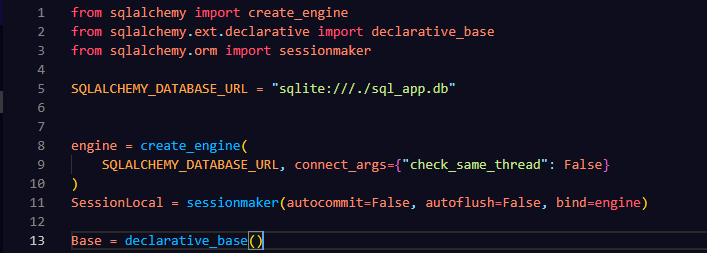
DataBase :



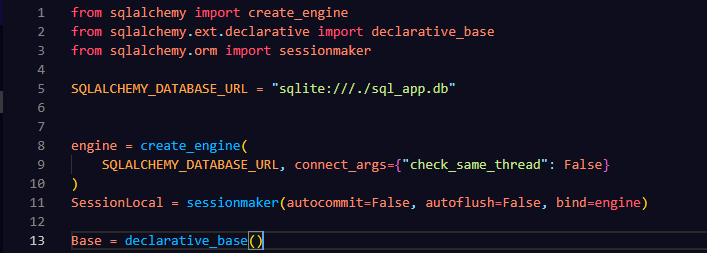
ابتدا فایل های مربوط به SQLAlchemy یا همان ORM را Import می کنیم.



آدرس مربوط به URL دیتابیس خود را ایجاد می کنیم( با توجه به نوع دیتابیس این آدرس می تواند متفاوت باشد)

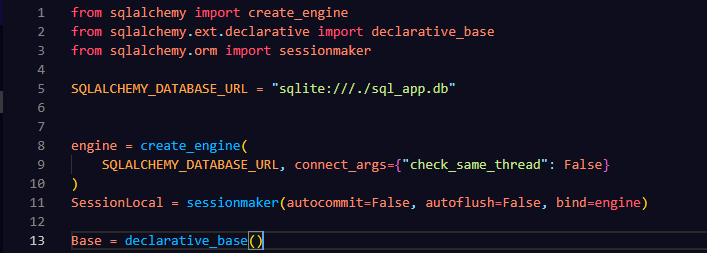


Engine هم جزء منطق دیتابیس ما می باشد و وظیفه ساختن دیتابیس ما با آدرس URL مربوطه می باشد.

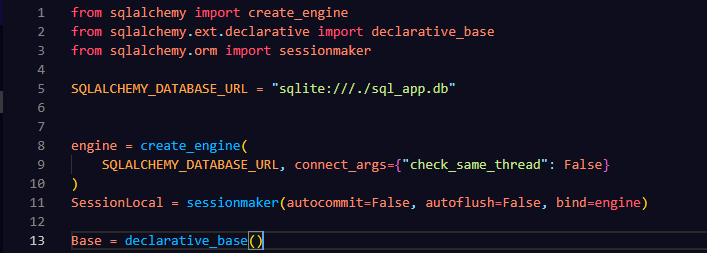


وظیفه Connect\_args این است که به طور پیش فرض SQLite تنها به یک موضوع اجازه ارتباط می دهد، اما در FASTAPI توابع معمولی می توانند با یک درخواست بیشتر از یک موضوع را پوشش دهند و به همین دلیل باید به SQLITE بفهمانیم که این محدودیت را اعمال نکند.

SeccionLocal برای ساخت Seccion یا همان ارتباط ما با دیتابیس می باشد.

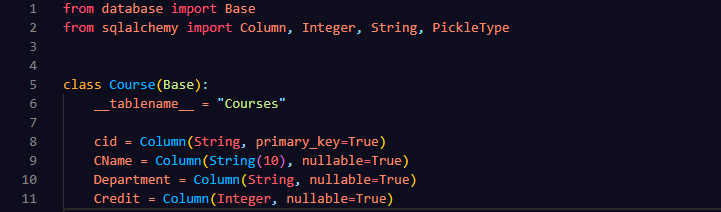


Base هم برای ساختن مدل های دیتابیس ما استفاده می شود.

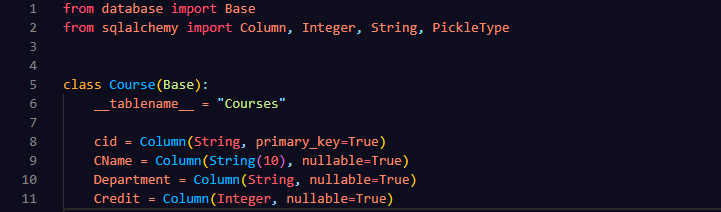


Models :

در این فایل هم مدل های SQLAlchemy از روی کلاس Base ما ساخته می شود.

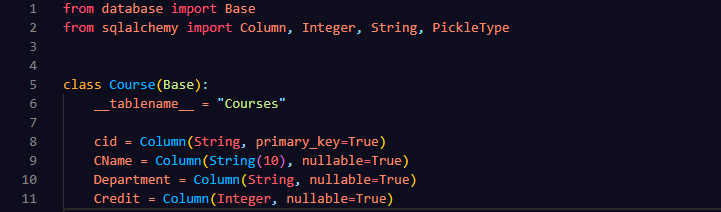


ابتدا از فایل دیتابیس Base را Import می کنیم ، سپس از SQLAlchemy نوع تایپ داده هایی که می خواهیم استفاده کنیم را Import می کنیم.

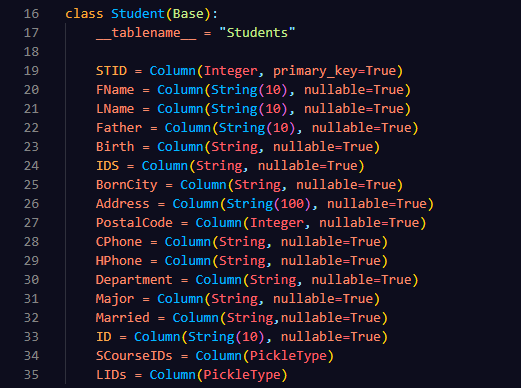


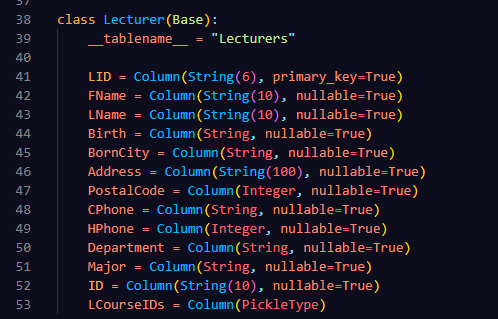
یک کلاس ایجاد می کنیم که از Base ارث بری می کند که در واقع این کلاس همان جدول ما در دیتابیس ما می باشد.

برای این جدول باید یک نام استفاده کنیم و همچنیم هرکدام از ویژگی های درس ، دانشجو و استاد یک ستون در جدول ما می باشد.برای هرکدام از این ستون ها باید نوع تایپ داده را مشخص کنیم و همچنین می توانیم از اتریبیوت های مختلف هم استفاده کنیم:



به طریق مشابه برای استاد و دانشجو هم جدول مربوطه را ایجاد میکنیم:





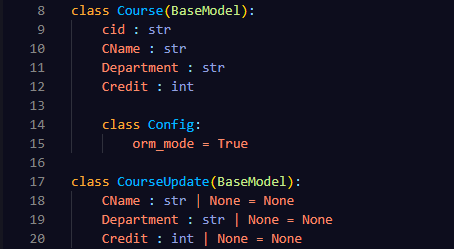
Schemas :

فایل Schemas هم برای ساختن مدل پایدانتیک ما ساخته شده است. در واقع فایل Schemas ورودی های برنامه ما را کنترل می کند. مثلا اگر نوع ورودی یک داده String باشد ولی کاربر غیر از این را وارد کند به کاربر نشان دهد که حتما این ورودی باید String باشد.

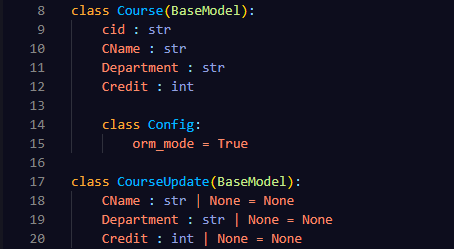
ابتد از پایدانتیک BaseModel را برای ساخت کلاس پایتونی Import می کنیم، هچنین از typing آپشنال و Any را Import می کنیم.Optional وارد کردن یک داده را اختیاری می کند و Any یک نوع ویژگی است که به داده اجازه می دهد هر نوع ورودی را دریافت کند.



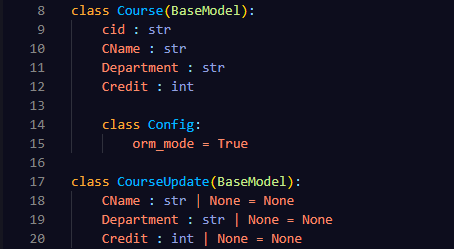
کلاس خود را ایجاد می کنیم و ورودی ها و نوع آنهارا مشخص می کنیم:



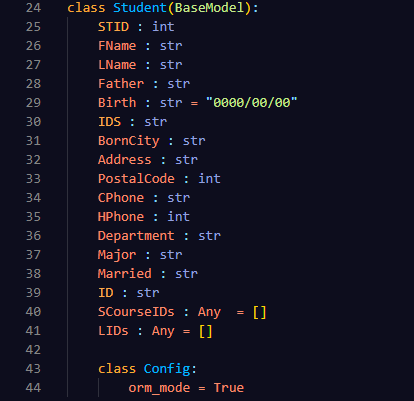
Orm\_mode = True هم برای این در زیر کلاس Config نوشته شده است که چون pydantic به صورت پیش فرض یک دیکشنری به عنوان ورودی میگیرد ولی با ایجاد این آپشن اجازه قبول کردن اینستنس از آن مدل هم میدهد.

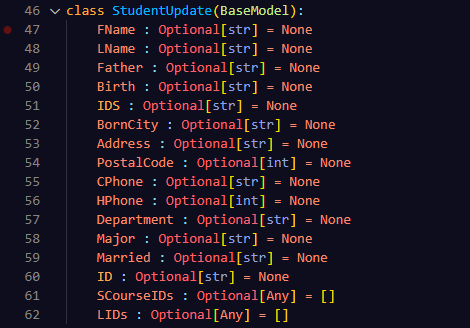


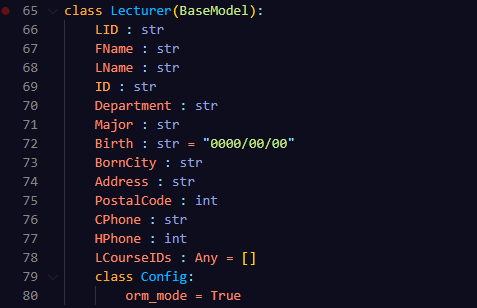
کلاس CourseUpdate هم برای گرفتن ورودی برای آپدیت کردن داده ها استفاده می شود و تفاوت زیادی با کلاس قبلی ندارد.تنها تفاوت آن این است که گرفتن داده هارا اختیاری می کند.برای مثال اگر کاربر بخواهد فقط ویژگی یک ستون از فیلد خود را تغییر بدهد مشکلی نداشته باشد.

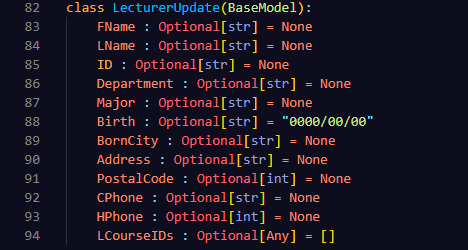


برای دانشجو و استاد هم مشابه Course کلاس هارا ایجاد می کنیم:



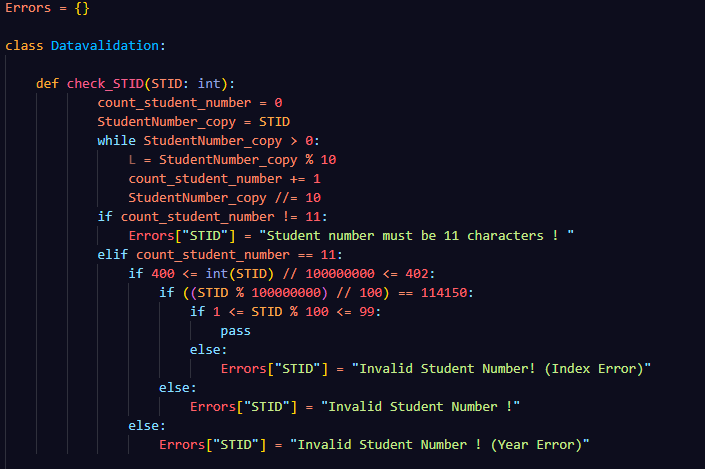




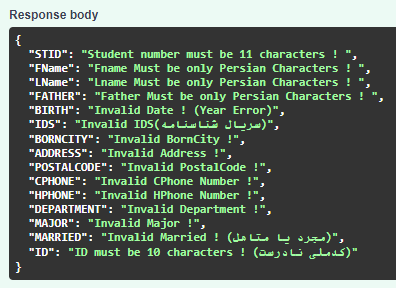


Validation :

فایل ولیدیشن برای صحت سنجی اطلاعاتی است که کاربر وارد می کند.برای مثال ساختار کد ملی باید مطابق استاندارد کشور ما باشد.ساختار کلی قسمت ولیدیشن من به این صورت است که یک دیکشنری با نام Error ایجاد کرده ام و اگر خطا یه اشتباهی در داده ورودی وجود داشته باشد ارور مربوط به آن داده را به دیکشنری اضافه می کند.

برای مثال صحت سنجی قسمت شماره دانشجویی به این صورت است:

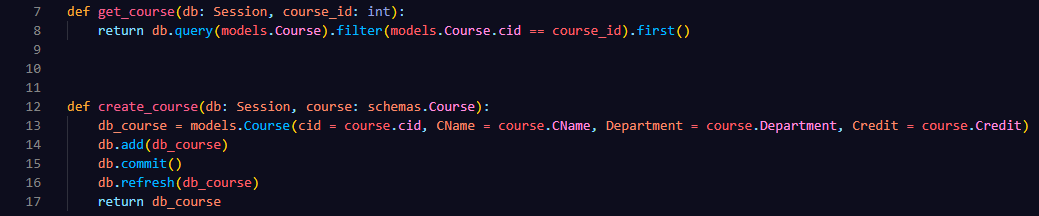
و ارور های برنامه را به این صورت بر می گرداند:



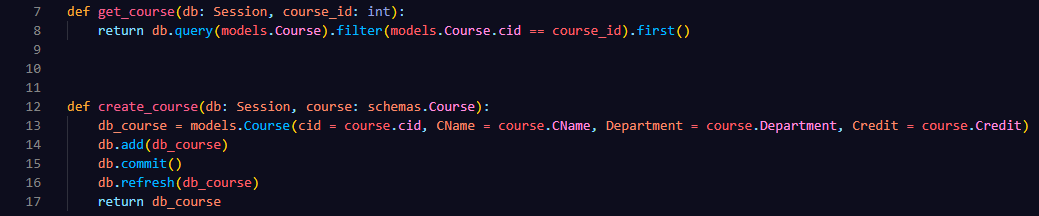
CRUD:

در این فایل فعالیت های ساخت ، خواندن ، حذف و آپدیت کردن دیتا انجام می شود.

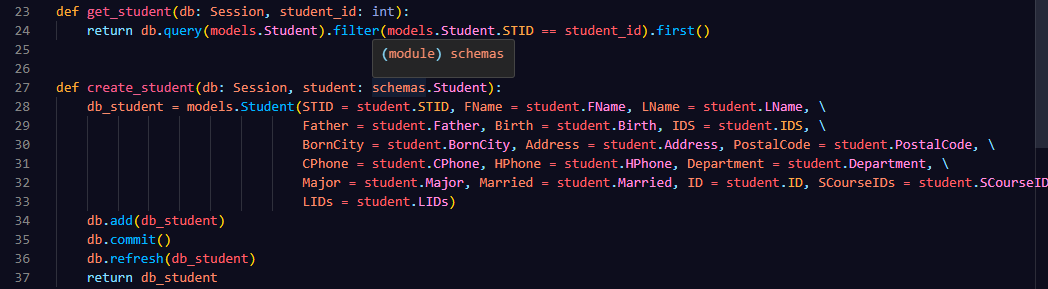
تابع get برای گرفتن دیتا از جدول های برنامه استفاده می شود.و ساختار آن به این صورت است که شماره دانشجویی یا کد درس یا کد استاد را که پرایمری کی جدول هستند را میگیرد و به جدول مربوطه کویری می زند و آن بخش از داده ها ر ا فیلتر می کند که اگر پرایمری کی وارد شده با پرایمری کی موجود در جدول برابر باشد به ما آن ردیف از جدول را بر می گرداند.

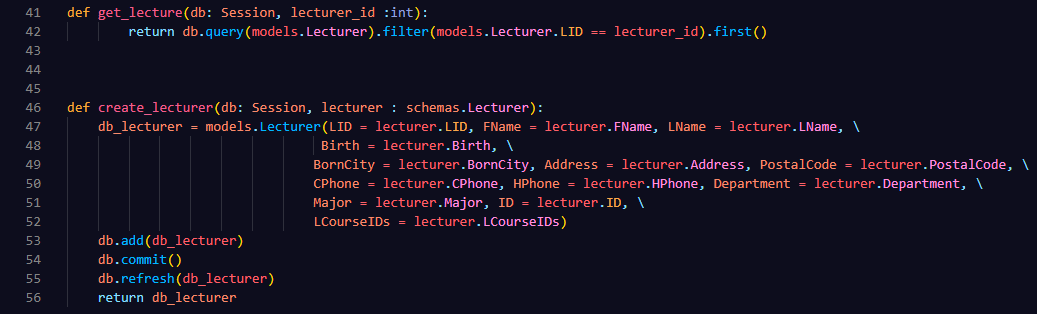


تابع create برای ردیف یا داده ها در جدول نوشته شده است. و به این صورت کار می کند که ورودی های کاربر را به صورت json دریافت می کند و به جدول مربوط به آن بخش کویری می زند و یکی یکی هر کدام از ستون های جدول را مقدار دهی می کند تا ردیف مربوط به آن پرایمری کی ساخته شود. و در سه خط بعدی دیتای وارد شده را اضافه و ثبت می کند و همچنین یک رفرش انجام می دهد تا دیتا ثب شود و در آخر هم آن ردیف را به ما بر می گرداند.



برای دیگر بخش ها (استاد و دانشجو) هم مشابه همین انجام می شود :

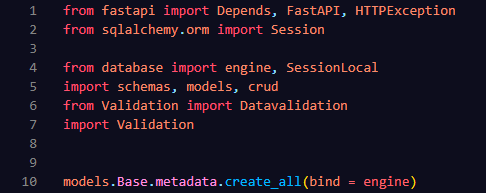




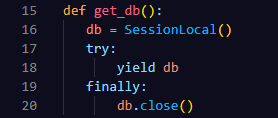
Main:

بخش main همان بخش API است.در این فایل با تمام فایل های پروژه تعامل داریم.

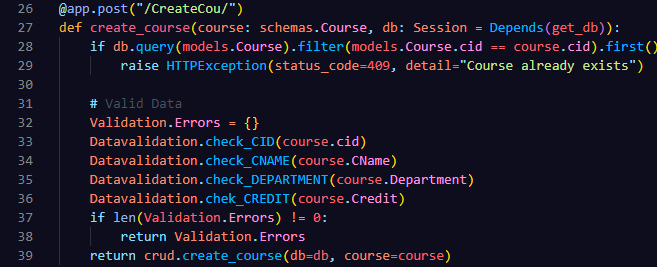
ابتدا فایل های مدنظر را import می کنیم:



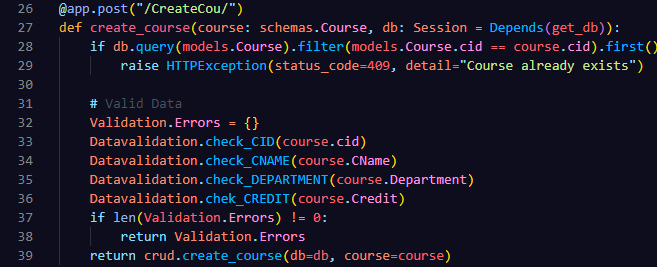
این بخش برای ارتباط با دیتابیس نوشته می شود.به این صورت است که هربار که برنامه با دیتابیس ارتباط برقرار می کند بعد از اتمام کار ارتباط همچنان باقی می ماند و قطع نمی شود ولی با وجود این بخش هربار که برنامه با دیتابیس تعامل پیدا می کند بعد از اتمام درخواست ارتباط را قطع می کند تا در صورت وجود درخواست های دیگر برنامه مشکلی نداشته باشد.



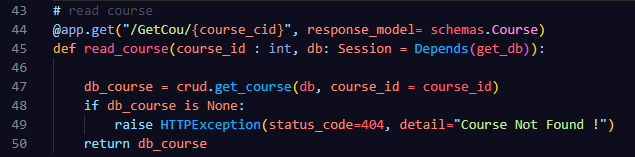
برای ساختن دیتا در جدول از متد Post استفاده می شود. و ساختار به این صورت است که داده های ورودی کاربر را به صورت json دریافت می کند و ابتدا چک می کند که پرایمری کی واردشده(شماره دانشجویی یا کد درس یا کد استاد) در جدول مربوط وجود دارد یا خیر و اگر وجود داشت ارور primary key already exists را بر می گرداند.



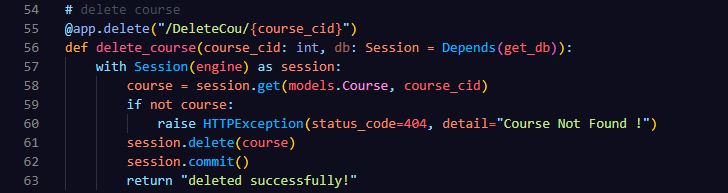
بعد از این بخش صحت سنجی دیتا باید انجام شود. به این صورت است که تابع صحت سنجی مربوط به هر داده از فایل ولیدیشن فراخوانی می شود و داده مربوطه به آن تابع از کلاس ولیدیشن ارسال می شود.و چون ارور ها به یک دیکشنری اضافه می شوند قبل از اینکه داده هارا برگرداند یک شرط ایجاد می کنیم که اگر دیکشنری Error خالی نبود دیکشنری Error ( که ارور های مربوطه در آن هستند) را برگرداند. اگر Error خالی بود به این معنی است که تمام اطلاعات به درستی وارد شده اند.



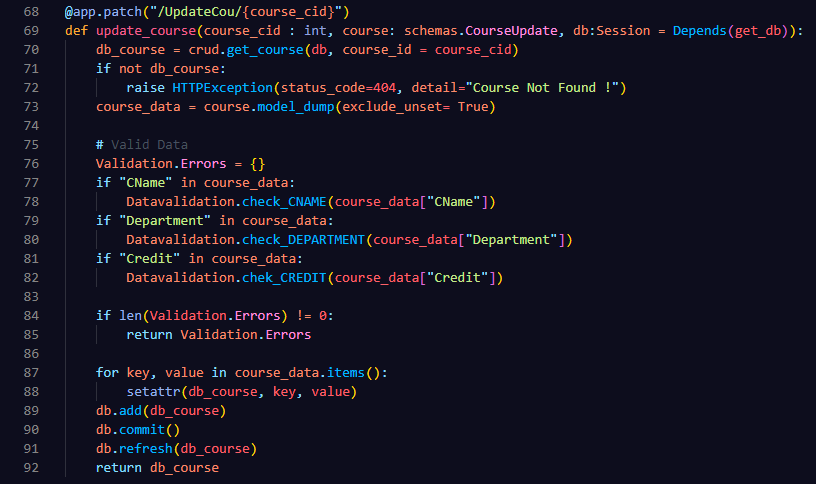
متد Get برای نمایش دیتا ایجاد شده است.ساختار آن به این صورت ایت که پرایمری کی را دریافت می کند و اطلاعات مربوط به آن ردیف را برمیگرداند.ساختار آن به این صورت است که پرایمری کی که کاربر وارد کرده است را به تابع get در فایل کراد ارسال می کند و اگر چیزی برگردانده نشود به این معنی است که دیتایی با پرایمری کی وارد شده وجود ندارد و ارور Primary Key Not Found ! را به کاربر نمایش می دهد. ولی گر دیتایی با پرایمری کی وارد شده وجود داشت اطاعات مربوطه را بر می گرداند.



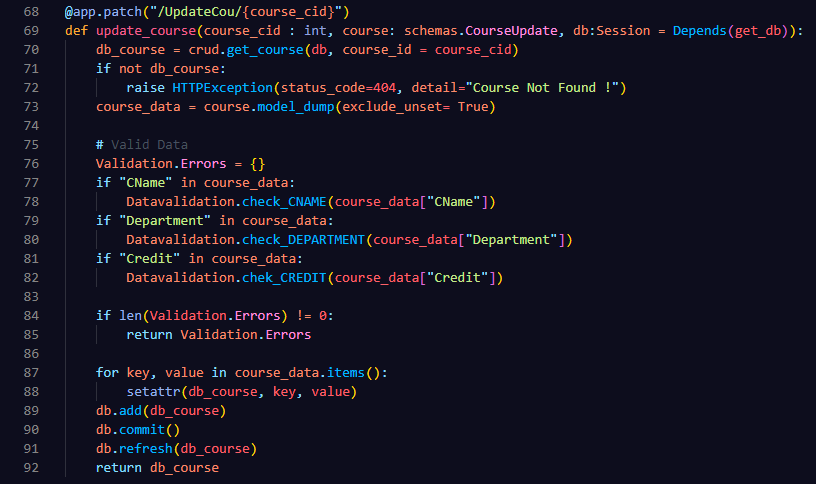
متد Delete برای حذف دیتا استفاده می شود.ساختار آن به این صورت است که پرایمری کی را به عنوان ورودی دریافت می کند و یک سشن با جدول مربوطه ایجاد می کند.از طریق پرایمری کی چک می کند که آیا اطلاعاتی با پرایمری کی وارد شده وجود دارد یا خیر ، اگر وجود نداشت ارور Not Found ! را به کاربر نمایش می دهد. در غیر اینصورت تمام داده های مربوط به آن ردیف را حذف می کند و تغییرات ایجاد شده را کامیت می کند و پیغام delete seccussfuly را نمایش می دهد.



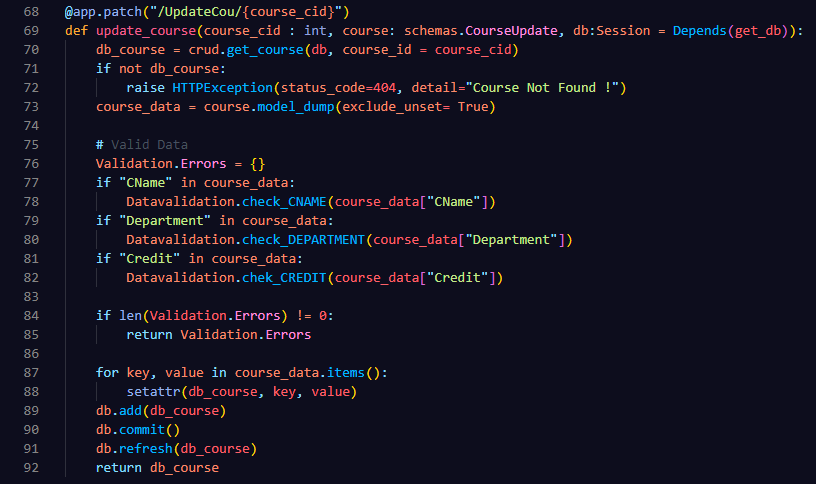
متد patch برای آپدیت داده ها استفاده می شود و تفاوت آن با متد put این است که در متد put اجبار بر این است که تمام داد های آن جدول باید تغییر داده شوند ولی در متد patch این گونه نیست.ساختار آن به این صورت است که پرایمری کی و ورودی جیسون که از طریق فایل اسکیماس کنترل شده است (چون بخش آپدیت وارد کردن تمام داده ها اختیاری است( را دریافت می کند.سپس از طریق get در فایل کراد وجود داشتن پرایمری کی را چک می کند که اگر وجود نداشت ارور not found را بر می گرداند. سپس دیتاهای وارد شده از طرف کاربر را با استفاده از model\_dump( تبدیل به فایل جیسون) دریافت می کند.



بعد از این عملیات نوبت به صحت سنجی دیتا وارد شده از طرف کاربر می باشد.روش صحت سنجی در این بخش این گونه است که چون داده های وارد شده به صورت یک دیکشنری(json) هستند برای هر داده یک شرط ایجاد می کنیم که اگر این داده در دیکشنری که از طرف کاربر وارد شده است وجود داشت صحت سنجی آن را انجام دهد(چون هدف ما آپدیت اطلاعات است و ممکن است کاربر فقط یک مورد از داده ها را تغییر دهد).در آخر اگر خطایی وجود داشت پیغام مربوط با آن را نمایش می دهد در غیر این صورت به خط های بعد می رود.



سپس از طریق یک حلقه داده های وارد شده را یکی یکی جایگزین داده های قبلی در جدول می شود.سپس داد های وارد شده اضافه می شوند در ادامه Commit و رفرش انجام می شود.



دیگر API ها هم مانند همین نوشته شده اند با این تفاوت که آن ها(دانشجو و استاد) یک بخش اضافه به نام کد دروس و کد اساتید دارند.

کد دروس(SCourseIDs) به این صورت است که باید چک کند کد درسی که کاربر وارد کرده است وجود دارد یا خیر.ساختار کد من به این صورت است که کد دروس و کد اساتید را به صورت یک لیست دریافت می کند و با یک حلقه کد های داخل لیست را چک میکند.برای مثال برای ایندکس شماره یک ، یک کویری به جدول مربوطه می زند و چک می کند که آیا کدی در جدول وجود دارد که با ایندکس شماره یک برابر باشد یا خیر.اگر وجود نداشت ارور مربوطه نمایش می دهد. برای کد اساتید هم مشابه همین انجام می شود.