به نام خدا

نام : مهسا قربانی راد

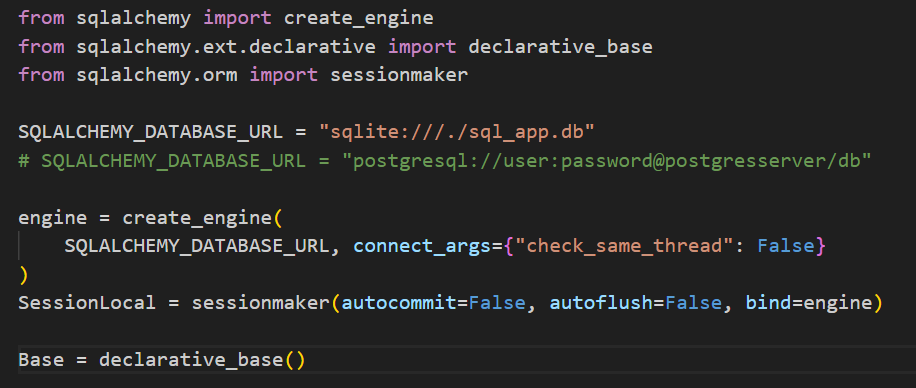
شماره دانشجویی : 40211415053

نام درس : برنامه نویسی پیشرفته

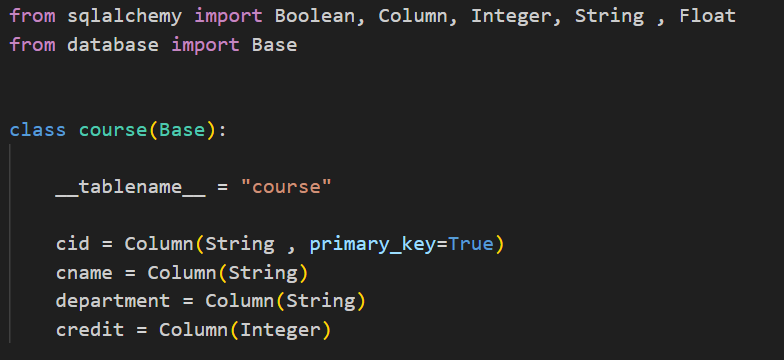
نام استاد : آرمین رشنو

تاریخ تحویل : 17/4/1403

ابتدا در فایل database تنظیمات مربوط به برقراری ارتباط با پایگاه داده را پیاده می کنیم.

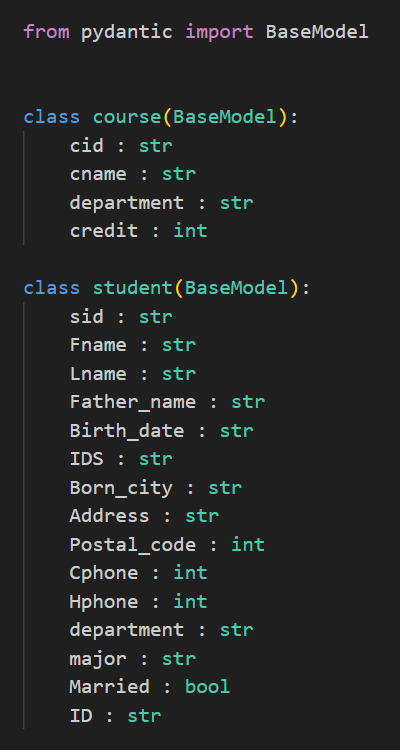


سپس جداول مورد نیاز را در فایل modelsایجاد می کنیم. ما در اینجا به سه جدول دانشجو (student) و استاد (lecturer) و درس (course) نیازمند هستیم. که من در اینجا جدول درس را برای نمونه آورده ام.

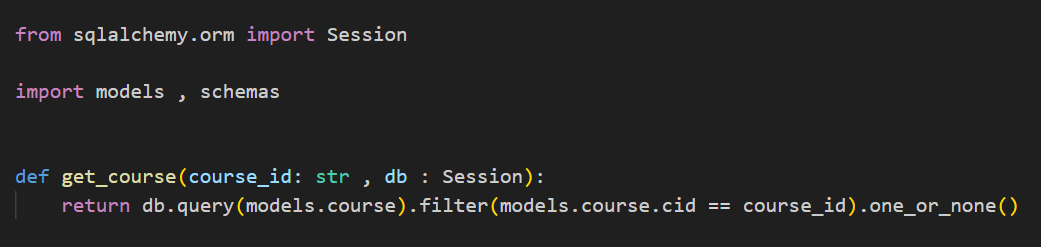


ستونی که با عبارت primary key مشخص شده است به این معنی است که این فیلد هنگام وارد کردن اطلاعات باید یکتا یا منحصر به فرد باشد. برای مثال در جدول درس فیلد cid یا همان course id که به معنای کد درس است primary key است. یعنی هیچ دو سطری با کد درس یکسان وجود ندارد.

در مرحله ی بعد فایل schemas را ایجاد می کنیم که در آن شِما یا همان قالبی که برای وارد کردن مقدار فیلد ها باید رعایت شود را مشخص می کنیم. این قالب را برای همه ی جداول پیاده می کنیم. من در اینجا جدول دانشجو یا student و درس یا course را برای نمونه آورده ام.



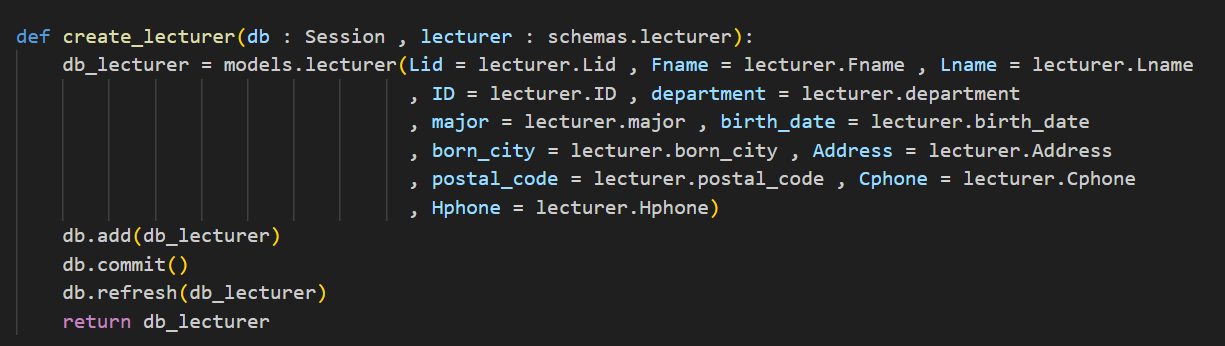
سپس باید فایل crud را ایجاد کنیم. در این فایل چهار عملیات اصلی که روی سطر های جداول انجام می شوند که شامل ایجاد یا create و خواندن یا read و بروزرسانی یا update و حذف یا delete می شوند را پیاده سازی می کنیم.

عملیات read برای جدول course به صورت زیر است:

با استفاده از تابع get\_course کد درس یا course\_id را از ماژول main (که ورودی ها با استفاده از توابع به کار رفته در آن و API ها از کاربر گرفته می شود) دریافت می کند.

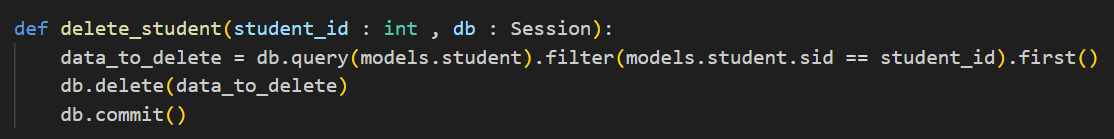
در قسمتdb.query(models.course) با استفاده از Session (پل ارتباطی بین کد ها و database است) یک درخواست (query) به database می زنیم و از آن می خواهیم با جست و جو در سطرهای جدول درس آن سطری که کد درسش با کد درس ورودی برابر است را برگرداند (.filter(models.course.cid == course\_id)) چون کد درس primary key بود بنابراین یا یک نتیجه برگردانده می شود یا هیچی (one\_or\_none به این قسمت اشاره دارد). البته می توان از(.first()) نیز همانطور که در کد های دیگر نوشتم , استفاده کرد که این تابع یعنی اگر چند نتیجه وجود داشت اولی را برگردان.

عملیات بعدی create یا ایجاد کردن است که نمونه ی آن برای جدول lecturer به صورت زیر می باشد:



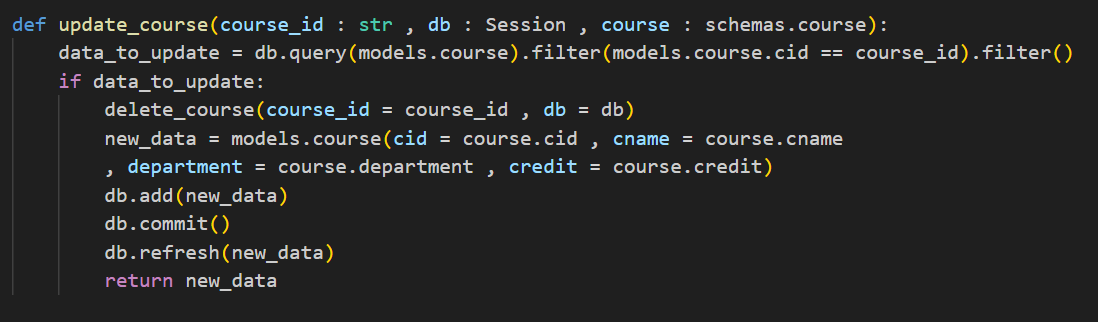
در اینجا با استفاده از تابع create\_lecturer ورودی های وارد شده توسط کاربر (به واسطه ی ارسال از طریق API ها به ماژول main و فراخوانی ماژول crud در main) را طبق مدل schemas بررسی می کند و با فراخوانی ماژول models ورودی ها را به عنوان یک سطر در جدول lecturer ذخیره می کند. هرکدام از فیلد ها را برابر ورودی هم نام خود قرار می دهد

(مثال: Lid = lecturer.id). در ادامه تغییرات ایجاد شده را با دستور (db\_lecturer).add ذخیره کرده و سپس با دستور .commit() آن ها را نهایی می کند. در نهایت به نشانه ی درست انجام شدن عملیات اطلاعات سطر ایجاد شده را برمی گرداند.

عملیات بعدی delete یا حذف کردن برای نمونه ی جدول student به صورت زیر است:

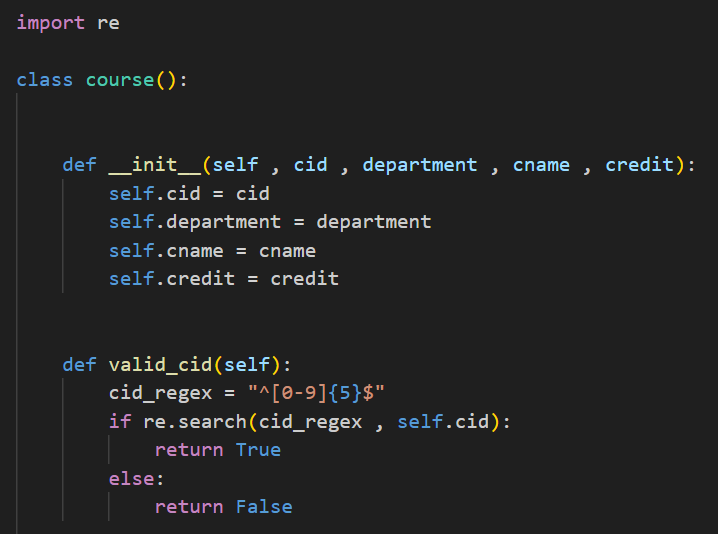
در اینجا تابع delete\_student شماره دانشجویی را از ماژول main (در ماژول main نیز با استفاده از API ها با کاربر در ارتباط هستیم) دریافت می کند. سپس از طریق متغیر data\_to\_update بررسی می کند آیا کاربری با این شماره دانشجویی موجود است یا نه. در ادامه سطر مربوط به او را حذف و تغییرات ایجاد شده را ثبت کند.

همانطور که در ادامه هنگام بررسی ماژول main خواهید دید , شرط وجود کاربری با شماره دانشجویی وارد شده , در تابعی در همان ماژول main بررسی شده است بنابراین در اینجا دیگر نیازی به بررسی این شرط نیست.

عملیات بعدی update است که برای نمونه جدول course را آورده ام.

در اینجا اطلاعات جدیدی که جهت جایگزین شدن با اطلاعات قبلی هستند و کد درس به عنوان ورودی به تابع داده شده است. ابتدا با استفاده از متغیر data\_to\_update بررسی می کند آیا سطری با کدرس داده شده وجود دارد سپس بواسطه ی یک شرط , اگر موجود بود آن را حذف کرده , یک سطر جدید را با اطلاعات داده شده ایجاد می کند (کد درس تغییر نمی کند). تغییرات را اضافه و ثبت نهایی کرده و جدول را برمی گرداند.

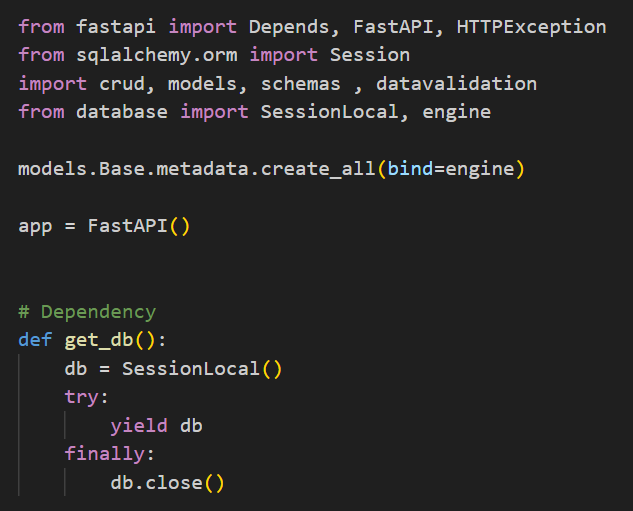
قبل از بررسی فایل main فایل datavalidation را بررسی می کنیم.



در این فایل داده های وارد شده را صحت سنجی می کنیم. برای برخی از داده های ورودی از Regular expression و برای برخی دیگر از همان نحو معمول پایتون استفاده کرده ام.

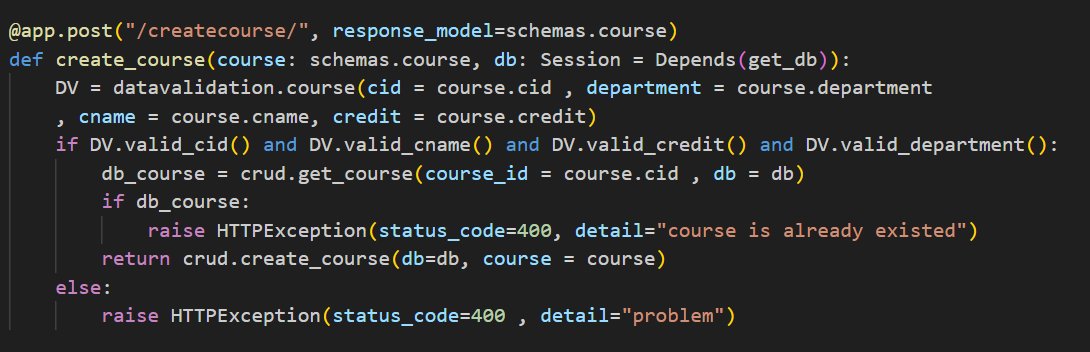
در تصویر بالا نمونه ی بررسی course id که به صورت مخفف cid نوشته شده را آورده ام. در این فایل برای صحت سنجی داده های هر جدول یک کلاس ایجاد کرده ام. اطلاعات ورودی از طریق تابع \_\_init\_\_ به کلاس ارسال شده است و از طریق توابع مختلف برای هر فیلد صحت سنجی اطلاعات مربوط به آن فیلد انجام می شود. صحت سنجی بقیه ی داده های جدول course و بقیه جداول در فایل datavalidation قابل مشاهده است .

حالا سراغ بررسی ماژول main می رویم.



در این فایل دیگر فایل های لازم import شده یک object از کلاس fastapi و همچنین یک session برای برقراری ارتباط با پایگاه داده ایجاد شده است.

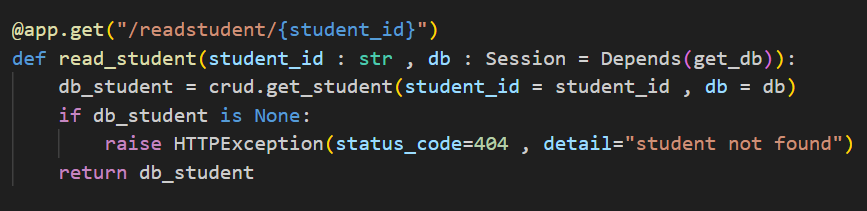
ابتدا نمونه ای از تابع read را برای جدول course می آوریم:



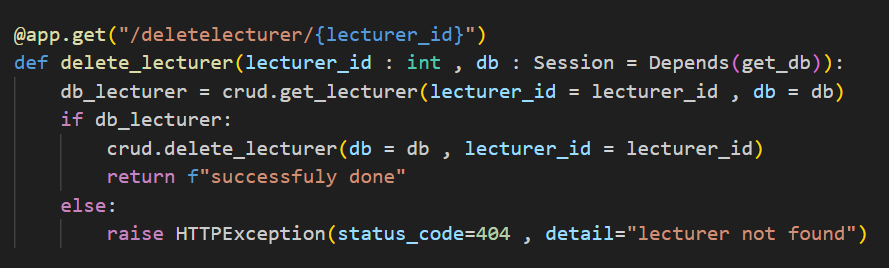
ما اطلاعات ورودی برای این تابع را با متد post ارسال می کنیم. در این تابع اطلاعات ورودی طبق مدل schems بررسی می شوند. درون تابع کلاس course را از فایل datavalidation فراخوانی کرده و اطلاعات را به آن ارسال می کنیم. در ادامه مفسر با تابع های موجود در کلاس course در ماژول datavalidation , صحت سنجی داده ها را انجام داده. با استفاده از یک شرط به مفسر می گوییم اگر صحت سنجی داده ها موفقیت آمیز بود (یعنی داده ها به درستی وارد شده بودند) ذیل یک درخواست به database بررسی کند آیا سطری با این کد درس موجود هست یا خیر. اگر موجود بود خطای course is already existed را برگرداند. اگر نه تابع create\_course را از ماژول crud فراخوانی کرده و عملیات ساخت سطری با اطلاعات ارسال شده را انجام دهد. اگر در صحت سنجی داده ها اشکالی وجود داشت (یک یا چندتا از داده ها به فرمت نادرست وارد شده اند) پیغام problem را برگرداند.

حالا نوبت به بررسی تابع read می رسد.

نمونه ی آن را برای جدول student آورده ایم.

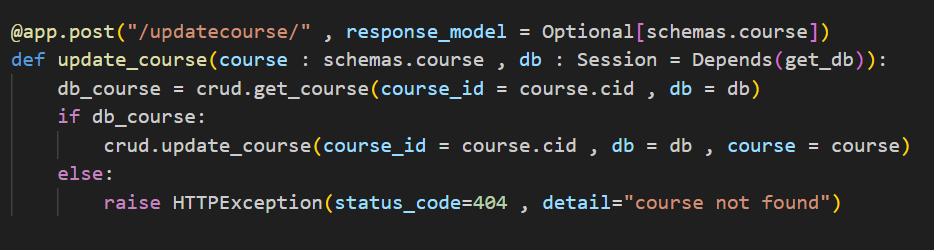


کد دانشجویی را به عنوان path parameter گرفته , درون تابع یک query به database می زند تا بررسی کند که آیا کاربری با شماره دانشجویی وارد شده وجود دارد یا خیر. در ادامه اگر کاربری با این شماره دانشجویی وجود نداشت خطای student not found را نشان می دهد و اگر کاربر موجود بود اطلاعاتش را بر می گرداند.

همچنین تابع delete\_lecturer را برای جدول استاد بررسی می کنیم.

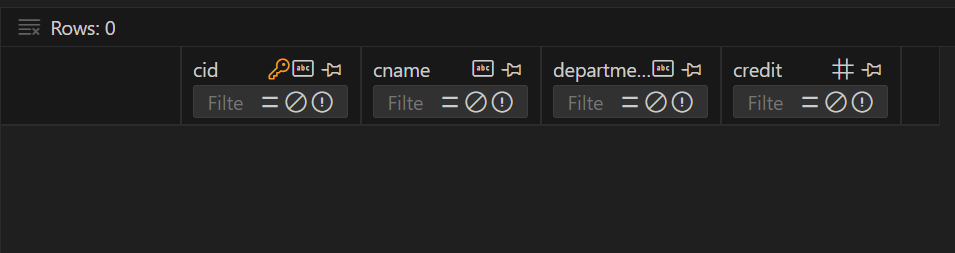
در اینجا ابتدا تابع get را از فایل crud برای بررسی موجود بودن استاد فراخوانی می کند. اگر اگر استادی با کد وارد شده موجود بود عملیات delete را روی آن اجرا می کند اگر موجود نبود ارور lecturer not found را نمایش می دهد.

در ادامه تابع update\_course را برای جدول course بررسی می کنیم.



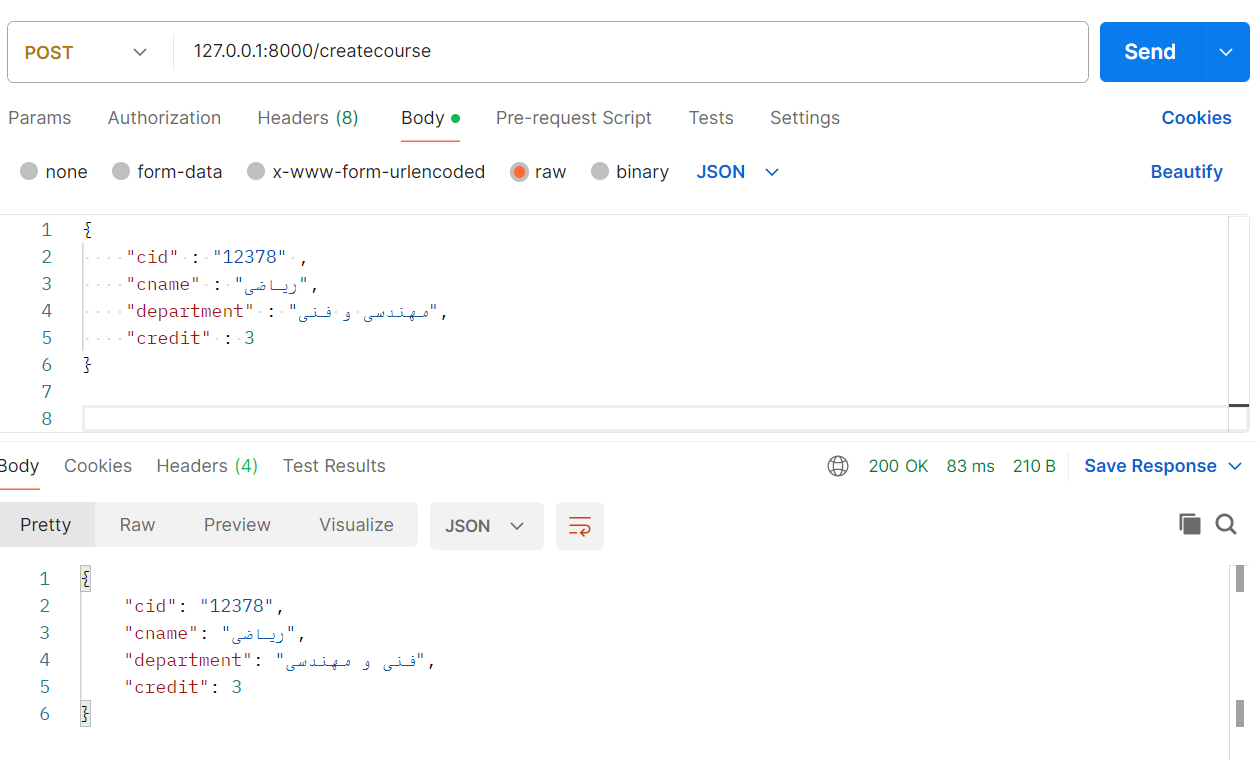
در این تابع ابتدا تابع get\_course را از ماژول crud فراخوانی کرده که در آن بررسی می کند آیا درسی با این کد وجود دارد یا خیر. اگر وجود داشت عملیات update را روی آن انجام می دهد اگر نه ارور course not found را برمی گرداند.

در نهایت هرکدام از عملیات ها را در documentation یا postman اجرا می کنیم و نتیجه ی ایجاد شده در پایگاه داده را نشان می دهیم.

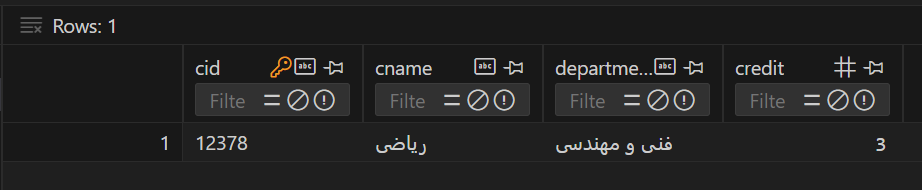
ابتدا عملیات ایجاد یا create را برای جدول course انجام می دهیم.

این حالت ابتدایی جدول است که هیچ اطلاعاتی در آن نیست.

حالا می خواهیم اطلاعات جدید به آن اضافه کنیم.



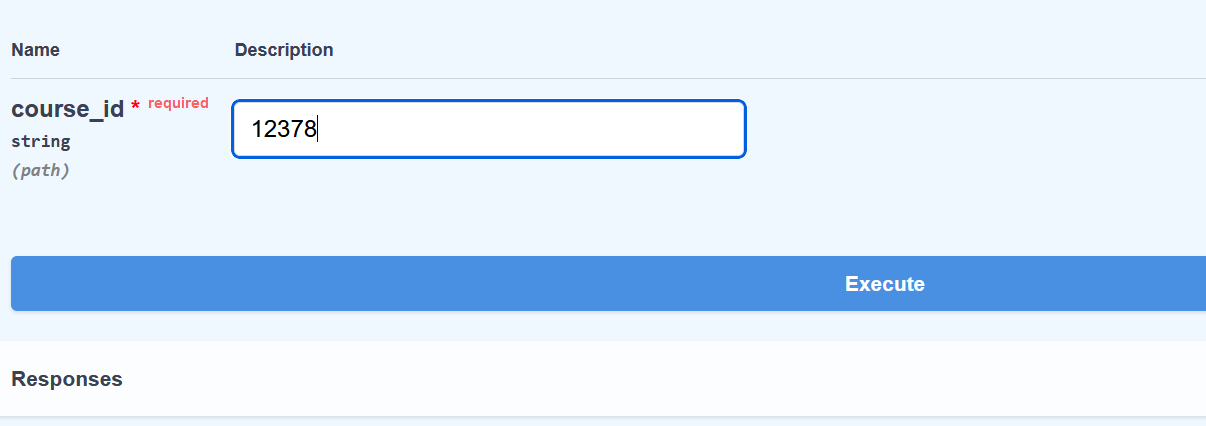
با استفاده از postman اطلاعات را ارسال کردیم و مفسر به نشانه ی درستی عملیات همانطور که برایش مشخص کرده بودیم سطر ایجاد شده را برگرداند.



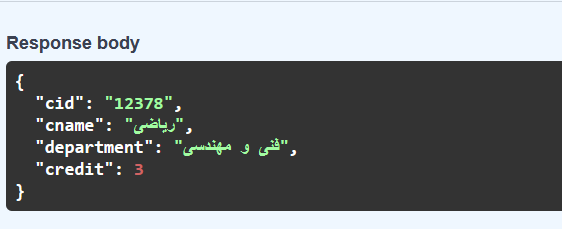
در database نیز سطر اضافه شد.

حالا می خواهیم عملیات read یا خواندن را اجرا کنیم.

با استفاده از documentation اینکار را انجام می دهیم.



کد درس را وارد می کنیم.

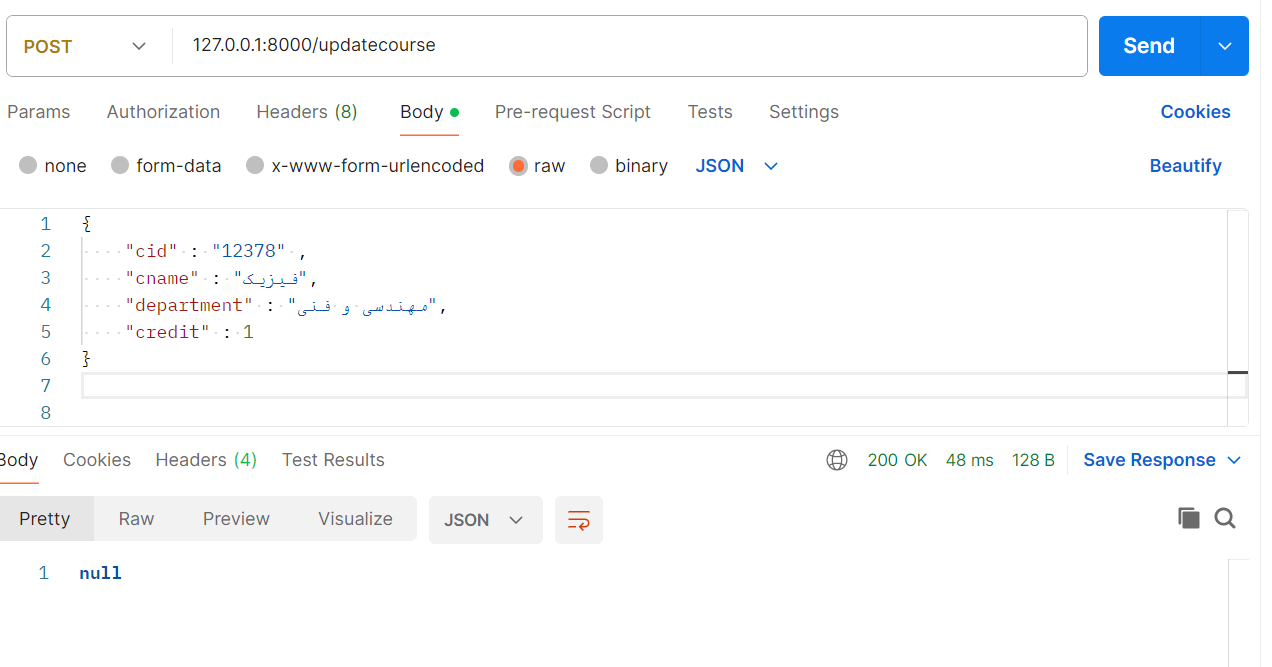


به درستی درس را خوانده و سطر را برمی گرداند.

حالا می خواهیم اطلاعات این سطر را برای ما updata کند.

منظور همان سطر با کد درس 12378 و نام: ریاضی هست.

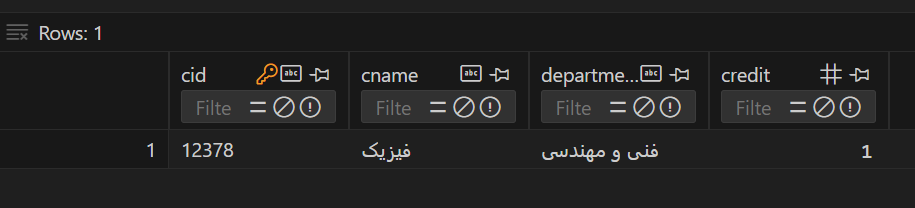
برای اینکار از postman استفاده می کنیم.



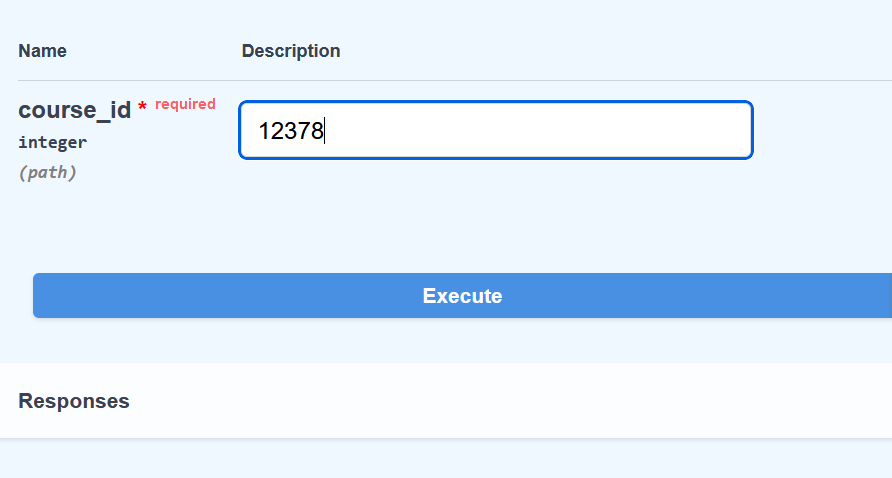
می خواهیم اطلاعات سطری با این کد درس را تغییر دهیم (کد درس تغییر نمی کند).

در جواب null را برمی گرداند اما اگر به database نگاه کنیم می بینیم تغییرات ایجاد شده است.

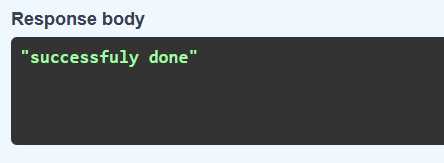
به این صورت:



حالا نوبت به عملیات delete می رسد.



کد درس را وارد کرده و پیام درستی عملیات را مشاهده می کنیم.



که نشان می دهد درس با موفقیت حذف شده است.

همانطور که در database قابل مشاهده است:

