

بسم الله الرحمن الرحيم



دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

Backend

درس:

مهندسی نرم افزار

استاد:

دکتر فاطمی

اعضای گروه:

محمد مهدی کتابچی، محمد جعفری، علی حسینی فرد، امین کیانی، متین عصب الظهور، آرمان

خلیلی، یزدان افرا

مدل های دیتابسی کنونی:

```

from django.db import models
from django.contrib.auth.models import User
from rest_framework.exceptions import ValidationError

11 usages  ↗ MJ202218
class StudentProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    stu_no = models.CharField(verbose_name="Student Number", max_length=10, blank=False, null=False)
    is_ta = models.BooleanField(default=False)
    phone_no = models.CharField(verbose_name="Phone Number", blank=False, null=False)

18 usages  ↗ MJ202218
class ProfessorProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    national_no = models.CharField(verbose_name="National Number", max_length=10, unique=True, blank=False)
    students = models.ManyToManyField(to='StudentProfile', through='StudentProfessor', related_name='professors', default=None)
    ↗ MJ202218
    def __str__(self):
        return f"{self.user} - {self.national_no}"
#intermediary model for adding some elemnts like datetime for relations if necessary
↗ MJ202218
class StudentProfessor(models.Model):
    student = models.ForeignKey(StudentProfile, on_delete=models.CASCADE, default=None)
    professor = models.ForeignKey(ProfessorProfile, on_delete=models.CASCADE, default=None)

#course model
3 usages  ↗ MJ202218
def validate_zero_or_one(value):...

```

```

class ZeroOrOneField(models.IntegerField):
    ↗ MJ202218
    def __init__(self, *args, **kwargs):
        kwargs['validators'] = [validate_zero_or_one]
        super(ZeroOrOneField, self).__init__(*args, **kwargs)

8 usages  ↗ MJ202218
class Course(models.Model):
    name = models.CharField(max_length=100, verbose_name="Course Name")
    term = models.IntegerField(verbose_name="Term")
    required_TAs = models.IntegerField(verbose_name="Required TAs")
    minPoint = models.IntegerField(verbose_name='minimum points')
    passCourse = ZeroOrOneField(verbose_name="should pass")
    description = models.TextField(verbose_name='description')
    professor = models.ForeignKey(to='ProfessorProfile', on_delete=models.CASCADE, related_name='courses')

    ↗ MJ202218
    def __str__(self):
        return f"{self.name} - Term {self.term} - Section {self.professor}"

6 usages  ↗ MJ202218
class Requests(models.Model):
    course = models.ForeignKey(to='Course', on_delete=models.CASCADE, related_name='course_id')
    student = models.ForeignKey(to='StudentProfile', on_delete=models.CASCADE, related_name='student_id')
    enter_year = models.IntegerField(verbose_name='sal vorodi')
    field_of_study = models.CharField(max_length=70)
    point = models.IntegerField(verbose_name='point of ta')
    gpa = models.FloatField(verbose_name='moadele daneshjoi')
    status = models.CharField(max_length=10, choices=[('accept', 'Accept'), ('decline', 'Decline'), ('uncertain', 'Uncertain')])

```

API ها:

Login

:URL <http://127.0.0.1:8000/users/login/>

method : POST

ورودی های الزامی :

Login

POST /users/login/

HTTP 400 Bad Request

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "username": [
    "This field is required."
  ],
  "password": [
    "This field is required."
  ]
}
```

خروجی: در حالت صحیح کد ۲۰۰ در حالت هر حالت نادرست ۴۰۰
خروجی درحالی که کاربر استاد باشد:

```
POST /users/login/
```

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "token": "82d574cedc113f5ee0a8a02eb4af80e076125721",
  "user_data": {
    "id": 1,
    "role": "professor",
    "username": "testuser",
    "first_name": "bro",
    "last_name": "yechi_dige",
    "national_no": "99546448",
    "email": "test@example.com"
  }
}
```

از قسمت role آن میتوان متوجه پروفسور بودن آن شد همچنین مقدار id متعلق به یوزری است که در پروفایل استاد استفاده شده. این در قسمت های بعدی استفاده زیادی خواهد داشت.

خروجی درحالی که کاربر دانشجو باشد:

Login

POST /users/login/

HTTP 200 OK

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "token": "d34b0d6f08bcafdb86b0022c6105e332111a11cc",
  "user_data": {
    "username": "testuserst",
    "first_name": "yechi",
    "last_name": "yechi_dige",
    "phone_no": "886454665",
    "stu_no": "1222334455",
    "is_ta": false,
    "email": "test@example.com"
  }
}
```

Professor register(sign-up)

url : <http://127.0.0.1:8000/users/professor-register/>

method : POST

ورودی های الزامی :

Professor Register

POST /users/professor-register/

HTTP 400 Bad Request

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "user": [
    "This field is required."
  ],
  "national_no": [
    "This field is required."
  ],
  "password2": [
    "This field is required."
  ]
}
```

ورودی های اصلی (ورودی های که به طور عادی انتظار آنها را داریم) :

```
{  
  "user": {  
    "username": "testuser",  
    "first_name": "yechi",  
    "last_name": "yechi_dige",  
    "email": "test@example.com",  
    "password": "testpassword"  
  },  
  "national_no": "1222334455",  
  "password2": "testpassword"  
}
```

خروجی در صورت اطلاعات تکراری یا غلط:

Professor Register

POST /users/professor-register/

HTTP 400 Bad Request

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "user": {
    "username": [
      "A user with that username already exists."
    ]
  },
  "national_no": [
    "professor profile with this National Number already exists."
  ]
}
```


خروجی در صورت درخواست صحیح (برگرداندن اطلاعات کاربری که ثبت شده)

```
"student successfully registered!"
```

Student register(sign-up)

url : <http://127.0.0.1:8000/users/student-register/>

method : POST

ورودی های الزامی :

Student Register

POST /users/student-register/

HTTP 400 Bad Request

Allow: POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "user": [
    "This field is required."
  ],
  "stu_no": [
    "This field is required."
  ],
  "phone_no": [
    "This field is required."
  ],
  "password2": [
    "This field is required."
  ]
}
```

ورودی های اصلی(ورودی های که به طور عادی انتظار آنها را داریم) :

```
{  
  "user": {  
    "username": "testuser9",  
    "first_name": "yechi",  
    "last_name": "yechi_dige",  
    "email": "test@example.com",  
    "password": "testpassword"  
  },  
  "stu_no": "1222334455",  
  "phone_no": "0991199853",  
  "password2": "testpassword"  
}
```

خروجی صحیح:

```
"professor successfully registered!"
```

خروجی در صورت اطلاعات غلط شبیه قبلی

Logout

url : <http://127.0.0.1:8000/users/logout/>

method :GET

ورودی های الزامی :

بدون اینکه وارد شده باشیم اجازه دسترسی نداریم و در ریکوئست گت باید توکن موجود باشد.

Logout

GET /users/logout/

HTTP 401 Unauthorized

Allow: GET, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

WWW-Authenticate: Token

```
{  
  "detail": "Authentication credentials were not provided."  
}
```

ورودی صحیح:

Authorization: token b2d617908123bd501e62f9bc8c81e3301e7e4aa5

باید هدری با این مشخصات به طوری که توکن بعد عبارت token باید با یک فاصله نوشته شود پاس داده شود

خروجی با اطلاعات غلط:

شبهه عکس ورودی های الزامی

خروجی با اطلاعات صحیح :

تنها کد ۲۰۰ برگردانده میشود.

ProfessorCourseAPIView

url : http

://127.0.0.1:8000/users/professor/get-lesson/

method :GET

درس‌های مربوط به هر استاد داده می‌شوند در صورت احراز هویت شده بودن.

ورودی های الزامی :

بدون اینکه وارد شده باشیم اجازه دسترسی نداریم و در ریکوئست گت باید توکن موجود باشد

مشابه عکس ورودی الزامی Logout

ورودی صحیح:

```
Authorization: token b2d617908123bd501e62f9bc8c81e3301e7e4aa5
```

باید هدری با این مشخصات به طوری که توکن بعد عبارت token باید با یک فاصله نوشته شود پاس داده شود

خروجی با اطلاعات غلط:

شبهه عکس ورودی های الزامی logout

خروجی با اطلاعات صحیح :

کد ۲۰۰ برگردانده میشود. همراه لیستی از دروس

```
1 #Login professor
2 Send Request
3 POST http://127.0.0.1:8000/users/professor-login/
4 Content-Type: application/json
5 { "username": "mohammadmahdi", "password": "zxcvbnm,./" }
6
7 ###
8 #Get courses professor
9 Send Request
10 GET http://127.0.0.1:8000/users/professor/mohammadmahdi
11 Authorization: token 10b8cdf9d18e6cdbae904a1f822215a02a3c1128
12
13 HTTP/1.1 200 OK
14 Date: Thu, 18 Apr 2024 23:24:27 GMT
15 Server: WSGIServer/0.2 CPython/3.12.0
16 Content-Type: application/json
17 Allow: GET, HEAD, OPTIONS
18 X-Frame-Options: DENY
19 Content-Length: 117
20 X-Content-Type-Options: nosniff
21 Referrer-Policy: same-origin
22 Cross-Origin-Opener-Policy: same-origin
23
24 [
25   {
26     "id": 2,
27     "name": "data structure",
28     "term": 1,
29     "required_TAs": 1,
30     "num_applicants": 1,
31     "num_tas": 1,
32     "section": 1,
33     "professor": 2
34   }
35 ]
```

CourseRegisterView

Method: post

url : http

://127.0.0.1:8000/users/professor/create-lesson/

ورودی ها:

```
data = {  
    "name": "data structure",  
    "term": 4022,  
    "required_TAs": 5,  
    "num_applicants": 10,  
    "num_tas": 3,  
    "section": 2,  
    "professor": 1 # Assuming ProfessorProfile has an auto-incremented id  
}
```

فیلد پروفسور همان آیدی ای را می گیرد که در قسمت لاگین بر می گردانیم.

خروجی:

```
f"lesson '{name}' is added to database",
```

جای name اسم درس قرار می گیرید

نکته:

اسم درس دوبار نمی تواند تکرار شود.

ProfessorDetailView

method: get

url : `http://127.0.0.1:8000/users/ professor/get-detail/<int:professor_id>`

ورودی:

شماره آیدی پروفسور (گرفته شده در لاگین)

خروجی:

```
{"professor":{"user":1,"national_no":"99546448","students":[]},"students":[]}
```

لیست دانشجو های اول دانشجو هایی هستند که متعلق به استادند. و لیست دانشجو های دوم ریز اطلاعات مربوط به آن دانشجویان.

StudentProfilePartialUpdateView

method: patch

url :

`http://127.0.0.1:8000/users/student/profile/<int:id>/`

اطلاعات دانشجویان به طور تکه ای آپدیت میکند یعنی لازم نیست حتما کل آن یکجا آپدیت شود قسمت های جدا هم آپدیت می شود. فقط ویژگی ها خاص دانشجو مثل

شماره تلفن و شماره دانشجویی و تی ای بودن می تواند آپدیت شود. اسم و فامیل و ایمیل و ... با آپدیت یوزر آپدیت میشود

ورودی:

مثلا میخواهم فقط stu_no بودن رو آپدیت کنم.

```
data = {  
  'stu_no': '99546448',  
}
```

اگر چیزی دیگه خواستم اونا رو هم اضافه می کنم. همچنین ای دی همونی هست که از لاگین گرفتیم.(مال یوزره)

خروجی:

اطلاعات جدید دانشجو

ProfessorProfilePartialUpdateView

method:patch

url

:http://127.0.0.1:8000/users/student/profile/<int:id>/

ورودی

شبهه آپدیت دانشجو یعنی مثلا چیزی مثل کدملی رو آپدیت می کنی نه اسم و فامیل چون اون مال آپدیت یوزر می شه.

خروجی

شبيه آپدیت دانشجو

UserPartialUpdateView

method:patch

url :http://127.0.0.1:8000/users/ update/<int:id>/

ورودی

```
data = {  
    'first_name':"bro",  
}
```

ویژگی های یوزر به طور بخش بخش یا کامل آپدیت می شود. آیدی هم همان آیدی ای است که موقع لاگین میگیریم.

خروجی

اطلاعات جدید یوزر

CourseDeleteView

method:delete

url :http://127.0.0.1:8000/users/ professor/delete-
lesson/<str:name>/

ورودی:

اسم درس را برای ورودی باید داده شود.

خروجی: ندارد

RequestView

method: get, post, patch

```
url: http://127.0.0.1:8000/users/request/< str:role >/< int:id >
```

در اینجا با روش *get* اگر استاد وارد شود اطلاعات درخواست‌های مرتبط با استاد نمایش داده می‌شود و اگر دانشجو وارد شود اطلاعات درخواست‌های دانشجو نمایش داده می‌شود.

The screenshot displays a web browser window with a REST client interface. At the top, there's a header area with a "Request" tab selected. Below this, the URL bar shows "GET /users/request/student/1". To the right of the URL bar are two buttons: "OPTIONS" and "GET".

The main content area shows the response details:

- Status: HTTP 200 OK
- Headers:
 - Allow: GET, POST, PATCH, HEAD, OPTIONS
 - Content-Type: application/json
 - Vary: Accept
- Body: A JSON array containing one object representing a student.

The JSON body is as follows:

```
[ {  
  "id": 1,  
  "course": 1,  
  "student": 1,  
  "enter_year": 1400,  
  "field_of_study": "computer engineering",  
  "point": 1,  
  "gpa": 15.5,  
  "status": "uncertain",  
  "studentFirstName": "",  
  "studentLastName": "",  
  "studentNo": "4003623011",  
  "courseName": "Software Engineering",  
  "courseDescription": "استادپاران بايد فعال باشند. استادپاران بايد فعال باشند. استادپاران بايد فعال باشند. استادپاران بايد فعال باشند. استادپاران بايد فعال باشند.",  
  "courseMinpoint": 15,  
  "courseTerm": 1400,  
  "professorFirstName": "",  
  "professorLastName": ""  
}]
```

خروجی برای کاربر دانشجو

Request

Request

OPTIONS

GET

GET /users/request/professor/1

HTTP 200 OK

Allow: GET, POST, PATCH, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
[
  {
    "id": 1,
    "course": 1,
    "student": 1,
    "enter_year": 1400,
    "field_of_study": "computer engineering",
    "point": 1,
    "gpa": 15.5,
    "status": "uncertain",
    "studentFirstName": "",
    "studentLastName": "",
    "studentNo": "4003623011",
    "courseName": "Software Engineering",
    "courseDescription": "ادایاران باید فعال باشند. استادیاران باید فعال باشند. استادیاران باید فعال باشند. استادیاران باید فعال باشند. استادیاران باید فعال باشند.",
    "courseMinpoint": 15,
    "courseTerm": 1400,
    "professorFirstName": "",
    "professorLastName": ""
  }
]
```

خروجی برای کاربر استاد

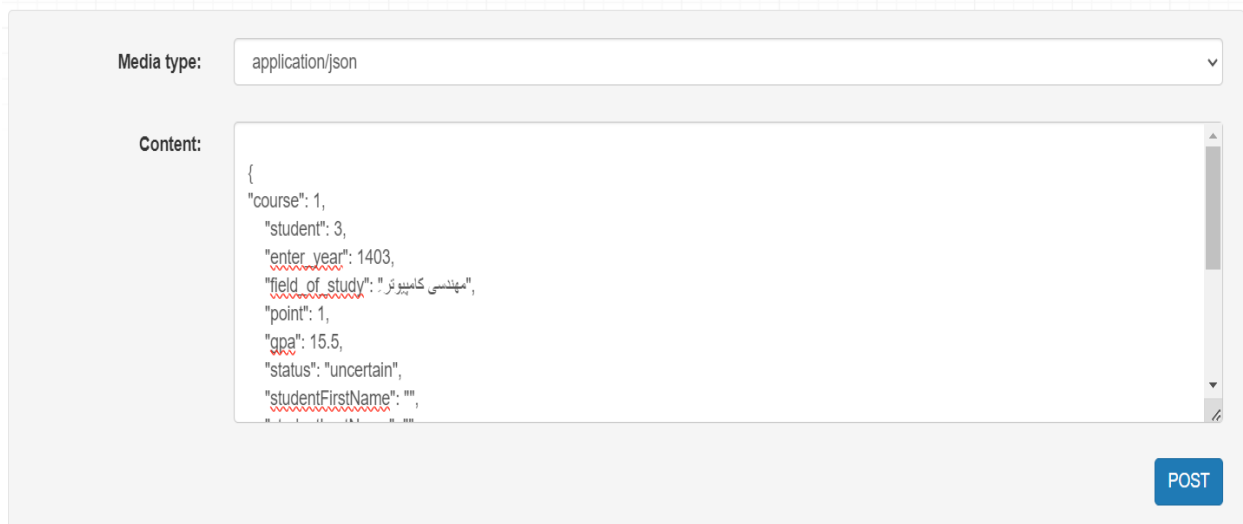
همچنین با روش *post* هم از سمت استاد و هم از سمت دانشجو امکان ذخیره درخواست‌ها فراهم می‌شود.

برای مثال فرض کنیم می‌خواهیم درخواست زیر را *post* کنیم:

API Documentation

```
{
  "course": 1,
  "student": 3,
  "enter_year": 1403,
  "field_of_study": "مهندسی کامپیوتر",
  "point": 1,
  "gpa": 15.5,
  "status": "uncertain",
  "studentFirstName": "",
  "studentLastName": "",
  "studentNo": "4003623012",
  "courseName": "ساختمان داده",
  "courseDescription": "استادیار باید وقت خالی زیادی داشته باشد",
  "courseMinpoint": 16,
  "courseTerm": 1404,
  "professorFirstName": "",
  "professorLastName": ""
}
```

برای اینکار این اطلاعات را به صورت جیسون در محیطی که *Django REST Framework* فراهم کرده می‌نویسیم و سپس روی دکمه *post* کلیک می‌کنیم:



The screenshot shows a REST client interface with a 'Media type' dropdown set to 'application/json'. The 'Content' field contains a JSON object with the following fields: 'course' (1), 'student' (3), 'enter_year' (1403), 'field_of_study' (مهندسی کامپیوتر), 'point' (1), 'gpa' (15.5), 'status' (uncertain), 'studentFirstName' (empty string), 'studentLastName' (empty string), 'studentNo' (4003623012), 'courseName' (ساختمان داده), 'courseDescription' (استادیار باید وقت خالی زیادی داشته باشد), 'courseMinpoint' (16), 'courseTerm' (1404), 'professorFirstName' (empty string), and 'professorLastName' (empty string). A blue 'POST' button is located at the bottom right of the interface.

نتیجه به صورت زیر است:

```
{
  "id": 4,
  "course": 1,
  "student": 3,
  "enter_year": 1403,
  "field_of_study": "مهندسی کامپیوتر",
  "point": 1,
  "gpa": 15.5,
  "status": "uncertain",
  "studentFirstName": "",
  "studentLastName": "",
  "studentNo": "4003623013",
  "courseName": "ساختمان داده",
  "courseDescription": "استادیار باید وقت خالی زیادی داشته باشد",
  "courseMinpoint": 16,
  "courseTerm": 1404,
  "professorFirstName": "",
  "professorLastName": ""
}
```

با استفاده از روش *patch* می‌توانیم وضعیت درخواست‌ها را عوض کنیم. برای مثال اگر بخواهیم درخواست ۲ را قبول کنیم به صورت زیر عمل می‌کنیم:

Media type:

Content:

```
{
  "id": 2,
  "status": "accept"
}
```

PATCH

نتیجه به صورت زیر می شود:

Request

Request

OPTIONS GET

PATCH /users/request/professor/1

HTTP 200 OK
Allow: GET, POST, PATCH, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id": 2,
  "course": 1,
  "student": 2,
  "enter_year": 1400,
  "field_of_study": "computer engineering",
  "point": 1,
  "gpa": 15.5,
  "status": "accept",
  "studentFirstName": "",
  "studentLastName": "",
  "studentNo": "4003623012",
  "courseName": "ساختمان داده",
  "courseDescription": "استادیار باید وقت خالی زیادی داشته باشد",
  "courseMinpoint": 16,
  "courseTerm": 1404,
  "professorFirstName": "",
  "professorLastName": ""
}
```

allCourseInStudent

method: get

url: <http://127.0.0.1:8000/users/student/home/>

در اینجا با روش *get* تمام دروس برای دانشجویانی که برای آنها توکن تشکیل شده است (لاگین کرده اند) نمایش داده می شود.

All Course In Student

GET /users/student/home/

HTTP 200 OK

Allow: OPTIONS, GET

Content-Type: application/json

Vary: Accept

```
[
  {
    "id": 1,
    "term": 1404,
    "required_TAs": 3,
    "minPoint": 16,
    "passCourse": 1,
    "description": "استادیار باید وقت خالی زیادی داشته باشد",
    "professor": 1,
    "professorName": "mohammad",
    "name": "ساختمان داده"
  },
  {
    "id": 2,
    "term": 1404,
    "required_TAs": 8,
    "minPoint": 16,
    "passCourse": 0,
    "description": "استادیاران باید به 4 گروه تقسیم بندی بشوند",
    "professor": 1,
    "professorName": "mohammad",
    "name": "مهندسی نرم افزار"
  }
]
```

authentication.py

کد:

```

from rest_framework import authentication
class BearerAuthentication(authentication.TokenAuthentication):
    """
    Simple token based authentication using utvsapitoken.
    Clients should authenticate by passing the token key in the
    'Authorization'
    HTTP header, prepended with the string 'Bearer '. For example:
    Authorization: Bearer 956e252a-513c-48c5-92dd-bfddc364e812
    """
    keyword = ['token', 'bearer']
    def authenticate(self, request):
        auth = authentication.get_authorization_header(request).split()
        if not auth:
            return None
        if auth[0].lower().decode() not in self.keyword:
            return None

        if len(auth) == 1:
            msg = _('Invalid token header. No credentials provided.')
            raise authentication.exceptions.AuthenticationFailed(msg)
        elif len(auth) > 2:
            msg = _('Invalid token header. Token string should not contain
spaces.')
            raise authentication.exceptions.AuthenticationFailed(msg)
        try:
            token = auth[1].decode()
        except UnicodeError:
            msg = _('Invalid token header. Token string should not contain
invalid characters.')
            raise
        authentication.TokenAuthentication.exceptions.AuthenticationFailed(msg)
        return self.authenticate_credentials(token)

```

کد بالا برای احراز هویت با استفاده از توکن‌های نوع *Bearer* و *token* مورد استفاده قرار می‌گیرد. این کلاس از کلاس پایه *TokenAuthentication* از کتابخانه *rest – framework* استفاده می‌کند.

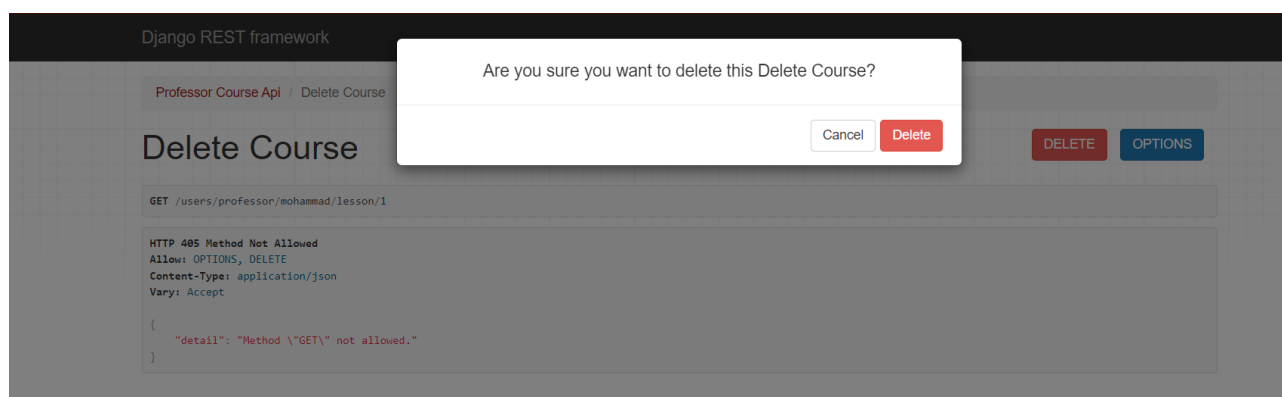
deleteCourse

method: **DELETE**

url: <http://127.0.0.1:8000/users/professor/<str:name>/lesson/<int:id>>

در اینجا با روش *delete* با استفاده از اسم استاد و آیدی درس می‌توانیم درس را حذف کنیم.

با کلیک روی *delete* درس مورد نظر با آیدی وارد شده حذف می‌شود.



نتیجه:

