

Topic: 4. Setting API EndPoints

Speaker: Personal / Notebook: API Development using Django Framework



For other resources on how to create simple API endpoints using Django, use this [reference in Medium](#).

1. Create an API app in your Django project:

```
$ python manage.py startapp api
```

2. Register this new app in your SETTINGS.PY

```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'rest_framework',
41     'students',
42     'api',
43 ]
44
45 MIDDLEWARE = [
46     'django.middleware.security.SecurityMiddleware',
47     'django.contrib.sessions.middleware.SessionMi
```

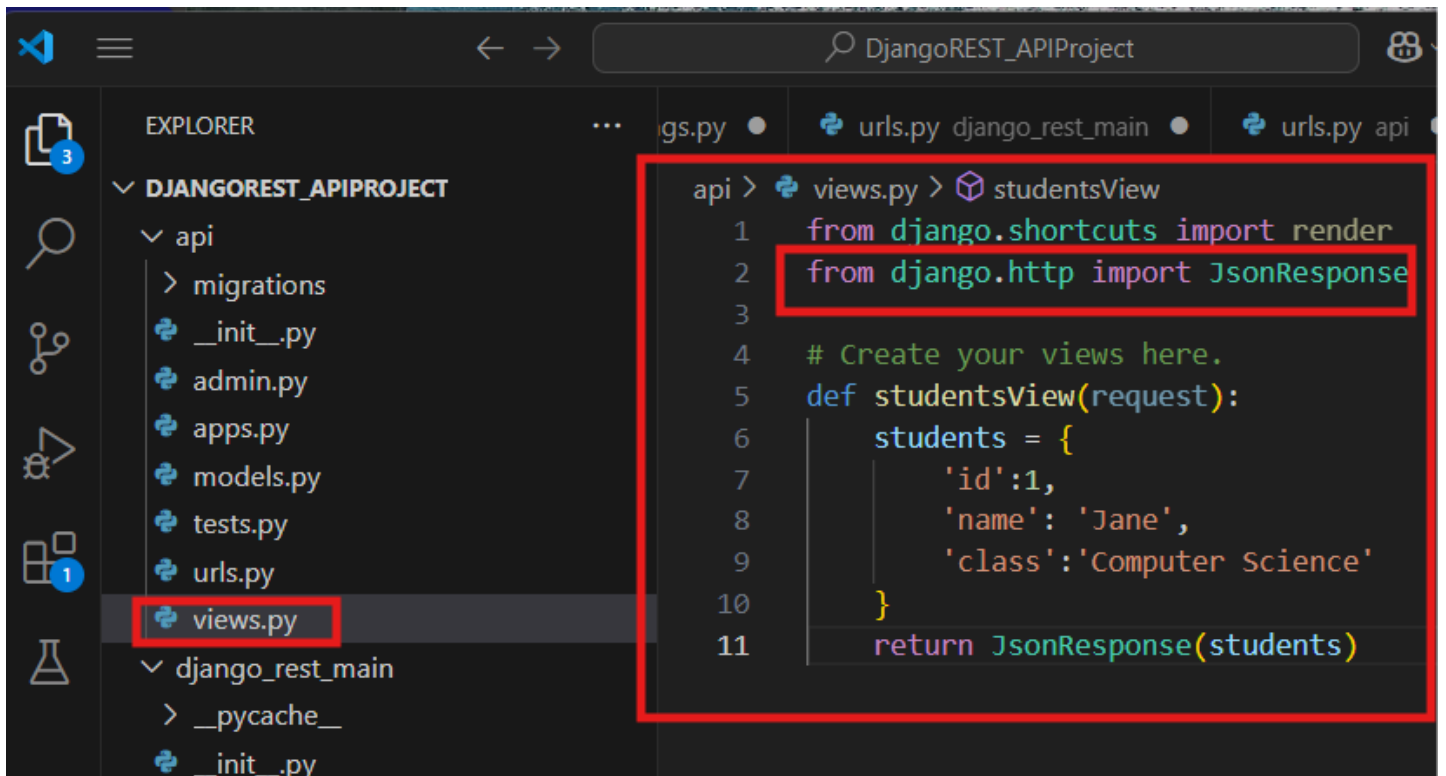
3. Now, update the main project's URLS.PY to include the URLS.PY of the newly created app.

```
13 including another URLCONF
14 1. Import the include() function: from django.urls
15 2. Add a URL to urlpatterns: path('blog/', include
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     # Web application endpoint
23     path('students/', include('students.urls')),
24
25     ...# API Endpoints
26     ...path('api/v1/', include('api.urls'))
27 ]
28
29
```

4. We then create a URLS.py in our API app and create the path.

```
api > urls.py > ...
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('students', views.studentsView)
6 ]
```

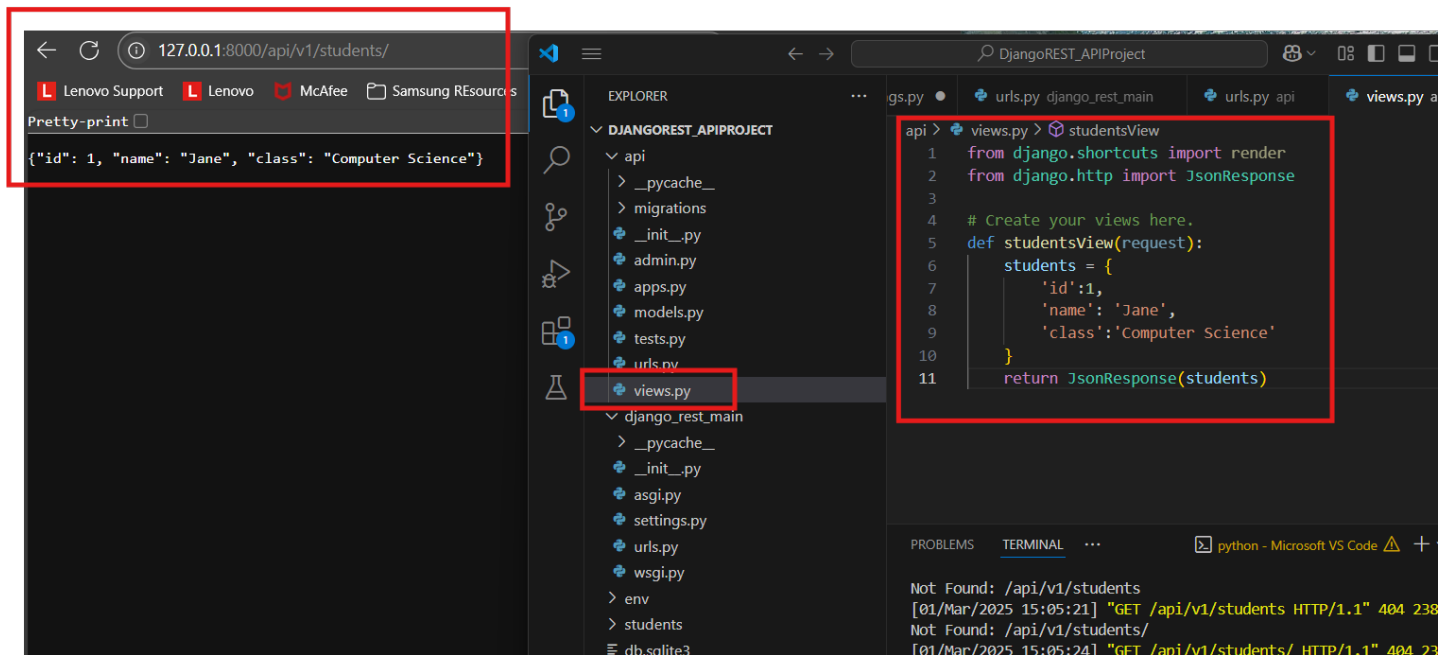
5. Update the API APP'S VIEWS.PY . Unlike regular views of web-based endpoints, APIs return JSON data.



6. Run the server and add the API endpoint:

```
$ python manage.py runserver
```

In your browser: 127.0.0.1:8000/api/v1/students/



7. Run the migrations command and create superuser.

```
$ python manage.py migrate
```

This will create user, auths default table.

File Edit Selection View Go Run Terminal Help

DjangoREST_APIProject

EXPLORER

DJANGOREST_APIPROJECT

api

- > __pycache__
- > migrations
- __init__.py
- admin.py
- apps.py
- models.py
- tests.py
- urls.py
- views.py

django_rest_main

- > __pycache__
- __init__.py
- asgi.py
- settings.py
- urls.py
- wsgi.py
- > env
- > students
- db.sqlite3
- manage.py

api > views.py > studentsView

```
5 def studentsView(request):
6     students = {
7         'id':1,
8         'name': 'Jane',
9         'class':'Computer Science'
10    }
11    return JsonResponse(students)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

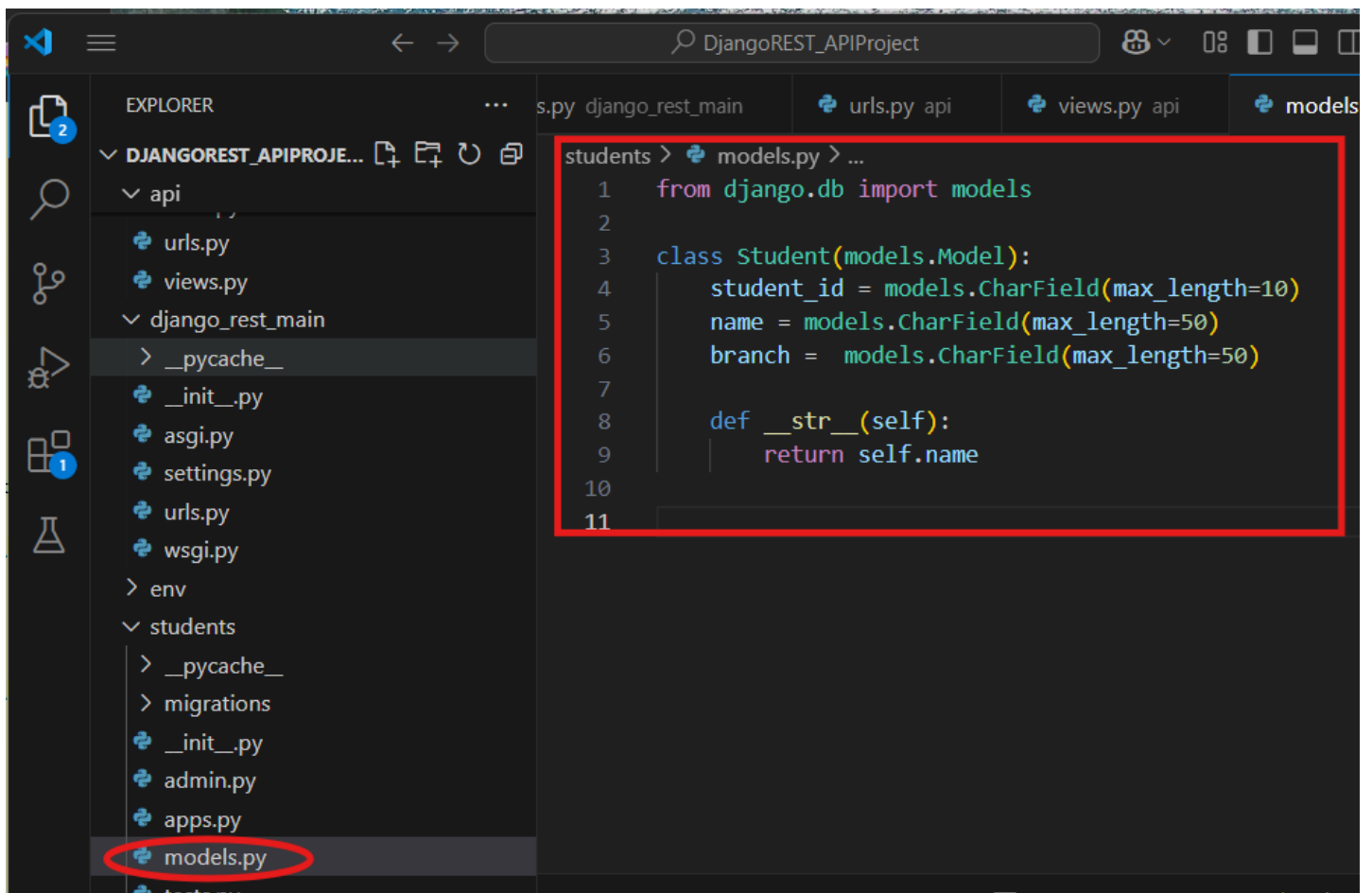
rosil@LearnCodeRepeat MINGW64 C:/Users/rosil/AppData/Local/Programs/Microsoft VS Code

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(env)
rosil@LearnCodeRepeat MINGW64 C:/Users/rosil/AppData/Local/Programs/Microsoft VS Code
$
```

8. Access the admin panel by running the server. Access it using:

<http://127.0.0.1:8000/admin/>

9. Create a student model in the STUDENTS app to create a new table.



10. Run the migrations.

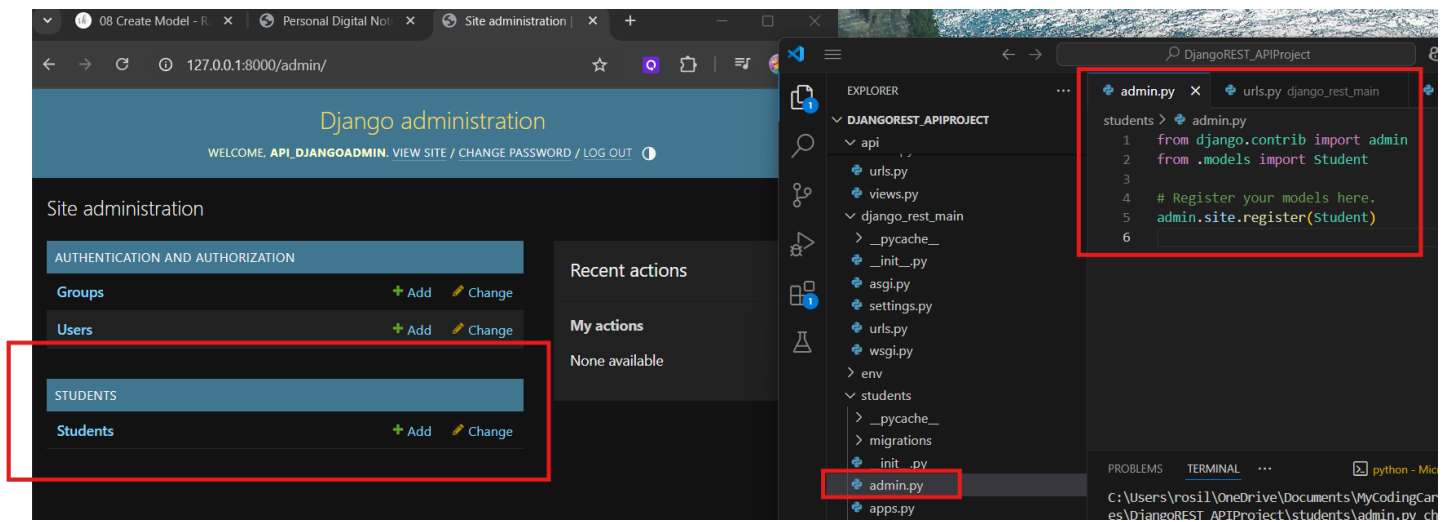
```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

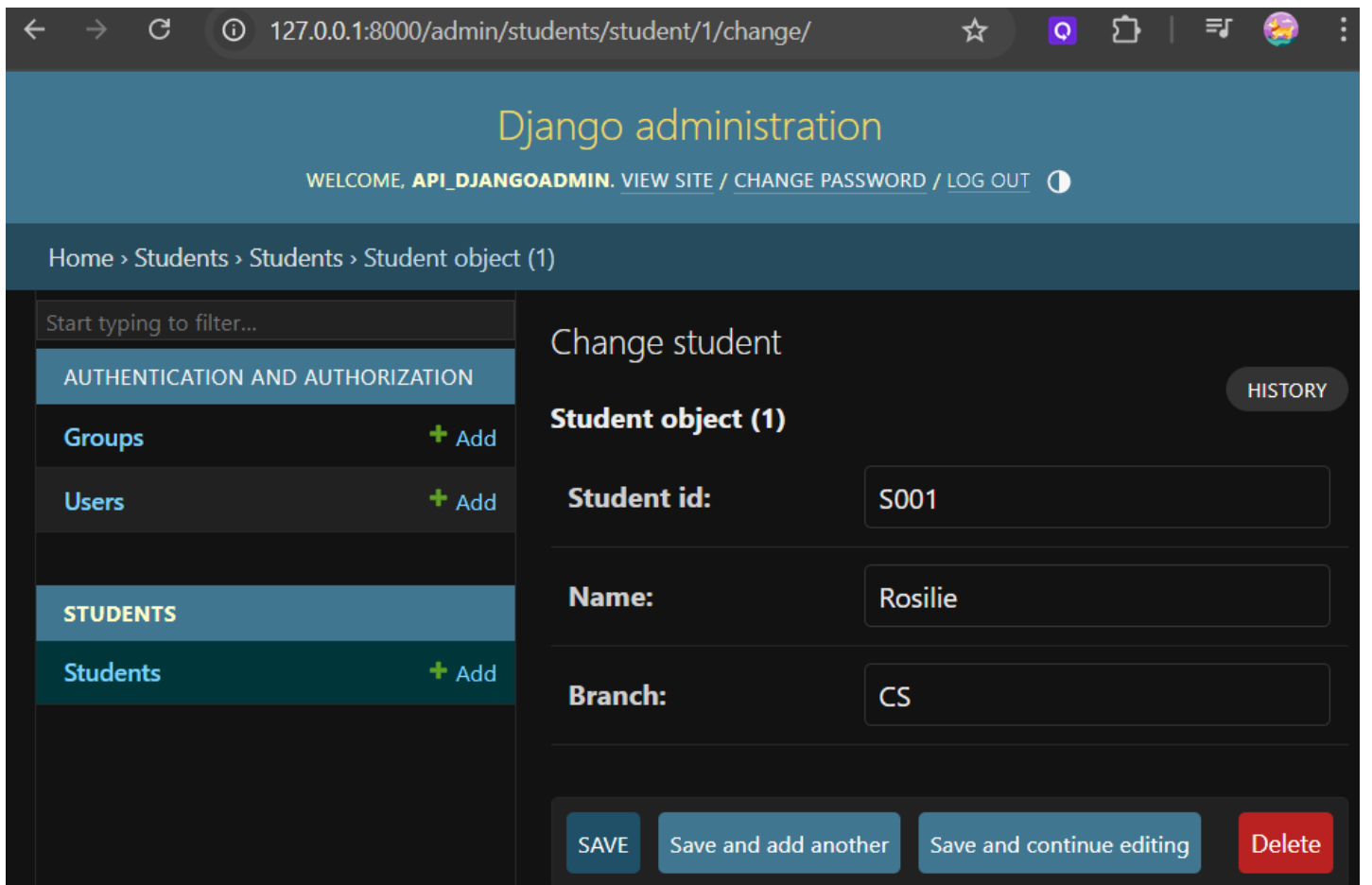
```
• $ python manage.py makemigrations
Migrations for 'students':
  students\migrations\0001_initial.py
    + Create model Student
(env)
```

```
(env)
rosil@LearnCodeRepeat MINGW64 C:/Users/rosil/AppData/Local/Programs/Mi
Code
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, students
Running migrations:
  Applying students.0001_initial... OK
(env)
```

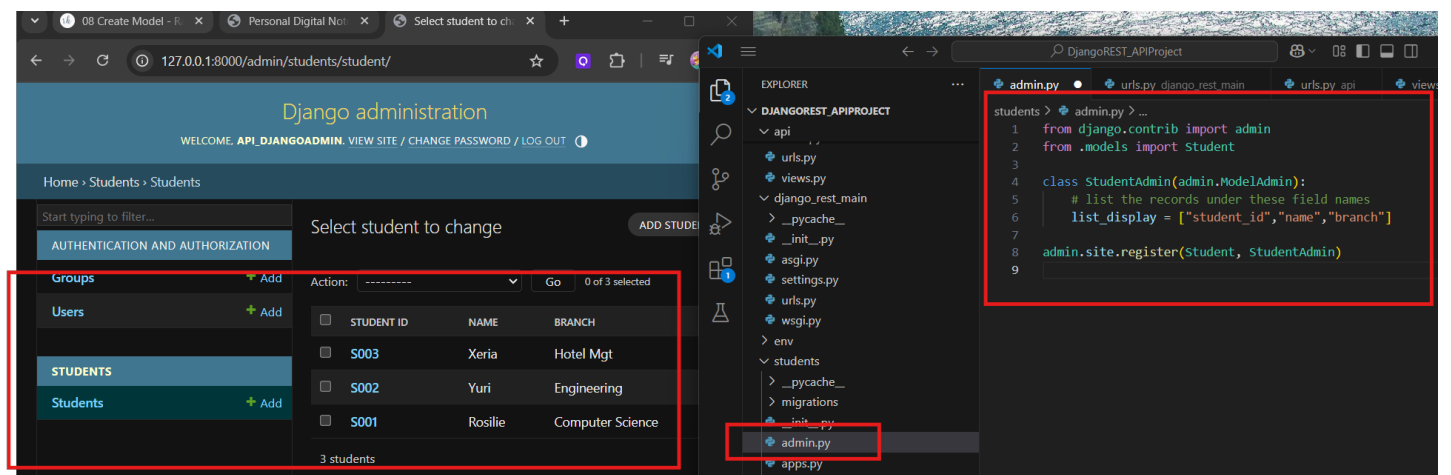
11. To make this table appear on your admin dashboard, register this in the ADMIN.PY of STUDENTS app.



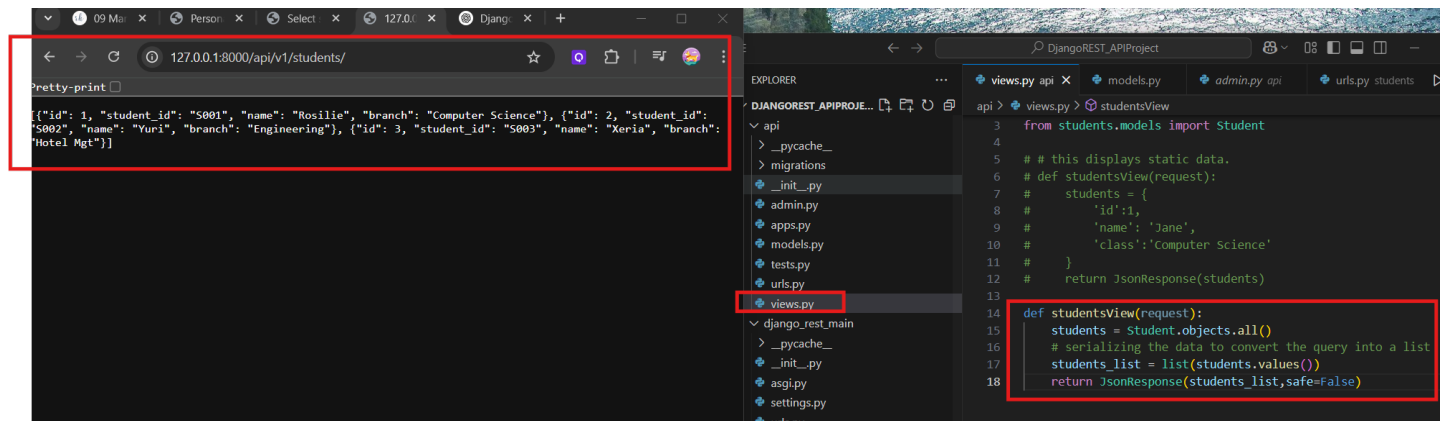
12. Create a sample record in the table.



13. To display the model student records using the admin dashboard:



14. To display the dynamic data from the database. We need to **SERIALIZE** to convert the returned query set into a list.



15. **SERIALIZERS** are like **TRANSLATORS** that convert certain data i.e **QUERYSET** from your database into other types of data like **JSON** data that can be used on **HTML**. While **DESERIALIZERS** will reverse the translation i.e from **JSON** file into **Query set** (database records).