

GUÍA PARA HACER UNA API REST CON PYTHON DJANGO Y MySQL

Jonny Luna Guerrero

Resumen

En esta guía encontrarás una metodología muy simple pero eficaz para realizar una API REST utilizando Python como lenguaje de programación, Django como framework y MySQL para hacer la base de datos, también se utilizan otras herramientas para configurar o testear la API REST tales como Thunder Client, jsonlint.com que es una página web donde comprobamos si el formato de datos obtenido en la API REST es JSON.

Los datos que se almacenaran en la base de datos o a los cuales se accederán de la misma se llaman: nombre el cual corresponde al nombre de una escuela o Institución Educativa, teléfono de la institución y nombre_rector que corresponde al nombre del Rector de la Institución Educativa

La API REST se realiza con cinco métodos los cuales reciben el nombre de:

GET list_escuelas el cual devuelve la lista de los registros o tuplas en formato JSON.

GET read_escuela el cual devuelve un solo registro buscado por su id.

POST add-escuela el cual permite adicionar o crear una nueva escuela o registro de la misma.

PUT update_escuela el cual permite editar una escuela o registro de la misma y

DELETE delete_escuela el cual permite eliminar una escuela o registro de la misma.

Se utilizó como IDE VS Code

Procedimiento:

1. Se crea una carpeta en el computador con el nombre que desee del proyecto en este caso lo llamaré API_ESCUELA.
2. Se abre la carpeta con VS Code.
3. En Python se abre una terminal y se crea un entorno virtual con el comando

virtualenv -p python3 env

```
PS C:\API_ESCUELA> virtualenv -p python3 env
created virtual environment CPython3.11.4.final.0-64 in 1137ms
  creator CPython3Windows(dest=C:\API_ESCUELA\env, clear=False, no_vcs_ignore=False
, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, v
ia=copy, app_data_dir=C:\Users\POWER\AppData\Local\pypa\virtualenv)
    added seed packages: pip==23.3.1, setuptools==68.2.2, wheel==0.41.3
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShell
Activator,PythonActivator
PS C:\API_ESCUELA>
```

4. Se activa el entorno virtual con el comando `.\env\Scripts\activate`

```
● PS C:\API_ESCUELA> .\env\Scripts\activate
○ (env) PS C:\API_ESCUELA> █
```

5. Se instala django con el comando `pip install Django==4.2.7`

En este caso se esta usando la versión mas reciente de Django obtenida de la página web oficial

```
● PS C:\API_ESCUELA> .\env\Scripts\activate
○ (env) PS C:\API_ESCUELA> pip install Django==4.2.7
Collecting Django==4.2.7
  Using cached Django-4.2.7-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.6.0 (from Django==4.2.7)
  Using cached asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
Collecting sqlparse>=0.3.1 (from Django==4.2.7)
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Collecting tzdata (from Django==4.2.7)
  Using cached tzdata-2023.3-py2.py3-none-any.whl (341 kB)
Using cached Django-4.2.7-py3-none-any.whl (8.0 MB)
Using cached asgiref-3.7.2-py3-none-any.whl (24 kB)
Installing collected packages: tzdata, sqlparse, asgiref, Django
```

6. Se verifica que se halla instalado Django correctamente con el comando `pip list`

```
(env) PS C:\API_ESCUELA> pip list
● Package      Version
-----
asgiref        3.7.2
Django         4.2.7
pip            23.3.1
setuptools     68.2.2
sqlparse       0.4.4
tzdata        2023.3
wheel          0.41.3
```

7. Se crea el proyecto de Django en este caso usaremos el comando `django-admin startproject Proyecto_Escuela`

```
● (env) PS C:\API_ESCUELA> django-admin startproject Proyecto_Escuela
○ (env) PS C:\API_ESCUELA> █
```

8. Se accede al proyecto con el comando `cd .\Proyecto_Escuela`

```
● (env) PS C:\API_ESCUELA> cd .\Proyecto_Escuela\
○ (env) PS C:\API_ESCUELA\Proyecto_Escuela> █
```

9. Con el comando `ls` verificamos si se encuentra el archivo `manage.py`

```
(env) PS C:\API_ESCUELA\Proyecto_Escuela> ls

Directorio: C:\API_ESCUELA\Proyecto_Escuela

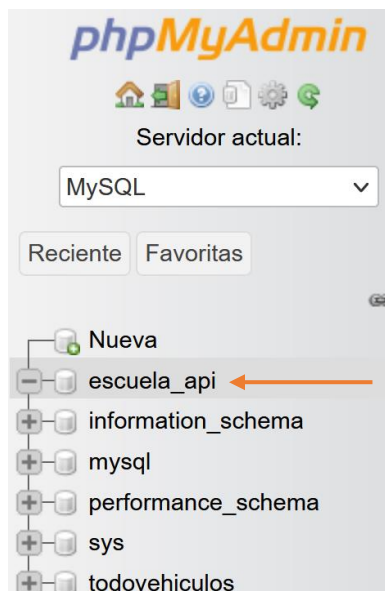
Mode                LastWriteTime         Length Name
----                -
d-----          23/11/2023   9:12 a. m.        Proyecto_Escuela
-a-----          23/11/2023   9:12 a. m.           694 manage.py
```

10. Se crea una aplicación con el nombre **api** usando el comando **django-admin startapp api**

```
(env) PS C:\API_ESCUELA\Proyecto_Escuela> django-admin startapp api
(env) PS C:\API_ESCUELA\Proyecto_Escuela>
```

Configuración de la base de datos

11. Se crea la base de datos en MySQL, en este caso se llamará **escuela_api**, su usuario es **root** y el password lo dejamos vacío, el puerto es 3306.



12. En el archivo **settings.py** de la aplicación agregamos la aplicación en **INSTALLED_APPS** en este caso agregamos **'api'**

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'api'
]

```

13. En el mismo archivo settings.py en DATABASES configuramos las variables de la base de datos.

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'HOST': 'localhost',
        'PORT': 3306,
        'USER': 'root',
        'PASSWORD': '',
        'NAME': 'escuela_api',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        }
    }
}

```

14. En el archivo **models.py** de la carpeta **migrations** se crea la clase **Escuela** con tres variables, **nombre** el cual recibirá el nombre de la Institución Educativa, **telefono** y **nombre_rector**

```

1  from django.db import models
2
3  # Create your models here.
4
5  class Escuela(models.Model):
6      nombre = models.CharField(max_length=50)
7      telefono = models.CharField(max_length=10)
8      nombre_rector = models.CharField(max_length=30)
9

```

15. En el archivo **admin.py** de la carpeta **migrations** se registra una Escuela

```

Proyecto_Escuela > api > admin.py
1  from django.contrib import admin
2  from .models import Escuela
3
4  # Register your models here.
5
6  admin.site.register(Escuela)
7

```

16. En la Terminal se instalan los conectores de la base de MySQL con el comando **pip install mysqlclient pymysql**

```

(env) PS C:\API_ESCUELA\Proyecto_Escuela> pip install mysqlclient pymysql
Collecting mysqlclient
  Downloading mysqlclient-2.2.0-cp311-cp311-win_amd64.whl.metadata (4.5 kB)
Collecting pymysql
  Downloading PyMySQL-1.1.0-py3-none-any.whl.metadata (4.4 kB)
  Downloading mysqlclient-2.2.0-cp311-cp311-win_amd64.whl (199 kB)
    200.0/200.0 kB 2.4 MB/s eta 0:00:00
  Downloading PyMySQL-1.1.0-py3-none-any.whl (44 kB)
    44.8/44.8 kB 2.2 MB/s eta 0:00:00
Installing collected packages: pymysql, mysqlclient
Successfully installed mysqlclient-2.2.0 pymysql-1.1.0
(env) PS C:\API_ESCUELA\Proyecto_Escuela>

```

Verificamos con **pip list**

```

Successfully installed mysqlclient-2.2.0 pymysql-1.1.0
(env) PS C:\API_ESCUELA\Proyecto_Escuela> pip list
Package      Version
-----
asgiref      3.7.2
Django       4.2.7
mysqlclient  2.2.0
pip          23.3.1
PyMySQL      1.1.0
setuptools   68.2.2
sqlparse     0.4.4
tzdata       2023.3
wheel        0.41.3

```

17. En la TERMINAL digitamos el comando **python manage.py migrate** para verificar que estamos conectados con la base de datos y además crear las tablas por defecto de django en la misma.

```

• (env) PS C:\API_ESCUELA\Proyecto_Escuela> 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.0005_alter_user_last_login_null... 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

```

Verificamos en la base de datos.

The screenshot shows the phpMyAdmin interface for a MySQL database named 'escuela_api'. The 'Estructura' (Structure) tab is selected, displaying a list of tables. The left sidebar shows the database hierarchy with 'escuela_api' selected. The main table list includes:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
auth_group	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	4.0 KB	-
auth_group_permissions	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	1.0 KB	-
auth_permission	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	24	MyISAM	utf8mb4_0900_ai_ci	7.1 KB	-
auth_user	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	4.0 KB	-
auth_user_groups	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	1.0 KB	-
auth_user_user_permissions	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	1.0 KB	-
django_admin_log	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	1.0 KB	-
django_content_type	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	6	MyISAM	utf8mb4_0900_ai_ci	9.1 KB	-
django_migrations	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	18	MyISAM	utf8mb4_0900_ai_ci	3.0 KB	-
django_session	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	MyISAM	utf8mb4_0900_ai_ci	1.0 KB	-
10 tablas	Número de filas	48	MyISAM	utf8mb4_0900_ai_ci	32.2 KB	0 B

18. Se crea un superusuario para acceder al panel de administración con el comando

Python manage.py createsuperuser con los siguientes valores:

Nombre de usuario: jlunag1970

Dirección de correo: jonnylunag@gmail.com

Password: 123456789

```

• (env) PS C:\API_ESCUELA\Proyecto_Escuela> Python manage.py createsuperuser
Username (leave blank to use 'power'): jlunag1970
Email address: jonnylunag@gmail.com
Password:
Password (again):
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
• (env) PS C:\API_ESCUELA\Proyecto_Escuela>

```

19. Se crea la migración del modelo usando el comando **python manage.py makemigrations**

```
● (env) PS C:\API_ESCUELA\Proyecto_Escuela> python manage.py makemigrations
Migrations for 'api':
  api\migrations\0001_initial.py
    - Create model Escuela
○ (env) PS C:\API_ESCUELA\Proyecto_Escuela> █
```

Verificamos en la carpeta **migrations** el archivo **0001_initial.py**

```
Proyecto_Escuela > api > migrations > 0001_initial.py > ...
10     dependencies = [
11     ]
12
13     operations = [
14         migrations.CreateModel(
15             name='Escuela',
16             fields=[
17                 ('id', models.BigAutoField(auto_created=True,
18                 primary_key=True, serialize=False, verbose_name='ID')),
19                 ('nombre', models.CharField(max_length=50)),
20                 ('telefono', models.CharField(max_length=10)),
21                 ('nombre_rector', models.CharField(max_length=30)),
22             ],
23         ),
24     ]
```

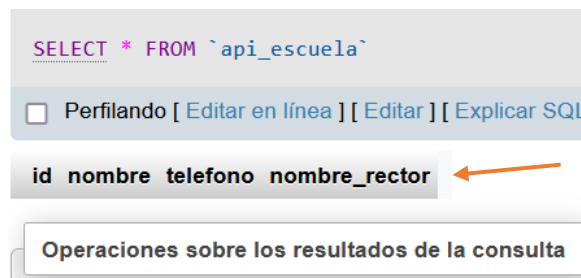
20. Se realiza de nuevo una migración con el comando **python manage.py migrate**

```
- Create model Escuela
● (env) PS C:\API_ESCUELA\Proyecto_Escuela> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, api, auth, contenttypes, sessions
Running migrations:
  Applying api.0001_initial... OK
○ (env) PS C:\API_ESCUELA\Proyecto_Escuela> █
```

Verificamos en la base de datos refrescando la estructura y observamos que ahora hay 11 tablas y la nueva tabla creada es la tabla **api_escuela**

Tabla	Acción	Fila
<input type="checkbox"/> api_escuela	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> auth_group	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> auth_group_permissions	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> auth_permission	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> auth_user	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> auth_user_groups	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> auth_user_user_permissions	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> django_admin_log	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> django_content_type	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> django_migrations	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
<input type="checkbox"/> django_session	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	
11 tablas	Número de filas	

Se observa en esta tabla las columnas de la misma.



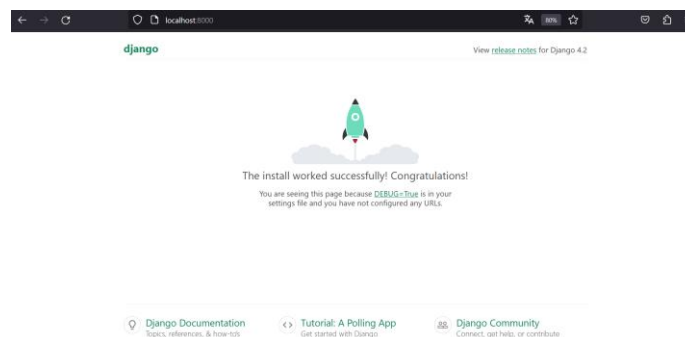
21. Con el comando **python manage.py runserver** corremos la aplicación en el servidor

```
(env) PS C:\API_ESCUELA\Proyecto_Escuela> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

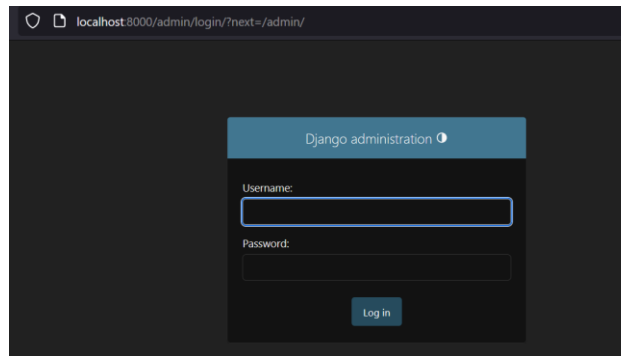
System check identified no issues (0 silenced).
November 26, 2023 - 08:45:35
Django version 4.2.7, using settings 'Proyecto_Escuela.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Se observa que la aplicación está corriendo en el servidor localhost puerto 8000.

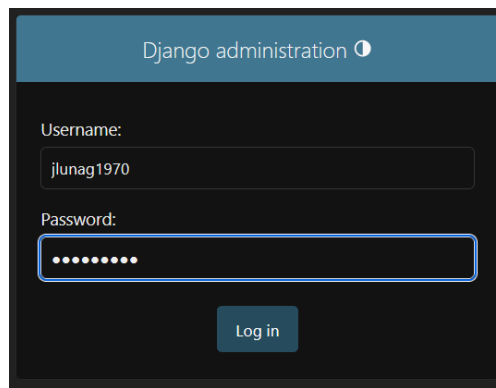
Para verificar esto accedemos al **localhost:8000**



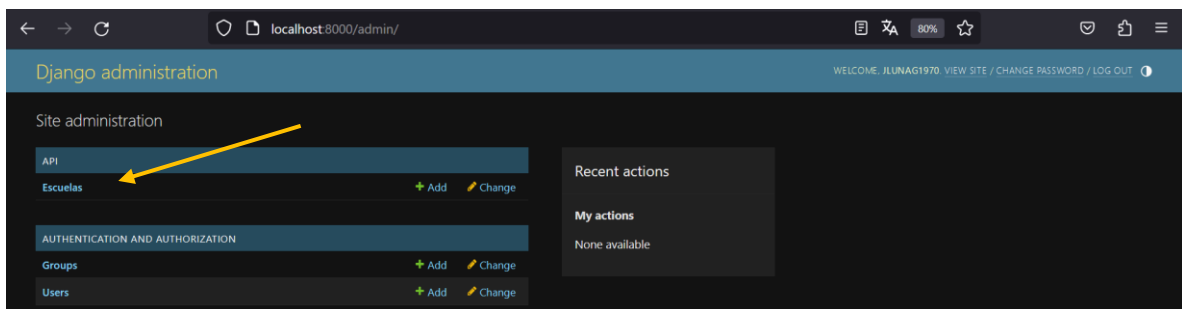
22. Accedemos al administrador de django con **localhost:8000/admin**



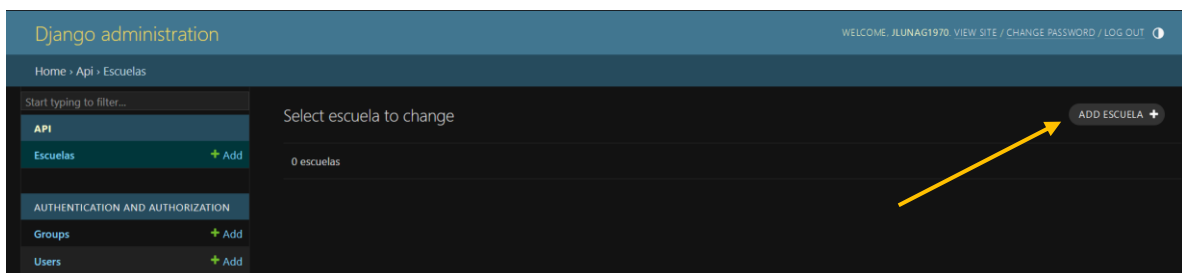
23. Ingresamos con el nombre de usuario y contraseña creada



Se observa que ya accedimos al administrador de django donde se encuentra nuestro modelo Escuela.



Accedemos al mismo para agregar una escuela en ADD ESCUELA.



Add escuela

Nombre:

Telefono:

Nombre rector:

SAVE Save and add another Save and continue editing

✓ The escuela "Escuela object (1)" was added successfully.

Select escuela to change

Action: Go 0 of 1 selected

☐ ESCUELA

☐ Escuela object (1)

1 escuela

Y verificamos en la base de datos

				id	nombre	telefono	nombre_rector
<input type="checkbox"/>	Editar	Copiar	Borrar	1	I.E. 20 de Julio	6056769211	Jonny Luna Guerrero
<input type="checkbox"/>	Seleccionar todo	Para los elementos que están marcados:					
<input type="checkbox"/>	Mostrar todo	Número de filas:	25	Filtrar filas:	Buscar en esta tabla		

Registramos otra escuela para tener dos en la base de datos

				id	nombre	telefono	nombre_rector
<input type="checkbox"/>	Editar	Copiar	Borrar	1	I.E. 20 de Julio	6056769211	Jonny Luna Guerrero
<input type="checkbox"/>	Editar	Copiar	Borrar	2	I.E. INEM	6052314897	Arides Sandoval

24. En el archivo views.py creamos la clase EscuelaView y en la misma creamos los métodos getters get, post, put y delete que posteriormente le haremos su codificación.

```

Proyecto_Escuela > api > views.py > EscuelaView > delete
1
2 from django.views import View
3
4 class EscuelaView(View):
5     def get(self, request):
6         pass
7     def post(self, request):
8         pass
9     def put(self, request):
10        pass
11    def delete(self, request):
12        pass
13

```

25. Creamos un archivo de nombre **urls.py** en la carpeta **api**

```

urls.py ...\Proyecto_Escuela  urls.py ...\api X
Proyecto_Escuela > api > urls.py > ...
1 from django.urls import path
2 from .views import EscuelaView
3
4 urlpatterns = [
5     path('escuelas/', EscuelaView.as_view(), name='escuelas_list')
6 ]

```

En el archivo **urls.py** del proyecto registramos el archivo url anterior

```

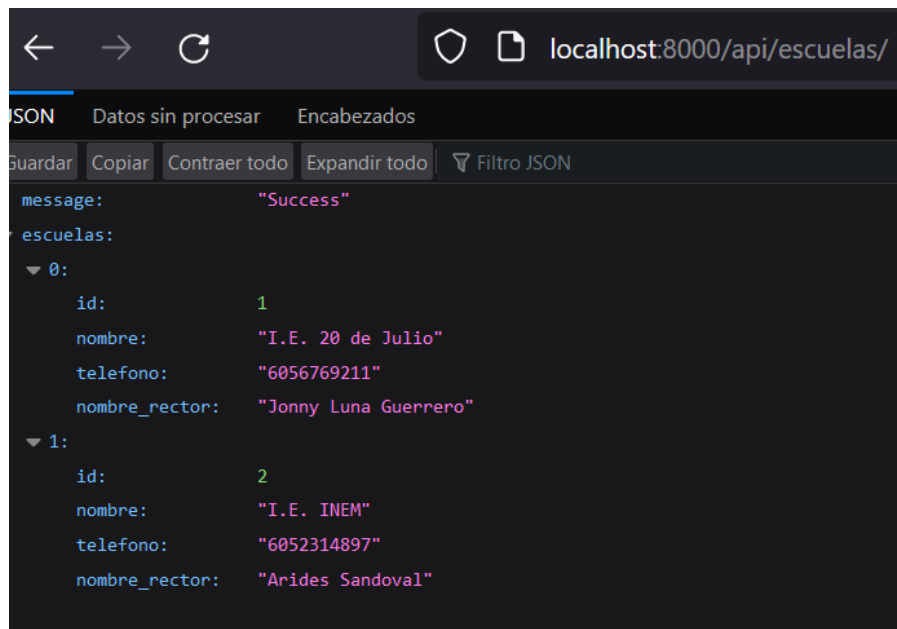
Proyecto_Escuela > Proyecto_Escuela > urls.py > ...
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('api/', include('api.urls'))
23 ]
24

```

26. En el archivo **views.py** Escribimos el método **get** para obtener los datos en el formato **Json**

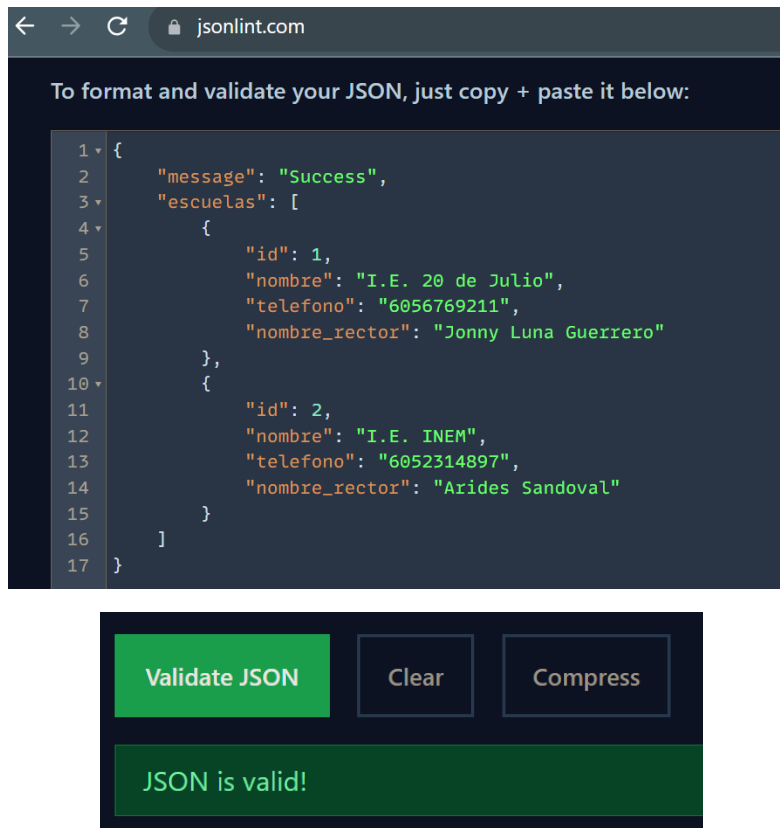
```
class EscuelaView(View):
    def get(self, request):
        escuelas = list(Escuela.objects.values())
        if len(escuelas)>0:
            datos = {'message': "Success", 'escuelas':escuelas}
        else:
            datos = {'message': "Escuelas no encontradas..."}
        return JsonResponse(datos)
```

Verificamos en el navegador con la ruta <http://localhost:8000/api/escuelas/>



Observamos que en efecto los datos han pasado al formato Json

27. En la pagina web <https://jsonlint.com/> validamos el formato JSON

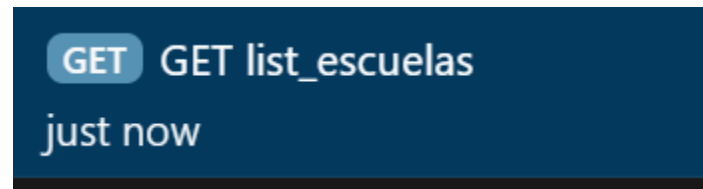


Y en efecto es un formato JSON

28. Instalamos en VS Code Thunder Client que es una aplicación para trabajar los datos como cliente RES API.



29. Una vez instalado Accedemos a Collections y creamos una nueva colección con el nombre **django_mysql_api** y creamos un New Request al que le ponemos como nombre **GET list_escuelas**

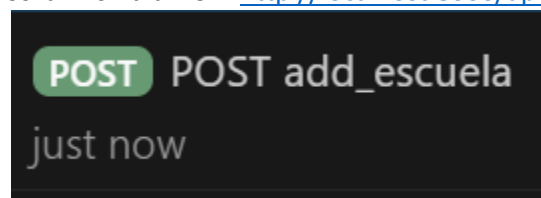


30. Copiamos la Url <http://localhost:8000/api/escuelas/> en el método GET y le damos SEND y se observa la respuesta con el código 200.

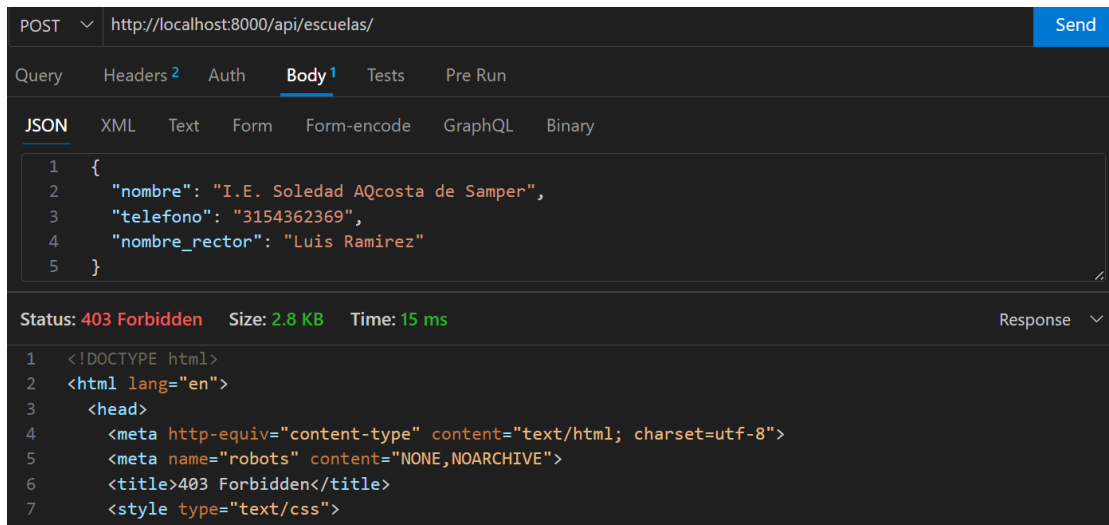
```
Status: 200 OK   Size: 239 Bytes   Time: 16 ms

1  {
2    "message": "Success",
3    "escuelas": [
4      {
5        "id": 1,
6        "nombre": "I.E. 20 de Julio",
7        "telefono": "6056769211",
8        "nombre_rector": "Jonny Luna Guerrero"
9      },
10     {
11       "id": 2,
12       "nombre": "I.E. INEM",
13       "telefono": "6052314897",
14       "nombre_rector": "Arides Sandoval"
15     }
16   ]
17 }
```

31. Creamos un nuevo Request para el método POST el cual llamaremos **add_escuela** y le insertamos la misma url <http://localhost:8000/api/escuelas/>



32. Enviamos a través del Body un nuevo registro y observamos un código 403 el cual es prohibido.



Este error lo corregimos en **views.py** creando en la clase **EscuelaView** los métodos `@method_decorator(csrf_exempt)` y

```
def dispatch(self, request, *args, **kwargs):
    return super().dispatch(request, *args, **kwargs)
```

```
class EscuelaView(View):
    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super().dispatch(request, *args, **kwargs)
```

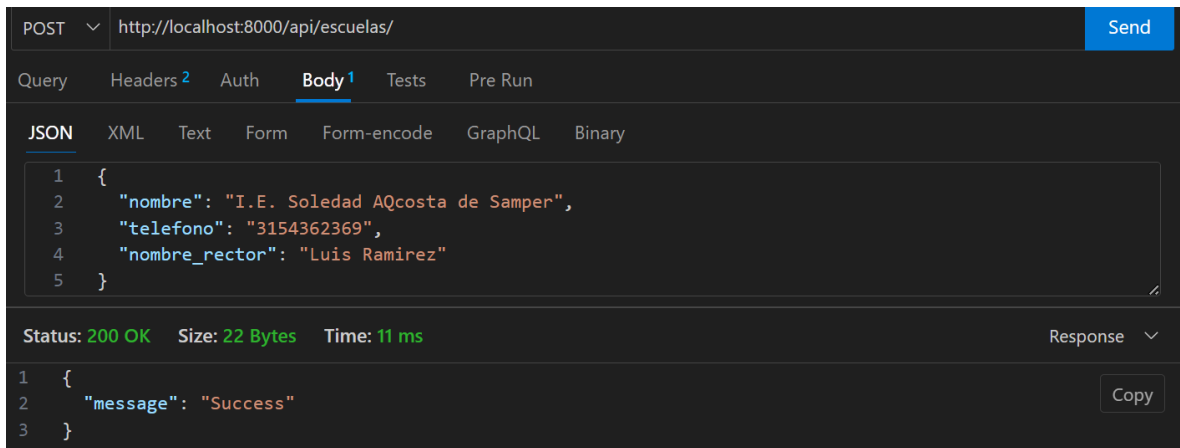
además de importar sus librerías

```
from django.utils.decorators import method_decorator
```

```
from django.views.decorators.csrf import csrf_exempt
```

```
from django.utils.decorators import method_decorator
from django.views import View
from django.views.decorators.csrf import csrf_exempt
```

Ahora enviamos nuevamente los datos y se observa que el error ha desaparecido.

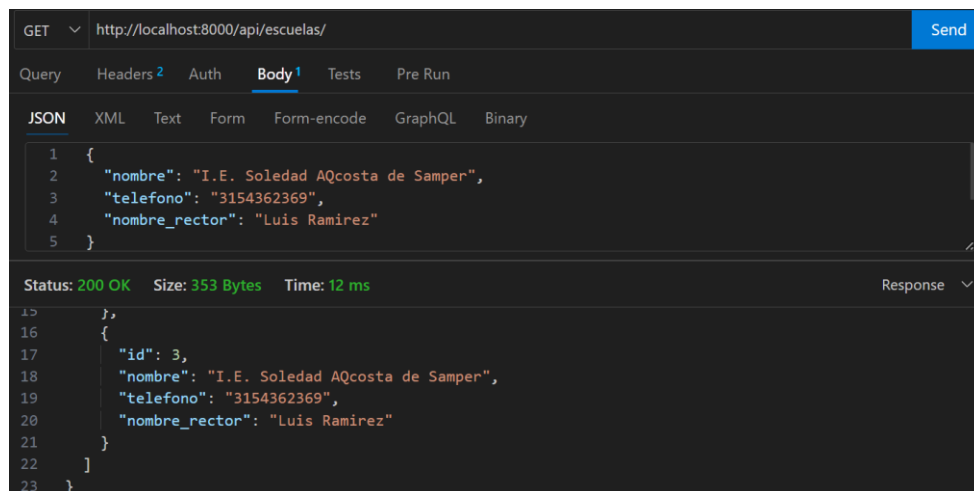


33. Modificamos el método POST en la clase EscuelaView para crear el método correctamente, además importamos la librería de JSON

```
6 import json
```

```
def post(self, request):
    jd = json.loads(request.body)
    Escuela.objects.create(nombre=jd['nombre'],telefono=jd['telefono'],nombre_rector=jd
    ['nombre_rector'])
    datos = {'message': "Success"}
    return JsonResponse(datos)
```

Nuevamente enviamos por el método POST y verificamos son el método GET y se observa que en efecto ya se registro la tercera institución.



Ahora verificamos en la base de datos

			id	nombre	telefono	nombre_rector
<input type="checkbox"/>				1	I.E. 20 de Julio	6056769211 Jonny Luna Guerrero
<input type="checkbox"/>				2	I.E. INEM	6052314897 Arides Sandoval
<input type="checkbox"/>				3	I.E. Soledad AQcosta de Samper	3154362369 Luis Ramirez

34. Se crea el método para buscar una escuela por id.

En el archivo **urls.py** de la **api**, se crea una nueva url, para que reciba un parámetro que es el id, esto se hace con la línea de código:

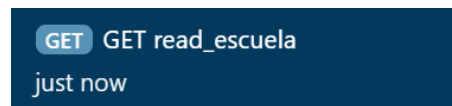
`path('escuelas/<int:id>', EscuelaView.as_view(), name='escuelas_procesos')`

```
urlpatterns = [
    path('escuelas/', EscuelaView.as_view(), name='escuelas_list'),
    path('escuelas/<int:id>', EscuelaView.as_view(), name='escuelas_procesos')
]
```

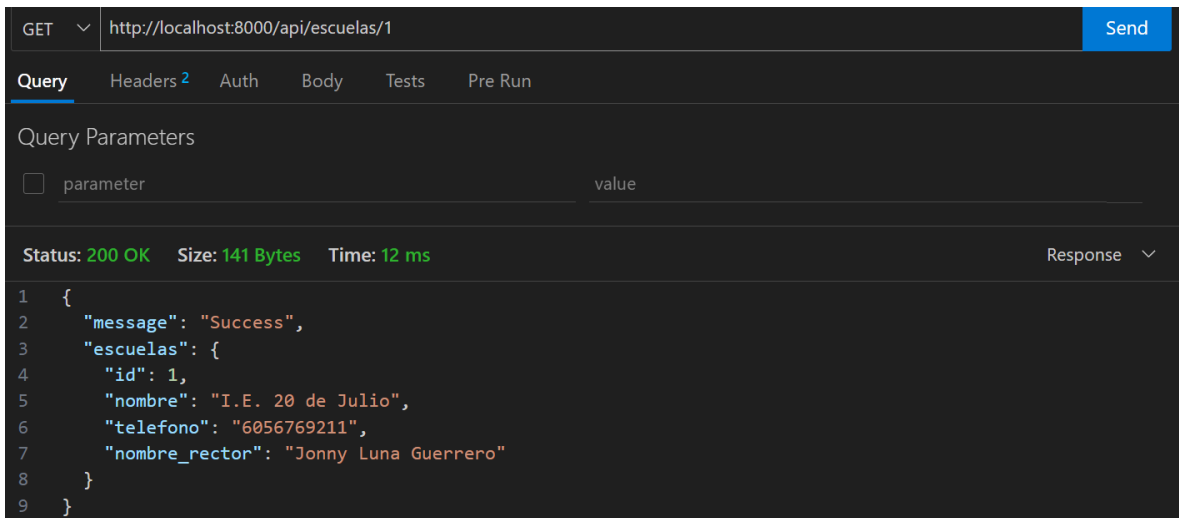
En la clase EscuelaView modificamos el procedimiento get

```
def get(self, request, id=0):
    if (id>0):
        escuelas = list(Escuela.objects.filter(id=id).values())
        if len(escuelas)>0:
            escuela = escuelas[0]
            datos = {'message': "Success", 'escuelas':escuela}
        else:
            datos = {'message': "Escuela no encontrada..."}
        return JsonResponse(datos)
    else:
        escuelas = list(Escuela.objects.values())
        if len(escuelas)>0:
            datos = {'message': "Success", 'escuelas':escuelas}
        else:
            datos = {'message': "Escuelas no encontradas..."}
        return JsonResponse(datos)
```

Se crea un New Request tipo GET con el nombre GET read_escuela.



Con la dirección web <http://localhost:8000/api/escuelas/1> en este caso el entero se colocó el número 1 como ejemplo.



```
GET http://localhost:8000/api/escuelas/1 Send
```

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

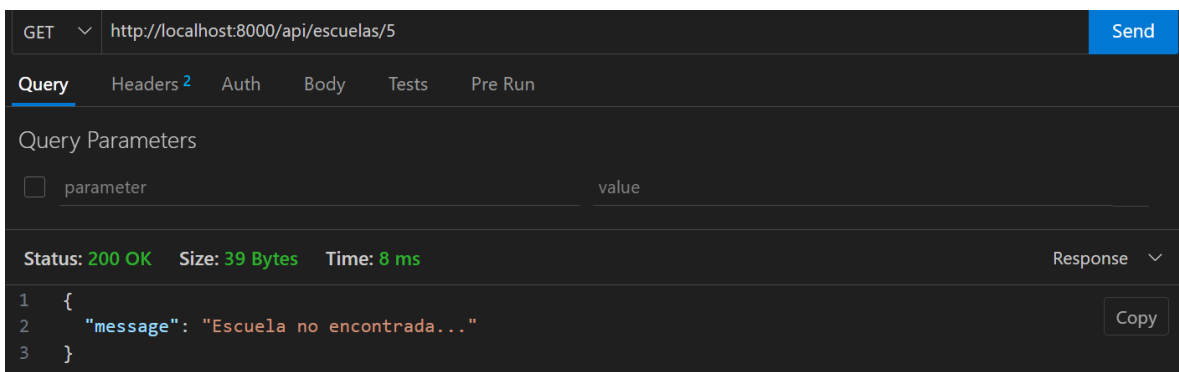
parameter	value
-----------	-------

Status: 200 OK Size: 141 Bytes Time: 12 ms Response

```
1 {
2   "message": "Success",
3   "escuelas": {
4     "id": 1,
5     "nombre": "I.E. 20 de Julio",
6     "telefono": "6056769211",
7     "nombre_rector": "Jonny Luna Guerrero"
8   }
9 }
```

Se observa que la respuesta es que trae el registro 1 correspondiente a la I.E. 20 de Julio.

Si lo probamos con el número 5 se obtiene el siguiente resultado



```
GET http://localhost:8000/api/escuelas/5 Send
```

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

parameter	value
-----------	-------

Status: 200 OK Size: 39 Bytes Time: 8 ms Response

```
1 {
2   "message": "Escuela no encontrada..."
3 }
```

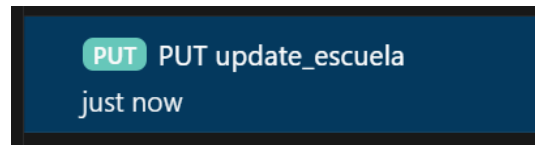
Copy

Se observa que esa escuela no existe. Lo cual es correcto.

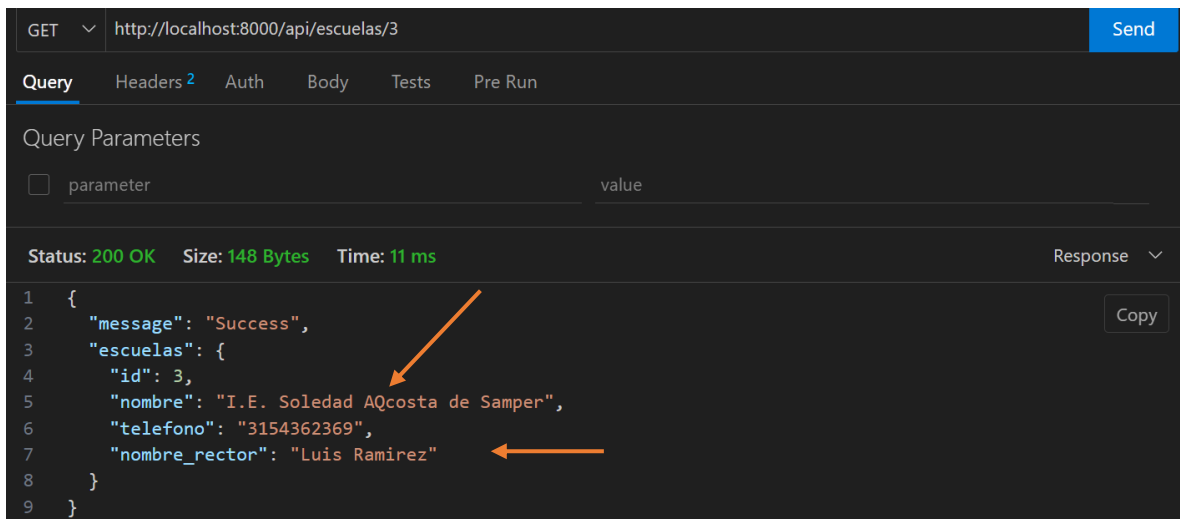
35. Creamos el método modificar o PUT

```
def put(self, request, id):
    jd = json.loads(request.body)
    escuelas = list(Escuela.objects.filter(id=id).values())
    if len(escuelas)>0:
        escuela = Escuela.objects.get(id=id)
        escuela.nombre = jd['nombre']
        escuela.telefono = jd['telefono']
        escuela.nombre_rector = jd['nombre_rector']
        escuela.save()
        datos = {'message': "Success"}
    else:
        datos = {'message': "Escuela no encontrada..."}
    return JsonResponse(datos)
```

Creamos un New Request tipo PUT con el nombre PUT update_escuela



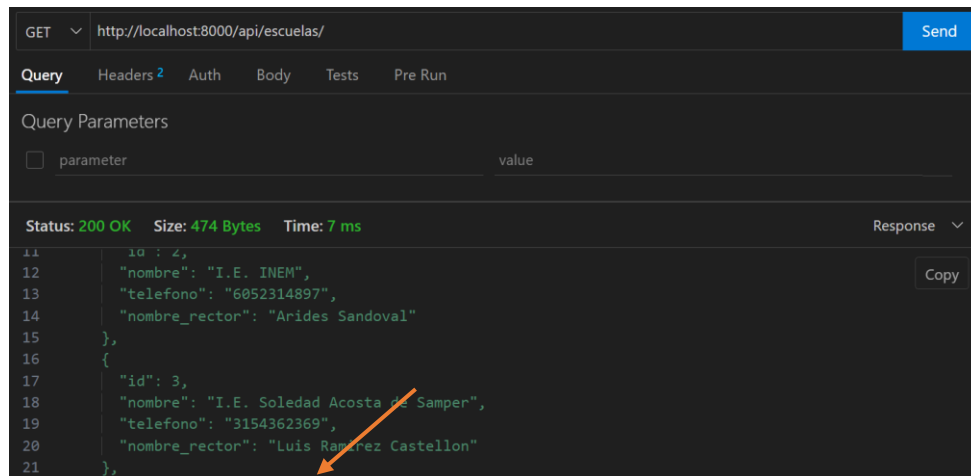
Se coloca la url y se prueba modificando el dato del registro 3, se observa que en el nombre existe un error, adem{as vamos a modificar el nombre del rector.



Se observa el registro modificado exitosamente



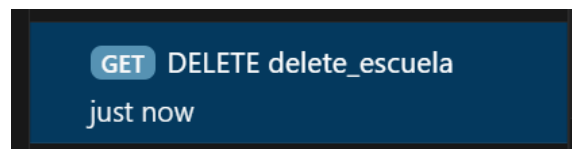
Verificamos con GET



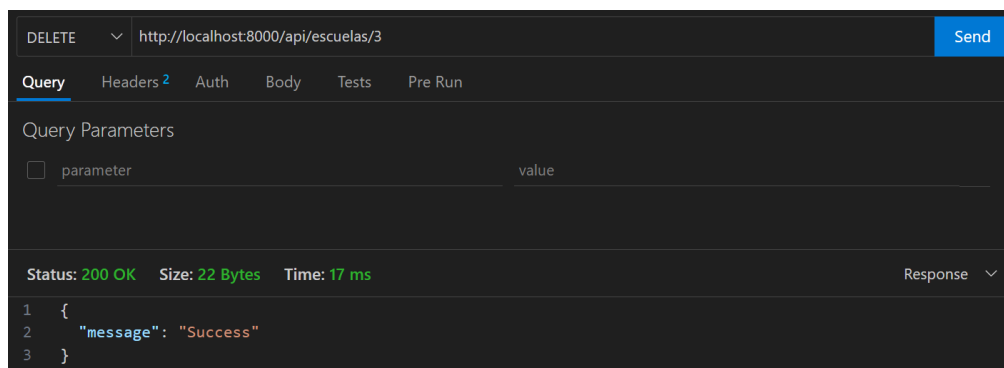
36. Creamos el método modificar o DELETE en

```
def delete(self, request, id):
    escuelas = list(Escuela.objects.filter(id=id).values())
    if len(esuelas)>0:
        Escuela.objects.filter(id=id).delete()
        datos = {'message': "Success"}
    else:
        datos = {'message': "Escuela no encontrada..."}
    return JsonResponse(datos)
```

Creamos un New Request tipo DELETE con el nombre DELETE delete_escuela

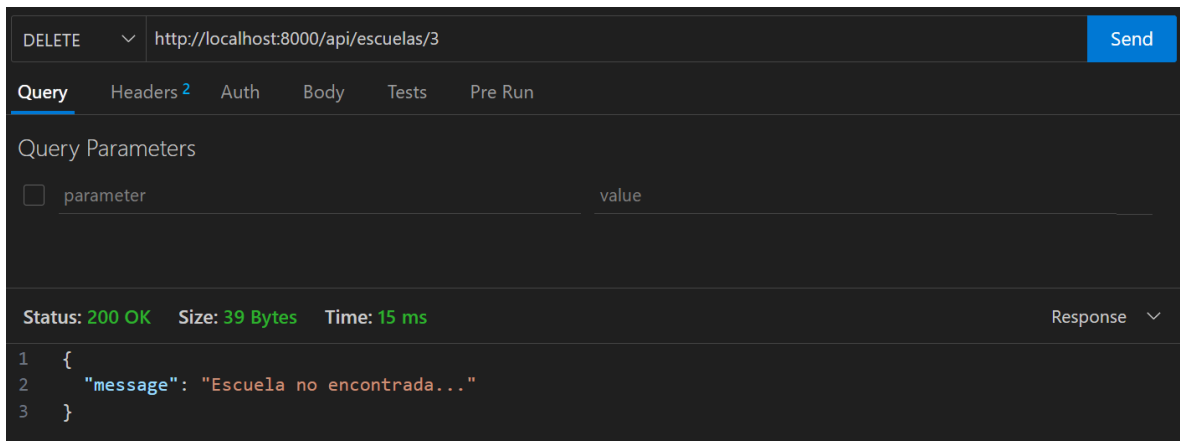


Se coloca la url y se prueba eliminando el registro 3.



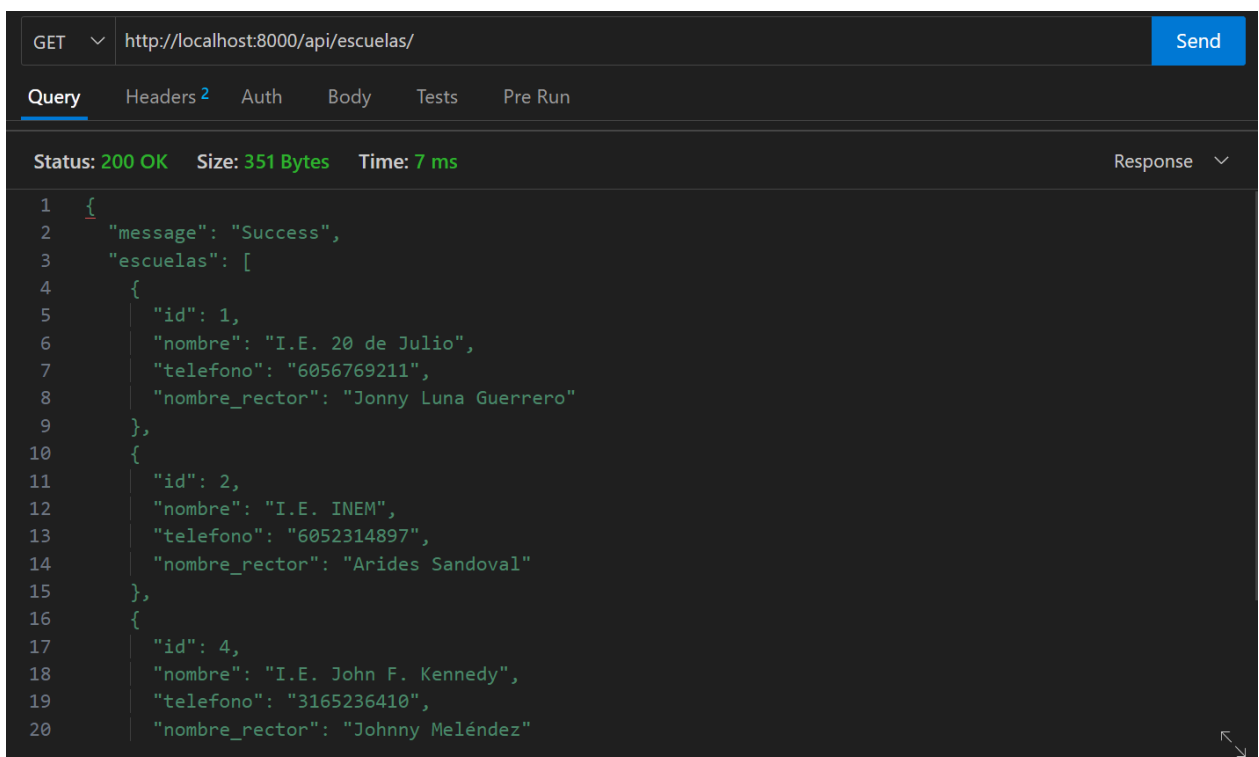
Se observa que fue eliminado correctamente.

Si intentamos eliminarlo de nuevo se observa que el mensaje ahora es “Escuela no encontrada...” debido a que ya se eliminó



Verificamos con el método GET para ver los registros existentes.

Se observa que ya no existe el id = 3



En la base de datos también podemos corroborar esto

← T →				▼ id	nombre	telefono	nombre_rector
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	I.E. 20 de Julio	6056769211	Jonny Luna Guerrero
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2	I.E. INEM	6052314897	Arides Sandoval
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	4	I.E. John F. Kennedy	3165236410	Johnny Meléndez

No existe el registro con id = 3