



Nombre: Dafne Gaviria Arcila

Actividad: Backend – Python 2

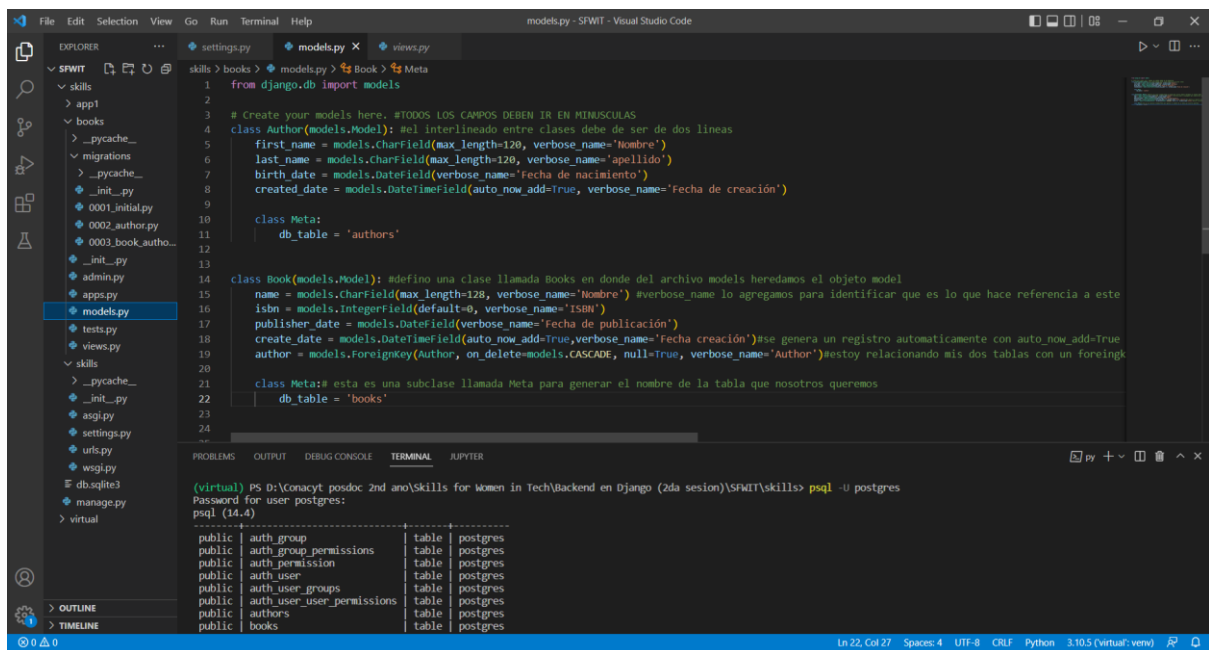
Durante las mentorías hemos visto cómo usar Django Rest Framework para poder crear nuestra API.

En esta tarea deberas crear un proyecto y desarrollar 4 endpoints: get, post, update, delete

Una vez creado el proyecto deberás de subirlo a un repositorio en tu cuenta de github y agregar el repo aquí como prueba. Toma por lo menos 4 capturas de pantalla de tu código y agrégalas aquí mismo.

1 GET-

1.1 Creación de los modelos:

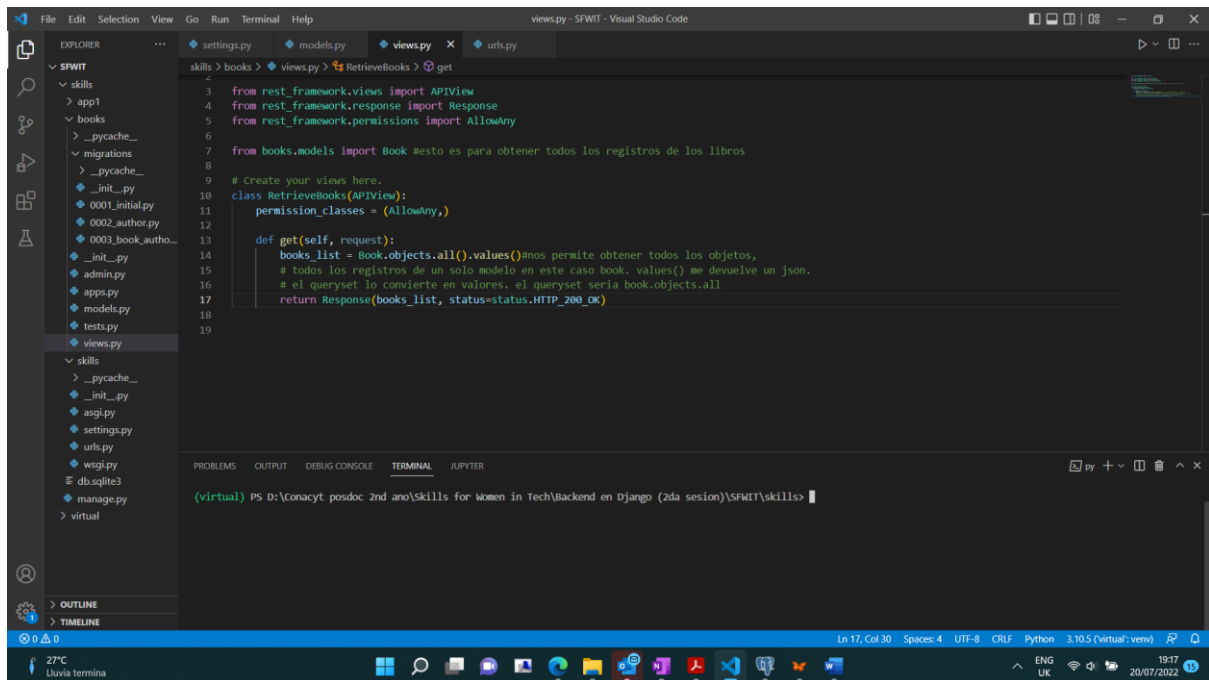


```
1 from django.db import models
2
3 # Create your models here. # TODOS LOS CAMPOS DEBEN IR EN MINUSCULAS
4 class Author(models.Model): # el interlineado entre clases debe de ser de dos lineas
5     first_name = models.CharField(max_length=120, verbose_name='Nombre')
6     last_name = models.CharField(max_length=120, verbose_name='Apellido')
7     birth_date = models.DateField(verbose_name='Fecha de nacimiento')
8     created_date = models.DateTimeField(auto_now_add=True, verbose_name='Fecha de creación')
9
10     class Meta:
11         db_table = 'authors'
12
13
14 class Book(models.Model): # defino una clase llamada Books en donde del archivo models heredamos el objeto model
15     name = models.CharField(max_length=128, verbose_name='Nombre') # verbose_name lo agregamos para identificar que es lo que hace referencia a este
16     isbn = models.IntegerField(default=0, verbose_name='ISBN')
17     publisher_date = models.DateField(verbose_name='Fecha de publicación')
18     create_date = models.DateTimeField(auto_now_add=True, verbose_name='Fecha creación') # se genera un registro automaticamente con auto_now_add=True
19     author = models.ForeignKey(Author, on_delete=models.CASCADE, null=True, verbose_name='Author') # estoy relacionando mis dos tablas con un foreign
20
21     class Meta: # esta es una subclase llamada Meta para generar el nombre de la tabla que nosotros queremos
22         db_table = 'books'
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
(virtual) PS D:\Conacyt posdoc 2nd ano\Skills for Women in Tech\Backend in Django (2da sesion)\SFWT\skills> psql -U postgres
Password for user postgres:
psql (14.4)
-----
public | auth_group | table | postgres
public | auth_group_permissions | table | postgres
public | auth_permission | table | postgres
public | auth_user | table | postgres
public | auth_user_groups | table | postgres
public | auth_user_user_permissions | table | postgres
public | authors | table | postgres
public | books | table | postgres
```

1.2 Creación de las vistas:

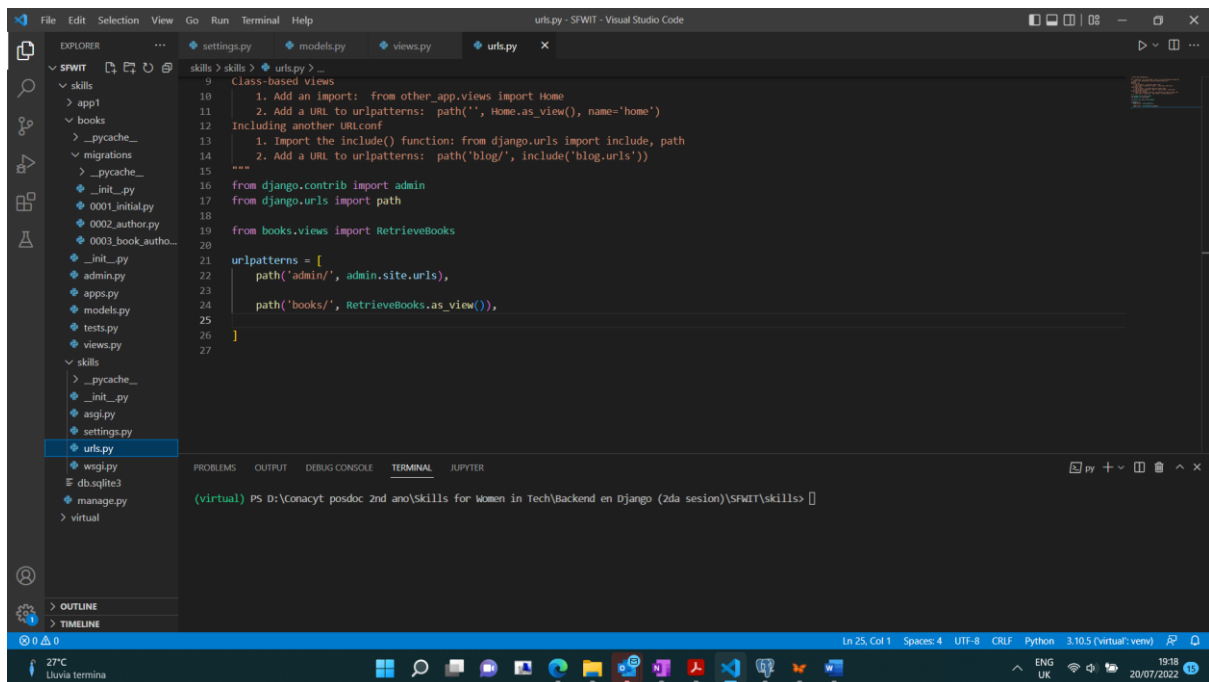


The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a project structure for 'SFWT'. The main editor window shows the 'views.py' file with the following code:

```
1 from rest_framework.views import APIView
2 from rest_framework.response import Response
3 from rest_framework.permissions import AllowAny
4
5 from books.models import Book #esto es para obtener todos los registros de los libros
6
7 # Create your views here.
8
9 class RetrieveBooks(APIView):
10     permission_classes = (AllowAny,)
11
12     def get(self, request):
13         books_list = Book.objects.all().values() #nos permite obtener todos los objetos,
14         # todos los registros de un solo modelo en este caso book. values() me devuelve un json.
15         # el queryset lo convierte en valores. el queryset seria book.objects.all
16         return Response(books_list, status=status.HTTP_200_OK)
```

The bottom panel shows the terminal with the command: `(virtual) PS D:\Conacyt posdoc 2nd ano\skills for Women in Tech\Backend en Django (2da sesion)\SFWT\skills>`

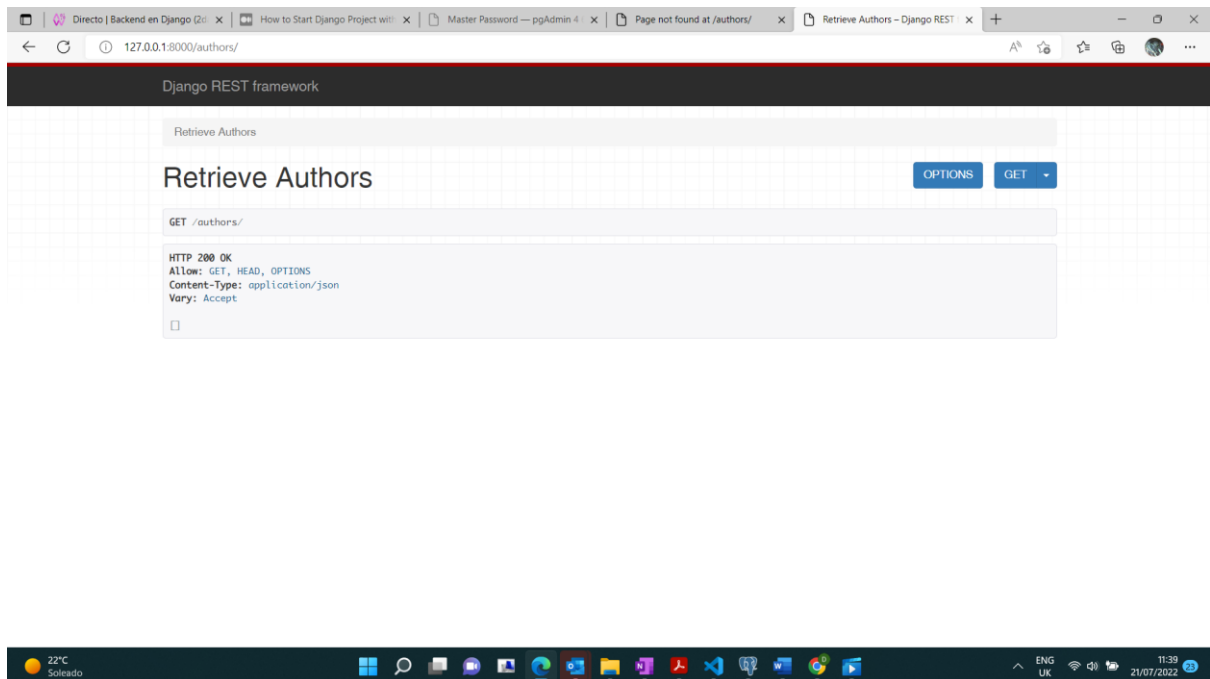
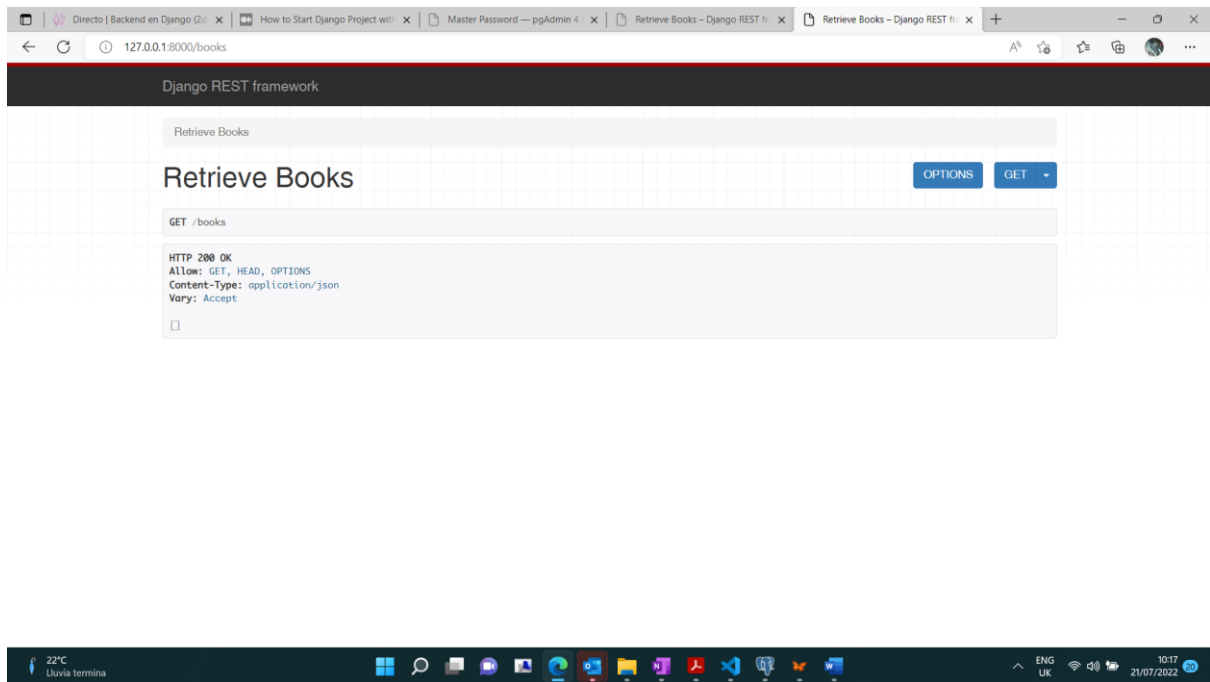
1.3 Modificando urls



The screenshot shows the Visual Studio Code interface with the file explorer on the left. The main editor window shows the 'urls.py' file with the following code:

```
1 from django.urls import path
2
3 # Import the include() function: from django.urls import include, path
4
5 # Add a URL to urlpatterns: path('blog/', include('blog.urls'))
6
7 urlpatterns = [
8     path('admin/', admin.site.urls),
9     path('books/', RetrieveBooks.as_view()),
10 ]
```

The bottom panel shows the terminal with the command: `(virtual) PS D:\Conacyt posdoc 2nd ano\skills for Women in Tech\Backend en Django (2da sesion)\SFWT\skills>`



2 POST

The screenshot shows the Visual Studio Code editor with the file `views.py` open. The Explorer sidebar on the left shows the project structure with folders `skills` and `books`. The `views.py` file is selected in the Explorer and the Editor. The code defines two Django APIView classes: `CreateAuthor` and `CreateBook`. Both classes have a `post` method that creates an author or book object and returns a `Response` with a 'Creado' message. The `permission_classes` attribute is set to `(allowAny,)` for both classes. The `PROBLEMS` panel at the bottom shows two errors: a `NameError` in `views.py` at line 19 and a `NameError` in `books/views.py` at line 38, both stating that the name `'APIView'` is not defined.

```
25
26 class CreateAuthor(APIView):
27     permission_classes = (allowAny,)
28
29     def post(self, request):
30         author_obj = Author.objects.create(
31             first_name = request.data.get('first_name',''),
32             last_name = request.data.get('last_name',''),
33             birth_date = request.data.get('birth_date',''),
34         )
35         return Response({'message':'Creado'}, status=status.HTTP_201_CREATED)
36
37
38 class CreateBook(APIView):
39     permission_classes = (AllowAny,)
40
41     def post(self, request):
42         book_obj = Book.objects.create(
43             name = request.data.get('name',''),
44             isbn = request.data.get('isbn',''),
45             publisher_date = request.data.get('publisher_date','1700-01-01'),
46             author_id = request.data.get('author_id','1'),
47         )
48         return Response({'mensaje':'Creado'}, status=status.HTTP_201_CREATED)
49
50
51
```

PROBLEMS

- File "D:\Conacyt posdoc 2nd ano\Skills for Women in Tech\Backend en Django (2da sesion)\SFMIT\skills\skills\views.py", line 19, in module: NameError: name 'APIView' is not defined. Did you mean: 'APIView'?
- File "D:\Conacyt posdoc 2nd ano\Skills for Women in Tech\Backend en Django (2da sesion)\SFMIT\skills\books\views.py", line 38, in module: NameError: name 'APIView' is not defined. Did you mean: 'APIView'?

The screenshot shows the Visual Studio Code editor with the file `urls.py` open. The Explorer sidebar on the left shows the project structure with folders `skills` and `books`. The `urls.py` file is selected in the Explorer and the Editor. The code includes the `include` function from `django.urls` and defines a list of `urlpatterns` for the application. The `PROBLEMS` panel at the bottom shows two errors: a `NameError` in `views.py` at line 19 and a `NameError` in `books/views.py` at line 38, both stating that the name `'APIView'` is not defined.

```
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 from django.contrib import admin
17 from django.urls import path
18
19 from books.views import (
20     RetrieveBooks,
21     RetrieveAuthors,
22     CreateAuthor,
23     CreateBook)
24
25 urlpatterns = [
26     path('admin/', admin.site.urls),
27     path('books/', RetrieveBooks.as_view()),
28     path('books/create/', CreateBook.as_view()),
29     path('authors/', RetrieveAuthors.as_view()),
30     path('authors/create/', CreateAuthor.as_view()),
31 ]
32
33
```

PROBLEMS

- File "D:\Conacyt posdoc 2nd ano\Skills for Women in Tech\Backend en Django (2da sesion)\SFMIT\skills\skills\views.py", line 19, in module: NameError: name 'APIView' is not defined. Did you mean: 'APIView'?
- File "D:\Conacyt posdoc 2nd ano\Skills for Women in Tech\Backend en Django (2da sesion)\SFMIT\skills\books\views.py", line 38, in module: NameError: name 'APIView' is not defined. Did you mean: 'APIView'?

Create Author - Django REST framework

Retrieve Authors Create Author

Create Author

OPTIONS

POST /authors/create/

HTTP 201 Created
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "message": "Creado"
}
```

Media type: application/json

Content:

POST

Retrieve Authors - Django REST framework

Retrieve Authors

Retrieve Authors

OPTIONS GET

GET /authors/

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

```
[
  {
    "id": 1,
    "first_name": "anne",
    "last_name": "rice",
    "birth_date": "1928-03-12",
    "created_date": "2022-07-21T17:13:49.626212"
  }
]
```

Create Book - Django REST fram x +

127.0.0.1:8000/books/create/

Django REST framework

Retrieve Books / Create Book

Create Book

OPTIONS

POST /books/create/

HTTP 201 Created
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "mensaje": "Creado"
}
```

Media type: application/json

Content:

25°C Soleado

Retrieve Books - Django REST fr x +

127.0.0.1:8000/books

Django REST framework

Retrieve Books

Retrieve Books

OPTIONS GET

GET /books

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

```
[
  {
    "id": 1,
    "name": "entrevista con el vampiro",
    "isbn": 1234,
    "publisher_date": "1969-02-02",
    "create_date": "2022-07-21T17:43:18.715043Z",
    "author_id": 1
  }
]
```

25°C Soleado

2.1 Agregando serializadores

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure for 'SFWT'. The main editor window shows the 'views.py' file with the following Python code:

```
13 # Create your views here.
14 class RetrieveBooks(APIView):
15     permission_classes = (AllowAny,)
16
17     def get(self, request):
18         books_list = Book.objects.all() #nos permite obtener todos los objetos,
19         # todos los registros de un solo modelo en este caso book. values() me devuelve un json.
20         # el queryset lo convierte en valores. el queryset seria book.objects.all
21         serializer = BookSerializer(books_list, many=True)
22         return Response(serializer.data, status=status.HTTP_200_OK)
23
24 class RetrieveAuthors(APIView):
25     permission_classes = (AllowAny,)
26
27     def get(self, request):
28         author_list = Author.objects.all()
29         serializer = AuthorSerializer(author_list, many=True)
30         #se agrega many para que me devuelva la lista de todos los objetos para serializar, es decir mas de uno que es lo que da el serializer
31         #es decir pude haber mas de un solo objeto en la lista
32         return Response(serializer.data) #asi se saa toda la info y se transforma en un objeto json
33
34 class createAuthor(APIView):
35     permission_classes = (AllowAny,)
36
37     def post(self, request):
38         data = request.data
39         serializer = AuthorSerializer(data=data)
40         serializer.is_valid(raise_exception=True)
41         serializer.save()
42         return Response(serializer.data, status=status.HTTP_201_CREATED)
43
```

The terminal at the bottom shows a successful GET request to the /books/3/ endpoint, returning a 404 status code.

```
[21/Jul/2022 18:32:48] "GET /books/3/ HTTP/1.1" 404 5361
```

The screenshot shows the Visual Studio Code editor with the file explorer on the left. The main editor window shows the 'views.py' file with the following Python code:

```
44 class CreateBook(APIView):
45     permission_classes = (AllowAny,)
46
47     def post(self, request):
48         serializer = BookSerializer(data=request.data)
49         serializer.is_valid(raise_exception=True)
50         serializer.save()
51         return Response(serializer.data, status=status.HTTP_201_CREATED)
52
53 class RetrieveAuthorsAPIView(APIView):
54     permission_classes = (AllowAny,)
55
56     def get(self, request, author_id):
57         author_obj = Author.objects.get(id=author_id)
58         serializer = AuthorSerializer(author_obj)
59         return Response(serializer.data)
60
61 class RetrieveBookAPIView(APIView):
62     permission_classes = (AllowAny,)
63
64     def get(self, request, book_id):
65         book_obj = get_object_or_404(Book, pk=book_id)
66         serializer = BookSerializer(book_obj)
67         return Response(serializer.data)
68
69
70
71
72
```

The terminal at the bottom shows a successful GET request to the /books/3/ endpoint, returning a 404 status code.

```
[21/Jul/2022 18:32:48] "GET /books/3/ HTTP/1.1" 404 5361
```

Create Author - Django REST framework

127.0.0.1:8000/authors/create/

Django REST framework

Retrieve Authors / Create Author

Create Author

OPTIONS

POST /authors/create/

HTTP 201 Created
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{
  "id": 2,
  "first_name": "julio",
  "last_name": "verne",
  "birth_date": "1800-01-01"
}
```

Media type: application/json

Content:

Retrieve Authors - Django REST framework

127.0.0.1:8000/authors/

Django REST framework

Retrieve Authors

Retrieve Authors

OPTIONS GET

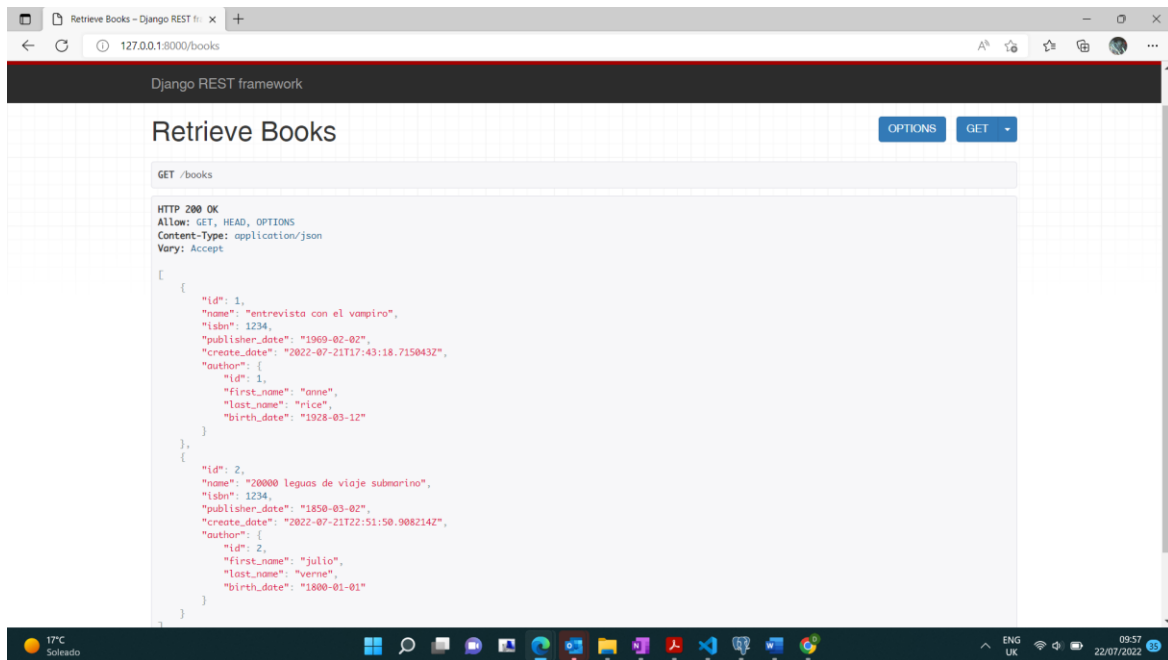
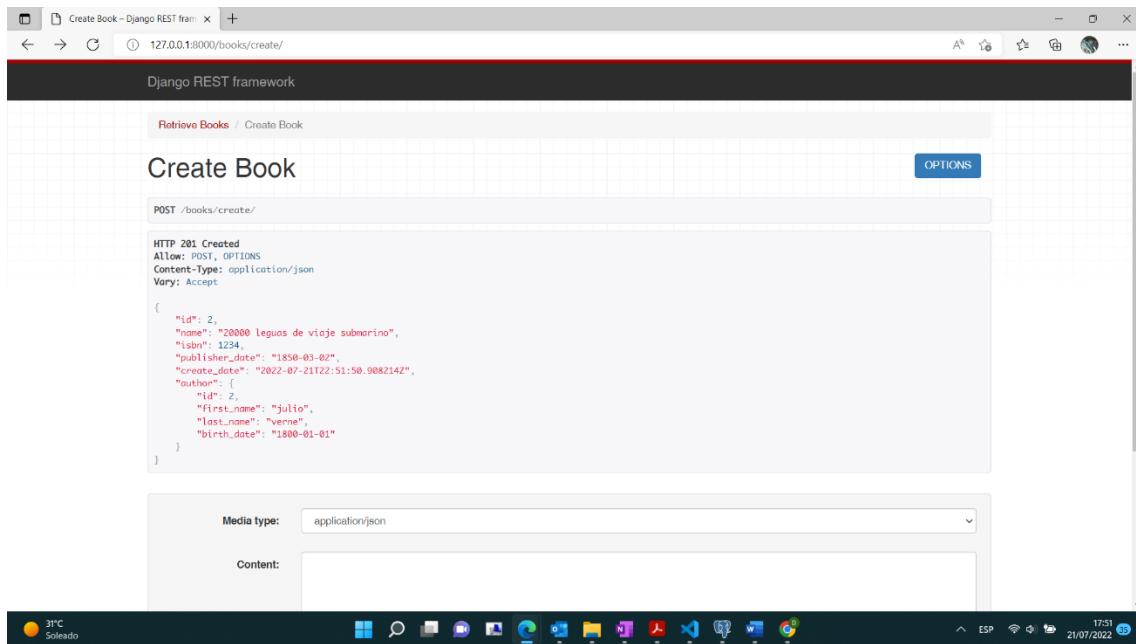
GET /authors/

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

```
[
  {
    "id": 1,
    "first_name": "anne",
    "last_name": "rice",
    "birth_date": "1928-03-12"
  },
  {
    "id": 2,
    "first_name": "julio",
    "last_name": "verne",
    "birth_date": "1800-01-01"
  }
]
```

31°C Soleado

16:51 21/07/2022



Retrieve Authors Api - Django REST framework

Retrieve Authors / Retrieve Authors Api

Retrieve Authors Api

OPTIONS GET

GET /authors/1/

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

{
  "id": 1,
  "first_name": "anne",
  "last_name": "rice",
  "birth_date": "1928-03-12"
}
```

Retrieve Book Api - Django REST framework

Retrieve Books / Retrieve Book Api

Retrieve Book Api

OPTIONS GET

GET /books/1/

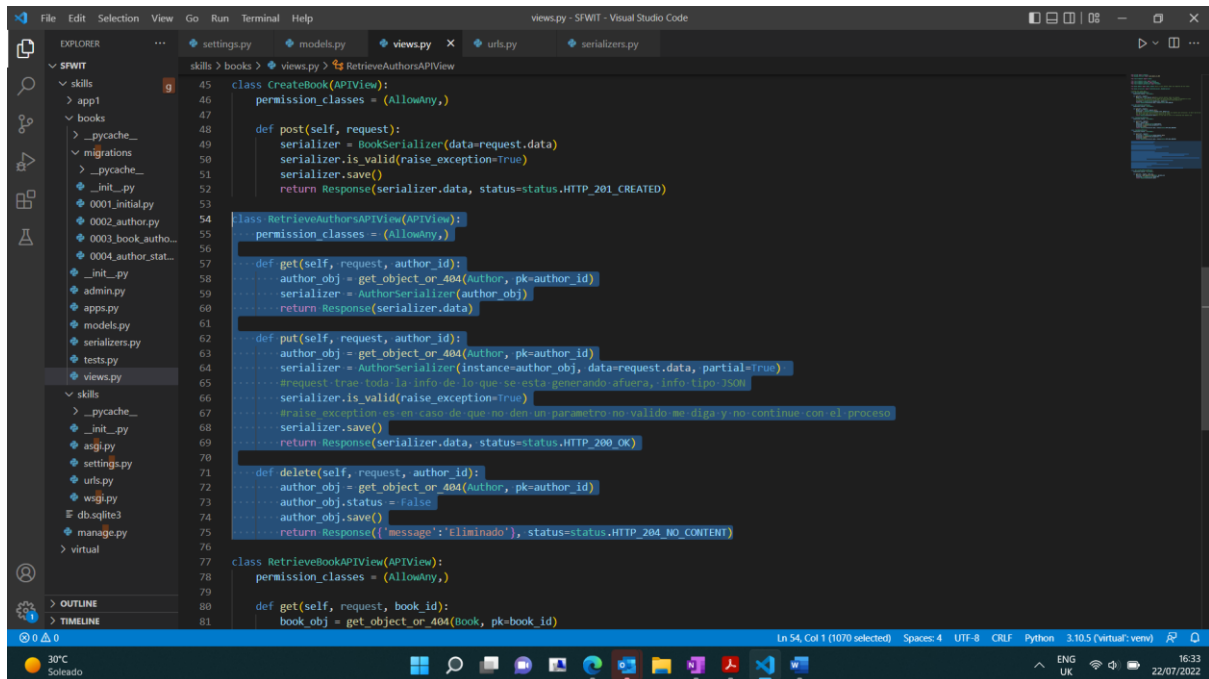
```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "id": 1,
  "name": "entrevista con el vampiro",
  "isbn": 1234,
  "publisher_date": "1969-02-02",
  "create_date": "2022-07-21T17:43:18.715043Z",
  "author": {
    "id": 1,
    "first_name": "anne",
    "last_name": "rice",
    "birth_date": "1928-03-12"
  }
}
```

30°C Parc soleado

ENG UK 18:26 21/07/2022

3 PUT –



```
class createBook(APIView):
    permission_classes = (AllowAny,)

    def post(self, request):
        serializer = BookSerializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)

class RetrieveAuthorsAPIView(APIView):
    permission_classes = (AllowAny,)

    def get(self, request, author_id):
        author_obj = get_object_or_404(Author, pk=author_id)
        serializer = AuthorSerializer(author_obj)
        return Response(serializer.data)

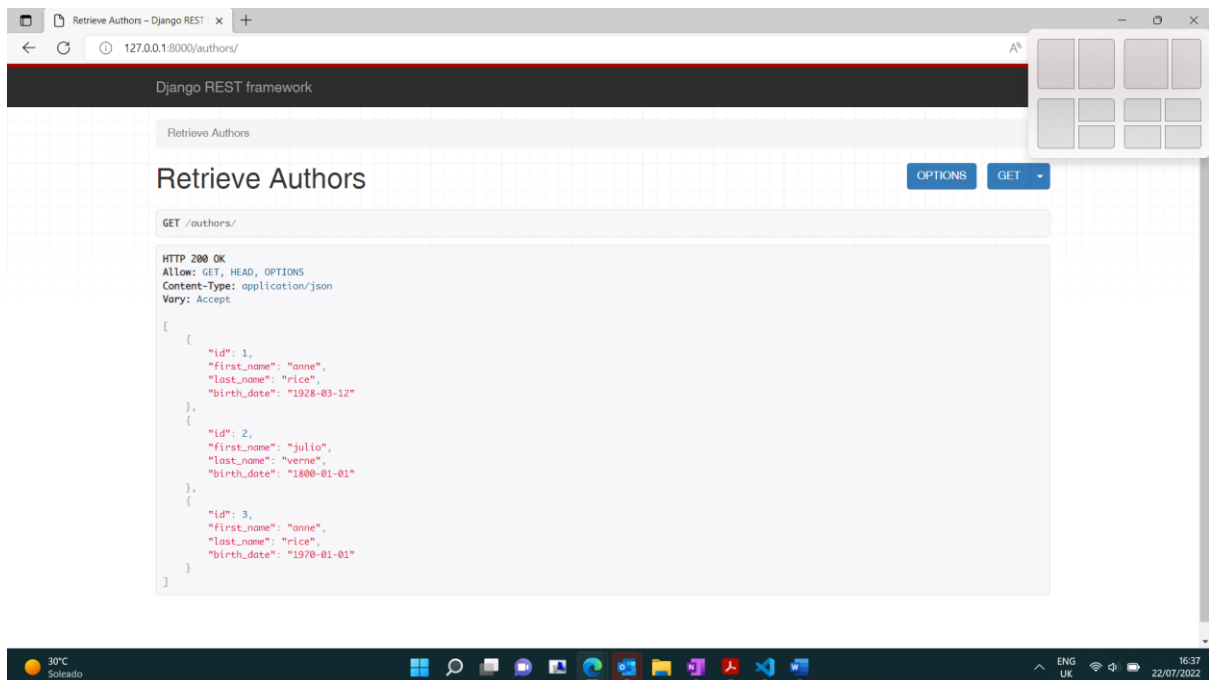
    def put(self, request, author_id):
        author_obj = get_object_or_404(Author, pk=author_id)
        serializer = AuthorSerializer(instance=author_obj, data=request.data, partial=True)
        #request trae toda la info de lo que se esta generando afuera, info tipo JSON
        serializer.is_valid(raise_exception=True)
        #raise exception es en caso de que no den un parametro no valido me diga y no continúe con el proceso
        serializer.save()
        return Response(serializer.data, status=status.HTTP_200_OK)

    def delete(self, request, author_id):
        author_obj = get_object_or_404(Author, pk=author_id)
        author_obj.status = False
        author_obj.save()
        return Response({'message': 'Eliminado'}, status=status.HTTP_204_NO_CONTENT)

class RetrieveBookAPIView(APIView):
    permission_classes = (AllowAny,)

    def get(self, request, book_id):
        book_obj = get_object_or_404(Book, pk=book_id)
```

Esta es mi lista de libros



Actualizo el Autor con ID 1

Retrieve Authors Api - Django REST framework

Retrieve Authors Api

DELETE OPTIONS GET

PUT /authors/1/

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

```
{
  "id": 1,
  "first_name": "anne",
  "last_name": "rice",
  "birth_date": "1985-02-03"
}
```

Media type: application/json

Content:

PUT

Retrieve Authors - Django REST framework

Retrieve Authors

OPTIONS GET

GET /authors/

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

```
[
  {
    "id": 2,
    "first_name": "julio",
    "last_name": "verne",
    "birth_date": "1800-01-01"
  },
  {
    "id": 3,
    "first_name": "anne",
    "last_name": "rice",
    "birth_date": "1970-01-01"
  },
  {
    "id": 1,
    "first_name": "anne",
    "last_name": "rice",
    "birth_date": "1985-02-03"
  }
]
```

4 DELETE-

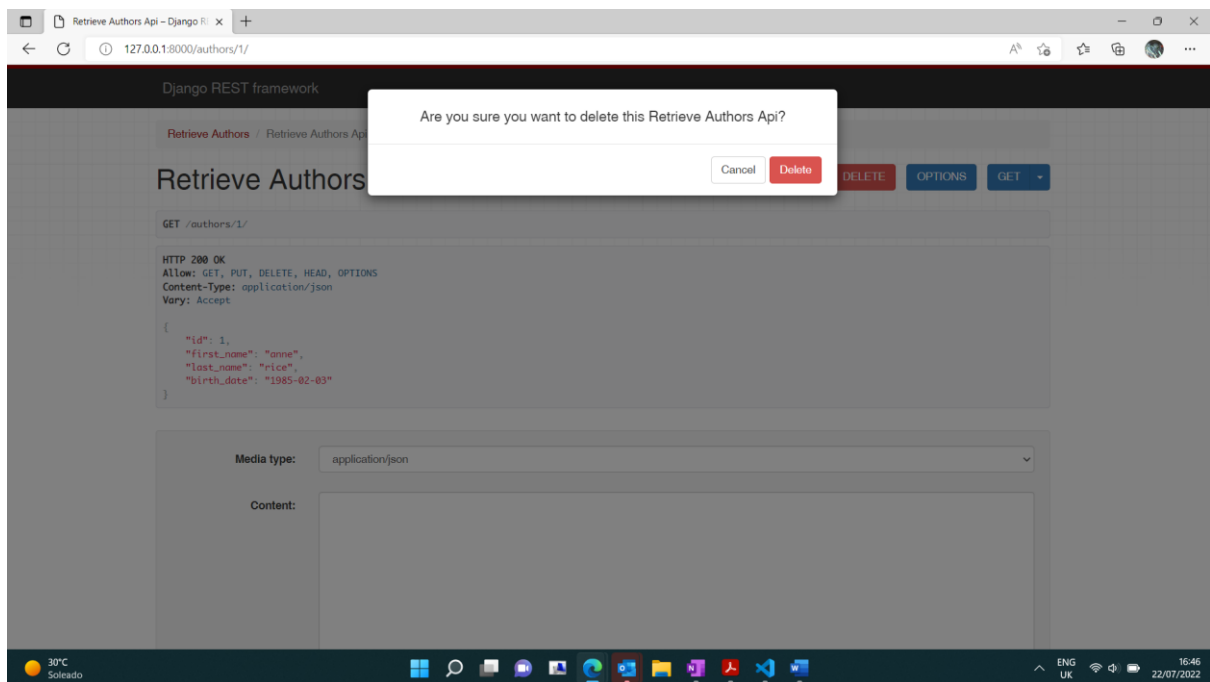
```
def get(self, request, author_id):
    author_obj = get_object_or_404(Author, pk=author_id)
    serializer = AuthorSerializer(author_obj)
    return Response(serializer.data)

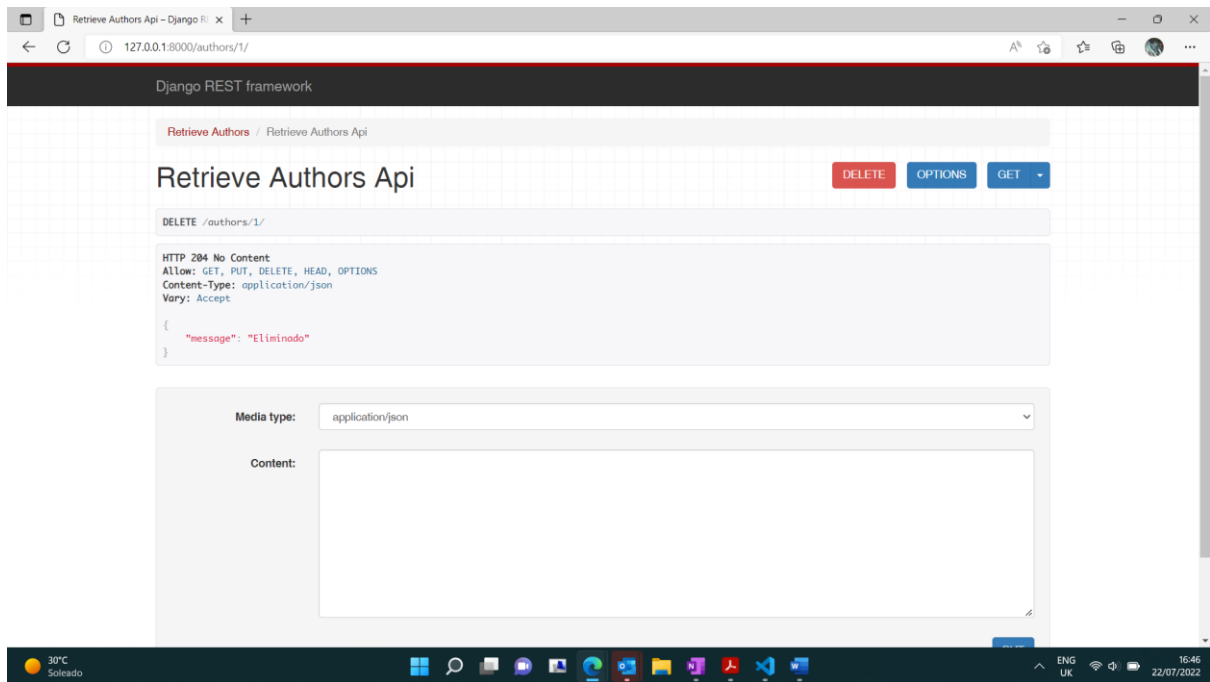
def put(self, request, author_id):
    author_obj = get_object_or_404(Author, pk=author_id)
    serializer = AuthorSerializer(instance=author_obj, data=request.data, partial=True)
    #request trae toda la info de lo que se esta generando afuera, info tipo JSON
    serializer.is_valid(raise_exception=True)
    #raise_exception es en caso de que no den un parametro no valido me diga y no continue con el proceso
    serializer.save()
    return Response(serializer.data, status=status.HTTP_200_OK)

def delete(self, request, author_id):
    author_obj = get_object_or_404(Author, pk=author_id)
    author_obj.status = False
    author_obj.save()
    return Response({'message': 'Eliminado'}, status=status.HTTP_204_NO_CONTENT)
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | JUPYTER

```
1.1" 304 0
[22/Jul/2022 09:56:59] "GET /static/rest_framework/img/grid.png HTTP/1.1" 304 0
(virtual) PS D:\Conacyt posdoc 2nd ano\Skills for Women in Tech\Backend en Django (2da sesion)\SFWET\skills> |
```





Quedando mi lista asi

