

Informe de Laboratorio 07

Tema: Django Rest Framework

Nota

Estudiante	Escuela	Asignatura
Andre David Delgado Allpan adelgadoal@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación web Semestre: III Código: 20231001

Laboratorio	Tema	Duración
07	Django Rest Framework	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 26 Junio 2023	Al 03 Julio 2023

1. URL de Repositorio Github

- URL para el laboratorio 01 en el Repositorio GitHub.
- <https://github.com/andre98652/pweb-lab8.git>

2. Ejercicio

- Practique desarrollando el ejercicio de iniciación:
- <https://www.django-rest-framework.org/tutorial/quickstart/>
- Para consumir el web-service puede usar el cliente SOAP UI Community:<https://www.soapui.org/downloads/soapui/>

3. Tarea

- En sus grupos de trabajo correspondientes. Elabore un servicio web que tenga un CRUD con el uso de este framework.
- Create - POST
- Read - GET
- Update - PUT

- Delete - DELETE
- Centrarce en el Core business de su aplicación web. Lo más importante y necesario que este disponible a través de un servicio web.
- Ejemplos: <https://reqbin.com/>, <https://www.googleapis.com/youtube/v3/playlistItems>
- Muestre la funcionalidad consumiendola desde el cliente Rest de su preferencia.

3.1. Informe de ejercicio

- Código de app quickstart-serializer.py

Listing 1: serializers.py

```
1 from django.contrib.auth.models import User, Group
2 from rest_framework import serializers
3
4
5 class UserSerializer(serializers.HyperlinkedModelSerializer):
6     class Meta:
7         model = User
8         fields = ['url', 'username', 'email', 'groups']
9
10
11 class GroupSerializer(serializers.HyperlinkedModelSerializer):
12     class Meta:
13         model = Group
14         fields = ['url', 'name']
```

- Código de app quickstart-views.py

Listing 2: views.py

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 from django.contrib.auth.models import User, Group
5 from rest_framework import viewsets
6 from rest_framework import permissions
7 from tutorial.quickstart.serializers import UserSerializer, GroupSerializer
8
9
10 class UserViewSet(viewsets.ModelViewSet):
11     """
12     API endpoint that allows users to be viewed or edited.
13     """
14     queryset = User.objects.all().order_by('-date_joined')
15     serializer_class = UserSerializer
16     permission_classes = [permissions.IsAuthenticated]
17
18
19 class GroupViewSet(viewsets.ModelViewSet):
20     """
21     API endpoint that allows groups to be viewed or edited.
```

```
22 """
23 queryset = Group.objects.all()
24 serializer_class = GroupSerializer
25 permission_classes = [permissions.IsAuthenticated]
```

- Código del proyecto-settings.py

Listing 3: settings.py

```
1 """
2 Django settings for tutorial project.
3
4 Generated by 'django-admin startproject' using Django 4.2.3.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/4.2/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/4.2/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-_skfz3k#kna(btmf8smruh)*%_2v3n3mc9gv6xh%*z4soj^xk#'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
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 ]
42
43
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
```

```
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52 ]
53
54 ROOT_URLCONF = 'tutorial.urls'
55
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': [],
60         'APP_DIRS': True,
61         'OPTIONS': {
62             'context_processors': [
63                 'django.template.context_processors.debug',
64                 'django.template.context_processors.request',
65                 'django.contrib.auth.context_processors.auth',
66                 'django.contrib.messages.context_processors.messages',
67             ],
68         },
69     },
70 ]
71
72 WSGI_APPLICATION = 'tutorial.wsgi.application'
73
74
75 # Database
76 # https://docs.djangoproject.com/en/4.2/ref/settings/#databases
77
78 DATABASES = {
79     'default': {
80         'ENGINE': 'django.db.backends.sqlite3',
81         'NAME': BASE_DIR / 'db.sqlite3',
82     }
83 }
84
85
86 # Password validation
87 # https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
88
89 AUTH_PASSWORD_VALIDATORS = [
90     {
91         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
92     },
93     {
94         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
95     },
96     {
97         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
98     },
99     {
100         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
101     },
102 ]
```

```
103
104
105 # Internationalization
106 # https://docs.djangoproject.com/en/4.2/topics/i18n/
107
108 LANGUAGE_CODE = 'en-us'
109
110 TIME_ZONE = 'UTC'
111
112 USE_I18N = True
113
114 USE_TZ = True
115
116
117 # Static files (CSS, JavaScript, Images)
118 # https://docs.djangoproject.com/en/4.2/howto/static-files/
119
120 STATIC_URL = 'static/'
121
122 # Default primary key field type
123 # https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
124
125 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
126 REST_FRAMEWORK = {
127     'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
128     'PAGE_SIZE': 10
129 }
```

- Código del proyecto-urls.py

Listing 4: urls.py

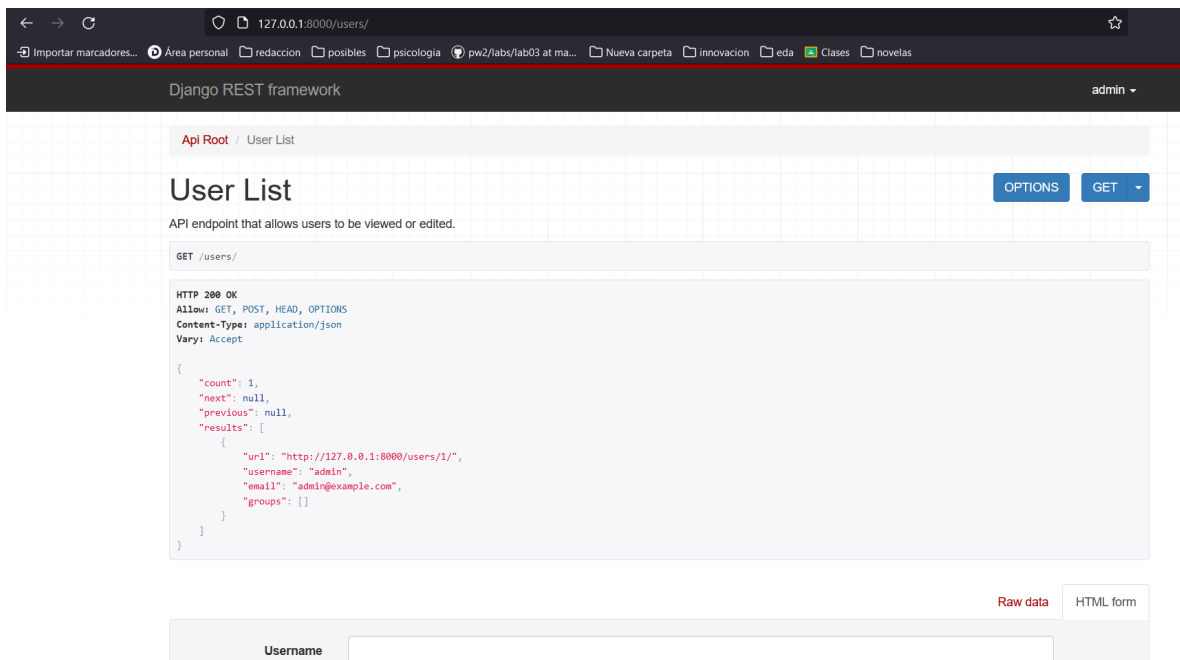
```
1 """
2 URL configuration for tutorial project.
3
4 The 'urlpatterns' list routes URLs to views. For more information please see:
5     https://docs.djangoproject.com/en/4.2/topics/http/urls/
6 Examples:
7 Function views
8     1. Add an import: from my_app import views
9     2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17
18
19 from django.urls import include, path
20 from rest_framework import routers
21 from tutorial.quickstart import views
22
23 router = routers.DefaultRouter()
```

```

24 router.register(r'users', views.UserViewSet)
25 router.register(r'groups', views.GroupViewSet)
26
27 # Wire up our API using automatic URL routing.
28 # Additionally, we include login URLs for the browsable API.
29 urlpatterns = [
30     path('', include(router.urls)),
31     path('api-auth/', include('rest_framework.urls', namespace='rest_framework'))
32 ]

```

■ Captura de la ejecución de ejercicio



The screenshot shows a web browser at the URL `127.0.0.1:8000/users/`. The page displays the Django REST framework API documentation for the `User List` endpoint. The breadcrumb navigation shows `Api Root / User List`. The endpoint is `GET /users/`. The response is shown in JSON format, indicating a successful `HTTP 200 OK` with `Allow: GET, POST, HEAD, OPTIONS` and `Content-Type: application/json`. The response body is a JSON object with the following structure:

```

{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://127.0.0.1:8000/users/1/",
      "username": "admin",
      "email": "admin@example.com",
      "groups": []
    }
  ]
}

```

At the bottom of the page, there is a form with a `Username` label and an input field.

4. Informe de tarea

- Se creo el proyecto api y la aplicacion crud
- En la app crud se trabajo con dos modelos autor y libros
- Código de app crud-models.py

Listing 5: models.py

```
1 from django.db import models
2
3 # Create your models here.
4
5 class Autor(models.Model):
6     nombre = models.CharField(max_length=100)
7     pais = models.CharField(max_length=100)
8
9     def __str__(self):
10         return self.nombre
11
12 class Libro(models.Model):
13     titulo = models.CharField(max_length=100)
14     autor = models.ForeignKey(Autor, on_delete=models.CASCADE)
15     descripcion = models.TextField()
16
17     def __str__(self):
18         return self.titulo
```

- Código de app crud-serializers.py

Listing 6: serializers.py

```
1 from rest_framework import serializers
2 from .models import Autor, Libro
3
4 class AutorSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Autor
7         fields = '__all__'
8
9 class LibroSerializer(serializers.ModelSerializer):
10     class Meta:
11         model = Libro
12         fields = '__all__'
```

- Código de app crud-views.py, con los metodos requeridos crear, obtener, actualizar y eliminar.

Listing 7: views.py

```
1 from rest_framework import viewsets, status
2 from .models import Autor, Libro
3 from .serializers import AutorSerializer, LibroSerializer
4 from rest_framework.decorators import api_view
```

```
5 from rest_framework.response import Response
6
7 @api_view(['POST'])
8 def crear_autor(request):
9     if request.method == 'POST':
10         serializer = AutorSerializer(data=request.data)
11         if serializer.is_valid():
12             serializer.save()
13             return Response(serializer.data, status=status.HTTP_201_CREATED)
14         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
15 @api_view(['POST'])
16 def crear_libro(request):
17     if request.method == 'POST':
18         serializer = LibroSerializer(data=request.data)
19         if serializer.is_valid():
20             serializer.save()
21             return Response(serializer.data, status=status.HTTP_201_CREATED)
22         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
23 @api_view(['GET'])
24 def obtener_lista_autores(request):
25     if request.method == 'GET':
26         autores = Autor.objects.all()
27         serializer = AutorSerializer(autores, many=True)
28         return Response(serializer.data, status=status.HTTP_200_OK)
29 @api_view(['GET'])
30 def obtener_lista_libros(request):
31     if request.method == 'GET':
32         libros = Libro.objects.all()
33         serializer = LibroSerializer(libros, many=True)
34         return Response(serializer.data, status=status.HTTP_200_OK)
35 @api_view(['GET'])
36 def obtener_detalle_autor(request, autor_id):
37     try:
38         autor = Autor.objects.get(id=autor_id)
39     except Autor.DoesNotExist:
40         return Response({"error": "El autor no existe."}, status=status.HTTP_404_NOT_FOUND)
41
42     if request.method == 'GET':
43         serializer = AutorSerializer(autor)
44         return Response(serializer.data, status=status.HTTP_200_OK)
45 @api_view(['GET'])
46 def obtener_detalle_libro(request, libro_id):
47     try:
48         libro = Libro.objects.get(id=libro_id)
49     except Libro.DoesNotExist:
50         return Response({"error": "El libro no existe."}, status=status.HTTP_404_NOT_FOUND)
51
52     if request.method == 'GET':
53         serializer = LibroSerializer(libro)
54         return Response(serializer.data, status=status.HTTP_200_OK)
55 @api_view(['PUT'])
56 def actualizar_autor(request, autor_id):
57     try:
58         autor = Autor.objects.get(id=autor_id)
59     except Autor.DoesNotExist:
60         return Response({"error": "El autor no existe."}, status=status.HTTP_404_NOT_FOUND)
```



```

61
62     if request.method == 'PUT':
63         serializer = AutorSerializer(author, data=request.data)
64         if serializer.is_valid():
65             serializer.save()
66             return Response(serializer.data, status=status.HTTP_200_OK)
67         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
68 @api_view(['PUT'])
69 def actualizar_libro(request, libro_id):
70     try:
71         libro = Libro.objects.get(id=libro_id)
72     except Libro.DoesNotExist:
73         return Response({"error": "El libro no existe."}, status=status.HTTP_404_NOT_FOUND)
74
75     if request.method == 'PUT':
76         serializer = LibroSerializer(libro, data=request.data)
77         if serializer.is_valid():
78             serializer.save()
79             return Response(serializer.data, status=status.HTTP_200_OK)
80         return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
81 @api_view(['DELETE'])
82 def eliminar_autor(request, autor_id):
83     try:
84         autor = Autor.objects.get(id=autor_id)
85     except Autor.DoesNotExist:
86         return Response({"error": "El autor no existe."}, status=status.HTTP_404_NOT_FOUND)
87
88     if request.method == 'DELETE':
89         autor.delete()
90         return Response(status=status.HTTP_204_NO_CONTENT)
91 @api_view(['DELETE'])
92 def eliminar_libro(request, libro_id):
93     try:
94         libro = Libro.objects.get(id=libro_id)
95     except Libro.DoesNotExist:
96         return Response({"error": "El autor no existe."}, status=status.HTTP_404_NOT_FOUND)
97
98     if request.method == 'DELETE':
99         libro.delete()
100         return Response(status=status.HTTP_204_NO_CONTENT)
101
102 class AutorViewSet(viewsets.ModelViewSet):
103     queryset = Autor.objects.all()
104     serializer_class = AutorSerializer
105
106 class LibroViewSet(viewsets.ModelViewSet):
107     queryset = Libro.objects.all()
108     serializer_class = LibroSerializer

```

- Código de app crud-urls.py

Listing 8: urls.py

```

1 from django.urls import path, include
2 from rest_framework.routers import DefaultRouter

```

```
3 from .views import *
4
5 # Crea un router y registra los ViewSets
6 router = DefaultRouter()
7 router.register(r'autores-api', AutorViewSet)
8 router.register(r'libros-api', LibroViewSet)
9
10 # Las URLs de la API estan determinadas automaticamente por el router
11 urlpatterns = [
12     path('', include(router.urls)),
13     path('libros/lista/', obtener_lista_libros, name='lista_libros'),
14     path('libros/nuevo/', crear_libro, name='crear_libro'),
15     path('libros/<int:libro_id>/', obtener_detalle_libro, name='detalle_libro'),
16     path('libros/<int:libro_id>/actualizar/', actualizar_libro, name='actualizar_libro'),
17     path('autores/lista/', obtener_lista_autores, name='lista_autores'),
18     path('autores/<int:autor_id>/', obtener_detalle_autor, name='detalle_autor'),
19     path('autores/nuevo/', crear_autor, name='crear_autor'),
20     path('autores/<int:autor_id>/actualizar/', actualizar_autor, name='actualizar_autor'),
21     path('autores/<int:autor_id>/eliminar/', eliminar_autor, name='eliminar_autor'),
22 ]
```

- Código del proyecto api-settings.py

Listing 9: settings.py

```
1 """
2 Django settings for api project.
3
4 Generated by 'django-admin startproject' using Django 4.2.3.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/4.2/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/4.2/ref/settings/
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-iqp49h7_@*05c2y!od!o!mx4@rb-@o-!msuxcgw!oj*m_p_eph'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
```

```
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     'crud',
42 ]
43
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52 ]
53
54 ROOT_URLCONF = 'api.urls'
55
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': [],
60         'APP_DIRS': True,
61         'OPTIONS': {
62             'context_processors': [
63                 'django.template.context_processors.debug',
64                 'django.template.context_processors.request',
65                 'django.contrib.auth.context_processors.auth',
66                 'django.contrib.messages.context_processors.messages',
67             ],
68         },
69     },
70 ]
71
72 WSGI_APPLICATION = 'api.wsgi.application'
73
74
75 # Database
76 # https://docs.djangoproject.com/en/4.2/ref/settings/#databases
77
78 DATABASES = {
79     'default': {
80         'ENGINE': 'django.db.backends.sqlite3',
81         'NAME': BASE_DIR / 'db.sqlite3',
82     }
83 }
84
85
86 # Password validation
```

```
87 # https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
88
89 AUTH_PASSWORD_VALIDATORS = [
90     {
91         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
92     },
93     {
94         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
95     },
96     {
97         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
98     },
99     {
100         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
101     },
102 ]
103
104
105 # Internationalization
106 # https://docs.djangoproject.com/en/4.2/topics/i18n/
107
108 LANGUAGE_CODE = 'en-us'
109
110 TIME_ZONE = 'UTC'
111
112 USE_I18N = True
113
114 USE_TZ = True
115
116
117 # Static files (CSS, JavaScript, Images)
118 # https://docs.djangoproject.com/en/4.2/howto/static-files/
119
120 STATIC_URL = 'static/'
121
122 # Default primary key field type
123 # https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
124
125 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

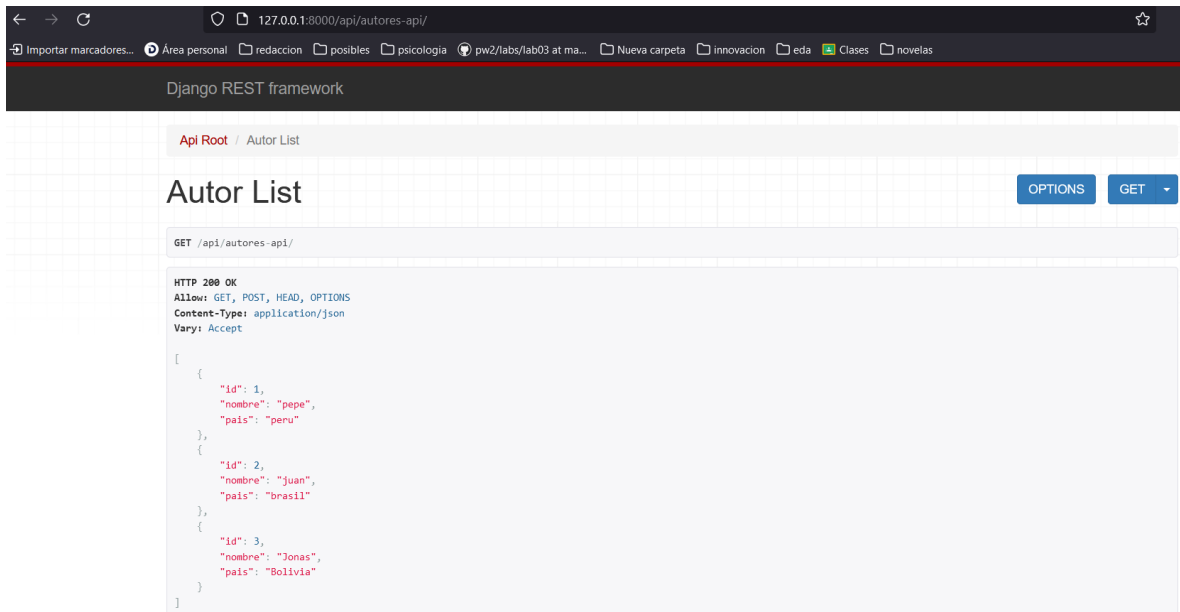
- Código del proyecto api-urls.py

Listing 10: urls.py

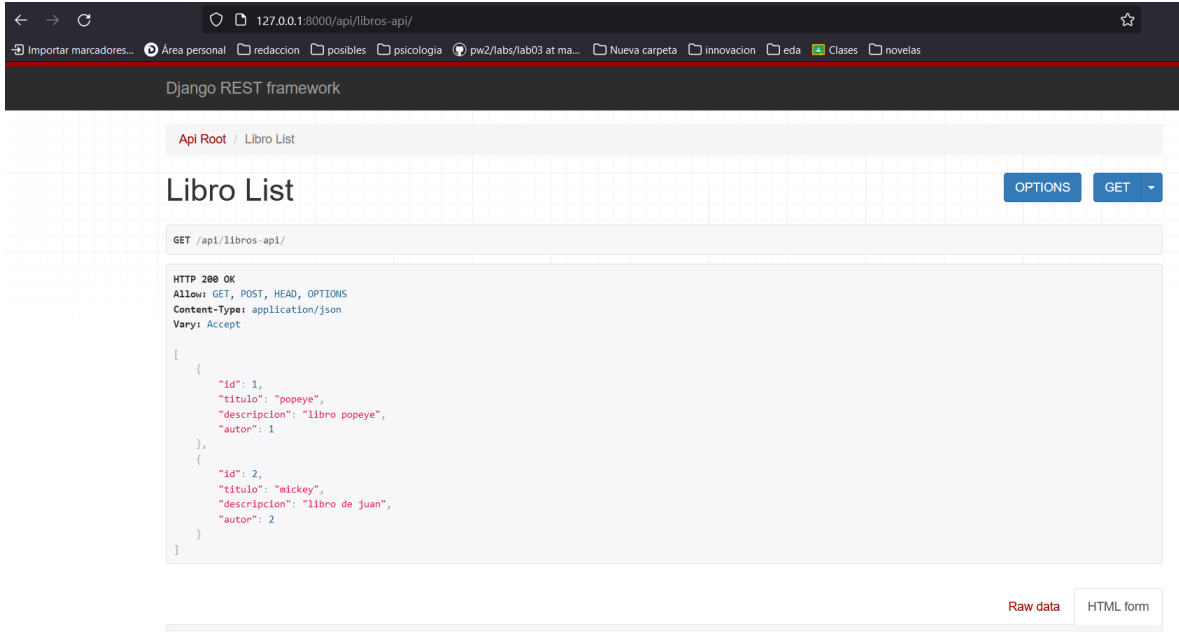
```
1 """
2 URL configuration for api project.
3
4 The 'urlpatterns' list routes URLs to views. For more information please see:
5     https://docs.djangoproject.com/en/4.2/topics/http/urls/
6 Examples:
7 Function views
8     1. Add an import: from my_app import views
9     2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
```

```
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from rest_framework.routers import DefaultRouter
20 from crud.views import AutorViewSet, LibroViewSet
21
22 router = DefaultRouter()
23 router.register(r'autores', AutorViewSet)
24 router.register(r'libros', LibroViewSet)
25
26 urlpatterns = [
27     path('admin/', admin.site.urls),
28     path('api/', include('crud.urls')),
29
30 ]
```

- Captura de la lista de autores por formato json pagina web



- Captura de la lista de libros por formato json pagina web



127.0.0.1:8000/api/libros-api/

Importar marcadores... Área personal redaccion posibles psicologia pw2/labs/lab03 at ma... Nueva carpeta innovacion eda Clases novelas

Django REST framework

Api Root / Libro List

Libro List

OPTIONS GET

GET /api/libros-api/

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

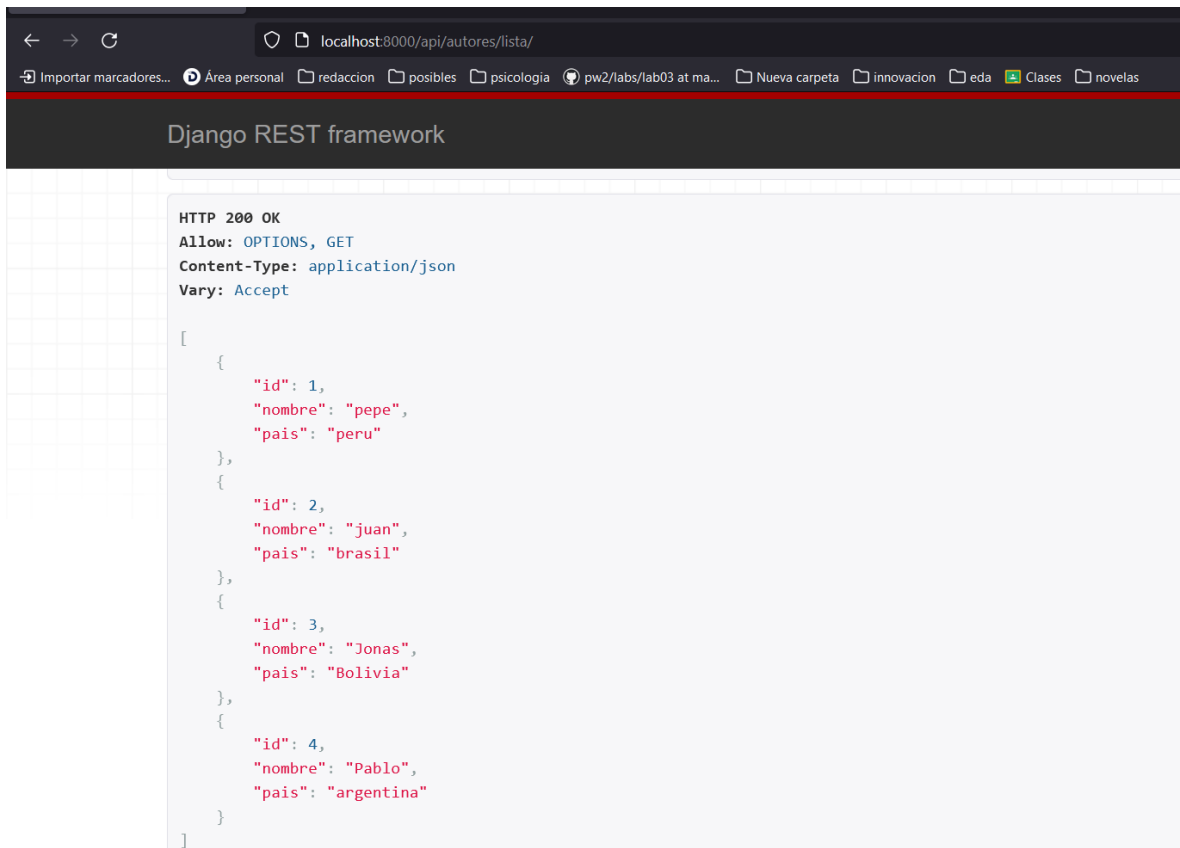
```
[
  {
    "id": 1,
    "titulo": "popeye",
    "descripcion": "libro popeye",
    "autor": 1
  },
  {
    "id": 2,
    "titulo": "mickey",
    "descripcion": "libro de juan",
    "autor": 2
  }
]
```

Raw data HTML form

5. Consumiendo la API

- curl para ingresar un autor, METODO POST

```
C:\Windows\system32>curl -X POST -H "Content-Type: application/json" -d '{"nombre": "Pablo",  
{"id":4,"nombre":"Pablo","pais":"argentina"}' http://localhost:8000/api/autores-api/  
C:\Windows\system32>
```



- curl para obtener la lista autores, METODO GET

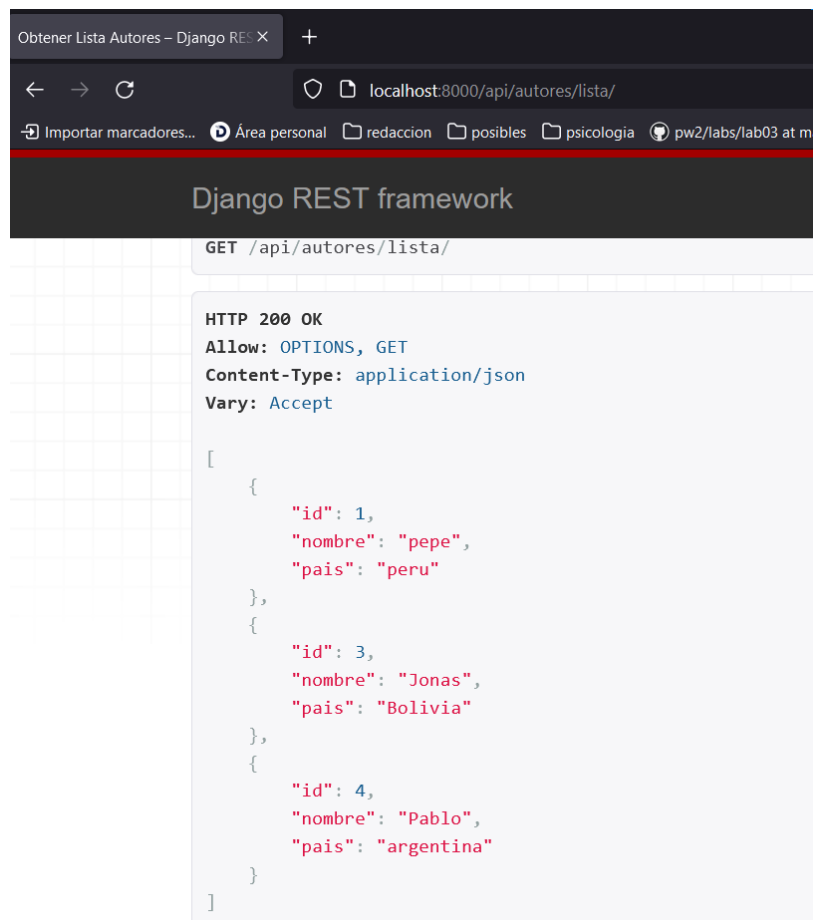
```
C:\Windows\system32>curl -X GET http://localhost:8000/api/autores-api/  
[{"id":1,"nombre":"pepe","pais":"peru"}, {"id":2,"nombre":"juan","pais":"brasil"}, {"id":3,  
"nombre":"Jonas","pais":"Bolivia"}, {"id":4,"nombre":"Pablo","pais":"argentina"}]  
C:\Windows\system32>
```

- curl para obtener autor por ID, METODO GET

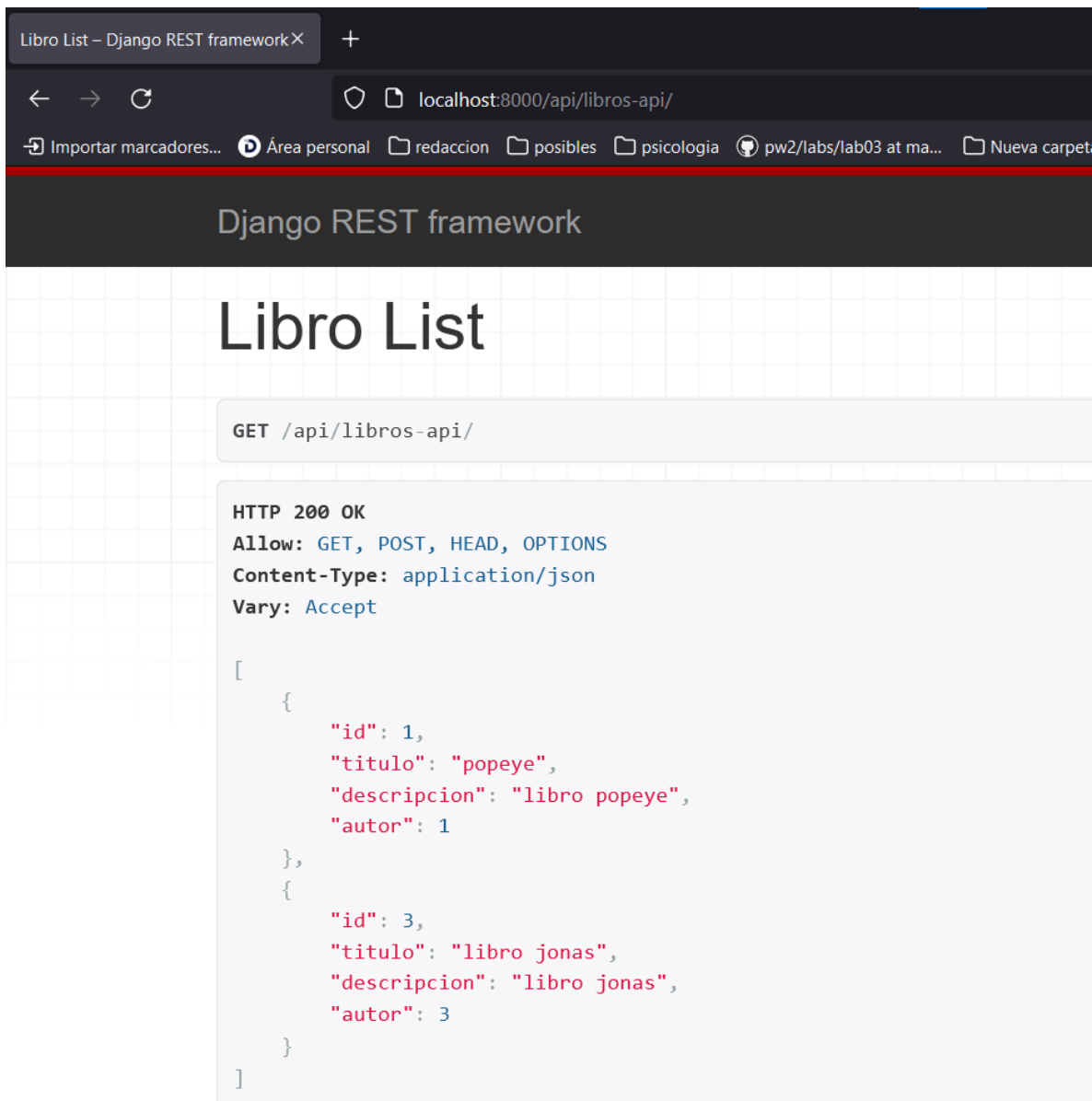
```
C:\Windows\system32>curl -X GET http://localhost:8000/api/autores-api/2/  
{  
  "id": 2,  
  "nombre": "juan",  
  "pais": "brasil"  
}  
C:\Windows\system32>
```

- curl para eliminar autor por ID, METODO DELETE

```
C:\Windows\system32>curl -X DELETE http://localhost:8000/api/autores-api/2/  
  
C:\Windows\system32>
```



- Al eliminar ese autor se elimina su libro



Libro List – Django REST frameworkX +

localhost:8000/api/libros-api/

Importar marcadores... Área personal redaccion posibles psicologia pw2/labs/lab03 at ma... Nueva carpeta

Django REST framework

Libro List

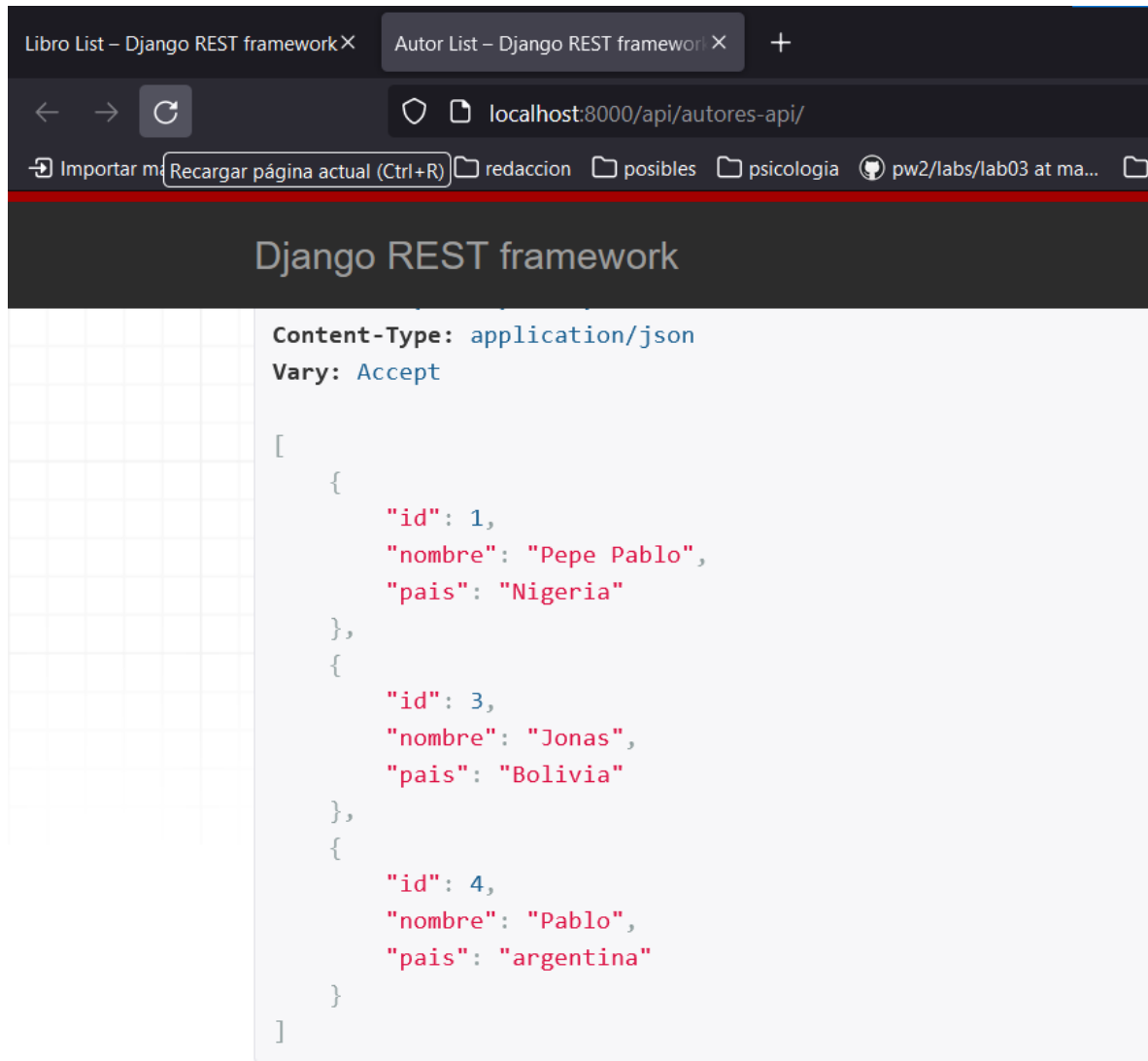
GET /api/libros-api/

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

```
[
  {
    "id": 1,
    "titulo": "popeye",
    "descripcion": "libro popeye",
    "autor": 1
  },
  {
    "id": 3,
    "titulo": "libro jonas",
    "descripcion": "libro jonas",
    "autor": 3
  }
]
```

- curl para actualizar autor por ID, METODO PUT

```
C:\Users\Lenovo>curl -X PUT -H "Content-Type: application/json" -d '{"nombre":"Pepe Pablo", "pais": "Nigeria"}' http://localhost:8000/api/autores-api/1/
{"id":1,"nombre":"Pepe Pablo","pais":"Nigeria"}
C:\Users\Lenovo>
```

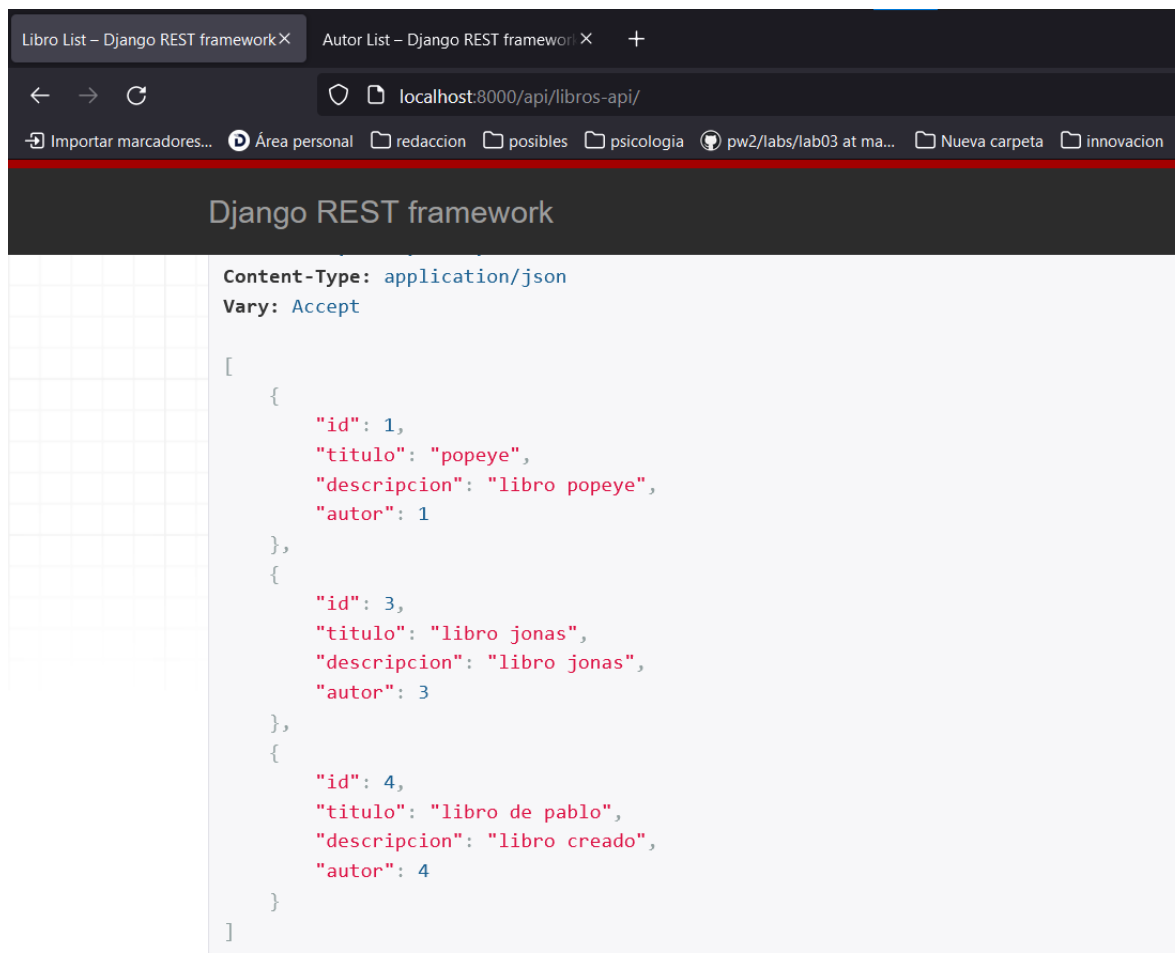


- curl para obtener lista de libros, METODO GET

```
C:\Users\Lenovo>curl -X GET http://localhost:8000/api/libros-api/
[{"id":1,"titulo":"popeye","descripcion":"libro popeye","autor":1},{ "id":3,"titulo":"libro jonas","descripcion":"libro jonas","autor":3}]
C:\Users\Lenovo>
```

- curl para ingresar libro, METODO POST

```
C:\Users\Lenovo>curl -X POST -H "Content-Type: application/json" -d '{"titulo": "libro de pablo",  
{"autor": 4, "descripcion": "libro creado"}' http://localhost:8000/api/libros-api/  
{"id":4,"titulo":"libro de pablo","descripcion":"libro creado","autor":4}  
C:\Users\Lenovo>
```

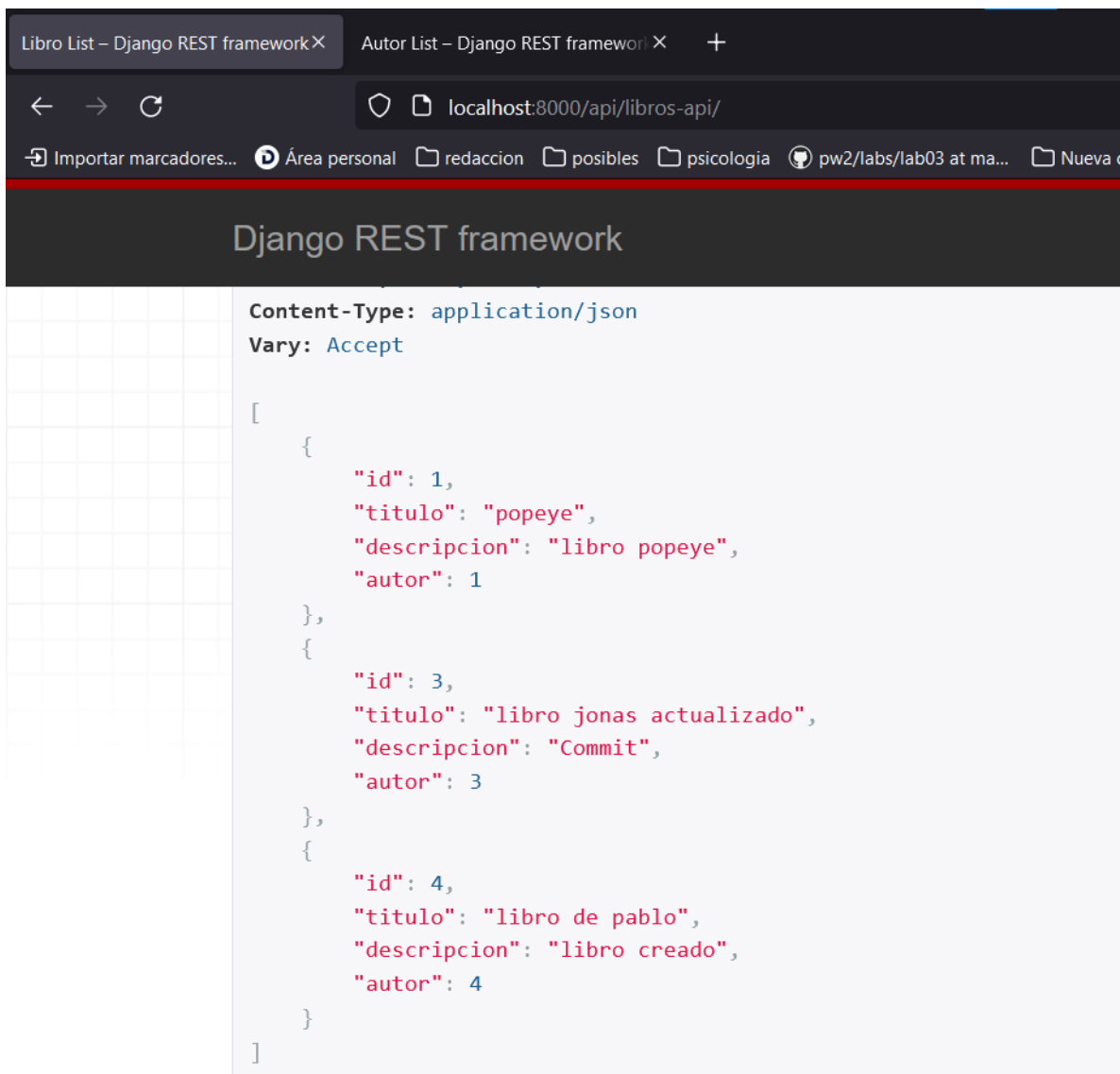


- curl para obtener libro por ID, METODO GET

```
C:\Users\Lenovo>curl -X GET http://localhost:8000/api/libros-api/1/  
{  
  "id": 1, "titulo": "popeye", "descripcion": "libro popeye", "autor": 1  
}  
C:\Users\Lenovo>
```

- curl para actualizar libro por ID, METODO PUT

```
C:\Users\Lenovo>curl -X PUT -H "Content-Type: application/json" -d '{"titulo": "libro jonas actualizado",  
  "autor": 3, "descripcion": "Commit"}' http://localhost:8000/api/libros-api/3/  
{  
  "id": 3, "titulo": "libro jonas actualizado", "descripcion": "Commit", "autor": 3  
}  
C:\Users\Lenovo>
```



Libro List – Django REST framework × Autor List – Django REST framework × +

← → ↻ localhost:8000/api/libros-api/

Importar marcadores... Área personal redaccion posibles psicologia pw2/labs/lab03 at ma... Nueva d

Django REST framework

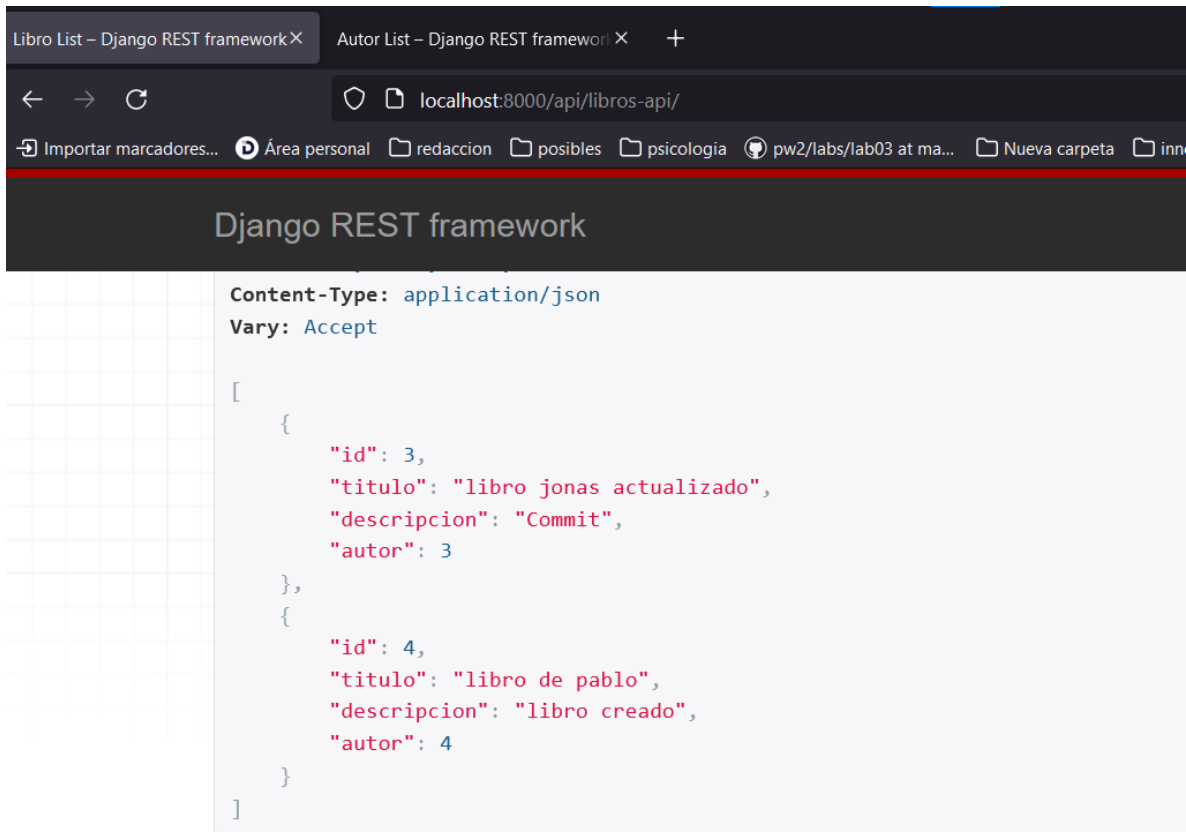
Content-Type: application/json
Vary: Accept

```
[  
  {  
    "id": 1,  
    "titulo": "popeye",  
    "descripcion": "libro popeye",  
    "autor": 1  
  },  
  {  
    "id": 3,  
    "titulo": "libro jonas actualizado",  
    "descripcion": "Commit",  
    "autor": 3  
  },  
  {  
    "id": 4,  
    "titulo": "libro de pablo",  
    "descripcion": "libro creado",  
    "autor": 4  
  }  
]
```

- curl para eliminar libro por ID, METODO DELETE

```
C:\Users\Lenovo>curl -X DELETE http://localhost:8000/api/libros-api/1/

C:\Users\Lenovo>
```



6. Cuestionario

- ¿Cuál fue la mayor dificultad del uso de este framework?

La mayor dificultad del uso de este framework fue la curva de aprendizaje inicial y la comprensión completa de sus conceptos y funcionamiento. Al ser un framework completo y robusto, Django tiene una amplia gama de características y funcionalidades que pueden resultar abrumadoras al principio.

Además, la configuración inicial del proyecto y la comprensión de la estructura de directorios pueden ser un desafío, especialmente para aquellos que son nuevos en el desarrollo web o en el uso de frameworks.

7. Rúbricas

7.1. Entregable Informe

Tabla 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y facil de leer.

7.2. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Tabla 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4			
Total		20		12	

8. Referencias

- <https://www.w3schools.com/java/default.asp>
- <https://www.geeksforgeeks.org/insertion-sort/>