Desarrollo de un API RESTful con Django

El framework Django permite desarrollar APIs REST de forma sencilla. En esta guía se indican los pasos para su desarrollo.

Nota: existe un framework específico para crear APIs REST llamado Django Rest Framework.

Caso de ejemplo:

El API mantendrá la entidad Videojuego, compuesta por los campos:

Nombre de campo	Características
Id	Numérico, PK, AI, NN
Título	Alfanumérico (100), NN
Género	Alfanumérico (100), NN
Plataforma	Alfanumérico (50), NN
Año	Numérico, NN

Pasos.

1. Creación del proyecto y de la aplicación.

```
django-admin startproject proyecto_vj
```

Desde dentro de la carpeta del proyecto:

```
python manage.py migrate
python manage.py startapp vj_app
```

2. Registro de la aplicación.

- Modificar el fichero settings.py
- Incluir en la lista **INSTALLED_APPS** el nombre de la aplicación

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

3. Arrancar el servidor.

```
python manage.py runserver
```

4. Creación del modelo.

- Modificar el fichero models.py de la aplicación para incluir las definiciones de los modelos.

```
from django.db import models

# Create your models here.

class Videojuego(models.Model):
    #id (se genera automáticamente)
    titulo=models.CharField(max_length=100,null=False)
    genero=models.CharField(max_length=100,null=False)
    plataforma=models.CharField(max_length=50,null=False)
    anyo=models.IntegerField(null=False)
```

- Generar las tablas:

```
python manage.py makemigrations
python manage.py migrate
```

5. Registrar el modelo para poder acceder a él desde la consola de administración.

- Modificar el fichero admin.py de la aplicación:

```
from django.contrib import admin

from .models import Videojuego

# Register your models here.

admin.site.register(Videojuego)
```

Recuerda. Para crear un usuario de administración se debe ejecutar el comando:

```
python manage.py createsuperuser
```

6. Creación de las vistas.

Se crean las vistas basadas en clases (CBV).

- Modificar el fichero views.py de la aplicación.
- Agregar los import necesarios:

```
from django.shortcuts import render

from django.views import View #Para realizar las vistas basadas
en clases en lugar de funciones
```

```
from vj_app.models import Videojuego #Modelo
from django.http import JsonResponse #Conversor a JSON

from django.utils.decorators import method_decorator #Para
desactivar la protección frente a Cross Site Request Forgery
(CSRF)
from django.views.decorators.csrf import csrf_exempt #Para
desactivar la protección frente a Cross Site Request Forgery
(CSRF)

from django.core.exceptions import ObjectDoesNotExist #Excepción
de objeto no encontrado
```

- Crear la clase que gestiona las vistas y los métodos del API REST.

```
@method_decorator(csrf_exempt, name='dispatch')
class VideojuegoView(View):
    def get(self, request):
        videojuegos = Videojuego.objects.all()
        return JsonResponse(list(videojuegos.values()),
safe=False)

    def post(self, request, titulo, genero, plataforma, anyo):
        Videojuego(titulo=titulo, genero=genero,
plataforma=plataforma, anyo=anyo).save()
        retorno = {"code":0,"message":"OK"}
        return JsonResponse(retorno)

    def put(self, request, id, titulo, genero, plataforma,
anyo):
        try:
            videojuego = Videojuego.objects.get(pk=id)
            videojuego.titulo = titulo
            videojuego.genero = genero
            videojuego.plataforma = plataforma
            videojuego.anyo = anyo
            videojuego.save()
            retorno = {"code":0,"message":"Ok"}
        except ObjectDoesNotExist:
            retorno = {"code":1,"message":"Entidad no
existente"}
        return JsonResponse(retorno)

    def delete(self, request, id):
        Videojuego.objects.get(pk=id).delete()
        retorno = {"code":0,"message":"Ok"}
        return JsonResponse(retorno)
```

7. Creación de las URLS.

La creación de las URLs necesita realizar dos acciones:

- Crear el fichero de URLs de la aplicación.
- Modificar el fichero de URLS del proyecto para que lea el proyecto anterior.

CREACIÓN DEL FICHERO DE URLs de la aplicación

MODIFICACIÓN DEL FICHERO URLs del proyecto:

```
from django.contrib import admin
from django.urls import path
from django.urls import include #Necesario para referenciar
otros orígenes de URLS

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include("vj_app.urls"))
]
```

CREACIÓN DEL FICHERO DE URLs de la aplicación

- Creación del fichero urls.py en la carpeta de la aplicación.
- Incluir las referencias a los métodos incluyendo los parámetros.

```
from django.urls import path
from vj_app.views import VideojuegoView

urlpatterns = [
path('videojuego/', VideojuegoView.as_view(),
name="find_all_videojuegos"),

path('videojuego/<str:titulo>/<str:genero>/<str:plataforma>/<int
:anyo>/', VideojuegoView.as_view(), name="create_videojuego"),

path('videojuego/<int:id>/<str:titulo>/<str:genero>/<str:platafo
rma>/<int:anyo>/', VideojuegoView.as_view(),
name="update_videojuego"),

path('videojuego/<int:id>/', VideojuegoView.as_view(),
name="delete_videojuego"),
]
```

8. Llamadas.

POST:

http://localhost:8000/api/videojuego/Metal Gear/Sigilo/PlayStation 1/1998/

PUT:

http://localhost:8000/api/videojuego/3/Metal Gear Solid/Sigilo y táctico/PlayStation/1998/

GET:

http://localhost:8000/api/videojuego/

DEL:

http://localhost:8000/api/videojuego/1/