C:\>cd C:\entornos\APIS

C:\entornos\APIS>python -m venv subastas\_clase

C:\entornos\APIS>subastas\_clase\Scripts\activate

(subastas\_clase) C:\entornos\APIS>pip install django

Configurar en pychar el interpreter y runserver

pip install djangorestframework

agregar: en installed app

**rest\_framework**

**también agregar:**

REST\_FRAMEWORK = {

'DEFAULT\_RENDERER\_CLASSES': [

'rest\_framework.renderers.JSONRenderer',

'rest\_framework.renderers.BrowsableAPIRenderer'],

}

Los serializadores me sirven para convertir desde un objeto a un formato de tipo jason y viceversa

Permite que el ususario me envie un objeto de tipo jason, validarlos y convertilos a objetos

python manage.py shell

>>> from apps.anuncio.models import Categoria

>>> from apps.anuncio.serializer import CategoriaSerializer

Ver si tengo categorías:

>>> c=Categoria.objects.first()

>>> c

<Categoria: Muebles>

**Creo el serializador:**

>>> serializer\_categoria=CategoriaSerializer(c)

**Accedo al atributo data**

>>> serializer\_categoria.data

**Y me muestra los EL OBJETO EN UN FORMATO DE TIPO DICCIONARIO**

{'id': 1, 'nombre': 'Muebles', 'activa': True}

Si quiero ver todas las categorías:

>>> categorias = Categoria.objects.all()

>>> categorias

<QuerySet [<Categoria: Muebles>, <Categoria: Automotor>, <Categoria: Computación>]>

>>>

También lo puedo hacer como una consulta y me devuelve una lista, un diccionario por cada objeto de ese queryset

>>> serializer\_categorias=CategoriaSerializer(categorias,many=True )

>>> serializer\_categorias.data

[{'id': 1, 'nombre': 'Muebles', 'activa': True}, {'id': 2, 'nombre': 'Automotor', 'activa': True}, {'id': 3, 'nombre': 'Computación', 'activa': True}]

Este no es el formato final que se envia a los usuarios, se convierte el tipo de dato lista en formato de tipo jason

Importamos JSONRenderer

>>> from rest\_framework.renderers import JSONRenderer

Aca me devuelve en un formato jason, es decir transformo por ejemplo las comillas simples en comillas dobles

>>> JSONRenderer().render(serializer\_categorias.data)

b'[{"id":1,"nombre":"Muebles","activa":true},{"id":2,"nombre":"Automotor","activa":true},{"id":3,"nombre":"Computaci\xc3\xb3n","activa":true}]'

from apps.anuncio.serializers import AnuncioSerializer

>>> from apps.anuncio.serializers import CategoriaSerializer

a\_serializer = AnuncioSerializer()

**acá al aplicar la función repr me da la definición de esa serialización**

>>> print(repr(a\_serializer))

AnuncioSerializer():

id = IntegerField(label='ID', read\_only=True)

titulo = CharField(max\_length=200)

precio\_inicial = DecimalField(decimal\_places=2, max\_digits=10)

fecha\_publicacion = DateTimeField(read\_only=True)

categorias = PrimaryKeyRelatedField(many=True, queryset=Categoria.objects.all(), required=False)

publicado\_por = PrimaryKeyRelatedField(read\_only=True)

oferta\_ganadora = PrimaryKeyRelatedField(allow\_null=True, read\_only=True)

>>>

**Obtener un anuncio**

>>> anuncio = Anuncio.objects.first()

>>> anuncio

<Anuncio: Escritorio de PC - Activo >

**Y vemos sus categorías**

>>> anuncio.categorias.all()

<QuerySet [<Categoria: Muebles>, <Categoria: Computación>]>

>>>

Ahora creamos un serializador

anuncio\_serializer = AnuncioSerializer(anuncio)

>>> anuncio\_serializer.data

{'id': 1, 'titulo': 'Escritorio de PC', 'precio\_inicial': '450000.00', 'fecha\_publicacion': '2025-04-01T15:34:14.489725Z', 'categorias': [1, 3], 'publicado\_por': 1, 'oferta\_ganadora': None}

>>>

Probando las operaciones:

POST

{

"titulo": "Mouse Logitech",

"descripcion": "Mouse inalámbrico con receptor USB",

"precio\_inicial": "2500.00",

"categorias": [1]

}

PUT

{

"titulo": "Nuevo título",

"precio\_inicial": 123.45

}

Tp4

from apps.anuncio.serializers import CategoriaSerializer

serializer = CategoriaSerializer(data={'nombre': 'Categoriaprincipal', 'activa': False})

serializer.is\_valid()

# False

serializer.errors

# {'nombre': [ErrorDetail(string="El nombre no puede contener la palabra 'categoria'.", code='invalid')]}

Probar el orden y filter:

para ver anuncios activos:

<http://127.0.0.1:8000/view-set/anuncios/?activo=true>

Filtrar por categoría:

<http://127.0.0.1:8000/view-set/anuncios/?categorias=1>

O por usuario:

<http://127.0.0.1:8000/view-set/anuncios/?publicado_por=3>

Ordenar por precio descendente:

http://127.0.0.1:8000/view-set/anuncios/?ordering=-precio\_inicial

Ordenar por fecha de inicio ascendente:

http://127.0.0.1:8000/view-set/anuncios/?ordering=fecha\_inicio

**Búsqueda**

Buscar por palabra clave en el título o descripción:

http://127.0.0.1:8000/view-set/anuncios/?search=Samsung

para categoría:

Filtro exacto:

/view-set/categoria/?nombre=automotor

Búsqueda parcial:

/view-set/categoria/?search=auto

Orden descendente:

/view-set/categoria/?ordering=-nombre

Con filter.py:

* Filtrar por rango de precio:

/view-set/anuncios/?precio\_min=1000&precio\_max=3000

* Filtrar por rango de fechas:

/view-set/anuncios/?fecha\_inicio\_min=2025-04-01T00:00:00&fecha\_inicio\_max=2025-05-01T23:59:59

* Combinado con búsqueda:

/view-set/anuncios/?search=televisor&precio\_min=5000

########## tp4##############################  
  
 ########## validaciones tp4 ######################  
"""  
 def validate(self, data):  
 fecha\_inicio = data.get('fecha\_inicio', None)  
 fecha\_fin = data.get('fecha\_fin', None)  
 precio = data.get('precio\_inicial', None)  
 categorias = data.get('categorias', [])  
  
 # a) que la fecha de inicio del anuncio sea posterior a la fecha actual  
 if fecha\_inicio and fecha\_inicio < timezone.now():  
 raise serializers.ValidationError({  
 'fecha\_inicio': 'La fecha de inicio debe ser posterior a la fecha y hora actual.'  
 })  
  
 # b) que la fecha fin sea posterior a la fecha de inicio  
 if fecha\_inicio and fecha\_fin and fecha\_fin <= fecha\_inicio:  
 raise serializers.ValidationError({  
 'fecha\_fin': 'La fecha de fin debe ser posterior a la fecha de inicio.'  
 })  
  
 # Validar precio mayor a cero  
 if precio is not None and precio < 0:  
 raise serializers.ValidationError({  
 'precio\_inicial': 'El precio inicial debe ser mayor a cero.'  
 })  
  
 # Validar que las categorías estén activas  
 for cat in categorias:  
 if not cat.activa:  
 raise serializers.ValidationError({  
 'categorias': f'La categoría "{cat.nombre}" está inactiva y no puede ser seleccionada.'  
 })  
  
 return data  
"""