Guía paso a paso: Crear una Books API con Django 5 + DRF + MySQL

Objetivo

Levantar una API REST que permita **listar libros**, **crear libros** y **consultar un libro por ID**, utilizando:

- **Django 5.x**
- **Django REST Framework (DRF)**
- **Base de datos MySQL** llamada `books_simple_drf_db`
- Expuesta en el puerto **8001** y accesible desde **0.0.0.0**

Paso 1: Crear el entorno virtual y las dependencias

```
python -m venv venv
source venv/bin/activate          # En Windows: venv\Scripts\activate
pip install "Django>=5.1" "djangorestframework>=3.16" PyMySQL
```

Paso 2: Crear el proyecto y la app

```
django-admin startproject books_project .
python manage.py startapp books
```

Esto genera la estructura de carpetas:

```
.
├── books/              # Nuestra app (modelos, vistas, urls...)
└── books_project/      # Configuración principal (settings.py, urls.py...)
```

Paso 3: Configurar `settings.py`

1. **Activar la app y DRF:**

```
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
```

```
    "django.contrib.staticfiles",
    "rest_framework",
    "books",
]
```

1. **Conexión a MySQL:**

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
        "NAME": "books_simple_drf_db",
        "USER": "example_user",
        "PASSWORD": "example_password",
        "HOST": "127.0.0.1",
        "PORT": "3306",
        "OPTIONS": {"charset": "utf8mb4"},
    }
}
```

1. **PyMySQL como driver MySQL:**

```
import pymysql
pymysql.install_as_MySQLdb()
```

1. **Hosts permitidos:**

```
ALLOWED_HOSTS = ["0.0.0.0",
                 "localhost"]
```

Paso 4: Crear el modelo `Book`

``

```
import secrets
from django.db import models

def _hex_id():
    return secrets.token_hex(12)  # 24 caracteres hexadecimales

class Book(models.Model):
    id     = models.CharField(primary_key=True, max_length=24,
default=_hex_id, editable=False)
    title  = models.CharField(max_length=255)
    author = models.CharField(max_length=255)
    pages  = models.PositiveIntegerField(default=0)
```

```
    class Meta:
        db_table = "books"
        ordering = ["title"]
```

Paso 5: Crear el serializador DRF

``

```
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
    class Meta:
        model  = Book
        fields = ["id", "title", "author", "pages"]
```

Paso 6: Crear las vistas (endpoints)

``

```
from rest_framework import generics
from .models import Book
from .serializers import BookSerializer

class BookListCreate(generics.ListCreateAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer

class BookRetrieve(generics.RetrieveAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
    lookup_field = "id"
```

Paso 7: Definir las URLs

``

```
from django.urls import path
from .views import BookListCreate, BookRetrieve

urlpatterns = [
    path("books/", BookListCreate.as_view()),
```

```
    path("books/<str:id>", BookRetrieve.as_view()),
]
```

``

```
from django.urls import include, path

urlpatterns = [
    path("", include("books.urls")),
]
```

Paso 8: Migraciones y Superusuario

```
python manage.py makemigrations books
python manage.py migrate --fake-initial
python manage.py createsuperuser
```

Esto te permitirá acceder al panel de administración en `http://localhost:8001/admin/`.

Paso 9: Registrar el modelo en el panel de administración

Para que el modelo `Book` sea visible en el admin, edita el archivo `` y añade:

```
from django.contrib import admin
from .models import Book

@admin.register(Book)
class BookAdmin(admin.ModelAdmin):
    list_display = ["id", "title", "author", "pages"]
```

Esto habilitará una sección "Books" en el panel de administración donde podrás consultar, crear o editar libros manualmente.

Paso 10: Arrancar el servidor

```
python manage.py runserver 0.0.0.0:8001
```

La API estará accesible desde cualquier máquina que pueda conectarse a esa IP y puerto.

Paso 11: Pruebas rápidas de la API

```
# Crear un libro
curl -X POST http://localhost:8001/books/ \
     -H "Content-Type: application/json" \
     -d '{"title":"Domain-Driven Design","author":"Eric Evans","pages":560}'

# Listar libros
curl http://localhost:8001/books/

# Obtener libro por id
curl http://localhost:8001/books/<id_hex>
```

Paso 12: Acceso desde el notebook

En tu notebook, solo tienes que apuntar a:

```
BASE_URL = "http://localhost:8001"  # Django DRF
```

Y el resto del código seguirá funcionando igual que con FastAPI o Flask.

Resumen

Ya tienes una mini API REST en Django 5 + DRF, conectada a MySQL, en el puerto 8001, compatible con tu cliente de pruebas en Python (el notebook).