

API REST con Django

REST Framework + MySQL

API functional con Django REST Framework

Python

Django

DRF

MySQL

REST API

Migraciones

AGENDA

A tener en cuenta

1

Instalación y entorno

Python, pip, virtualenv, Django y paquetes necesarios

2

Configurar MySQL

Conexión a la base de datos desde Django

3

Crear el proyecto y app

Estructura de archivos, settings y apps

4

Modelos y Migraciones

Definir tablas con ORM y ejecutar migraciones

5

Serializers

Convertir modelos a JSON con DRF

6

Views y URLs

ViewSets, Routers y endpoints REST

7

Probar la API

Endpoints CRUD con ejemplos reales

PASO 1

Instalación y Entorno Virtual

1. Crear entorno virtual (opcional)

```
python -m venv venv  
source venv/bin/activate  
# Windows:  
venv\Scripts\activate
```

2. Instalar Django y DRF

```
pip install django djangorestframework
```

3. Instalar conector MySQL

```
pip install mysqlclient  
# Alternativa:  
pip install PyMySQL
```

4. Crear proyecto Django e instancia

```
django-admin startproject mi_api  
cd mi_api  
python manage.py startapp api
```

requirements.txt

```
Django>=4.3  
djangorestframework>=3.20  
mysqlclient>=2.2
```

Siempre trabaja dentro del entorno virtual para evitar conflictos de versiones entre proyectos.

PASO 2

Configurar MySQL en Django

Crear la base de datos en MySQL:

```
CREATE DATABASE api_db CHARACTER SET utf8mb4 COLLATE  
utf8mb4_unicode_ci;  
CREATE USER 'api_user'@'localhost' IDENTIFIED BY  
'tu_password';  
GRANT ALL PRIVILEGES ON api_db.* TO  
'api_user'@'localhost';  
FLUSH PRIVILEGES;
```

Configuración en settings.py:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'api_db',  
        'USER': 'api_user',  
        'PASSWORD': 'password',  
        'HOST': 'localhost',  
        'PORT': '3306',  
    }  
}
```

Agregar DRF a INSTALLED_APPS:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    # Agregar:  
    'rest_framework',  
    'api', # la app  
]
```

✓ Buena práctica

Usar python-decouple o django-environ para guardar credenciales en un archivo .env y no exponerlas en el código.

PASO 3

Crear Proyecto y Aplicación

Comandos:

```
# Crear proyecto Django  
django-admin startproject mi_api .  
  
# Crear la app  
python manage.py startapp api
```

Estructura de archivos:

```
miapi/  
└── manage.py  
miapi/  
    ├── settings.py  
    ├── urls.py  
    └── wsgi.py  
productos/  
    ├── models.py  
    ├── serializers.py  
    ├── views.py  
    └── urls.py
```

models.py

Define las tablas de la base de datos como clases Python (ORM)

serializers.py

Convierte objetos del modelo a JSON y viceversa

views.py

Contiene la lógica de los endpoints (GET, POST, PUT, DELETE)

urls.py

Conecta las URLs con las vistas correspondientes

PASO 4

Modelos y Migraciones

api/models.py

```
from django.db import models

class Producto(models.Model):
    nombre = models.CharField(max_length=100)
    precio = models.DecimalField(
        max_digits=10, decimal_places=2
    )
    creado = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.nombre
```

Ejecutar migraciones:

```
# 1. Crear archivos de migración
python manage.py makemigrations

# 2. Aplicar a la base de datos
python manage.py migrate
```

Tipos de campo más usados

CharField	Texto corto (max_length requerido)
TextField	Texto largo sin límite
IntegerField	Número entero
DecimalField	Decimal con precisión
BooleanField	Verdadero / Falso
DateTimeField	Fecha y hora
ForeignKey	Relación muchos a uno

PASO 5

Serializers — Modelo a JSON

api/serializers.py

```
from rest_framework import serializers
from .models import Producto

class ProductoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Producto
        fields = '__all__'
```

¿Qué hace el serializer?

- ▶ Serialización: Modelo Python → JSON (para GET)
- ▶ Deserialización: JSON → Modelo Python (para POST/PUT)
- ▶ Validación automática de datos recibidos
- ▶ Manejo de relaciones entre modelos

Respuesta JSON generada:

```
[  
  {  
    "id": 1,  
    "nombre": "Laptop Pro",  
    "precio": "1299.99",  
    "activo": true,  
    "creado_en": "2024-01-15T10:30:00Z"  
  },  
  {  
    "id": 2,  
    "nombre": "Mouse Inalámbrico",  
    "precio": "45.00",  
    "creado_en": "2024-01-16T09:15:00Z"  
  }  
]
```

PASO 6

Views y URLs — Los Endpoints

api/views.py

```
from rest_framework import viewsets
from .models import Producto
from .serializers import ProductoSerializer

class ProductoViewSet(viewsets.ModelViewSet):
    queryset = Producto.objects.all()
    serializer_class = ProductoSerializer
```

api/urls.py

```
from rest_framework.routers import DefaultRouter
from django.urls import path, include
from . import views

router = DefaultRouter()
router.register(r'productos', views.ProductoViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

```
python manage.py runserver Puerto
python manage.py createsuperuser
```

miapi/urls.py

```
from django.contrib import admin
from django.urls import path, include

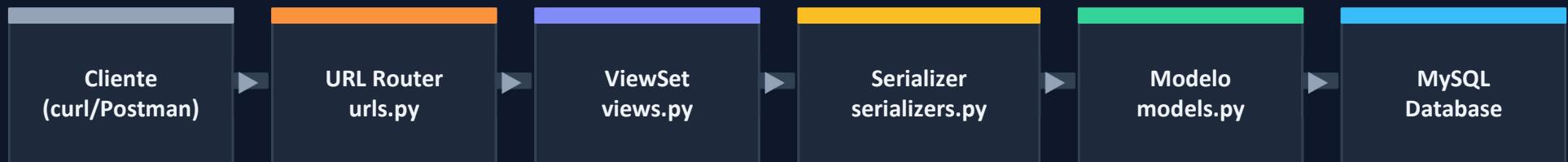
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('api.urls')),
    path('api-auth/', include('rest_framework.urls')),
]
```

Endpoints generados automáticamente:

GET	/api/productos/	Lista todos los productos
POST	/api/productos/	Crea un producto
GET	/api/productos/{id}/	Obtiene un producto
PUT	/api/productos/{id}/	Actualiza completo
PATCH	/api/productos/{id}/	Actualiza parcial
DELETE	/api/productos/{id}/	Elimina producto

RESUMEN

Flujo Completo de la API REST



Verificar

- | | |
|---|--|
| <ul style="list-style-type: none">✓ Entorno virtual activado e instalación de paquetes✓ Base de datos MySQL creada y configurada en settings.py✓ Proyecto Django y app “api” creados✓ Modelo Producto definido con campos apropiados | <ul style="list-style-type: none">✓ Migraciones ejecutadas con makemigrations + migrate✓ Serializer configurado con ModelSerializer✓ ViewSet y Router registrados en urls.py✓ Servidor corriendo y endpoints respondiendo |
|---|--|