# https://chatgpt.com/share/671ba890-e380-8011-bafe-9e62d13750b0

# Tutorial: Crear un CRUD simple con API en Django

Este tutorial te guiará a través de la creación de un CRUD (Crear, Leer, Actualizar, Eliminar) simple utilizando Django y Django REST Framework. Usaremos una base de datos MySQL a través de Workbench y un entorno virtual (venv) para la gestión del proyecto.

## 1. Configuración del Entorno

1.1 Crea un entorno virtual y actívalo:

bash

Copiar código

python -m venv venv

venv\Scripts\activate

1.2 Instala Django y Django REST Framework:

bash

Copiar código

pip install django djangorestframework mysqlclient

1.3 Crea un nuevo proyecto y una nueva aplicación:

bash

Copiar código

django-admin startproject BookCritiqueAPI

cd BookCritiqueAPI

python manage.py startapp reviews

## 2. Configuración del Proyecto

2.1 Modifica settings.py para incluir la aplicación y el REST Framework:

python

Copiar código

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'rest\_framework',

'reviews',

]

2.2 Configura la base de datos en settings.py:

python

Copiar código

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'nombre\_de\_tu\_base\_de\_datos',

'USER': 'tu\_usuario',

'PASSWORD': 'tu\_contraseña',

'HOST': 'localhost',

'PORT': '3306',

}

}

## 3. Creación del Modelo

3.1 En reviews/models.py, define el modelo Review:

python

Copiar código

from django.db import models

class Review(models.Model):

title = models.CharField(max\_length=100)

author = models.CharField(max\_length=50)

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.title

## 4. Creación del Serializador

4.1 Crea el archivo serializers.py en la carpeta reviews:

python

Copiar código

from rest\_framework import serializers

from .models import Review

class ReviewSerializer(serializers.ModelSerializer):

class Meta:

model = Review

fields = '\_\_all\_\_'

## 5. Creación de la Vista

5.1 En reviews/views.py, crea el ReviewViewSet:

python

Copiar código

from rest\_framework import viewsets

from .models import Review

from .serializers import ReviewSerializer

class ReviewViewSet(viewsets.ModelViewSet):

queryset = Review.objects.all()

serializer\_class = ReviewSerializer

## 6. Configuración de URLs

6.1 En BookCritiqueAPI/urls.py, configura las rutas:

python

Copiar código

from django.contrib import admin

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from reviews.views import ReviewViewSet

router = DefaultRouter()

router.register(r'reviews', ReviewViewSet)

urlpatterns = [

path('', include(router.urls)),

path('admin/', admin.site.urls),

]

## 7. Ejecutar Migraciones

7.1 Ejecuta las migraciones para crear las tablas:

bash

Copiar código

python manage.py makemigrations

python manage.py migrate

## 8. Probar la API

8.1 Inicia el servidor de desarrollo:

bash

Copiar código

python manage.py runserver

8.2 Prueba las siguientes operaciones:

* **GET**: Accede a http://127.0.0.1:8000/reviews/ para obtener todas las reseñas.
* **POST**: Envía una solicitud POST a http://127.0.0.1:8000/reviews/ con un cuerpo JSON para agregar una reseña.
* **PUT**: Para actualizar una reseña, usa http://127.0.0.1:8000/reviews/<id>/.
* **DELETE**: Para eliminar una reseña, usa http://127.0.0.1:8000/reviews/<id>/.