**Primer proyecto con Django Rest Framework**

Antes que nada, **creamos una carpeta** (donde se guardara el proyecto) ya la arrastramos a VSC (Visual Studio Code)

**Creamos** nuestro entorno virtual

**TERM->**python -m venv env

**Activamos** el entorno Virtual

**TERM->**.\env\Scripts\activate

*En esta parte PUEDE que tengan un error de activación del Entorno Virtual, pero se soluciona con un par de líneas en el terminal.*

*SI NO TIENE UN ERROR HASTA AQUI, NO ES NECESARIO EJECUTAR ESTAS LINEAS Y PUEDE CONTINUAR CON LA SIGUIENTE SECCION.*

TERM->Set-ExecutionPolicy Unrestricted -Force

TERM->Set-ExecutionPolicy Unrestricted -Scope Process

Mas información de lo que estamos haciendo acá podemos encontrar en la página de Microsoft:

[ENLACE](https://learn.microsoft.com/es-es/powershell/module/microsoft.powershell.security/set-executionpolicy?view=powershell-7.4)

Actualizamos PIP en el terminal

**TERM**->python.exe -m pip install --upgrade pip

Hacemos una instalacion de Django

**TERM**->pip install Django

Iniciamos el proyecto de Django (Ojo al punto es para que nos cree la carpeta/proyecto en el mismo nivel de carpetas que ENV y API), se crean los archivos básicos para el proyecto Django.

TERM->django-admin startproject drf .

Creamos la aplicación API, con los modelos (BD), vistas (paginas) y administrador.

**TERM**->django-admin startapp api

Agregamos en el archivo **settings.py** en la sección INSTALLED\_APPS una línea para indicar que hemos creado una nueva aplicación llamada API (Ojo con las comas, ya que es una lista)

**VSC**->'api'

Luego creamos nuestro modelo (BD) en el archivo **models.py**

Con esto agregamos los campos y tipos de datos al modelo. Ojo que aparte de estos campos se crea también un campo que se llama ID para tener en cuenta a la hora de ver la tabla creada.

**VSC**-> (todo este código va en el archivo y es el cuerpo de la tabla y se escribe dentro del archivo **models.py** después de la línea "# Create your models here.")

class programmer(models.Model):

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

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

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

age = models.PositiveSmallIntegerField()

is\_active = models.BooleanField(default=True)

Luego agregamos el modelo al archivo **admin.py** para dejar que nuestro modelo y nuestra tabla "programmer" sea importada y poder ver el modelo desde nuestro panel de administración.

**VSC**->from .models import programmer

También registramos el modelo "programmer" para que sea visible desde nuestro panel de administración dentro del archivo **admin.py**, después de la línea "# Register your models here.".

**VSC**--> admin.site.register(programmer)

Esto es OPCIONAL, ejecutamos la lista de programas instalados en el ENV para verificar que todo este OK y tengamos los programas necesarios instalados.

**TERM**->pip list

Ejecutamos las migraciones del modelo a la BD en este caso SQLite3 por defecto (se puede cambiar por Mysql, Oracle,etc.).

**TERM**->python manage.py migrate #acà creamos la BD

**TERM**->python manage.py makemigrations # se crea la migración para el modelo PROGRAMMERS

**TERM**->python manage.py migrate # se aplica la migración dentro de la BD.

Creamos el super usuario en el terminal, con esto ya podremos luego ingresar al administrador que estamos creando.

Usted cambie su usuario por su nombre (Utilice la misma contraseña de este ejemplo para que vea el WARNING y entienda que hay que usar una más fuerte en ambiente de producción).

**TERM**->python manage.py createsuperuser

user=profe

pass=123456

Acá debemos confirmar con ‘Y’ la creación del usuario.

Levantamos el server con el siguiente comando:

**TERM**-> python manage.py runserver

Listo con esto ya tenemos nuestro SERVER corriendo Django con su proyecto API y su BD en SQLite3.

Ahora podemos ir a nuestro navegador y escribir la ruta que nos da (http://127.0.0.1:8000/) para probar que estemos visibles y tenemos acceso al modelo programador que creamos y vamos bien hasta ahora.

Luego, detenemos el servidor desde el terminal con las teclas CONTROL+C para continuar con DJANGO REST FRAMEWORK.

Ahora instalamos DjangoRestFramework en nuestro terminal

**TERM**->pip install djangorestframework

Actualizamos el archivo **settings.py** y agregamos una línea en las INSTALLED\_APPS (Ojo con las comas, ya que es una lista)

**VSC**->'rest\_framework',

Ahora vamos a proveer los datos a la API, para eso vamos a serializar los datos para crear un JSON y tener acceso a los datos en formato JSON

Creamos en la carpeta API un archivo llamado **serializer.py** (según la documentación de DJANGO en su página web).

**VSC**->

from rest\_framework import serializers

from .models import programmer

class ProgrammerSerializer(serializers.ModelSerializer):

class Meta:

model = programmer

# fields = ('fullname','languaje','is\_active’) acá podemos traer cualquier atributo del modelo o campo

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

# con la opción de '\_\_all\_\_' nos traemos todo para ver y tener acceso a todo el registro de cada programador

Con esta clase ya tenemos las vistas del Modelo necesarias para realizar los CRUD.

Ahora vamos a modificar el archivo **views.py** de la carpeta API.

Ingresamos el siguiente código.

VSC->

# from django.shortcuts import render <-- esta libreria no la usamos por ahora

from rest\_framework import viewsets

from .serializer import ProgrammerSerializer

from .models import programmer

# Create your views here.

class ProgrammerViewSet(viewsets.ModelViewSet):

# acá creamos una consulta o QUERY a nuestra tabla, trayendo todos los campos como un objeto.

queryset = programmer.objects.all()

# Agregamos la clase ProgrammerSerializer que ya tiene el modelo serializado para mostrar

serializer\_class = ProgrammerSerializer

Creamos otro archivo dentro de API con el nombre **urls.py** y ponemos este código.

**VSC**->

from django.urls import path, include

from rest\_framework import routers

from api import views

router = routers.DefaultRouter() # este elemento enrutador permite manejar múltiples rutas.

# esta es la base del conjunto de rutas o la raíz de las rutas

# acá se manejan las rutas o ENDsPOINTS que pueda tener tu API

router.register(r'programmers', views.ProgrammerViewSet)

# la r permite que no se interprete como un salto de línea o como un escape de carácter

# usamos la r para indicar que no tome los caracteres como \n o \t que es un salto de línea o una tabulación, es un formato tipo RAW de python.

# 'programmers' es un ENDPOINT

urlpatterns = [

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

# la ruta base va a incluir todos los elementos que tenga el router que hemos creado en URLS

# esta es la lista de URLS que maneja ROUTER en sus elementos URLS

]

Ahora incluimos o agregamos esta URLS en las vistas del proyecto (**url.py** de la carpeta drf)

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('api/', include('api.urls'))

]

Con esto seria todo en cuanto a configuración de nuestra API en Django Rest Framework

Ahora podemos probar todo en las siguientes rutas:

<http://127.0.0.1:8000/>

<http://127.0.0.1:8000/api/>

<http://127.0.0.1:8000/admin/>

Los dejo solo con esto mientras para que se puedan familiarizar con su primera API en DjangoRestFramework.

Revisen la documentación directa del sitio en <https://www.django-rest-framework.org/>