**Creando una API con DRF y JWT**

API RESTful usando JWT para la autenticación en Django Rest Framework (DRF) y que soporte un CRUD completo para el modelo “Tarea”:

**1. Configuración del Proyecto**

Primero, asegúrate de tener Django y Django Rest Framework instalados:

Terminal:

pip install django djangorestframework djangorestframework-simplejwt

Luego, crea un nuevo proyecto y una nueva aplicación en Django si aún no lo has hecho:

Terminal:

django-admin startproject gestor\_tareas

cd gestor\_tareas

python manage.py startapp tareas

**2. Configuración de JWT**

Agrega `rest\_framework\_simplejwt` a tus aplicaciones instaladas en el archivo `settings.py`:

Codigo python:

# gestor\_tareas/settings.py

INSTALLED\_APPS = [

...

'rest\_framework',

'tareas',

'rest\_framework\_simplejwt',

]

REST\_FRAMEWORK = {

'DEFAULT\_AUTHENTICATION\_CLASSES': (

'rest\_framework\_simplejwt.authentication.JWTAuthentication',

),

}

Configura los tokens JWT en el mismo archivo:

Codigo python:

# gestor\_tareas/settings.py

from datetime import timedelta

SIMPLE\_JWT = {

'ACCESS\_TOKEN\_LIFETIME': timedelta(minutes=5),

'REFRESH\_TOKEN\_LIFETIME': timedelta(days=1),

'ROTATE\_REFRESH\_TOKENS': True,

'BLACKLIST\_AFTER\_ROTATION': True,

}

**3. Definir el Modelo**

Ya tenemos el modelo `Tarea` en `tareas/models.py`. Aquí está para referencia:

Codigo python:

# tareas/models.py

from django.db import models

from django.contrib.auth.models import User

class Tarea(models.Model):

nombre = models.CharField(max\_length=100, verbose\_name="Nombre de la Tarea")

completada = models.BooleanField(default=False, verbose\_name="¿Completada?")

usuario = models.ForeignKey(User, on\_delete=models.CASCADE, verbose\_name="Usuario")

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

return self.nombre

**4. Crear los Serializers**

Crea un archivo `serializers.py` en la aplicación `tareas` y define un serializer para el modelo `Tarea`.

Codigo python:

# tareas/serializers.py

from rest\_framework import serializers

from .models import Tarea

class TareaSerializer(serializers.ModelSerializer):

class Meta:

model = Tarea

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

**5. Crear las Vistas**

Crea un archivo `views.py` en la aplicación `tareas` y define las vistas para el CRUD.

Codigo python:

# tareas/views.py

from rest\_framework import generics

from rest\_framework.permissions import IsAuthenticated

from .models import Tarea

from .serializers import TareaSerializer

class TareaListCreateView(generics.ListCreateAPIView):

queryset = Tarea.objects.all()

serializer\_class = TareaSerializer

permission\_classes = [IsAuthenticated]

def get\_queryset(self):

return Tarea.objects.filter(usuario=self.request.user)

class TareaRetrieveUpdateDestroyView(generics.RetrieveUpdateDestroyAPIView):

queryset = Tarea.objects.all()

serializer\_class = TareaSerializer

permission\_classes = [IsAuthenticated]

def get\_queryset(self):

return Tarea.objects.filter(usuario=self.request.user)

**6. Configurar las URLs**

Crea un archivo `urls.py` en la aplicación `tareas` y define las rutas para las vistas.

Codigo python:

# tareas/urls.py

from django.urls import path

from .views import TareaListCreateView, TareaRetrieveUpdateDestroyView

urlpatterns = [

path('tareas/', TareaListCreateView.as\_view(), name='tarea-list-create'),

path('tareas/<int:pk>/', TareaRetrieveUpdateDestroyView.as\_view(), name='tarea-retrieve-update-destroy'),

]

Luego, incluye las rutas de `tareas` en el archivo `urls.py` del proyecto principal:

Codigo python:

# gestor\_tareas/urls.py

from django.contrib import admin

from django.urls import path, include

from rest\_framework\_simplejwt import views as jwt\_views

urlpatterns = [

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

path('api/token/', jwt\_views.TokenObtainPairView.as\_view(), name='token\_obtain\_pair'),

path('api/token/refresh/', jwt\_views.TokenRefreshView.as\_view(), name='token\_refresh'),

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

]

**7. Migraciones y Creación de Usuario**

Aplica las migraciones para crear las tablas en la base de datos:

Terminal:

python manage.py makemigrations

python manage.py migrate

Crea un superusuario para poder probar la autenticación:

Terminal:

python manage.py createsuperuser

**8. Probar la API**

**1. Obtener Tokens JWT:** Envía una solicitud POST a `/api/token/` con las credenciales del usuario (username y password) para obtener un `access` token y un `refresh` token.

**2. Usar el Token JWT:** Usa el `access` token para autenticar las solicitudes a las vistas de la API. Incluye el token en el encabezado `Authorization` con el formato `Bearer <token>`.

**Ejemplo de solicitud para obtener el token:**

Terminal:

curl -X POST http://localhost:8000/api/token/ -d '{"username": "admin", "password": "admin"}' -H "Content-Type: application/json"

**Ejemplo de solicitud para crear una tarea:**

Terminal:

curl -X POST http://localhost:8000/api/tareas/ -H "Authorization: Bearer <access\_token>" -H "Content-Type: application/json" -d '{"nombre": "Tarea 1", "completada": false}'

**Resumen**

Una API RESTful en Django Rest Framework que utiliza JWT para la autenticación. La API soporta operaciones CRUD completas para el modelo `Tarea`, y todas las respuestas están en formato JSON. Las vistas están protegidas con autenticación JWT, asegurando que solo los usuarios autenticados puedan acceder a las funcionalidades de la API.