**Documentación del Proyecto Ticket System**

**Autor:**

Julian David Hernandez Quintero

Andres Mauricio Cano

**Fecha: 7/09/2024**

[Fecha Actual]

**Índice**

1. Introducción
2. Configuración del Entorno
3. Creación del Proyecto
4. Desarrollo
   * Modelos
   * Serializers
   * Vistas
   * URLs
5. Modificaciones y Problemas
6. Resultados
7. Conclusiones

**Introducción**

El proyecto "Ticket System" es una aplicación web diseñada para gestionar tickets y comentarios asociados. Utiliza Django y Django REST Framework para proporcionar una API que permite la creación, visualización, actualización y eliminación de tickets y comentarios.

**Configuración del Entorno**

**Instalación de Dependencias**

1. **Python 3.12**
2. **Django**

bash

Copiar código

pip install django

1. **Django REST Framework**

bash

Copiar código

pip install djangorestframework

**Configuración de la Base de Datos**

* Usar SQLite (configurado por defecto en settings.py).

**Configuración del Proyecto**

1. Crear el proyecto Django:

bash

Copiar código

django-admin startproject TicketSystem

cd TicketSystem

1. Crear la aplicación tickets:

bash

Copiar código

python manage.py startapp tickets

**Creación del Proyecto**

**Creación de la Aplicación tickets**

1. Crear la aplicación tickets:

bash

Copiar código

python manage.py startapp tickets

1. Configuración de settings.py:
   * Agregar 'tickets' a INSTALLED\_APPS.
2. Creación de Modelos en tickets/models.py:

python

Copiar código

from django.db import models

class Ticket(models.Model):

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

description = models.TextField()

class Comment(models.Model):

ticket = models.ForeignKey(Ticket, related\_name='comments', on\_delete=models.CASCADE)

content = models.TextField()

**Desarrollo**

**Modelos**

1. **Modelo Ticket**

python

Copiar código

from django.db import models

class Ticket(models.Model):

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

description = models.TextField()

1. **Modelo Comment**

python

Copiar código

class Comment(models.Model):

ticket = models.ForeignKey(Ticket, related\_name='comments', on\_delete=models.CASCADE)

content = models.TextField()

**Serializers**

1. **Serializador TicketSerializer**

python

Copiar código

from rest\_framework import serializers

from .models import Ticket

class TicketSerializer(serializers.ModelSerializer):

class Meta:

model = Ticket

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

1. **Serializador CommentSerializer**

python

Copiar código

from .models import Comment

class CommentSerializer(serializers.ModelSerializer):

class Meta:

model = Comment

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

**Vistas**

1. **Vistas de Ticket**

python

Copiar código

from rest\_framework import viewsets

from .models import Ticket

from .serializers import TicketSerializer

class TicketViewSet(viewsets.ModelViewSet):

queryset = Ticket.objects.all()

serializer\_class = TicketSerializer

1. **Vistas de Comment**

python

Copiar código

from .models import Comment

from .serializers import CommentSerializer

class CommentViewSet(viewsets.ModelViewSet):

queryset = Comment.objects.all()

serializer\_class = CommentSerializer

**URLs**

1. **Archivo urls.py en tickets**

python

Copiar código

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import TicketViewSet, CommentViewSet

router = DefaultRouter()

router.register(r'tickets', TicketViewSet)

router.register(r'comments', CommentViewSet)

urlpatterns = [

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

]

1. **Archivo urls.py en TicketSystem**

python

Copiar código

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('api/', include('tickets.urls')), # Incluye las URLs de la aplicación tickets bajo el prefijo 'api/'

]

**Modificaciones y Problemas**

**Error al Cargar el Servidor**

* **Problema:** Error de conexión ERR\_CONNECTION\_REFUSED.
* **Solución:** Verificar la configuración del servidor y asegurarse de que el puerto esté disponible.

**Error en urls.py**

* **Problema:** NameError: name 'include' is not defined.
* **Solución:** Asegurarse de que el módulo include esté importado correctamente.

**Resultados**

**Página Principal**

* Captura de pantalla de la interfaz web.

**Solicitud a la API**

* Ejemplo de solicitud GET a /api/tickets/ y la respuesta:

bash

Copiar código

curl http://127.0.0.1:8000/api/tickets/

**Conclusiones**

Este proyecto proporcionó una experiencia práctica valiosa en la creación de una API utilizando Django y Django REST Framework. Se enfrentaron desafíos y se resolvieron problemas relacionados con la configuración y la implementación de funcionalidades. El proyecto se completó con éxito y se logró una comprensión más profunda de la creación de APIs en Django.