KEEPCODING
Tech School

mini guía API REST FRAMEWORK

Crear proyecto

1. Crear carpeta contenedor
2. Crear entorno virtual

```
python -m venv <path_nombre entorno virtual>
```

3. Activar entorno virtual

```
linux/mac
      source <path_nombre entorno virtual>/bin/activate
win
      <path_nombre entorno virtual>\Scripts\activate
```

4. Instalar dependencias

```
pip install django
```

5. Crear aplicación

```
django-admin startproject <nombre_proyecto>
```

6. Desactivar el entorno virtual y borrar la carpeta <path_nombre entorno virtual>

```
deactivate
```

Abrir proyecto con VSC

1. Abrir carpeta (File > Open Folder> <nombre_proyecto>
2. Recrear el entorno virtual, pasos 2, 3 y 4 de Crear proyecto
3. Lanzar las migraciones

```
python manage.py migrate
```

4. Crear el superusuario


```
python manage.py createsuperuser
```

5. Lanzar el servidor

```
python manage.py runserver
```

Crear app para la API

1. Crear nueva aplicación

```
python manage.py startapp <nombre_app>
```

2. instalar app en el fichero <nombre_proyecto>/settings

```
INSTALLED_APPS = {
   …
   ‘nombre_app’,
   …
}
```

3. crear modelo <Model> in `<nombre_app>/models.py`

```
from django.db import models

# Create your models here.
class Entry(models.Model):
    datetime = models.DateTimeField()
    concept = models.CharField(max_length=30)
    amount = models.FloatField()
```

4. Integrar users en panel de administración in `<nombre_app>/admin.py`

```
from django.contrib import admin
from entries.models import Entry

admin.site.register(Entry)
```

5. Crear migraciones y migrar


```
python manage.py makemigrations
python manage.py migrate
```

Instalar django-rest-framework

1. Instalar con pip

```
pip install djangorestframework
```

2. Registrar app en `<nombre_proyecto>/settings` a ser posible delante de la anterior

```
INSTALLED_APPS = {
   …
   “rest_framework”,
   …
}
```

Crear endpoints - GET - POST

1. Crear los serializers (equivalente a los formularios)
    a. Crearlo en carpeta API
    b. Elegir los campos que mostrar/recibir - Serializer

```
from dataclasses import dataclass
from rest_framework import serializers
from entries.models import Entry

class EntrySerializer(serializers.Serializer):
    id = serializers.ReadOnlyField()
    concept = serializers.CharField()
    amount = serializers.FloatField()
    datetime = serializers.DateTimeField()
```

    c. métodos - Update (PUT)
2. Crear APIView ListaEntidades
    a. GET lista


```
from rest_framework.views import APIView
from rest_framework.response import Response

from entries.models import Entry
from entries.api.serializers import EntrySerializer

class EntryListAPI(APIView):
    def get(self, request):
        entries = Entry.objects.all()

        serializer = EntrySerializer(entries, many=True)

        return Response(serializer.data)
```

b. Asociar url en <nombre_proyecto>/urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/v1/entries/', EntryListAPI.as_view(), name="entry_list_api")
]
```


3. Añadir a APIview el método de creación
    a. POST crear
    b. Modificar serializer para que grabe

```
from dataclasses import dataclass
from rest_framework import serializers
from entries.models import Entry

class EntrySerializer(serializers.Serializer):
    id = serializers.ReadOnlyField()
    concept = serializers.CharField()
    amount = serializers.FloatField()
    datetime = serializers.DateTimeField()

    def create(self, validated_data):
        # Voy a salvar a validated_data
        instance = Entry(
            datetime=validated_data.get("datetime"),
            concept=validated_data.get("concept"),
            amount=validated_data.get("amount")
        )

        instance.save()
        return instance
```

Crear endpoint detalle entidad - PUT DELETE

1. Crear una nueva APIView que recibirá el id del modelo a modificar/borrar
    a. usar shortcut de django `get_object_or_404`

```
…
from django.shortcuts import get_object_or_404
…
class EntryDetailAPI(APIView):
    def put(self, request, pk):
        #entry = Entry.objects.get(pk=pk)
        entry = get_object_or_404(Entry)
        serializer = EntrySerializer(instance=entry, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(data=serializer.data, status=200)

        return Response(status=400, data=serializer.errors)
```


2. Añadir método update en serializer

```
…
def update(self, instance, validated_data):
        # Voy a modificar instancia con validated data
        instance.datetime = validated_data.get("datetime")
        instance.concept = validated_data.get("concept")
        instance.amount = validated_data.get("amount")

        instance.save()
        return instance
```

3. Crear url en <nombre_proyecto>/urls.py

```
    path('api/v1/entries/<int:pk>', EntryDetailAPI.as_view(),
name="entry_detail_api"),
```

