#### Creación de una API

Una API (Interfaz de Programación de Aplicaciones) construida con Django utilizando **Django REST Framework (DRF)** se caracteriza por lo siguiente:

- 1. **Rápida integración**: DRF facilita la creación de API RESTful al ofrecer herramientas predefinidas como vistas genéricas, serializadores y autenticación.
- Serialización de datos: Utiliza serializadores para convertir datos entre formatos, como JSON o XML, y las estructuras internas de Python (como diccionarios y objetos).
   Esto permite enviar y recibir datos entre el servidor y el cliente de forma eficiente.
- 3. **Vistas basadas en clases y funciones**: Permite la creación de **vistas** usando clases o funciones para manejar las solicitudes HTTP (GET, POST, PUT, DELETE). Las vistas genéricas proporcionan un enfoque estándar para operaciones CRUD.
- Autenticación y permisos: Ofrece soporte para diversos métodos de autenticación (como token, JWT, OAuth) y permisos (para controlar el acceso a recursos), mejorando la seguridad de la API.
- 5. **Navegación**: DRF incluye una interfaz web de navegación que facilita la prueba de la API a través de un navegador, lo que mejora la experiencia de desarrollo.
- 6. **Paginación**: Proporciona **paginación** de resultados para evitar la sobrecarga al retornar grandes cantidades de datos.
- 7. **Filtrado y búsqueda**: Permite filtrar y buscar datos en los modelos, ayudando a personalizar las consultas en la API.
- 8. **Soporte de versionado**: Puedes gestionar diferentes versiones de tu API, permitiendo que distintas versiones de la misma API convivan y se mantengan durante un tiempo.

# ¿Cómo funciona una API?



Estas características hacen que Django REST Framework sea una excelente opción para construir APIs escalables, seguras y fáciles de mantener.

https://www.django-rest-framework.org/

Para crear una API, se genera la estructura del proyecto como ya es sabido.

- 1. Creación del entorno virtual y activación del mismo
- 2. Instalación de las dependencias del archivo requirements.txt

```
asgiref==3.8.1

Django==5.1.2

django-allauth==65.1.0

django-filter==24.3

django-model-utils==5.0.0

django-rest-auth==0.9.5

djangorestframework==3.15.2

Markdown==3.7

six==1.16.0

sqlparse==0.5.1

tzdata==2024.2

requests==2.32.3
```

- 3. Creación del proyecto
  - a. django-admin startproject proyecto\_api
  - b. Cambiarse al directorio del proyecto
  - c. Creación de la app
    - i. django-admin startapp app api
- 4. Configuración del archivo settings.py

```
INSTALLED_APPS = [
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'rest_framework',
   'rest_framework.authtoken',
   'app_api',
]
```

5. El siguiente apartado indica al archivo la configuración de la autenticación que usará django rest framework para consumir la API

```
REST_FRAMEWORK = {
'DEFAULT_PERMISSION_CLASES':[
'rest_framework.permissions.IsAuthenticated',
],
'DEFAULT_AUTHENTICATION_CLASSES': [
'rest_framework.authentication.TokenAuthentication',
'rest_framework.authentication.BasicAuthentication',
'rest_framework.authentication.SessionAuthentication',
], }
```

```
TEMPLATES = [
  {
     'BACKEND': 'django.template.backends.django.DjangoTemplates',
     'DIRS': ['templates'],
     'APP DIRS': True,
     'OPTIONS': {
       'context_processors': [
         'django.template.context_processors.debug',
         'django.template.context processors.request',
         'django.contrib.auth.context processors.auth',
         'django.contrib.messages.context processors.messages',
       ],
     },
  },
1
LOGIN URL = "login/"
   6. Creación del modelo en la app (models.py)
       from django.db import models
       from model utils.models import TimeStampedModel, SoftDeletableModel
       # Create your models here.
       class Categoria(SoftDeletableModel,models.Model):
              cod categoria = models.IntegerField(primary key=True)
              descripcion_categoria = models.CharField(max_length=50, null=False,
       blank=False)
              def str (self):
                     return self.descripcion_categoria
       class Producto(SoftDeletableModel,models.Model):
              codigo = models.IntegerField(primary key=True)
              descripcion = models.CharField(max_length=100, null=False, blank=False)
              precio = models.IntegerField(null=False, blank=False)
              stock = models.IntegerField(null=False, blank=False)
              categoria = models.ForeignKey(Categoria, on_delete=models.CASCADE)
         def str (self):
              return f"{self.codigo} {self.descripcion}"
```

## 7. Registro de los modelos (admin.py)

from django.contrib import admin from app\_api.models import Producto, Categoria # Register your models here.

admin.site.register(Producto) admin.site.register(Categoria)

## 8. Crear migraciones

C:\Users\HP\proyectos\proyecto\_api>python manage.py makemigrations Migrations for 'app api':

app\_api\migrations\0001\_initial.py

- + Create model Categoria
- + Create model Producto

## 9. Migrar a la BD

C:\Users\HP\proyectos\proyecto\_api>python manage.py makemigrations Migrations for 'app\_api':

app\_api\migrations\0001\_initial.py

- + Create model Categoria
- + Create model Producto

C:\Users\HP\proyectos\proyecto\_api>python manage.py migrate

Operations to perform:

Apply all migrations: admin, app\_api, auth, authtoken, contenttypes, sessions Running migrations:

Applying contenttypes.0001\_initial... OK

Applying auth.0001 initial... OK

Applying admin.0001\_initial... OK

Applying admin.0002 logentry remove auto add... OK

Applying admin.0003\_logentry\_add\_action\_flag\_choices... OK

Applying app api.0001 initial... OK

Applying contenttypes.0002\_remove\_content\_type\_name... OK

Applying auth.0002 alter permission name max length... OK

Applying auth.0003 alter user email max length... OK

Applying auth.0004\_alter\_user\_username\_opts... OK

Applying auth.0005 alter user last login null... OK

Applying auth.0006\_require\_contenttypes\_0002... OK

Applying auth.0007 alter validators add error messages... OK

Applying auth.0008\_alter\_user\_username\_max\_length... OK

Applying auth.0009 alter user last name max length... OK

Applying auth.0010 alter group name max length... OK

Applying auth.0011\_update\_proxy\_permissions... OK

Applying auth.0012 alter user first name max length... OK

Applying authtoken.0001\_initial... OK

Applying authtoken.0002 auto 20160226 1747... OK

```
Applying authtoken.0003_tokenproxy... OK
Applying authtoken.0004_alter_tokenproxy_options... OK
Applying sessions.0001 initial... OK
```

#### 10. Crear el super usuario

C:\Users\HP\proyectos\proyecto\_api>python manage.py createsuperuser

# Username (leave blank to use 'hp'): admin

Email address:

Password:

Password (again):

This password is too short. It must contain at least 8 characters.

This password is too common.

This password is entirely numeric.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

## 11. Dentro de la aplicación crear el archivo serializers.py

```
from rest_framework import serializers
from app_api.models import Producto, Categoria

class ProductoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Producto
        #fields = "__all__"
        fields = ["codigo","descripcion","precio","stock", "categoria"]

class CategoriaSerializer(serializers.ModelSerializer):
    class Meta:
        model = Categoria
        fields = ["cod_categoria","descripcion_categoria"]
```

## 12. Crear las vistas

from django.contrib.auth.decorators import login\_required

from django.contrib.auth.models import User

from django.shortcuts import render,loader,redirect

from django.contrib.auth.forms import AuthenticationForm,authenticate

from django.contrib.auth import login,logout

from django.http import HttpResponse

from rest framework.authentication import SessionAuthentication,

BasicAuthentication

from rest framework.permissions import IsAuthenticated

from rest\_framework.response import Response

from rest framework.views import APIView

from app\_api.serializers import ProductoSerializer, CategoriaSerializer

from app api.models import Producto, Categoria

```
from rest framework import status
from django.http import Http404
from rest framework import viewsets
from rest_framework.authtoken.views import ObtainAuthToken
from rest framework.authtoken.models import Token
# Create your views here.
@login required
def index(request):
       return render (request, "index.html",{})
def loginPage(request):
       template = loader.get template('login.html')
       context = {'form':AuthenticationForm}
       if request.method == "GET":
              return HttpResponse(template.render(context, request))
       else:
              usuario = request.POST["username"]
              clave = request.POST["password"]
              user = authenticate( request, username=usuario, password=clave)
              if user is None:
                      context["error"] = "Usuario o contraseña incorrectos"
                      return HttpResponse(template.render(context, request))
              else:
                      login(request, user)
                      template = loader.get template('index.html')
                      return HttpResponse(template.render(context, request))
def logout (request):
       logout(request)
       template = loader.get template('index.html')
       context = {}
       return HttpResponse(template.render(context, request))
###########
class Producto APIView(APIView):
       permission classes = [IsAuthenticated]
       def get(self, request, format=None, *args, **kwargs):
              producto = Producto.objects.all()
              serializer = ProductoSerializer(producto, many=True)
              return Response(serializer.data)
       def post(self, request, format=None):
              serializer = ProductoSerializer(data=request.data)
              if serializer.is valid():
```

```
serializer.save()
                    return Response(serializer.data, status=status.HTTP 201 CREATED)
             else:
             return Response(serializer.errors,
      status=status.HTTP 400 BAD REQUEST)
      class Producto APIView Detail(APIView):
             def get_objetc(self, pk):
                    try:
                           return Producto.objects.get(pk=pk)
                    except Producto.DoesNotExist:
                           raise Http404
             def get(self, request, pk, format=None):
                    producto = self.get objetc(pk)
                    serializer = ProductoSerializer(producto)
                    return Response(serializer.data)
             def put(self, request, pk, Format=None):
                    producto = self.get_objetc(pk)
                    serializer = ProductoSerializer(producto, data=request.data)
                    if serializer.is valid():
                           serializer.save()
                           return Response(serializer.data,
      status=status.HTTP 201 CREATED)
                    else:
                           return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
             def delete(self, request, pk, format=None):
                    producto = self.get objetc(pk)
                    producto.delete()
                    return Response(status=status.HTTP_204_NO_CONTENT)
      class Categoria APIView(APIView):
             permission_classes = [IsAuthenticated]
             def get(self, request, format=None, *args, **kwargs):
                    categoria = Categoria.objects.all()
                    serializer = CategoriaSerializer(categoria, many=True)
                    return Response(serializer.data)
             def post(self, request, format=None):
                    serializer = CategoriaSerializer(data=request.data)
                    if serializer.is_valid():
                    serializer.save()
                    return Response(serializer.data, status=status.HTTP 201 CREATED)
                    else:
```

```
return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)
class Categoria APIView Detail(APIView):
       def get objetc(self, pk):
       try:
              return Categoria.objects.get(pk=pk)
       except Categoria.DoesNotExist:
              raise Http404
       def get(self, request, pk, format=None):
              categoria = self.get objetc(pk)
              serializer = CategoriaSerializer(categoria)
              return Response(serializer.data)
       def put(self, request, pk, Format=None):
              categoria = self.get objetc(pk)
              serializer = CategoriaSerializer(producto, data=request.data)
              if serializer.is valid():
                     serializer.save()
                     return Response(serializer.data,
status=status.HTTP 201 CREATED)
              else:
                     return Response(serializer.errors,
status=status.HTTP 400 BAD REQUEST)
       def delete(self, request, pk, format=None):
              categoria = self.get_objetc(pk)
              categoria.delete()
              return Response(status=status.HTTP_204_NO_CONTENT)
class LoginAuthToken(ObtainAuthToken):
       def post(self, request, *args, **kwargs):
              serializer = self.serializer_class(data=request.data,
context={'request':request})
              serializer.is valid(raise exception=True)
              user = serializer.validated data['user']
              token, created = Token.objects.get_or_create(user=user)
              return Response({'token':token.key,'user id':user.pk,
       'email':user.email}
       )
```

## 13. Archivo urls.py

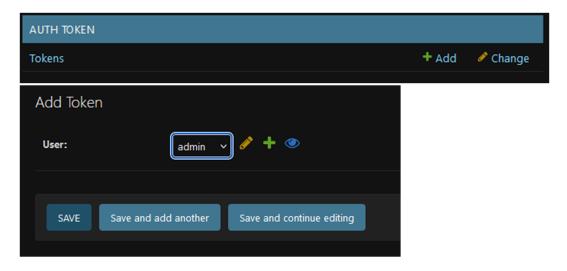
```
from django.contrib import admin
           from django.urls import path, include
           from app api.views import index, loginPage, logout
           from app api.views import LoginAuthToken
           ## faltan las urls de la api
           urlpatterns = [
             path('admin/', admin.site.urls),
             path(",index),
             path('login/',loginPage),
             path('logout/',logout_),
             path(", include("app_api.urls")),
             path('api-token-auth/', LoginAuthToken.as view()),
14. Creacion de urls en app api (urls.py)
           from django.urls import path
           from app_api.views import *
           app name='app api'
           urlpatterns = [
             path('api/producto/', Producto_APIView.as_view()),
             path('api/producto/<int:pk>/',Producto APIView Detail.as view()),
             path('api/categoria/', Categoria APIView.as view()),
             path('api/categoria/<int:pk>/',Categoria APIView Detail.as view()),
          ]
15. Creación del directorio templates y sus correspondientes archivos.
   index.html
```

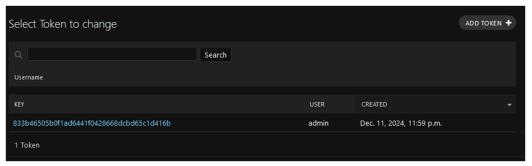
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <a href="/api/producto">Ir a Productos</a>
</body>
</html>
```

# login.html

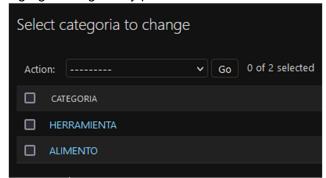
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <form action="" method="POST">
    {% csrf_token %}
    {{error}}
    {{form}}
    <input type="Submit" name="enviar" id="enviar" Value="Enviar">
  </form>
</body>
</html>
```

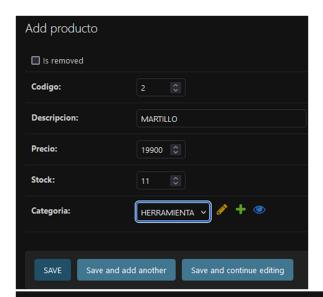
- 16. Crear un token para el usuario en la BD
  - a. Entrar al admin de django

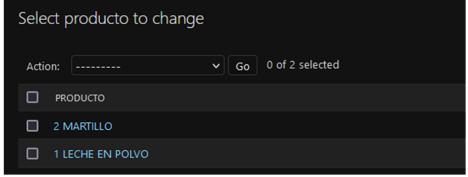




17. Agregar categorías y productos antes de realizar el acceso a la API







```
18. Archivo tests.py (copiar el token creado en el paso anterior para probar)
      from django.test import TestCase
      # Create your tests here.
      import requests
      # El token dependerá del usuario creado en el momento en la BD
      headers = {'Authorization':'Token
      833b46505b0f1ad6441f0428668dcbd65c1d416b'}
      def getData():
         url = "http://127.0.0.1:8000/api/producto/"
         datos = requests.get(url, headers=headers)
         print(datos)
        elementos = datos.json()
         return elementos
      for x in getData():
         print(x)
   19. Realizar la prueba
      El proyecto debe estar corriendo
 System check identified no issues (0 silenced).
 December 11, 2024 - 21:08:59
 Django version 5.1.2, using settings 'proyecto_api.settings'
 Starting development server at http://127.0.0.1:8000/
 Quit the server with CTRL-BREAK.
En otra ventana:
      python manage.py test app_api
      C:\Users\HP\proyectos\proyecto_api>python manage.py test app_api
      <Response [200]>
      {'codigo': 1, 'descripcion': 'LECHE EN POLVO', 'precio': 990, 'stock': 11,
      'categoria': 1}
      {'codigo': 2, 'descripcion': 'MARTILLO', 'precio': 19900, 'stock': 11, 'categoria':
      {'codigo': 10, 'descripcion': 'POROTOS', 'precio': 1990, 'stock': 150, 'categoria':
```

1}

Found 0 test(s).

System check identified no issues (0 silenced).

El resultado en el servidor al realizar la consulta usando requests.

```
System check identified no issues (0 silenced).

December 11, 2024 - 21:13:57

Django version 5.1.2, using settings 'proyecto_api.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

[11/Dec/2024 21:14:02] "GET /api/producto/ HTTP/1.1" 200 239
```