

Perfecto, vamos a armar un **proyecto Django sencillo** para que tengas un ejemplo práctico. Será un mini-blog donde puedes publicar y listar artículos.

Estructura del proyecto

Supongamos que ya tienes tu entorno virtual activo y Django instalado.

1. **Crear el proyecto base:**
 2. `django-admin startproject mi_blog .`
 3. **Crear una aplicación dentro del proyecto:**
 4. `python manage.py startapp blog`
 5. **Registrar la app en `settings.py`:** Abre `mi_blog/settings.py` y en `INSTALLED_APPS` agrega:

```
6. INSTALLED_APPS = [  
7.     'django.contrib.admin',  
8.     'django.contrib.auth',  
9.     'django.contrib.contenttypes',  
10.    'django.contrib.sessions',  
11.    'django.contrib.messages',  
12.    'django.contrib.staticfiles',  
13.    'blog',    # nuestra app  
14. ]
```
-

Modelo de datos

En `blog/models.py` define un modelo sencillo para artículos:

```
from django.db import models  
  
class Artículo(models.Model):  
    titulo = models.CharField(max_length=200)  
    contenido = models.TextField()  
    fecha_publicacion = models.DateTimeField(auto_now_add=True)  
  
    def __str__(self):  
        return self.titulo
```

Aplica migraciones:

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

Admin de Django

En `blog/admin.py` registra el modelo para gestionarlo desde el panel admin:

```
from django.contrib import admin
from .models import Artículo

admin.site.register(Articulo)
```

Crea un superusuario:

```
python manage.py createsuperuser
```

Vistas y URLs

En `blog/views.py` crea una vista para listar artículos:

```
from django.shortcuts import render
from .models import Artículo

def lista_articulos(request):
    articulos = Artículo.objects.all()
    return render(request, 'blog/lista.html', {'articulos': articulos})
```

En `blog/urls.py` (crea este archivo):

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.lista_articulos, name='lista_articulos'),
]
```

En `mi_blog/urls.py` incluye las rutas de la app:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')),
]
```

Plantilla HTML

Crea `blog/templates/blog/lista.html`:

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Mi Blog</title>
</head>
<body>
  <h1>Artículos</h1>
  <ul>
    {% for articulo in articulos %}
      <li><strong>{{ articulo.titulo }}</strong> - {{
articulo.fecha_publicacion }}</li>
      <p>{{ articulo.contenido }}</p>
    {% endfor %}
  </ul>
</body>
</html>
```

Ejecutar el servidor

```
python manage.py runserver
```

Abre en tu navegador: **http://127.0.0.1:8000/**
Verás la lista de artículos que vayas creando desde el panel admin.

☒ Con esto tienes un **mini-blog funcional**:

- Panel admin para crear artículos.
- Página principal que lista todo lo publicado.

¿Quieres que te prepare también un ejemplo para **crear y editar artículos desde formularios web** (no solo desde el admin)?

Hoy
paso a paso instalrr django
Ver mi razonamiento

Instalación paso a paso de Django

Antes de empezar, respira: esto no es una carrera. Instalar Django bien es sentar las bases para lo que vas a crear después. Vamos paso a paso.

Requisitos previos

- **Python instalado:** Django necesita Python 3.10+. Comprueba tu versión:

Código

```
python --version
```

o

Código

```
python3 --version
```

Si no lo tienes, descárgalo desde el sitio oficial o con el gestor de paquetes de tu sistema. Python incluye SQLite, suficiente para empezar sin configurar una base de datos aparte.

Crear y activar un entorno virtual

- **Crear carpeta de proyecto:**

Código

```
mkdir mi_proyecto_django  
cd mi_proyecto_django
```

- **Crear entorno virtual:**

- En Windows:

Código

```
python -m venv .venv
```

- En macOS/Linux:

Código

```
python3 -m venv .venv
```

- **Activar entorno virtual:**

- En Windows (PowerShell):

Código

```
.\.venv\Scripts\Activate.ps1
```

En CMD:

Código

```
.\.venv\Scripts\activate.bat
```

- En macOS/Linux:

Código

```
source .venv/bin/activate
```

- **Verifica que está activo:** Deberías ver “(.venv)” al inicio de tu línea de comandos.

Instalar Django

- **Actualizar pip (opcional pero recomendable):**

Código

```
python -m pip install --upgrade pip
```

o

Código

```
python3 -m pip install --upgrade pip
```

- **Instalar la última versión estable de Django:**

Código

```
pip install django
```

- **Confirmar instalación:**

Código

```
python -m django --version
```

Debería mostrar el número de versión instalado.

Crear tu primer proyecto y app

- **Crear un proyecto:**

Código

```
django-admin startproject config .
```

El punto final crea los archivos directamente en la carpeta actual para evitar un directorio anidado.

- **Migraciones iniciales (configura la base SQLite por defecto):**

Código

```
python manage.py migrate
```

- **Ejecutar el servidor de desarrollo:**

Código

```
python manage.py runserver
```

Abre `http://127.0.0.1:8000` en tu navegador. Verás la página de bienvenida de Django si todo está correcto.

- **Crear una app dentro del proyecto (opcional, pero es el siguiente paso lógico):**

Código

```
python manage.py startapp core
```

Luego añade "core" a `INSTALLED_APPS` en `config/settings.py` y crea tus vistas, urls y modelos.

Consejos y problemas comunes

- **Comando python vs python3:** En algunas máquinas, "python" apunta a Python 2; usa "python3" si "python --version" no es 3.x.
- **Permisos en macOS/Linux:** Si ves errores al activar el venv, asegúrate de usar "source" y que el shell sea bash/zsh.
- **PowerShell en Windows:** Si aparece una política de ejecución bloqueando `Activate.ps1`, abre PowerShell como administrador y ejecuta:

Código

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope  
CurrentUser
```

- **Base de datos:** SQLite viene lista para usar; no necesitas configurar PostgreSQL/MySQL para empezar.

¿Quieres que lo hagamos específico para tu sistema?

Dime si estás en Windows, macOS o Linux, y qué versión de Python tienes, y te ajusto los comandos exactos. Si prefieres video, también puedo recomendarte uno en español con el paso a paso visual. Para documentación oficial y la guía rápida, esta referencia es útilDjango.

django para mi version 3.13.7

Perfecto, tienes **Python 3.13.7** instalado 🎉. Django funciona sin problema en esa versión, así que puedes seguir los pasos de instalación con confianza. Te muestro el proceso adaptado específicamente para tu versión:

Instalación de Django en Python 3.13.7

1. Verifica tu versión de Python

En tu terminal escribe:

```
bash
python3 --version
```

Debe mostrar:

```
Código
Python 3.13.7
```

2. Crea un entorno virtual

Esto mantiene tu proyecto aislado de otros paquetes:

```
bash
python3 -m venv .venv
```

Activa el entorno:

- **Linux/macOS:**

```
bash

source .venv/bin/activate
```

- **Windows (PowerShell):**

```
bash

.\.venv\Scripts\Activate.ps1
```

3. Actualiza pip

```
bash
python -m pip install --upgrade pip
```

4. Instala Django

```
bash
pip install django
```

Confirma la instalación:

```
bash
python -m django --version
```

Deberías ver la última versión estable (por ejemplo, 5.x).

5. Crea tu primer proyecto

```
bash
django-admin startproject config .
```

6. Aplica migraciones iniciales

```
bash
python manage.py migrate
```

7. Ejecuta el servidor de desarrollo

```
bash
python manage.py runserver
```

Abre en tu navegador: **http://127.0.0.1:8000**

☑ Con esto ya tienes Django funcionando en **Python 3.13.7**. El siguiente paso sería crear una **app** dentro del proyecto:

```
bash
python manage.py startapp core
```

👉 ¿Quieres que te muestre también cómo instalar una versión **específica de Django** (por ejemplo, 4.x o 5.x) para tu Python 3.13.7, o prefieres siempre la última disponible?

```
PS C:\Users\IUTEPI\Nueva carpeta> .\venv\Scripts\Activate .\venv\Scripts\Activate : No
se puede cargar el archivo C:\Users\IUTEPI\Nueva carpeta\venv\Scripts\Activate.ps1
porque la ejecución de scripts está deshabilitada en este sistema. Para obtener más
información, consulta el tema about_Execution_Policies en
https://go.microsoft.com/fwlink/?LinkID=135170. En línea: 1 Carácter: 1 +
.\venv\Scripts\Activate + ~~~~~ + CategoryInfo          : SecurityError: (:)
[], PSSecurityException + FullyQualifiedErrorId : UnauthorizedAccess
```

Ese error es muy común en **Windows PowerShell**: significa que la política de ejecución de scripts está bloqueando el archivo `Activate.ps1` de tu entorno virtual. Vamos a solucionarlo paso a paso:

Solución al error de ejecución de scripts en PowerShell

1. Abre PowerShell como administrador

- Haz clic derecho en el ícono de PowerShell → **Ejecutar como administrador**.

2. Cambia la política de ejecución

Ejecuta este comando:

```
powershell  
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

- **RemoteSigned** permite ejecutar scripts locales (como el de tu venv) sin problemas.
- El parámetro `-Scope CurrentUser` aplica el cambio solo a tu usuario, no al sistema completo.

Cuando te pregunte, escribe:

Código
Y

y presiona **Enter**.

3. Activa tu entorno virtual

Ahora vuelve a tu carpeta del proyecto y activa el entorno:

```
powershell  
.\venv\Scripts\Activate
```

Deberías ver algo como:

```
Código  
(venv) PS C:\Users\IUTEPI\Nueva carpeta>
```

4. Alternativas si no quieres cambiar la política

- Usa **CMD** en lugar de PowerShell:

```
cmd  
  
.\venv\Scripts\activate.bat
```

- O usa **Git Bash** si lo tienes instalado:

```
bash
```

```
source venv/Scripts/activate
```

✓ Con esto tu entorno virtual debería activarse sin problemas. ¿Quieres que te muestre también cómo **revertir** el cambio de política si luego prefieres dejarlo como estaba?

nuevo. En línea: 1 Carácter: 1 + django-admin startproject config . + ~~~~~ +
CategoryInfo : ObjectNotFound: (django-admin:String) [], CommandNotFoundException +
FullyQualifiedErrorId : CommandNotFoundException

Ese error significa que **PowerShell no encuentra el comando** `django-admin`, lo que indica que Django no está instalado en tu entorno virtual o que el entorno no está activado correctamente. Vamos a resolverlo paso a paso:

Diagnóstico y solución

1. Verifica que tu entorno virtual esté activo

Cuando activas el venv, tu línea de comandos debería mostrar algo como:

Código
(venv) PS C:\Users\IUTEPI\Nueva carpeta>

Si no aparece `(venv)`, el entorno no está activo. Actívalo con:

```
powershell  
.\venv\Scripts\Activate
```

2. Instala Django dentro del entorno

Con el venv activo, instala Django:

```
powershell  
pip install django
```

3. Comprueba que `django-admin` existe

Ejecuta:

```
powershell  
django-admin --version
```

Si devuelve un número (por ejemplo, `5.1.3`), ya está instalado.

4. Alternativa si aún no funciona

En algunos casos, el comando no se reconoce aunque esté instalado. Usa el módulo de Python directamente:

```
powershell
python -m django --version
python -m django startproject config .
```

5. Confirmar instalación

Después de crear el proyecto, deberías ver una carpeta `config` con archivos como:

- `manage.py`
- `config/settings.py`
- `config/urls.py`

☑ Con esto deberías poder crear tu proyecto sin problemas. ¿Quieres que te muestre también cómo **forzar la instalación de Django en tu venv** y verificar que el ejecutable `django-admin` está en la carpeta `Scripts`?

dame un proyecto sencillo para ejemplo

Perfecto, vamos a armar un **proyecto Django sencillo** para que tengas un ejemplo práctico. Será un mini-blog donde puedes publicar y listar artículos.

Estructura del proyecto

Supongamos que ya tienes tu entorno virtual activo y Django instalado.

1. Crear el proyecto base:

```
bash

django-admin startproject mi_blog .
```

2. Crear una aplicación dentro del proyecto:

```
bash

python manage.py startapp blog
```

3. Registrar la app en `settings.py`: Abre `mi_blog/settings.py` y en `INSTALLED_APPS` agrega:

```
python
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'blog',      # nuestra app  
]
```



Modelo de datos

En `blog/models.py` define un modelo sencillo para artículos:

python

```
from django.db import models  
  
class Artículo(models.Model):  
    titulo = models.CharField(max_length=200)  
    contenido = models.TextField()  
    fecha_publicacion = models.DateTimeField(auto_now_add=True)  
  
    def __str__(self):  
        return self.titulo
```

Aplica migraciones:

bash

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



Admin de Django

En `blog/admin.py` registra el modelo para gestionarlo desde el panel admin:

python

```
from django.contrib import admin  
from .models import Artículo  
  
admin.site.register(Articulo)
```

Crea un superusuario:

bash

```
python manage.py createsuperuser
```



Vistas y URLs

En `blog/views.py` crea una vista para listar artículos:

python

```
from django.shortcuts import render
from .models import Artículo

def lista_articulos(request):
    articulos = Artículo.objects.all()
    return render(request, 'blog/lista.html', {'articulos': articulos})
```

En `blog/urls.py` (crea este archivo):

python

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.lista_articulos, name='lista_articulos'),
]
```

En `mi_blog/urls.py` incluye las rutas de la app:

python

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')),
]
```



Plantilla HTML

Crea `blog/templates/blog/lista.html`:

html

```
<!DOCTYPE html>
<html>
<head>
    <title>Mi Blog</title>
</head>
<body>
    <h1>Artículos</h1>
    <ul>
        {% for articulo in articulos %}
            <li><strong>{{ articulo.titulo }}</strong> - {{
articulo.fecha_publicacion }}</li>
            <p>{{ articulo.contenido }}</p>
        {% endfor %}
    </ul>
</body>
</html>
```



Ejecutar el servidor

bash

```
python manage.py runserver
```

Abre en tu navegador: **http://127.0.0.1:8000/** Verás la lista de artículos que vayas creando desde el panel admin.



Con esto tienes un **mini-blog funcional**:

- Panel admin para crear artículos.
- Página principal que lista todo lo publicado.

¿Quieres que te prepare también un ejemplo para **crear y editar artículos desde formularios web** (no solo desde el admin)?