

# Documentación Django (Phyton)

---

## Tabla de contenidos

---

- [Documentación Django \(Phyton\)](#)
- [Tabla de contenidos](#)
- [Introducción](#)
  - [Instalación en Linux](#)
    - [Explicación de los comandos](#)
  - [Creación de un proyecto](#)
    - [Explicación de los ficheros al crear un proyecto Django](#)
  - [Ejecución del servidor de desarrollo](#)
  - [Instalación del IDE PyCharm](#)

# Introducción

---

## Tabla de contenidos

**Django** es un framework web de alto nivel de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador (MVC).

El modelo-vista-controlador es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

- **Modelo:** Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).
- **Vista:** Presenta el modelo en un formato adecuado para interactuar (usualmente la interfaz de usuario) y también puede encargarse de filtrar la entrada de datos y enviarla al modelo.
- **Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud de información (por ejemplo, editar un documento o un registro en una base de datos).

## Instalación en Linux

### Tabla de contenidos

Python viene instalado por defecto en la mayoría de distribuciones GNU/Linux. Para comprobar si lo tenemos instalado, abrimos una terminal y escribimos:

```
python3 --version
```

En el caso de que no lo tengamos instalado, podemos instalarlo desde los repositorios oficiales de nuestra distribución. En el caso de Ubuntu, escribimos:

```
sudo apt install python3
```

Una vez verificado que tenemos instalado Python podemos empezar a instalar Django primero debemos instalar python3.10-venv para poder crear entornos virtuales. Para ello, escribimos:

```
sudo apt install python3.10-venv
```

Una vez instalado, creamos un entorno virtual llamado django y lo activamos:

```
$ python3 -m venv django
$ source django/bin/activate
(django)$ pip install django
(django)$ python -m django --version
4.2.7
```

## Explicación de los comandos

- **python3 -m venv django:** Creamos un entorno virtual llamado django.
- **source django/bin/activate:** Activamos el entorno virtual esto hay que hacerlo cada vez que queramos trabajar con Django.
- **pip install django:** Instalamos Django.
- **python -m django --version:** Comprobamos la versión de Django instalada.

Si en la terminal nos aparece (django) delante del nombre de nuestro usuario, significa que el entorno virtual está activado. En caso contrario, debemos activarlo con el comando `source django/bin/activate`. Para desactivar el entorno virtual, escribimos `deactivate`.

# Creación de un proyecto

## Tabla de contenidos

En primer lugar vamos a ver como crear un proyecto Django. Para ello, nos situamos en el directorio donde queremos crear el proyecto y escribimos:

```
(django)$ django-admin startproject mysite
```

Esto creará un directorio llamado mysite con la siguiente estructura:

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    asgi.py  
    wsgi.py
```

## Explicación de los ficheros al crear un proyecto Django

- **manage.py:** Es un script que ayuda con la gestión del sitio. Con él podemos arrancar un servidor de desarrollo, crear aplicaciones, crear migraciones de la base de datos, etc.
- El directorio **mysite/** es un paquete de Python para nuestro proyecto.
- **settings.py:** Contiene la configuración del proyecto.
- **urls.py:** Contiene las definiciones de las URLs del proyecto.
- **wsgi.py:** Es un punto de entrada para los servidores web compatibles con WSGI para servir el proyecto.
- **asgi.py:** Es un punto de entrada para los servidores web compatibles con ASGI para servir el proyecto.

## Ejecución del servidor de desarrollo

### Tabla de contenidos

Para arrancar el servidor de desarrollo, nos situamos en el directorio donde se encuentra el fichero `manage.py` y escribimos:

```
(django)$ python manage.py runserver
```

Ahora si todo a ido bien, podemos acceder a nuestro servidor de desarrollo desde un navegador web en la dirección `http://127.0.0.1:8000/` en mi caso esa es la dirección, pero puede variar en función de la configuración de nuestro equipo. Si todo ha ido bien, veremos una página de bienvenida de Django.

Ahora bien si queremos que el servidor de desarrollo sea accesible desde cualquier dirección IP y además en un puerto determinado, escribimos:

```
(django)$ python manage.py runserver 0.0.0.0:8000
```

## Instalación del IDE PyCharm

### Tabla de contenidos

Ahora que tenemos todo lo necesario para desarrollar nuestra aplicación web con Django vamos a instalar PyCharm que es un IDE multiplataforma creado por JetBrains. Es uno de los mejores IDEs para proyectos que utilizan el lenguaje de programación Python, primero debemos descargarlo desde la página oficial de [JetBrains](#). Una vez descargado, nos situamos en el directorio donde se encuentra el fichero descargado y escribimos:

```
$ tar -xzf pycharm-2023.2.5.tar.gz
```

Esto creará un directorio llamado `pycharm-2023.2.5` con la siguiente estructura:

```
pycharm-2023.2.5/  
  bin/  
  lib/  
  jbr/  
  license/
```

```
plugins/  
help/  
debug-eggs/
```

Ahora para ejecutar PyCharm, nos situamos en el directorio `pycharm-2023.2.5/bin` y escribimos:

```
$ ./pycharm.sh
```

Esto nos abrirá una ventana de bienvenida de PyCharm, donde podemos crear un proyecto nuevo o abrir uno existente.

Es recomendable crear un alias para ejecutar PyCharm desde cualquier directorio. Para ello, abrimos una terminal y escribimos:

```
$ sudo nano ~/.bashrc
```

Esto nos abrirá el fichero `.bashrc` en el editor de texto `nano`. Ahora debemos añadir la siguiente línea al final del fichero:

```
alias pycharm="{tu sitio de instalacion}/pycharm-2023.2.5/bin/pycharm.sh"
```

Ahora guardamos los cambios y cerramos el editor de texto. Para que los cambios surtan efecto, escribimos:

```
$ source ~/.bashrc
```

Ahora podemos ejecutar PyCharm desde cualquier directorio escribiendo `pycharm` en la terminal.