

## DEPLOY APPLICACION DJANGO

Vamos a visualizar cómo podemos desplegar una aplicación de django en producción.

Lo primero que debemos hacer es utilizar recursos gratuitos, para ello, necesitamos un Hosting y una base de datos gratuita.

Para el Hosting, utilizaremos **Python Anywhere**

<https://www.pythonanywhere.com/>

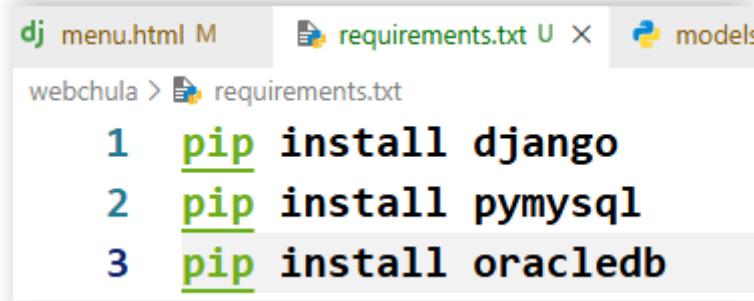
Para la base de datos, utilizaremos Free Sql Database

<https://www.freesqldatabase.com/>

La base de datos gratuita es de tipo **MySql**, por lo que debemos modificar nuestro servicio de Oracle para utilizar MySql.

Sobre nuestro proyecto **webchula** creamos un nuevo fichero llamado **requirements.txt**

Incluimos todos los elementos que necesitemos dentro de **Python**:



```
1 pip install django
2 pip install pymysql
3 pip install oracledb
```

Instalamos **pymysql** dentro de nuestro **Terminal** de **VS Code**.

El siguiente paso es modificar nuestro fichero **models.py** para que se conecte con **MySql** en lugar de **Oracle**

### MODELS.PY

- 1) Renombramos la clase **ServiceSeries** a **ServiceSeriesOracle**
- 2) Copiamos **ServiceSeriesOracle** dentro del mismo fichero y cambiamos el nombre a **ServiceSeries**
- 3) Importamos la librería de **mysql**:

```

dj menu.html M requirements.txt U models.py M X views.py
webchula > television > models.py > ServiceSeries > insertarPersonaje
1 from django.db import models
2 import oracledb
3 import pymysql

```

- 4) Modificamos el objeto **connection** para conectar con **MySQL** y nuestro servidor gratuito creado anteriormente.

```

class ServiceSeries:
    def __init__(self):
        self.connection = pymysql.connect(host='sql17.freesqldatabase.com', port=3306,
                                         user='sql17818176', password='FzMNGLTjrq', database='sql17818176')

```

- 5) Las consultas con parámetros debemos modificarlas, ya que MySQL llama a todos los parámetros igual: **%s**

```

def getPersonajesSerie(self, idserie):
    sql = "select * from PERSONAJES where IDSERIE=%s"

def insertarPersonaje(self, nombre, imagen, idserie):
    sql = """
        insert into PERSONAJES values
        ((select max(IDPERSONAJE) + 1 from PERSONAJES), %s, %s, %s)
    """

```

- 6) Ejecutamos nuestra aplicación y veremos que debería ser funcional

El siguiente paso es configurar el fichero **settings.py** para habilitar nuestro servidor en producción.

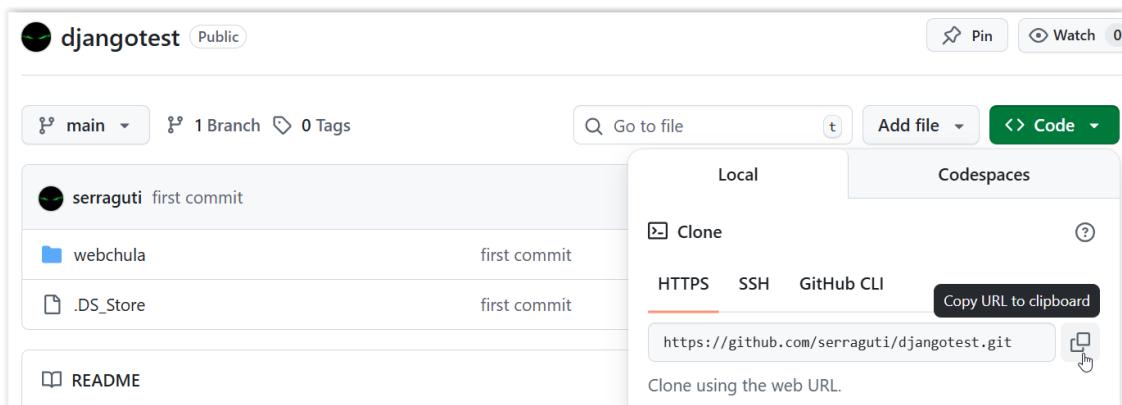
Dentro de la clave **ALLOWED\_HOST** debemos incluir todos los servidores que vayamos a utilizar:

**Nota:** Debemos poner nuestro USUARIO de Python Anywhere

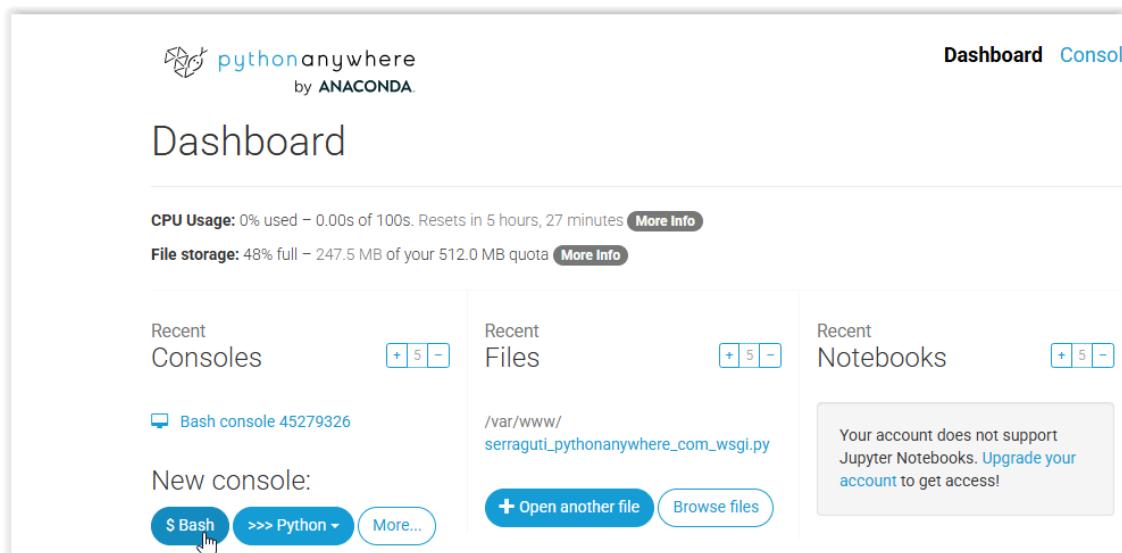
```
ALLOWED_HOSTS = ["127.0.0.1", "localhost", "serraguti.pythonanywhere.com"]
```

Con todo listo, ponemos todo en GitHub de forma **public**, luego lo cambiamos a **private**, ya que tiene claves y todo...

Dentro de **GitHub**, en nuestro nuevo repositorio, copiamos la URL de nuestro **.git**

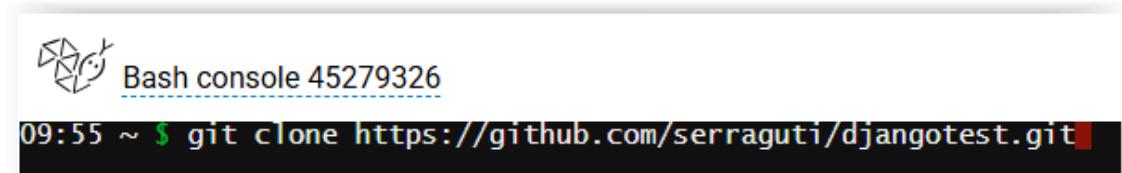


El siguiente paso es ir a Python Anywhere y dentro de la página principal, pulsar sobre **BASH**



Dentro de la consola, escribimos el siguiente comando:

**git clone "NUESTRA URL GITHUB.git"**



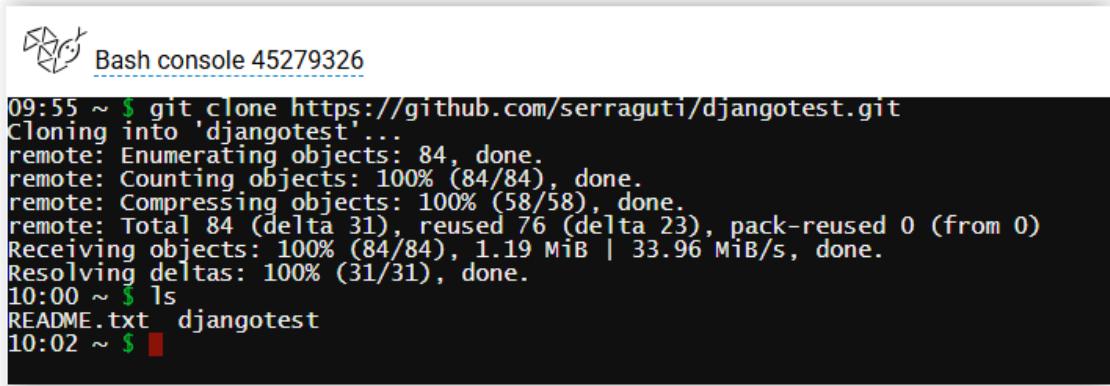
El siguiente paso **IMPORTANTE** es poner nuestro GitHub en modo privado, ya que tenemos las claves de conexión a nuestro **MySQL** personal.

Dentro del **BASH** ya podremos ver nuestra carpeta.

Si escribimos el comando **ls** veremos los directorios.

Para entrar en un directorio **cd NombreDirectorio**

Para salir de un directorio **cd ..**



```
Bash console 45279326
09:55 ~ $ git clone https://github.com/serraguti/djangotest.git
Cloning into 'djangotest'...
remote: Enumerating objects: 84, done.
remote: Counting objects: 100% (84/84), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 84 (delta 31), reused 76 (delta 23), pack-reused 0 (from 0)
Receiving objects: 100% (84/84), 1.19 MiB | 33.96 MiB/s, done.
Resolving deltas: 100% (31/31), done.
10:00 ~ $ ls
README.txt djangotest
10:02 ~ $
```

A continuación, nos crearemos un entorno virtual aislado para instalar las librerías necesarias para publicar nuestra aplicación.

Escribimos el siguiente comando:

**python -m venv .venv**

Posteriormente, lo activamos.

**source .venv/bin/activate**



```
RENDERME.CAT djangotest
10:02 ~ $ python -m venv .venv
10:04 ~ $ source .venv/bin/activate
(.venv) 10:04 ~ $
```

Una vez que tenemos listo el entorno, instalamos las librerías necesarias para el funcionamiento de nuestra aplicación.(Las tenemos dentro de **Requirements.txt** para no olvidarnos)

**pip install django**  
**pip install oracledb**  
**pip install pymysql**

```
Bash console 45279326
(.venv) 10:08 ~ $ pip install django
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: django in ./venv/lib/python3.13/site-packages (6.0.2)
Requirement already satisfied: asgiref>=3.9.1 in ./venv/lib/python3.13/site-packages (from django) (3.11.1)
Requirement already satisfied: sqlparse>=0.5.0 in ./venv/lib/python3.13/site-packages (from django) (0.5.5)
(.venv) 10:08 ~ $ pip install oracledb
Looking in links: /usr/share/pip-wheels
Collecting oracledb
  Downloading oracledb-3.4.2-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (7.7 kB)
Requirement already satisfied: cryptography<=3.2.1 in ./venv/lib/python3.13/site-packages (from oracledb) (46.0.5)
Requirement already satisfied: typing_extensions<=4.14.0 in ./venv/lib/python3.13/site-packages (from oracledb) (4.15.0)
Requirement already satisfied: cffi>=2.0.0 in ./venv/lib/python3.13/site-packages (from cryptography>=3.2.1->oracledb) (2.0.0)
Requirement already satisfied: pycparser in ./venv/lib/python3.13/site-packages (from cffi>=2.0.0-cryptography>=3.2.1->oracledb) (3.0)
  Downloading oracledb-3.4.2-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (2.4 MB)
    2.4/2.4 MB 8.4 Mb/s eta 0:00:00

Installing collected packages: oracledb
Successfully installed oracledb-3.4.2
(.venv) 10:10 ~ $ pip install pymysql
Looking in links: /usr/share/pip-wheels
Collecting pymysql
  Downloading pymysql-1.1.2-py3-none-any.whl.metadata (4.3 kB)
  Downloading pymysql-1.1.2-py3-none-any.whl (45 kB)
Installing collected packages: pymysql
Successfully installed pymysql-1.1.2
(.venv) 10:10 ~ $
```

Ya casi lo tenemos. Salimos del **bash** y volvemos al **Dashboard** de Python anywhere

## Entramos dentro de Web

The dashboard shows:

- CPU Usage: 7% used – 7.60s of 100s. Resets in 5 hours, 9 minutes ([More Info](#))
- File storage: 50% full – 257.8 MB of your 512.0 MB quota ([More Info](#))
- Recent activity: Recent Consoles, Recent Files, Recent Notebooks, All Web apps.
- Welcome message: Welcome, [serraguti](#)
- Upgrade Account button

Pulsamos sobre **Add a new web app**

The screen shows:

- Dashboard, Consoles, Files, Web, Tasks, Databases buttons.
- A blue button: **Add a new web app**.
- A message box: You have no web apps. To create a PythonAnywhere-hosted web app, click the 'Add a new web app' button to the left.

Nos preguntará si queremos hacer Upgrade, **ni caso** y pulsamos en **Next**

Seleccionamos **django** como Framework

Create new web app

## Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » [Django](#)
- » [web2py](#)
- » [Flask](#)
- » [Bottle](#)
- » [Manual configuration](#) (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

[Cancel](#) [« Back](#) [Next »](#)

Seleccionamos la versión de Python (la última...)

Create new web app

## Select a Python version

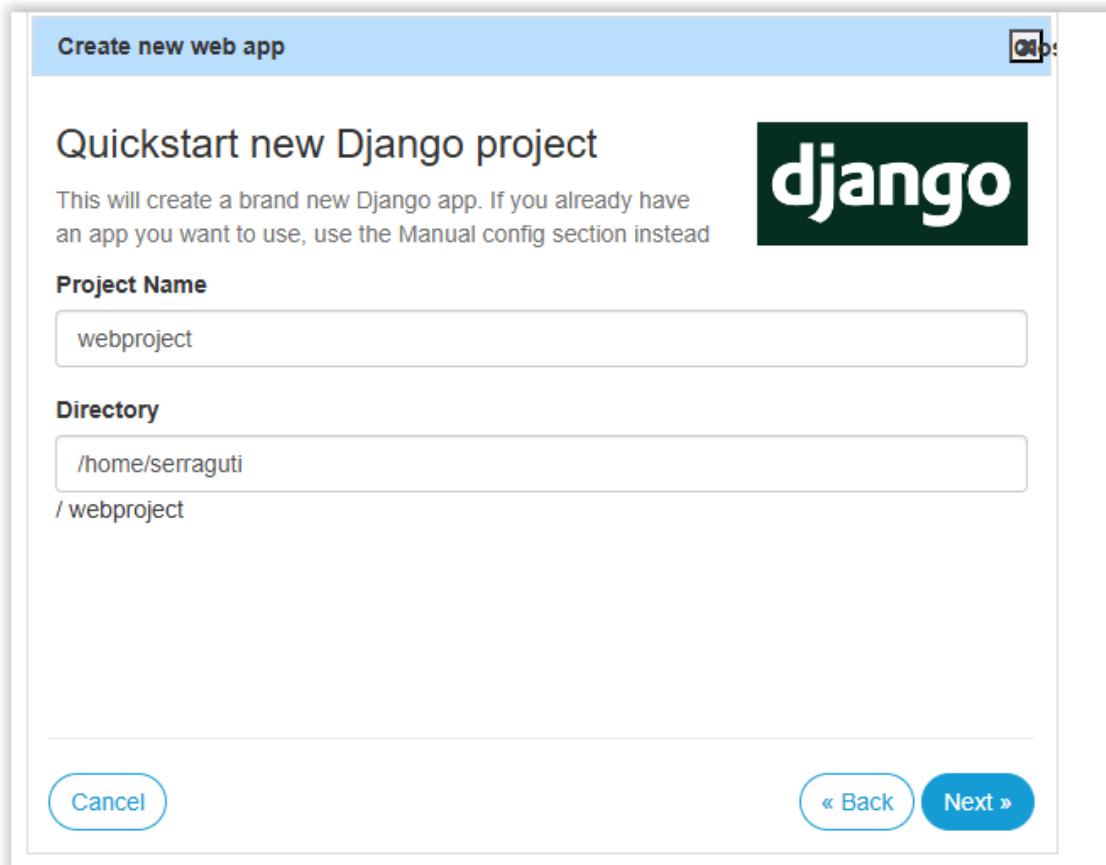
- » [Python 3.9](#) (Django 4.2.16)
- » [Python 3.10](#) (Django 5.0.9)
- » [Python 3.11](#) (Django 5.0.9)
- » [Python 3.12](#) (Django 5.0.9)
- » [Python 3.13](#) (Django 5.1.3)

**Note:** If you'd like to use a different version of Django to the default version, you can use a virtualenv for your web app. There are [instructions here](#).

[Cancel](#) [« Back](#) [Next »](#)

Lo siguiente que nos pregunta es por el nombre de nuestro proyecto, no es necesario este paso ya que vamos a configurar el nuestro propio.

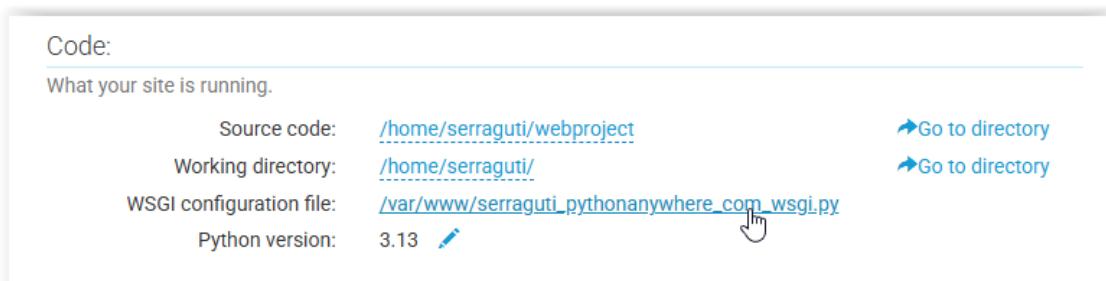
Ponemos cualquier nombre (**webproject**) y **Next**



Comenzará a implementar el servidor...

Por último, debemos indicar un par de características.

Entramos dentro de WSGI:



Sustituimos nuestro nombre temporal (**webproject**) por la ruta a nuestro proyecto real: **webchula**

```
import os
import sys

# add your project directory to the sys.path
project_home = '/home/serraguti/djangotest/webchula' ←
- if project_home not in sys.path:
    sys.path.insert(0, project_home)

# set environment variable to tell django where your settings.py is
os.environ['DJANGO_SETTINGS_MODULE'] = 'webchula.settings' ←

# serve django via WSGI
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Por último, indicamos dónde tenemos los ficheros **static**

**/home/serraguti/djangotest/webchula/television/static**

#### Static files:

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
/static/	/home/serraguti/djangotest/webchula/television/static	
/media/	/home/serraguti/webproject/media	
Enter URL	Enter path	

Por último, indicamos el nombre de nuestro entorno virtual **.venv**

#### Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

[Enter path to a virtualenv, if desired](#)



### Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

