Guía

Proyecto Django.

Primero usaremos una consola para crear una carpeta que nos servirá de guía.

Creamos una carpeta y nos entramos en ella para seguir trabajando en ella.

mkdir ~/guia  
cd ~/guia



Ahora procedemos a instalar Pycharm con estos comandos.

sudo snap install pycharm-community –classic

Una vez instalado podemos abrirlo desde la terminal usando

pycharm-community

Cuando abrimos un programa desde la terminal este queda linkado a la terminal, esto significa que si cerramos la terminal también cerraremos el programa. Para evitar este comportamiento podemos añadir *&* al final de cualquier comando. De esta forma podemos seguir usando la terminal, aunque el comando que hayamos usado no haya terminado.

pycharm-community &

Ahora instalaremos python3, pero primero hay que hacer unas verificaciones con estos comandos,

python3

nos saldrá esto

nos saldrá esto:

Python 2.7.17 (default, Jul 20 2020, 15:37:01)

[GCC 7.5.0] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> quit()

Luego verificamos con este comando.

Python

nos saldrá esto:

Python 2.7.17 (default, Jul 20 2020, 15:37:01)

[GCC 7.5.0] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> quit()

Antes de continuar instalaremos python3 con el siguiente comando:

sudo apt install python3-pip

Uso e importación de librerías a Python en Ubuntu



Hay que instalar dos pip,pip y pip3, uno para cada version. Tenemos que ejecutarlo de esta manera.

Ejercicio: instala la librería antigravity e importarla en un interprete de python.

Ejercicio: importa la librería this. Esta librería ya viene incluida en python.

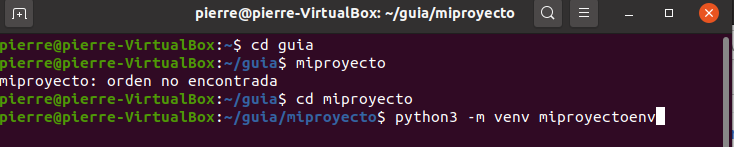
Comando: $ pip3 install antigravity

Comando: $ pip3 install this

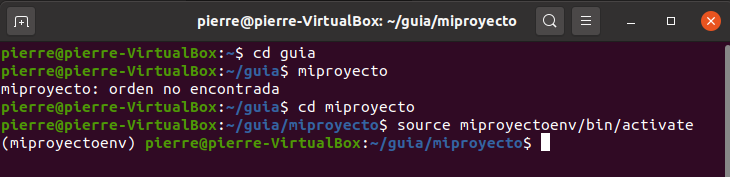
Luego tenemos que instalar esta herramienta ejecutando el siguiente comando

 sudo apt-get install python3-venv Una vez instalado vamos a crear una carpeta para nuestro proyecto, la vamos a llamar *miproyecto*, esto lo hacemos usando el comando mkdir miproyecto. Y seguidamente entramos en esta carpeta ejecutando cd miproyecto. Este será el directorio donde trabajaremos. Para saber la ruta donde nos encontramos podemos ejecutar el comando pwd, ejecútalo y haz una nota mental de donde estamos para poder volver a esta carpeta en cualquier momento.

Luego dentro de la carpeta de nuestro proyecto creamos un entorno virtual, esto lo hacemos con el siguiente comando python3 -m venv miproyectoenv. Miproteyectoenv es simplemente el nombre que hemos decidido darle a nuestro entorno virtual y no implica nada.



Ahora ya tenemos un entorno virtual, para entrar en él (activarlo), lo haremos de la siguiente forma: source miproyectoenv/bin/activate. Si el comando ha funcionado deberíamos ver (miproyectoenv) al inicio de nuestra línea de comandos, así:



Para este proyecto vamos a necesitar dos paquetes de python *django* y *psycopg2*. Instalamos estos paquetes usando pip3. Es importante que estemos dentro del entorno virtual cuando los instalemos.

Antes de instalar psycopg2 se debe instalar postgres, esto lo podemos hacer de la siguiente forma:

Primero:

$ sudo apt update  
$ sudo apt install libpq-dev postgresql postgresql-contrib

Segundo:

pip3 install django

pip3 install psycopg2

$ sudo apt install tree

Ojo En caso de haberlo instalado fuera del entorno virtual desinstala el paquete en cuestión usando

|  |
| --- |
| $ pip3 uninstall nombre\_paquete |

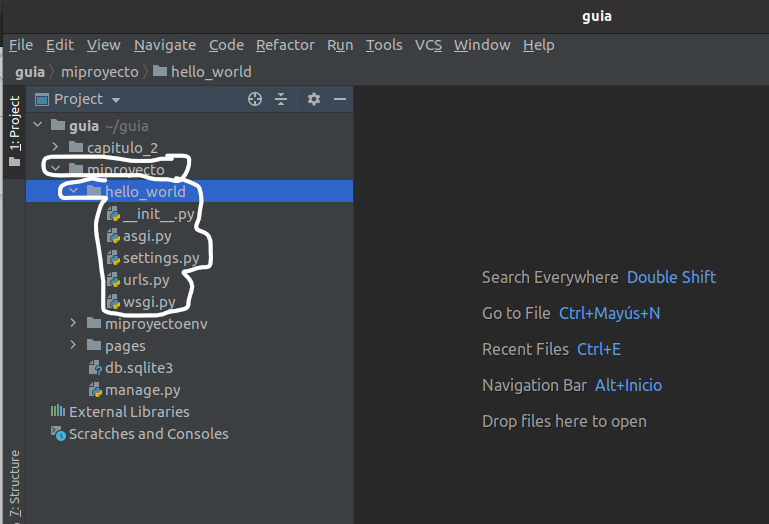
Crear un proyecto

Empecemos creando un proyecto con Django. Abrimos una terminal, nos situamos en la carpeta donde creamos nuestro entorno virtual, activamos el entorno virtual, y ejecutamos el siguiente comando.

|  |
| --- |
| $ django-admin.py startproject hello\_world . |

*Peligro: El punto del final indica que queremos que el proyecto sea creado en el directorio “.”, en otras palabras, que sea creado en el directorio donde nos encontramos. Es importante ya que sin el Django crea una nueva carpeta por defecto.*

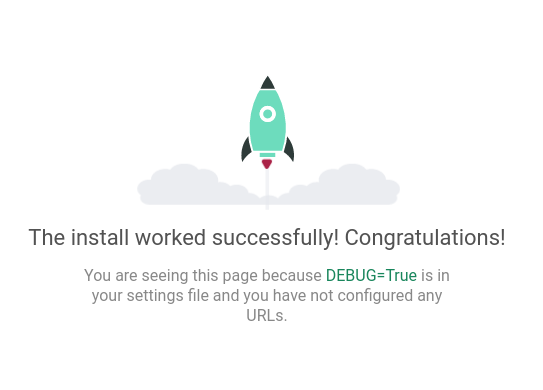
*Al ejecutar estos comandos abrimos en pycharm y vemos que quede o este como en la imagen:*

**

*AHORA* vamos a comprobar si todo funciona correctamente. Para ello vamos a usar el programa manage.py este programa será nuestra interfaz para comunicarnos con las herramientas de Django. Para

empezar, vamos a arrancar su servidor web y ver si todo funciona.

Vamos a nuestro navegador preferido y vamos a la siguiente dirección: <http://127.0.0.1:8000/>



Si vemos el siguiente mensaje es que ya tenemos nuestro proyecto Django preparado y listo para empezar a trabajar con él.

*Nota*: *La dirección* [*http://127.0.0.1:8000/*](http://127.0.0.1:8000/) *y* [*http://localhost:8000/*](http://localhost:8000/)

## Crear una app

Django usa los conceptos de proyectos y aplicaciones para mantener el código limpio y estructurado. Un proyecto de Django está formado por una o más aplicaciones. Por ejemplo, una tienda online podría tener una app para el login, otra para los pagos, y otra para gestionar todos los productos de la tienda. Cada una focalizada en una funcionalidad distinta.

Vamos a crear nuestra primera aplicación y la vamos a llamar pages. Para ello vamos a volver a hacer uso de la herramienta manage.py de django.

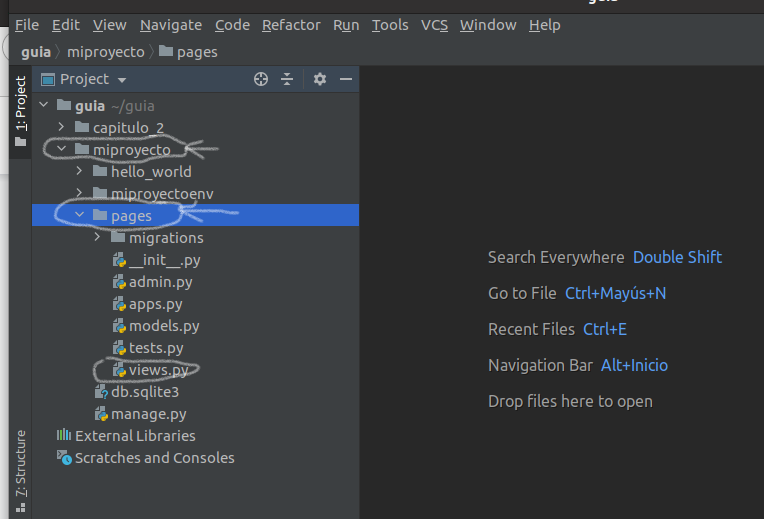
|  |
| --- |
| $ python3 manage.py startapp pages |

Este comando nos crea una carpeta llamada pages con la siguiente estructura

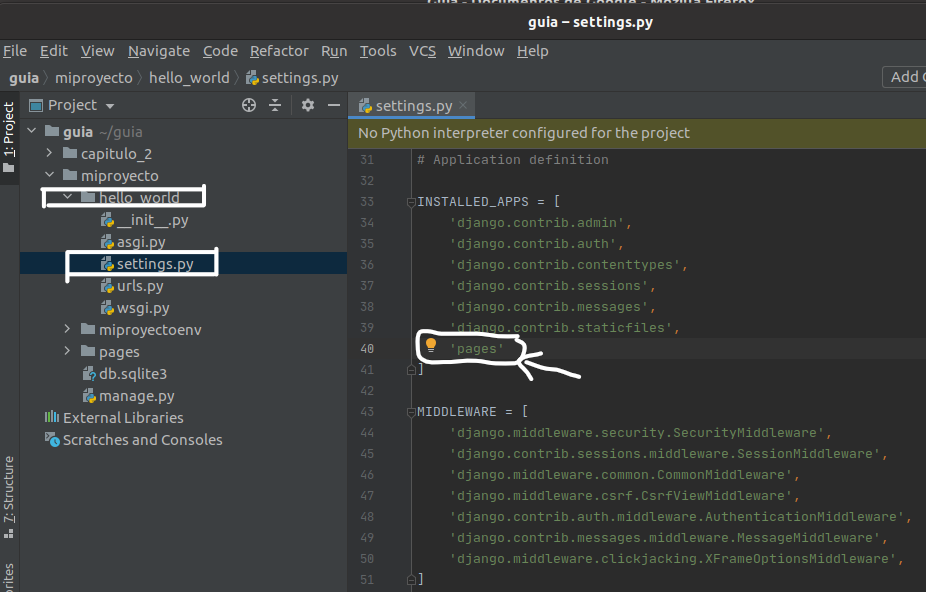
|  |
| --- |
| (miproyectoenv) joan:~/tests/miproyecto$ tree pages/ pages/ ├── admin.py ├── apps.py ├── \_\_init\_\_.py ├── migrations │   └── \_\_init\_\_.py ├── models.py ├── tests.py └── views.py |

Teniendo nuestro primer proyecto y nuestra primera app iniciadas ya podemos empezar a escribir código.

También verificamos en Pycharm que tenga ya creada la carpeta page como vemos en la imagen:



Ahora en pycharm desplegamos y abrimos en stting.py, y ponemos page como esta en la imagen esto nos ayuda a que creemos nuestra app.



Esto es necesario porque Django está pensado para trabajar con muchas aplicaciones a la vez, y rehusar aplicaciones entre diferentes proyectos. Haciendo uso de este archivo de configuración se puede de forma fácil desactivar o activar aplicaciones dentro de un proyecto.

Ahora en pycharm en la page-----views.py. copiamos estos comandos.

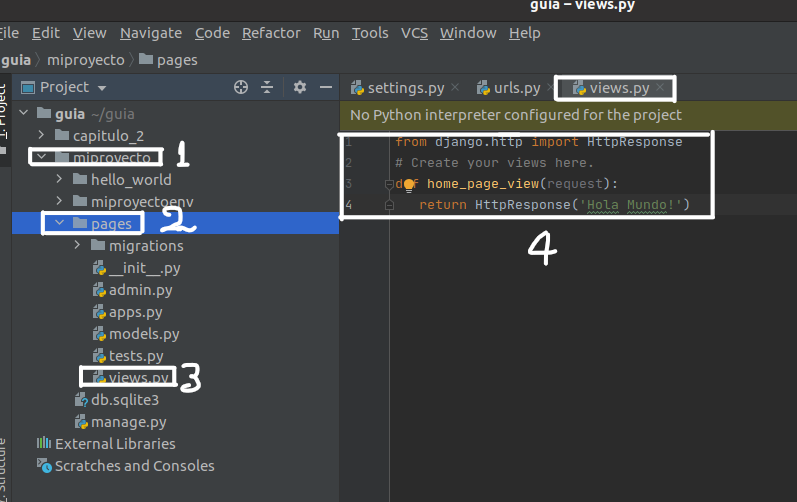
from django.http import HttpResponse

# Create your views here.

def home\_page\_view(request):

return HttpResponse('Hola Mundo!')

debe quedar asi:



Después vamos al archivo miproyecto/url.py y allí añadimos una nueva url que nos lleve a nuestra nueva vista. Esto lo hacemos de la siguiente manera:

from django.contrib import admin

from django.urls import path

from pages import views

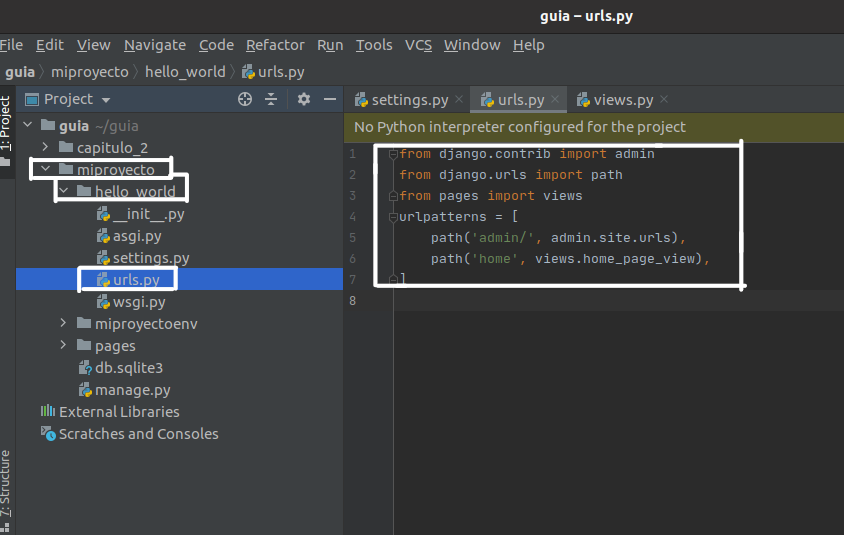
urlpatterns = [

path('admin/', admin.site.urls),

path('home', views.home\_page\_view),

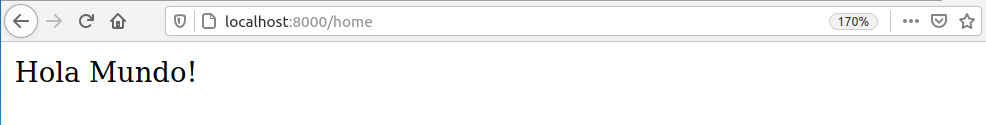
]

debe quedar asi:



Fijate que hemos tenido que importar de la app pages el archivo views. Una vez importado podemos llamar a sus funciones escribiendo *view.nombre\_funcion*.

¡Vamos a probarlo! Vamos a <http://localhost:8000/home>

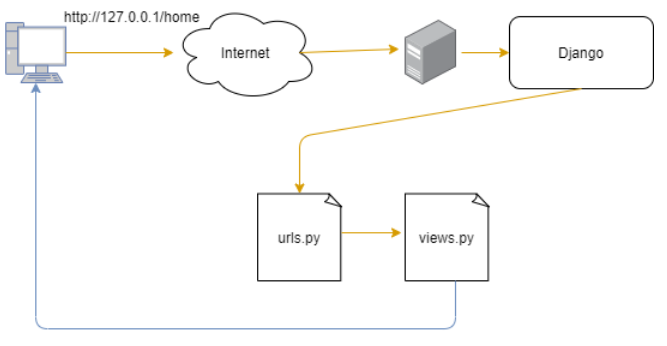


¡Perfecto! Buen trabajo, hemos conseguido crear nuestra primera página web con python usando Django.

#### **Resumen capítulo 1**

Vamos a recapitular cosas que hemos aprendido.

* Como instalar Pycharm
* Como usar entornos virtuales
* Como instalar Django
* Como crear un proyecto y una app en Django
* Como Django sirve las peticiones url.py -> views.py

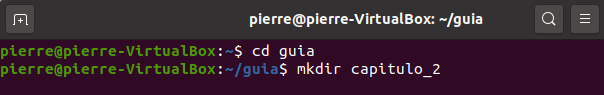


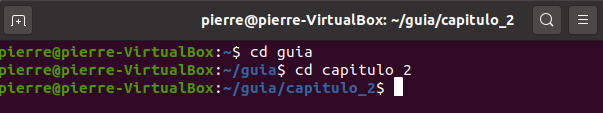
## Capítulo 2

### **Templates**

En el capítulo anterior hemos conseguido crear nuestra primera página con django. Vamos a repasar lo aprendido y expandir un poco más.

Primero de todo vamos a crear una carpeta dentro de guia llamada capitulo\_2, dentro de ella creamos run nuevo proyecto llamado cap2\_project, y un entorno virtual.





Ahora ejecutamos este comando:

python3 -m venv cap2\_projectenv

luego este otro:

source cap2\_projectenv/bin/actívate

luego instalamos en este modo:

Primero:

sudo apt update

sudo apt install libpq-dev postgresql postgresql-contrib

Segundo:

pip3 install django

pip3 install psycopg2

django-admin.py startproject cap2\_project .

python3 manage.py startapp plantillas

después este otro:

mkdir plantillas/templates

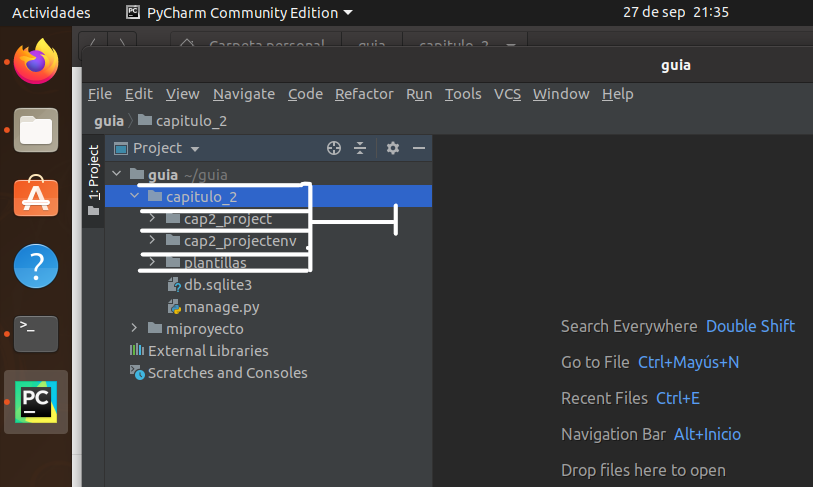
ahora este también:

touch plantillas/templates/hola\_mundo.html

en la carpeta capitulo\_2 dentro debería estar dos carpetas como estas:



Y en pycharm debería verse de esta manera:



En el archivo hola\_mundo.html ponemos el código HTML. Y ahora desde plantillas/views.py

vamos a llamar a este archivo que contiene el html en vez de tener el código html mezclado

con el código Python:

from django.shortcuts import render

import random

# Create your views here.

def home\_page\_view(request):

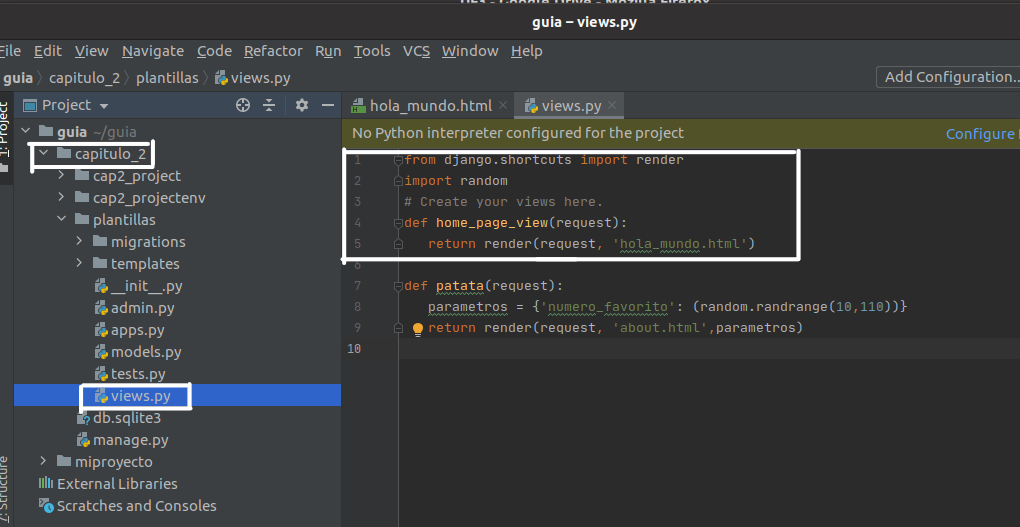
return render(request,'hola\_mundo.html')

def patata(request):

parámetros = {‘numero\_favorito’: (random.randrange(10,110))}

return render(request, 'about.html',parametros)

Tiene que quedar, así como se ve en la imagen:



Luego en hola\_mundo.html.

<!DOCTYPE html>

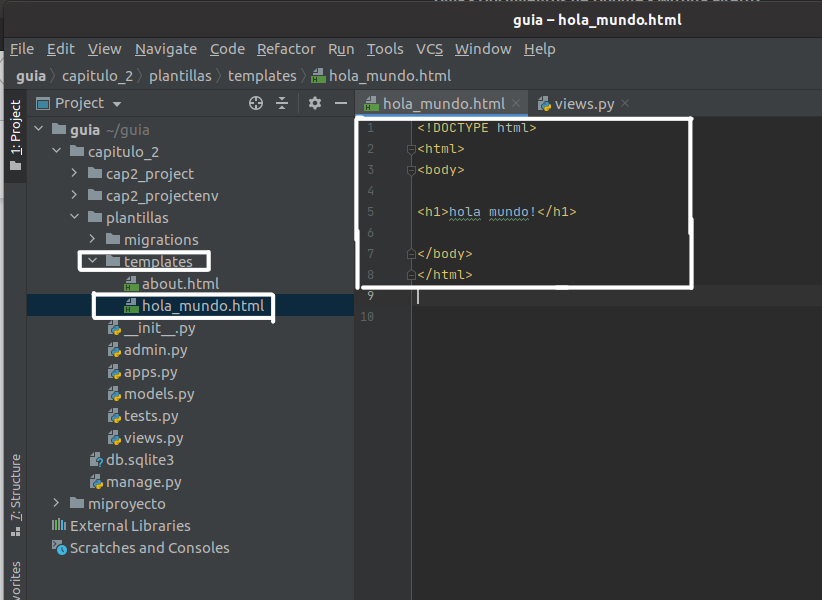
<html>

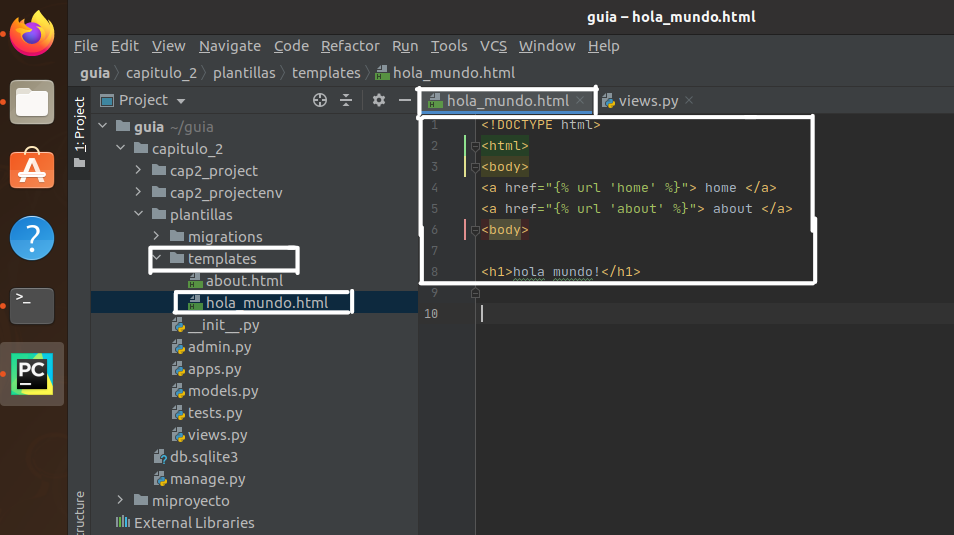
<body>

<h1>hola mundo!</h1>

</body>

</html>





<!DOCTYPE html>

<html>

<body>

<a href="{% url 'home' %}"> home </a>

<a href="{% url 'about' %}"> about </a>

<body>

<h1>hola mundo!</h1>

En la URL.py

Ponemos en el archivo *cap2\_project/urls.py* y allí añadimos un atributo *name* a cada url, este atributo le da un nombre identificador a la url.

from django.contrib import admin

from django.urls import path

from plantillas import views

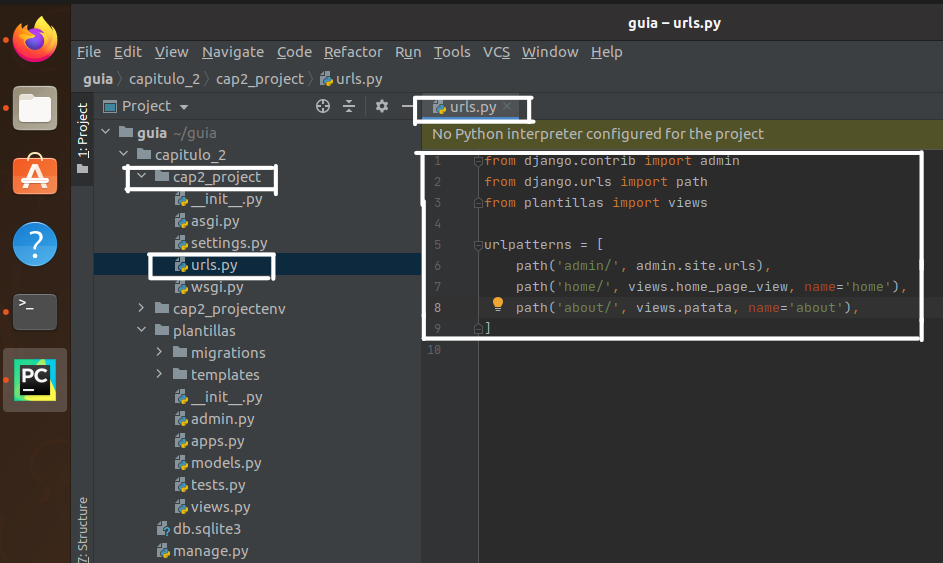
urlpatterns = [

path('admin/', admin.site.urls),

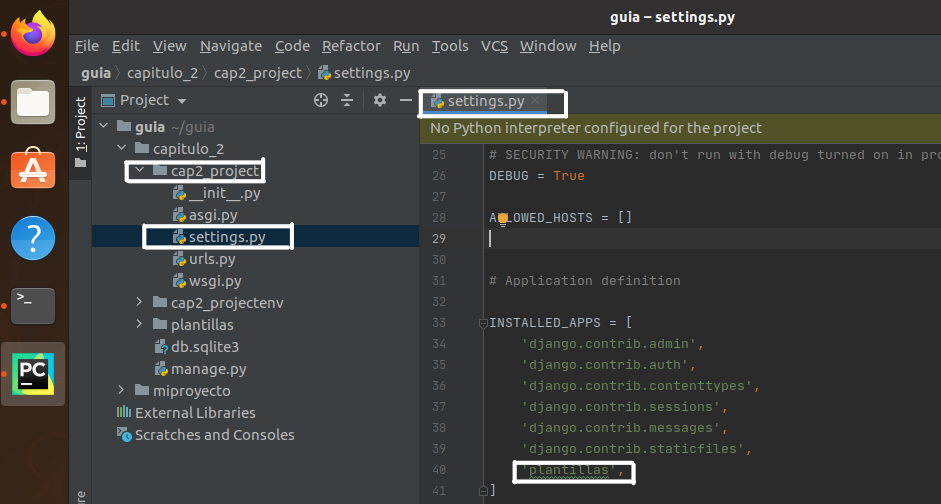
path('bienvenido', views.home\_page\_view, name='home'),

path('about', views.about\_page\_view, name='about'),

]



Crear una app tal como se ve en la imagen:



Y por último antes de probar la web y la app creamos about.html.

<!DOCTYPE html>

<html>

<body>

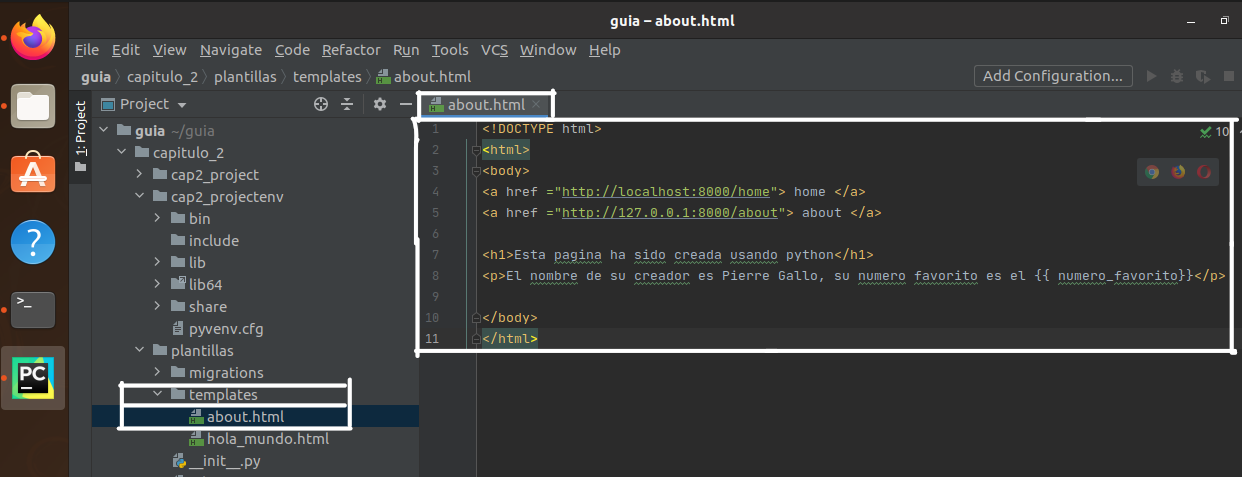
<a href="http://localhost:8000/home"> home </a>

<a href="http://127.0.0.1:8000/about"> about </a>

<h1>Esta pagina ha sido creada usando Python</h1>

<p>El nombre de su creador es Pierre Gallo, su número favorito es el {{ numero\_favorito }}</p>

</body>

</html> 

## Capítulo 3

### **Static files**

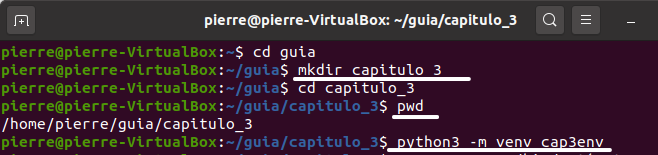
Empezamos este capítulo creando una nueva carpeta capitulo\_3, un nuevo un proyecto llamado cap3\_project y una app llamada cap3\_app, hacemos “hola mundo” con el proyecto, como ya sabéis repetitio est mater studiorum.

mkdir capitulo\_3

cd capitulo\_3

pwd

python3 -m venv cap3env



source cap3env/bin/activate

sudo apt update

sudo apt install libpq-dev postgresql postgresql-contrib

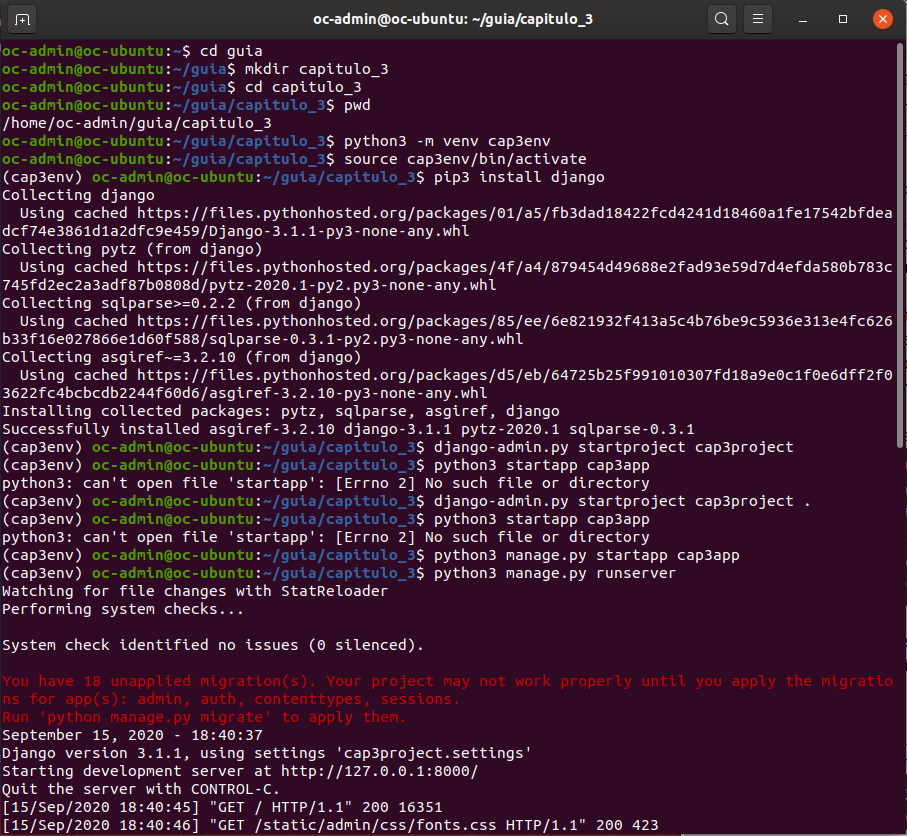
pip3 install django

pip3 install psycopg2

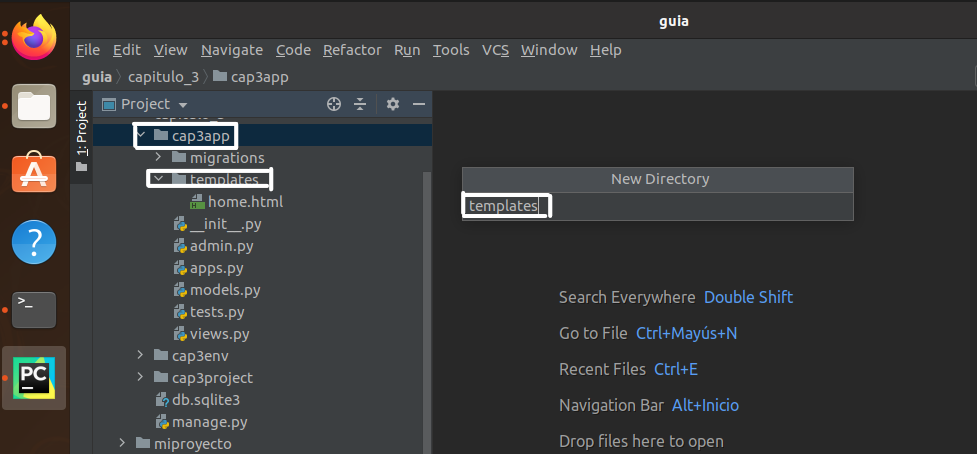
django-admin.py startproject cap3project .

python3 manage.py startapp cap3app

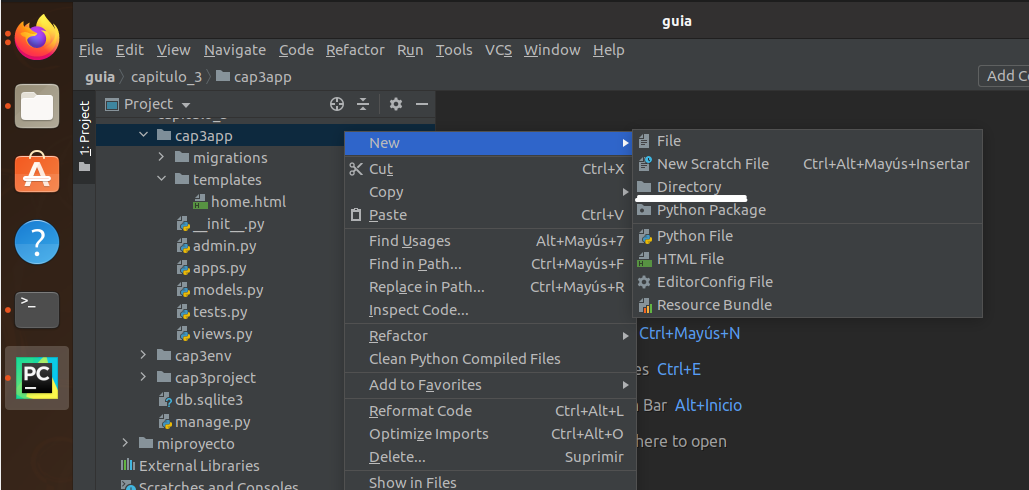
python3 manage.py runserver

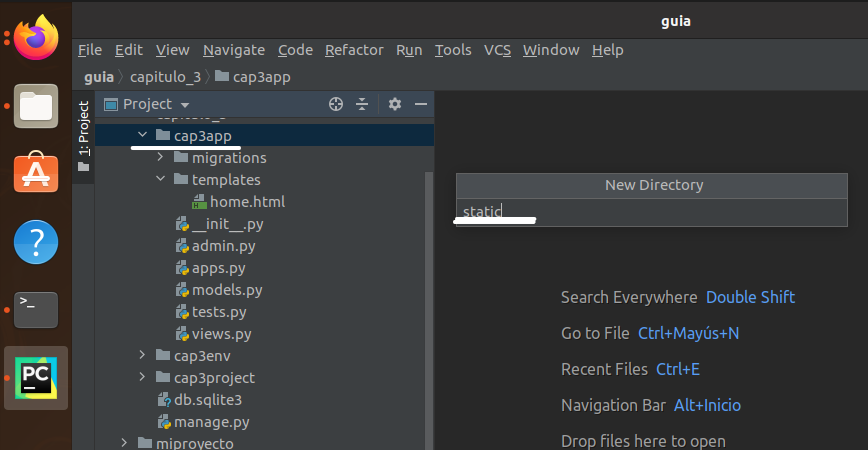


Ahora creamos una carpeta en cap3app con el nombre templates y en esta carpeta debemos alojar a home.html como se ve en la imagen

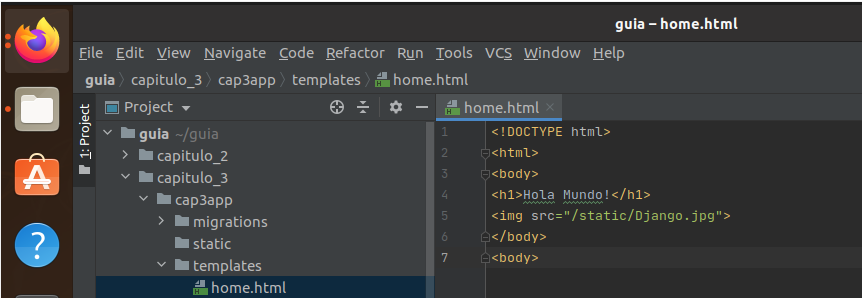


Volvemos a crear otra carpeta de nombre STATIC aquí en esta carpeta almacenaremos todas las imágenes para las webs.

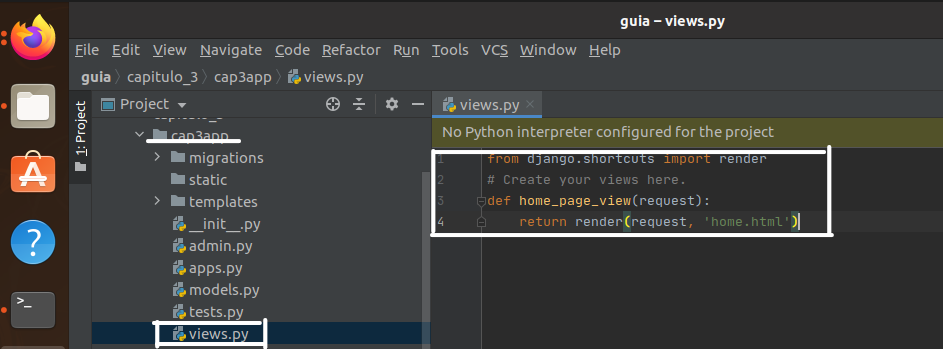




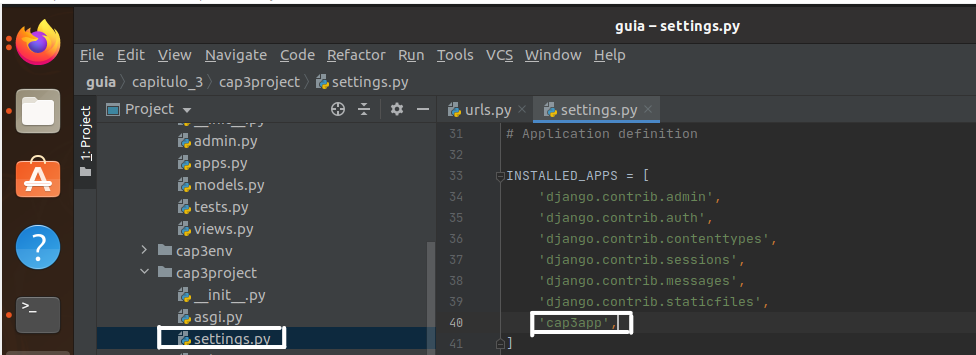
Abrimos home.html. Y ponemos el siguiente código:



Ahora en cap3app views.py.

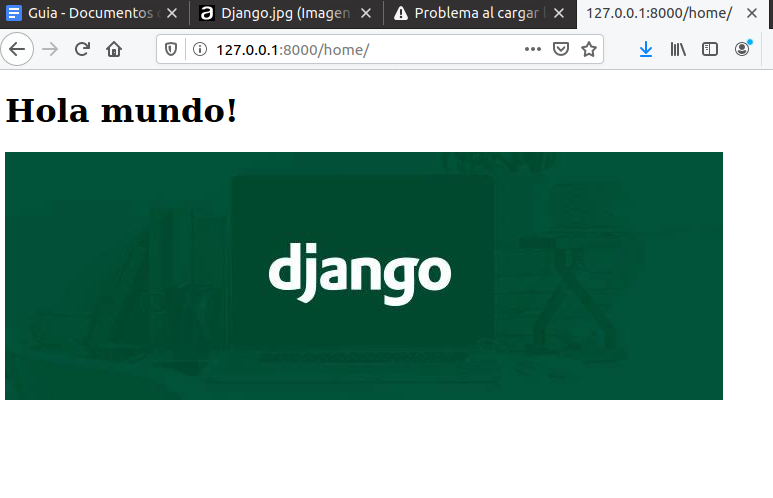


Y por último ponemos en SETTING.PY agregamos una app con el nombre ‘cap3app’ como se ve en la imagen:



Comprobamos:

127.0.0.1:8000/home/



Creamos nuestra web de recetas:

