



Clase 17. Python

Portfolio (parte 1)

***RECUERDA PONER A GRABAR LA
CLASE***





OBJETIVOS DE LA CLASE

- Realizar los comandos básicos de Django.
- Utilizar MVC creando primeras vistas.
- Aplicar el uso de templates.

CRONOGRAMA DEL CURSO

Clase 16



Git - GitHub



GITHUB

Clase 17



Django - Portfolio (parte 1)



¿EN QUÉ AÑO NACISTE?



TEMPLATE

Clase 18



Django - Portfolio (parte 2)



AGREGANDO CARGADORES



NUESTRO PRIMER MVT

Django

¿Qué es Django?

¿Qué es Django?

Ya sabemos programar en Python, es momento de empezar a aplicar un poco de todo esto para poder crear un **proyecto web**.

Para realizar un proyecto web necesitaremos usar un **framework** que nos facilite esta tarea, el más sencillo de todos los frameworks web de Python es Flask. Pero en este apartado usaremos **Django** que es mucho más completo y es de **código abierto**.

¿Qué es un Framework?

Es un entorno de trabajo, un ecosistema que nos facilita crear algo, en este caso un sitio web, sin mayores esfuerzo.



Fundamentos de Django (MVC)

¿QUÉ ES EL MODELO VISTA CONTROLADOR?



Modelo Vista Controlador

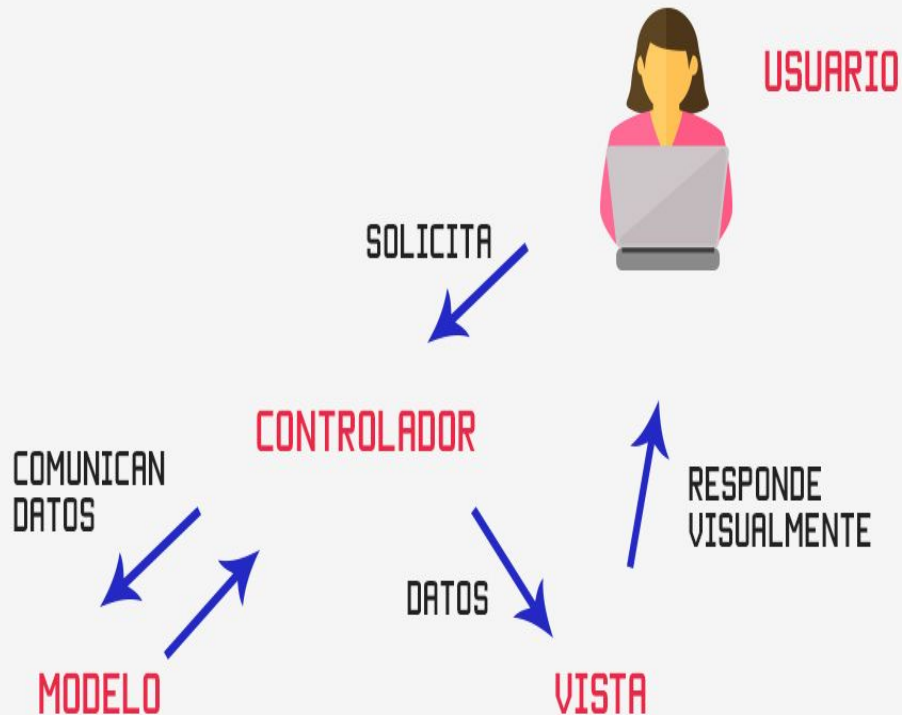
Django se base en el **patrón de diseño web**, denominado, modelo vista controlador.

Modelo: Maneja los datos.

Vista: Lo que el usuario ve.

Controlador: Interacción entre los datos y lo que ve el usuario.

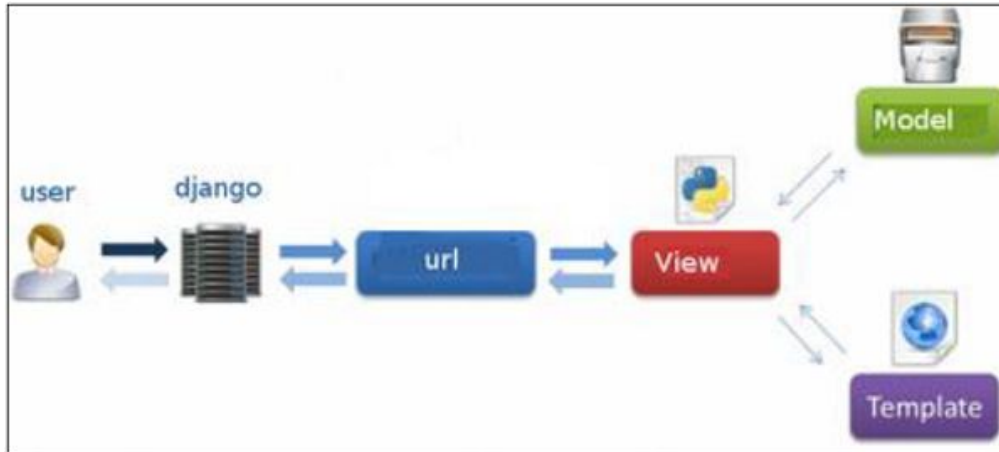
Modelo Vista Controlador



Esto nos permite que la aplicación sea más funcional, mantenible, escalable y colaborativa.

Modelo Vista Controlador

Este es el patrón de diseño clasico Web para cualquier proyecto web en cualquier lenguaje.



Django modifica este patrón por el modelo MTV, Model template, View.

Instalando Django



Instalando Django

Antes que nada tenemos que tener Python instalado, nosotros ya lo tenemos.

- Vamos a la página de Django: www.djangoproject.com
- Descargamos la última o anteúltima versión.
- Con Windows, podemos instalarlo bajo el comando `pip install Django`



Instalando Django

```
Microsoft Windows [Versión 10.0.17763.615]  
(c) 2018 Microsoft Corporation. Todos los derechos reservados.  
C:\WINDOWS\system32>pip install Django
```

1. Desde el cmd o desde la terminal de VCS

```
Collecting Django==2.2.3  
  Downloading https://files.pythonhosted.org/packages/39/b0/2138c31...  
/Django-2.2.3-py3-none-any.whl (7.5MB)  
    100% |#####| 7.5MB 141kB/s  
Collecting pytz (from Django==2.2.3)  
  Downloading https://files.pythonhosted.org/packages/3d/73/fe30c2d...  
/pytz-2019.1-py2.py3-none-any.whl (510kB)  
    100% |#####| 512kB 1.4MB/s  
Collecting sqlparse (from Django==2.2.3)  
  Downloading https://files.pythonhosted.org/packages/ef/53/900f7d2...  
/sqlparse-0.3.0-py2.py3-none-any.whl  
Installing collected packages: pytz, sqlparse, Django  
Successfully installed Django-2.2.3 pytz-2019.1 sqlparse-0.3.0
```

2. El proceso suele tardar alrededor de 3 a 5 minutos.



3. Comprobar que se instalo bien:

```
import django
```

django.VERSION

```

Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.1-py3-none-any.whl (42 kB)
Installing collected packages: sqlparse, pytz, asgiref, django
Successfully installed asgiref-3.4.1 django-3.2.6 pytz-2021.1 sqlparse-0.4.1
PS C:\Users\nico\Desktop> python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.VERSION
(3, 2, 6, 'final', 0)

```


Crear proyecto



Pasos para crear nuestro proyecto

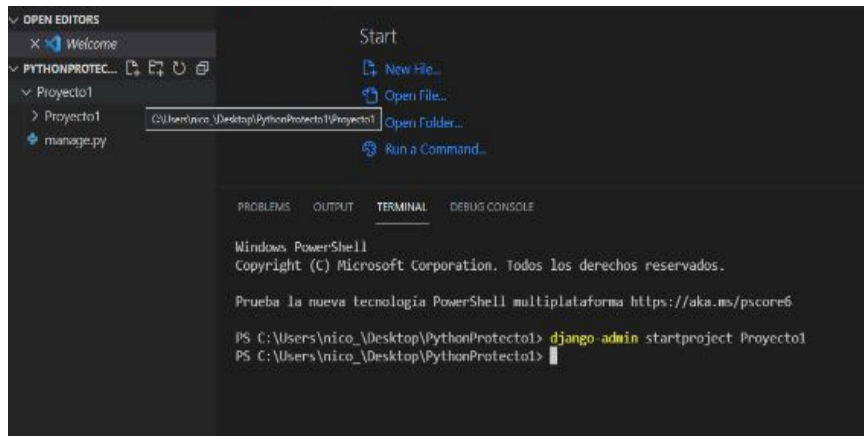


- 1 - Crear una carpeta, en mi caso, escritorio, **PythonProyecto1**
- 2- Iniciar VSC o tu editor (o incluso cmd) y pararte en esa carpeta.
- 3 - Escribir **`django-admin startproject Proyecto1`**

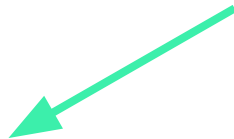
“Donde Proyecto1 será una nueva carpeta que se creará en
PythonProyecto1”



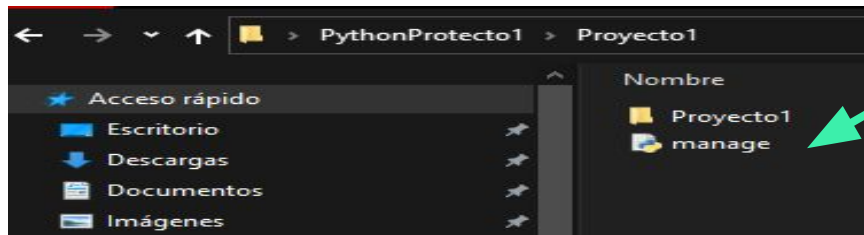
Pasos para crear nuestro proyecto



Hasta acá deberíamos tener algo así.



Con este comando se creará una carpeta y un archivo **manage**.





Manage es importante porque nos permite interactuar con los proyectos, con algunos comandos muy simples. Por ejemplo:
ayudas,migraciones, testeos, etc.

En la carpeta **Proyecto1** veremos que hay varios archivos **.py**.
__init__.py para que sepa que es un paquete, **__settings.py** para manipular la configuración, y **url/wsgi** que pronto sabremos su uso.



Pasos para crear nuestro proyecto



```
PS C:\Users\nico\Desktop\PythonProtecto1> cd Proyecto1
PS C:\Users\nico\Desktop\PythonProtecto1\Proyecto1> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, 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 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 sessions.0001_initial... OK
PS C:\Users\nico\Desktop\PythonProtecto1\Proyecto1> |
```

4- Nos paramos en nuestro proyecto,
Proyecto1, `cd Proyecto1`.

5- Tipeamos `python manage.py migrate`.

6- Verificamos con:

`python manage.py runserver`



Pasos para crear nuestro proyecto



django

Ver [notas de la versión](#) de Django 3.2



¡La instalación funcionó correctamente!
¡Felicidades!

Estás viendo esta página porque `DEBUG = True` está en tu
archivo de configuración y no has configurado ninguna
URL.

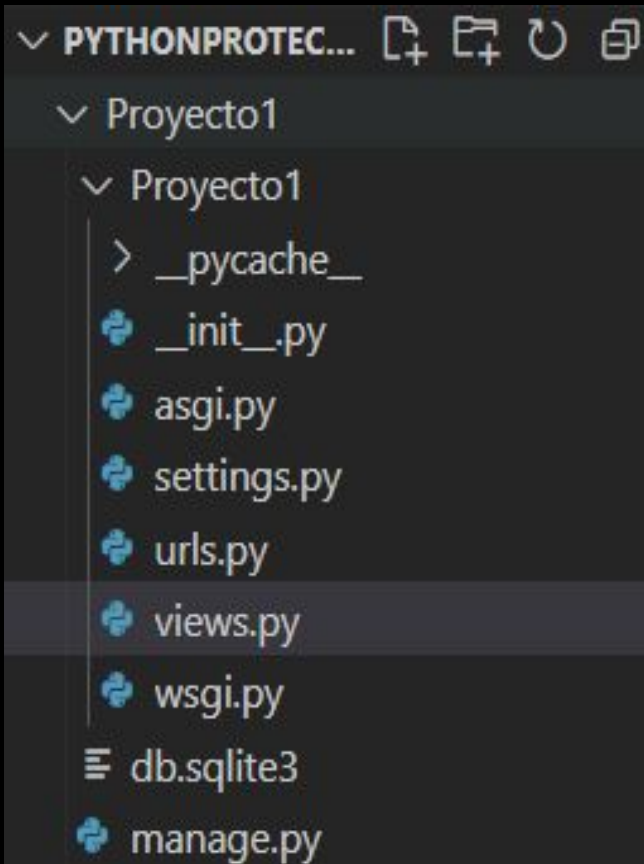
¡Tenemos nuestro primer
proyecto funcionando! 🙌



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

Nuestro primer view



Nuestro primer view



Lo primero que tendremos que hacer es crear un archivo para esta nueva vista o view.

Usualmente se suele llamar **view.py**, debemos crearlo en la carpeta del proyecto, es decir en la misma ruta que tenemos **urls**, **__init__**, **wsgi**, etc.

Debería quedar así



Nuestro primer view



Vamos a nuestro archivo **views.py**, e importamos los elementos de un **Response** de la siguiente manera:

```
from django.http import  
HttpResponse
```

Nuestro primer view



Luego iniciamos creando nuestra primer vista, por medio de un método que recibe como parámetro la **request** y nos da por resultado un **response**:



```
def saludo(request):  
    return HttpResponse("Hola  
Django - Coder")
```



Nuestro primer view

Listo, ahora solo necesitamos avisarle a Python y Django cual será la URL que nos llevará a la vista que creamos. Eso lo hacemos en el archivo **urls.py**.

```
from django.contrib import admin
from django.urls import path
from Proyecto1.views import saludo #Para acceder a la vista hay que importar el modulo y el método

urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo), #ojo la url no hace falta que se llame saludo/ el nombre es libre,
                           #pero la vista sí debe llamarse por su nombre
]
```



Nuestro primer view

Para esto debemos primero importar **Proyecto1.views** y generar el vínculo entre una url y la vista, nos debería quedar así:

```
from django.contrib import admin
from django.urls import path
from Proyecto1.views import saludo #Para acceder a la vista hay que importar el modulo y el método

urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo), #ojo la url no hace falta que se llame saludo/ el nombre es libre,
                           #pero la vista sí debe llamarse por su nombre
]
```



Nuestro primer view

Solo resta probar que funcione lo que hicimos 👁️👁️

Para esto nos apoyamos en lo visto anteriormente y arrancamos el servidor de la siguiente manera:

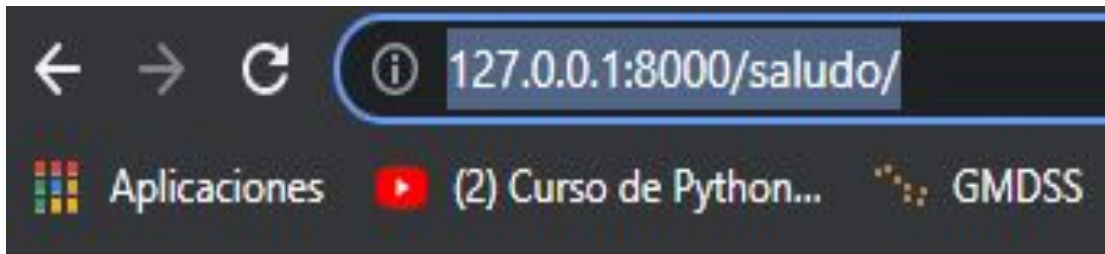
```
python manage.py runserver
```



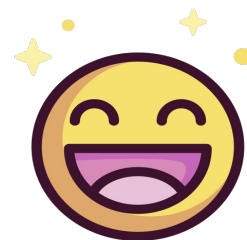
Nuestro primer view

Luego entramos a: <http://127.0.0.1:8000/saludo/>

Ya podemos ver nuestra primer página, o mejor dicho nuestra vista



Hola Django - Coder



Agregar otra view



Agregar otra view

Hagamos lo mismo para ver lo sencillo que ha sido

```
from django.http import HttpResponse

def saludo(request): #Nuestra primera vista :)
    return HttpResponse("Hola Django - Coder")
```

```
def segunda_vista(request):
    return HttpResponse("<br><br>Ya entendimos esto, es muy simple :) ")
```

```
from django.contrib import admin
from django.urls import path
from Proyecto1.views import saludo, segunda_vista

urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo), #ojo la url no hace falta el nombre de la vista
                             #pero la vista sí debe tener el nombre
    path('segundavista/', segunda_vista),
]
```

Podemos poner comandos HTML 🧐

Pasaje de parámetros

¿Qué es?

Muchas veces queremos mostrar en una vista el resultado de algún proceso interno realizado en Python, esto es básicamente **enviar parametros por medio de la vista.**

Veamos cómo se hace, es muy simple y parecido a la anterior.



Ejemplo



Así es el proceso de creación de una nueva vista que muestra **DíaDeHoy**.

```
def diaDeHoy(request):  
    dia = datetime.datetime.now()  
  
    documentoDeTexto = f"Hoy es día: <br> {dia}"  
  
    return HttpResponse(documentoDeTexto)
```

```
from django.contrib import admin  
from django.urls import path  
from Proyecto1.views import saludo, segunda_vista, diaDeHoy  
  
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('saludo/', saludo), #ojo la url no hace falta que  
                             #pero la vista sí debe llamar  
  
    path('segundavista/', segunda_vista),  
    path('diaDeHoy/', diaDeHoy),  
]
```



Hoy es día:
2021-09-01 20:07:54.838810

Parámetros desde la url

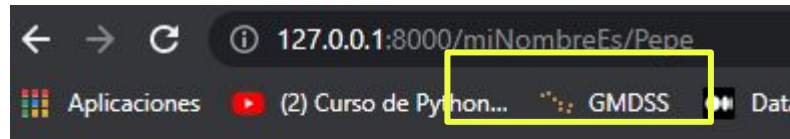
¿Qué es?



Mi vista puede trabajar sobre algún dato que nos envíen por **url**, veamos cómo funciona así la respuesta aparece sola:

```
def miNombreEs(self, nombre):  
    documentoDeTexto = f"Mi nombre es: <br><br> {nombre}"  
    return HttpResponse(documentoDeTexto)
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('saludo/', saludo), #ojo la url no ha  
                             #pero la vista s  
    path('segundavista/', segunda_vista),  
    path('diaDeHoy/', diaDeHoy),  
    path('miNombreEs/<nombre>', miNombreEs),  
]
```



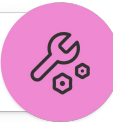
Mi nombre es:

Pepe



¿EN QUÉ AÑO NACISTE?

Calcular el año de nacimiento.



¿En qué año naciste?

Enviar por parametro tu edad y calcular el año de nacimiento
(más o menos uno, para no entrar con los meses).

Tiempo estimado: 10 min.

Plantillas Django

Plantillas

¿Qué son?

Son archivos que nos permiten separar la vista de la estética, es decir guardar en un archivo separado de todo lo que guardabamos en “documento”, para así enviar por la **HttpsResponse**.

Entonces, ¿podríamos decir que así se crea un template? 😊

¡Exacto! 🙌

Recordemos que Django se basaba en **Modelo, Vista, Template**.



TEMPLATE

Creando nuestro primer template

TIEMPO: 20 MIN



ACUERDOS



Presencia



Escucha Activa



**Apertura al
aprendizaje**



Todas las voces

template

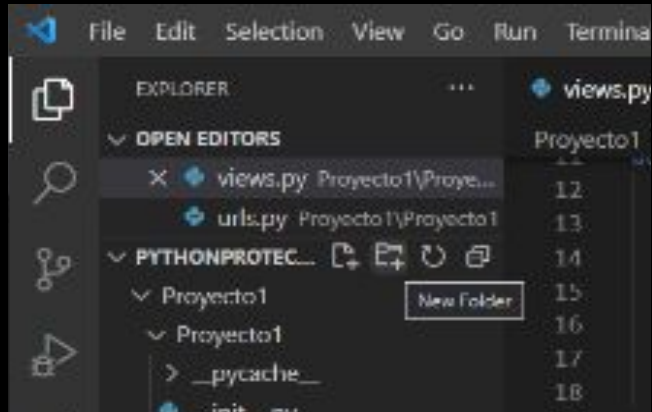


Consigna: Vamos a crear nuestro primer template. Esto necesitará de un “template”, propiamente dicho, es decir lo que se muestra. Además necesitaremos un “contexto”, esto es para manejar contenido que cambia, por ejemplo nuestro nombre en el ejercicio anterior. Y por último crear un “render” es decir una transformación a pagina web.

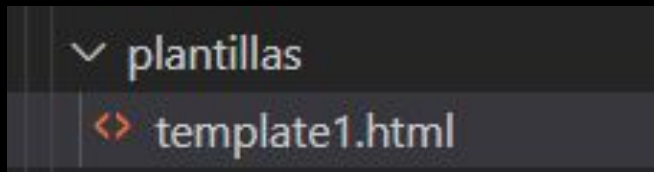
NOTA: usaremos los breakouts rooms. El tutor/a tendrá el rol de facilitador/a.



1)



2)



Paso a paso

- 1) Dentro del proyecto creamos una carpeta que se puede llamar plantillas.
- 2) Dentro de esa nueva carpeta creamos un archivo `template1.html`.



Paso a paso

3) Dentro de `template1`, escribimos `html:5` y apretamos enter.
Se crea automáticamente un esqueleto.

```
views.py  template1.html  urls.py  
Proyecto1 > Proyecto1 > plantillas > template1.html  
1  html:5
```




Paso a paso

views.py X template1.html ● urls.py

Proyecto1 > Proyecto1 > plantillas > template1.html > html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

4) Dentro de `<body>`

`</body>`, escribimos lo

que queremos que se vea
en nuestra página web.

```
<!DOCTYPE html>
v <html lang="en">
v <head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
v <body>

  Excelente!!!!!!.... Es muy fácil usar plantillas.

</body>
</html>
```

Paso a paso



```
def probandoTemplate(self):  
  
    miHtml = open("C:/Users/nico/Desktop/PythonProtecto1/Proyecto1/Proyecto1/plantillas/template1.html")  
  
    plantilla = Template(miHtml.read()) #Se carga en memoria nuestro documento, template1  
    ##OJO importar template y contex, con: from django.template import Template, Context  
  
    miHtml.close() #Cerramos el archivo  
  
    miContexto = Context() #EN este caso no hay nada ya que no hay parametros, IGUAL hay que crearlo  
  
    documento = plantilla.render(miContexto) #Aca renderizamos la plantilla en documento  
  
    return HttpResponse(documento)
```

5) a- Llamamos a **template1** desde una nueva vista (**probandoTemplate**),



Paso a paso

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('saludo/', saludo),    #ojo la url no hace  
                                #pero la vista sí  
  
    path('segundavista/', segunda_vista),  
    path('diaDeHoy/', diaDeHoy),  
    path('miNombreEs/<nombre>', miNombreEs),  
    path('probandoTemplate/', probandoTemplate),  
]
```

5) b- Aquí crearemos el contexto y el render.

← → ↺ ⓘ 127.0.0.1:8000/probandoTemplate/

Aplicaciones (2) Curso de Python... GMDSS

Excelente !!!!!!! Es muy fácil usar plantillas.

¿Qué hemos logrado?

Logramos separar lo que construye a la página web, es decir el **HTML**, del procesamiento de los datos dado por la vista. De esta manera podemos tener un diseñador trabajando en el HTML y nosotros desarrollando en **Python/Django** sin saber nada de HTML.



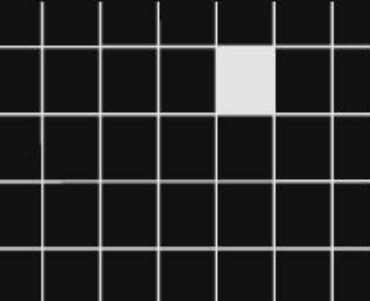
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Crear primer proyecto en Django.
 - Crear primeras vistas
 - Relacionar un template con una vista.
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE