



DESARROLLO CON PYTHON

Contenido de formación



Programación



Pyhton



Paradigma POO



Bases de datos SQL



Tkinter



Django



Flask

1. Primeros Instalación de SW
2. Variables y tipos de datos
3. Operadores (aritméticos, asignación)
4. Entrada y salida de datos
5. Estructuras de control
 - 5.1 Condicionales (operadores lógicos, operadores de comparación)
 - 5.2 Bucles (estructuras interactivas for, while)
6. Bloque de ejercicios de aplicación de conceptos
7. Funciones (parámetros, return, invocación, lambda)
 - 7.1 Variables locales y globales, funciones y métodos predefinidos
8. Lista y tuplas (creación, índices, recorrer y mostrar listas, listas multidimensionales)
9. Diccionarios y sets
10. Bloque de ejercicios aplicación de conceptos
11. Módulos y paquetes (creación y funcionalidad)
12. Sistemas de archivos y directorios
13. Manejo de errores (captura de excepciones, errores personalizados)
14. Programación orientada a objetos
15. Bases de datos SQLite
16. Bases de datos MySQL
17. Proyecto con Python
18. Interfaces graficas con Tkinter (aplicación de escritorio Python – Tkinter)
19. Desarrollo Web con Django
20. Interfaces graficas con Flask (aplicación de escritorio Python – Flask)



Django

Python es el quinto lenguaje de programación en popularidad en estos momentos. Se trata de un lenguaje que se emplea, sobre todo, en la programación para las distintas aplicaciones de análisis de datos e IA, pero que también es importante para la programación web. En este contexto, el trabajo de los programadores es mucho más sencillo con herramientas como Django, un **framework** específico para el trabajo con Python.

Django es un framework de desarrollo para Python que se emplea para la creación de páginas web. Se trata de una herramienta de código abierto y gratuita que cuenta con una comunidad amplia y que comparte recursos constantemente. Además, Django también cuenta con funciones de pago que pueden facilitar más el trabajo de los desarrolladores.

Django es una herramienta que se puede usar para el desarrollo full-stack de aplicaciones y páginas web, así como para el desarrollo de servidores. Está considerado como el mejor framework para el desarrollo de aplicaciones web con Python y es uno de los marcos de desarrollo más demandados por los programadores que trabajan con este lenguaje en el desarrollo web.

Creado por Adrian Holovaty y Simon Willison mientras estaban trabajando en PHP y necesitaban algo que les facilitase actualizar una web de manera rápida para cumplir con los plazos. En ese momento se pasaron a Python y, en 2005, lanzaron Django para que la programación fuese aún más sencilla.

Estas son algunas de las características por las que Django es uno de los frameworks más usados para el desarrollo web:

Completo

Django proporciona casi todo lo que los programadores necesitan y pueden querer usar. Se trata de una herramienta que sigue unos principios de diseño consistentes y que cuenta con una buena base de documentación para facilitar el trabajo de los desarrolladores.

Escalable

Django funciona por componentes sustituibles e intercambiables. Eso significa que se puede escalar con bastante facilidad. En este sentido, un ejemplo de uso de Django lo tenemos en Instagram o Disqus, plataformas que lo han empleado para mejorar sus servidores gracias a la escalabilidad del entorno de desarrollo.

Versátil

Este entorno de desarrollo se ha empleado para la creación de todo tipo de páginas web. Desde sistemas que son puramente para administración de contenidos, como puede ser, por ejemplo, una wiki; hasta redes sociales o páginas webs de noticias.

Y es que, Django es compatible con cualquier framework que se emplea del lado del cliente, por lo que puede mandar contenidos en cualquier formato.

Seguro

Django facilita la detección y solución de posibles fallos en la seguridad en las páginas web del lado del servidor. Proporciona una administración segura de usuarios y contraseñas y evita errores que son comunes en el diseño y desarrollo back-end.

Portátil

Django se puede usar en cualquier sistema y plataforma y, además, está respaldado por muchos de los proveedores de hosting que, además, suelen proporcionar la documentación necesaria para implementarlo.

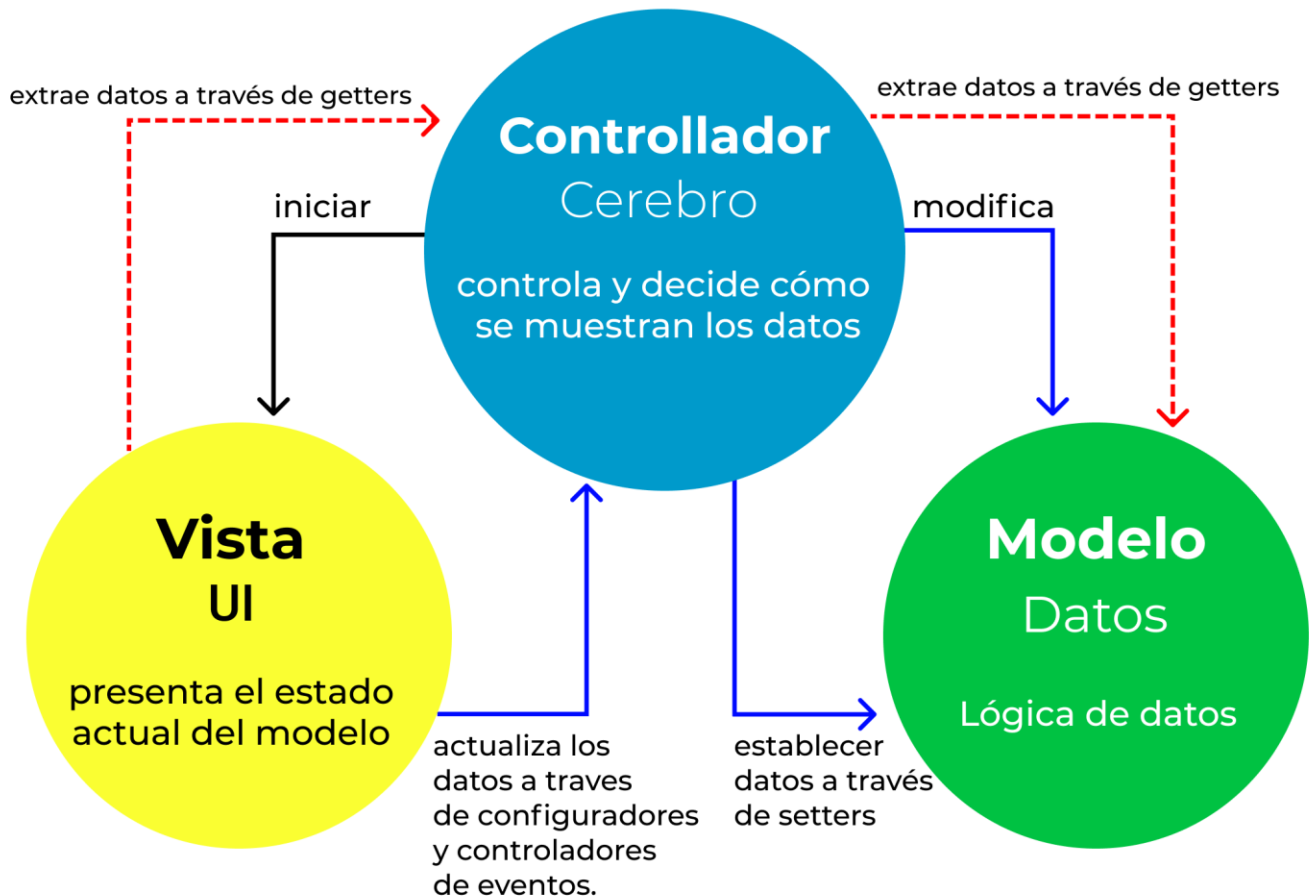


Mantenimiento

Django facilita el mantenimiento y la velocidad del mismo en el desarrollo web. Fomenta la creación de código reutilizable, lo cual simplifica, en gran medida, el trabajo de los desarrolladores web.

Usamos el modelo, vista controlador

Patrones de Arquitectura MVC



Instalación de Django

Cmd

1. Comprobar si tenemos Python

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\MGV>python --version
Python 3.10.7

C:\Users\MGV>
```





2. Instalar Django

```
Símbolo del sistema
C:\Users\MGV>pip install Django==4.1.6
Collecting Django==4.1.6
  Downloading Django-4.1.6-py3-none-any.whl (8.1 MB)
    ----- 8.1/8.1 MB 1.4 MB/s eta 0:00:00
Collecting asgiref<4,>=3.5.2
  Downloading asgiref-3.6.0-py3-none-any.whl (23 kB)
Collecting sqlparse>=0.2.2
  Downloading sqlparse-0.4.3-py3-none-any.whl (42 kB)
    ----- 42.8/42.8 2.0 MB/s eta 0:00:00
    kB
Collecting tzdata
  Downloading tzdata-2022.7-py2.py3-none-any.whl (340 kB)
    ----- 340.1/340.1 1.8 MB/s eta 0:00:00
    kB
Installing collected packages: tzdata, sqlparse, asgiref, Django
Successfully installed Django-4.1.6 asgiref-3.6.0 sqlparse-0.4.3 tzdata-2022.7

[notice] A new release of pip available: 22.3.1 -> 23.0
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\MGV>
```

Creamos en VsCode una carpeta 22-django luego vamos al cmd para ubicarnos en la carpeta creada

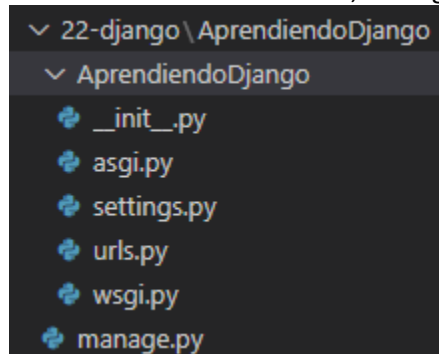
```
c:\xampp\htdocs\sena-python>cd 22-django

c:\xampp\htdocs\sena-python\22-django>
```

Iniciamos un nuevo proyecto en Django **AprendiendoDjango**

```
Símbolo del sistema
c:\xampp\htdocs\sena-python\22-django>django-admin startproject AprendiendoDjango
c:\xampp\htdocs\sena-python\22-django>
```

Como se visualiza en VsCode, manage.py es un archivo de comandos



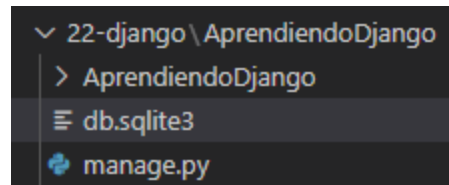
Vamos a visualizar el proyecto un navegador web, para lo cual debo migrar el proyecto, y lo que hace es generar una base de datos con la funcionalidad de las aplicaciones que vienen por defecto dentro de Django.



```
c:\xampp\htdocs\sena-python\22-django\AprendiendoDjango>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

c:\xampp\htdocs\sena-python\22-django\AprendiendoDjango>
```

Y visualizamos el cambio en VsCode



Ahora arrancamos el servidor local de Django, esta consola siempre la debemos tener abierta en segundo plano

```
Seleccionar Símbolo del sistema - python manage.py runserver

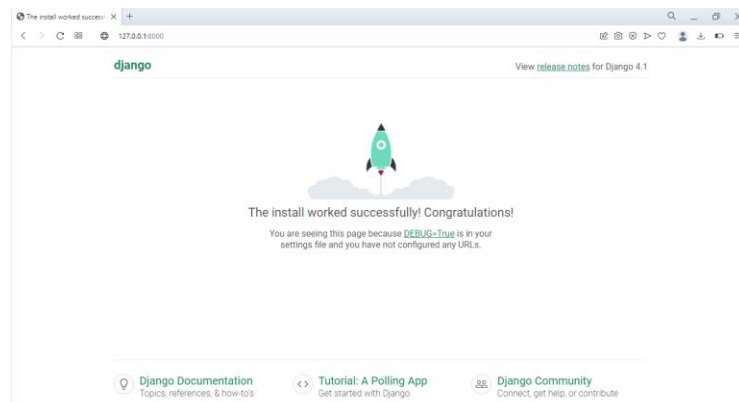
c:\xampp\htdocs\sena-python\22-django\AprendiendoDjango>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 11, 2023 - 13:01:19
Django version 4.1.6, using settings 'AprendiendoDjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

En donde se ejecuta

Dirección del servidor

Ahora lo corremos en el navegador **<http://127.0.0.1:8000/>**:



Instructor: Ing. Moisés García Vargas



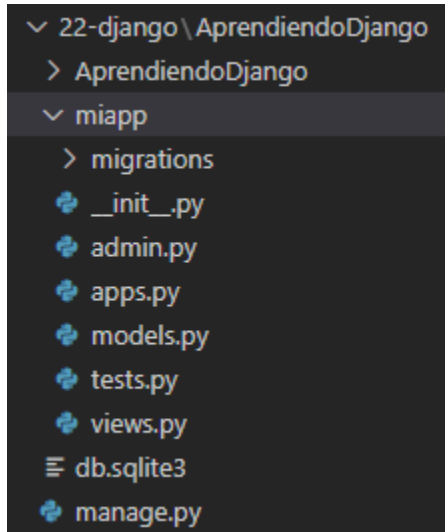


3. Crear apps con Django

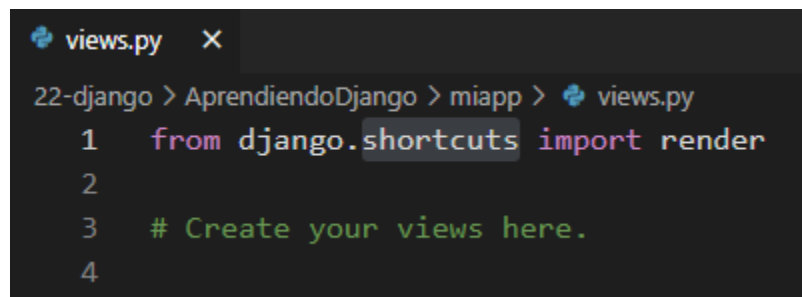
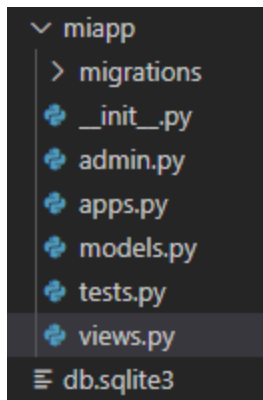
Simbolo del sistema

```
c:\xampp\htdocs\sena-python\22-django\AprendiendoDjango>python manage.py startapp miapp  
c:\xampp\htdocs\sena-python\22-django\AprendiendoDjango>
```

Como de visualiza en VsCode



4. Creación de vistas y rutas, esto es dentro del fichero **views.py**



Diferencia entre MVC y MVT

MVC = MODELO

VISTA

CONTROLADOR

MVT = MODELO

TEMPLATE

VISTA → así se maneja en Django

Las siglas en ingles MTV corresponden a Model (Modelo), Template (Plantilla) y View (Vista). En este post me referiré a las plantillas como templates, ya que su uso en el español es bastante común.

En la practica el patrón MTV es muy similar al MVC a tal punto que se puede decir que Django es un framework MVC. Realmente este no se desvía demasiado del patrón Modelo Vista Controlador, simplemente lo implementa de una manera distinta y para evitar confusiones es llamado MTV.





Modelo Vista Controlador (MVC)

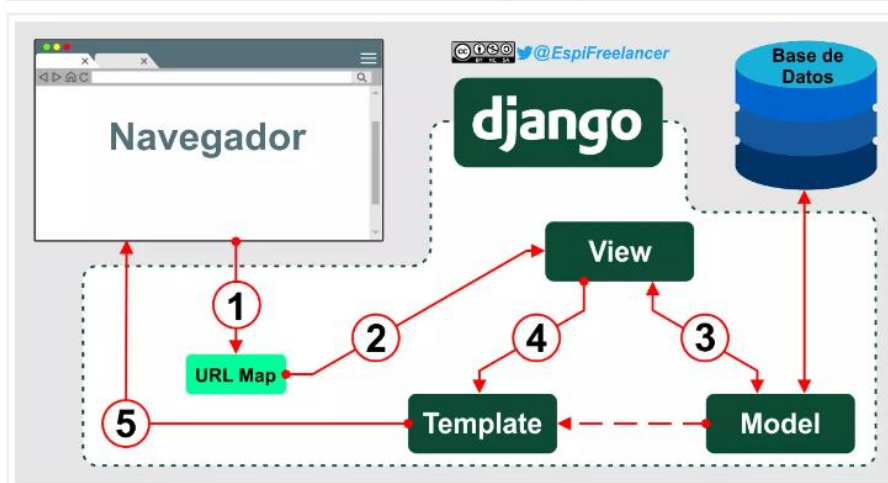
- Modelo: Es el que se encarga de manipular la información de la aplicación, la cual usualmente esta almacenada en la base de datos.
- Vista: Decide qué información mostrar y cómo mostrarla.
- Controlador: Es quien responde a las peticiones, decide que vista usar y si es necesaria información accede al modelo.

Model Template Vista (MTV)

En Django, el controlador sigue estando presente, nada más que de una manera intrínseca, ya que todo el framework Django es el controlador.

- Modelo: Maneja todo lo relacionado con la información, esto incluye como acceder a esta, la validación, relación entre los datos y su comportamiento.
- Vista: Es un enlace entre el modelo y el template. Decide qué información será mostrada y por cual template.
- Template: Decide como será mostrada la información.

Funcionamiento MTV de Django

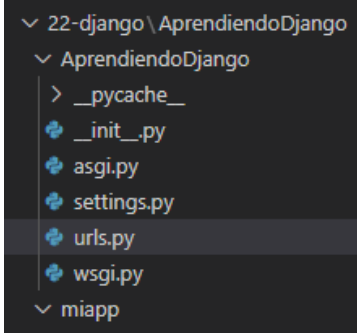


Creación de nuestra primera vista

```
views.py X
22-django > AprendiendoDjango > miapp > views.py
1  from django.shortcuts import render, HttpResponseRedirect
2
3  # Create your views here.
4  #request es un parametro que permite recibir datos de una URL
5  #Se le pasa a cada una de nuestras vistas
6
7  def hola_mundo(request):
8      #respuesta HTTP que se debe importar HttpResponseRedirect
9      return HttpResponseRedirect("Hola Mundo con Django!!!")
10
```

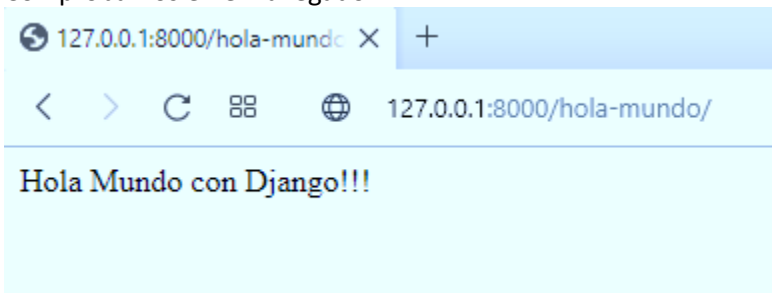


Configuración de la URL



```
urls.py x
22-django > AprendiendoDjango > AprendiendoDjango > urls.py
12 including another URLconf
13 1. Import the include() function: from django.urls import include
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 #Importar app con mis vistas
20 from miapp import views
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     #creacion de ruta
25     path('hola-mundo/', views.hola_mundo, name="hola_mundo")
26 ]
```

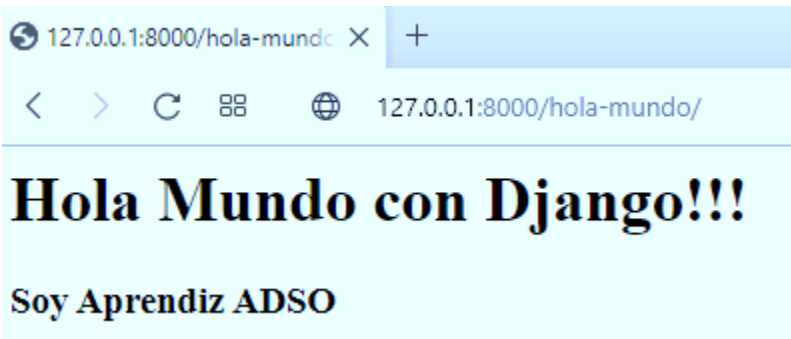
Comprobamos en el navegador



Algunos cambios

```
views.py x
22-django > AprendiendoDjango > miapp > views.py
5 #Se le pasa a cada una de nuestras vistas
6
7 def hola_mundo(request):
8     #respuesta HTTP que se debe importar HttpResponse
9     return HttpResponse("""
10     <h1>Hola Mundo con Django!!!</h1>
11     <h3>Soy Aprendiz ADSO</h3>
12     """)
13
```



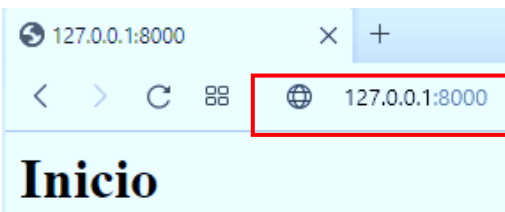


Creación de un método Inicio → *index*

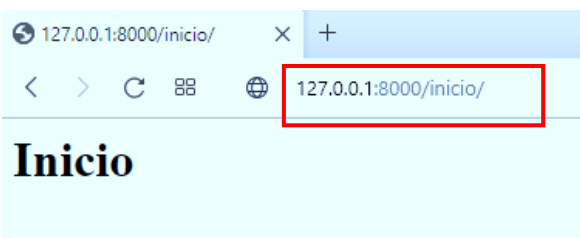
```
views.py X
22-django > AprendiendoDjango > miapp > views.py
1 from django.shortcuts import render, HttpResponse
2
3 def index(request):
4     return HttpResponse("""
5         <h1>Inicio</h1>
6     """)
7
8 # Create your views here.
9 #request es un parametro que permite recibir datos de una URL
10 #Se le pasa a cada una de nuestras vistas
```

Ahora para mostrarla y convertirla en la página de inicio

```
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     #ruta para pagina de inicio -> index
25     path('', views.index, name="index"),
26     #creacion de ruta
27     path('hola-mundo/', views.hola_mundo, name="hola_mundo")
28 ]
```



```
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     #ruta para pagina de inicio -> index
25     path('', views.index, name="index"),
26     #Otro cambio
27     path('inicio/', views.index, name="inicio"),
28     #creacion de ruta
29     path('hola-mundo/', views.hola_mundo, name="hola_mundo")
30 ]
```





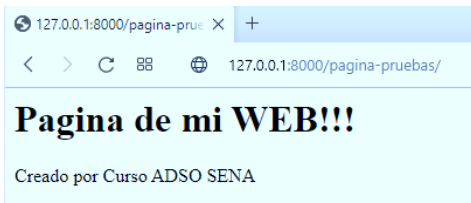
Ahora vamos a crear una que se llame pagina

```
17 def pagina(request):
18     return HttpResponse("""
19         <h1>Pagina de mi WEB!!!</h1>
20         <p>Creado por Curso ADSO SENA</p>
21     """)
```

views

```
27 path('inicio/', views.index, name="inicio"),
28 #creacion de ruta
29 path('hola-mundo/', views.hola mundo, name="hola mundo")
30 path('pagina-pruebas/', views.pagina, name="pagina")
31 ]
```

urls



Navegación entre rutas

Creación de plantilla Layout

Definición del Views

```
1 from django.shortcuts import render, HttpResponse
2
3 layout = """
4 <h1>Sitio WEb con Django | Aprendices ADSO</h1>
5 <hr/>
6 <ul>
7     <li>
8         <a href="/inicio">Inicio</a>
9     </li>
10    <li>
11        <a href="/hola-mundo">Hola Mundo</a>
12    </li>
13    <li>
14        <a href="/pagina-pruebas">Página de Pruebas</a>
15    </li>
16 </ul>
17 <hr/>
18 """
19
20 def index(request):
```

Y esto se le concatena a todos losHttpResponse(layout+

```
20 def index(request):
21     return HttpResponse(layout+"""
22     <h1>Inicio</h1>
```





Parámetros en rutas:

Creamos este método

```
40 def contacto(request, nombre):
41     return HttpResponse(layout+f"<h2>Contacto {nombre}</h2>")
```

Pasar parámetro

```
30 path('pagina-pruebas/', views.pagina, name="pagina"),
31 path('contacto/<str:nombre>', views.contacto, name="contacto")
```



Plantillas y templates en Django

Creamos dentro de **miapp** una carpeta llamada **templates** y creamos dentro un fichero que se llamara **index.html** igual que el método.

```
<> index.html x views.py
22-django > AprendiendoDjango > miapp > templates > <> index.html
1 <h1> Inicio</h1>
2 <p>ADSO SENA 2023 - Regional CAUCA:</p>
```

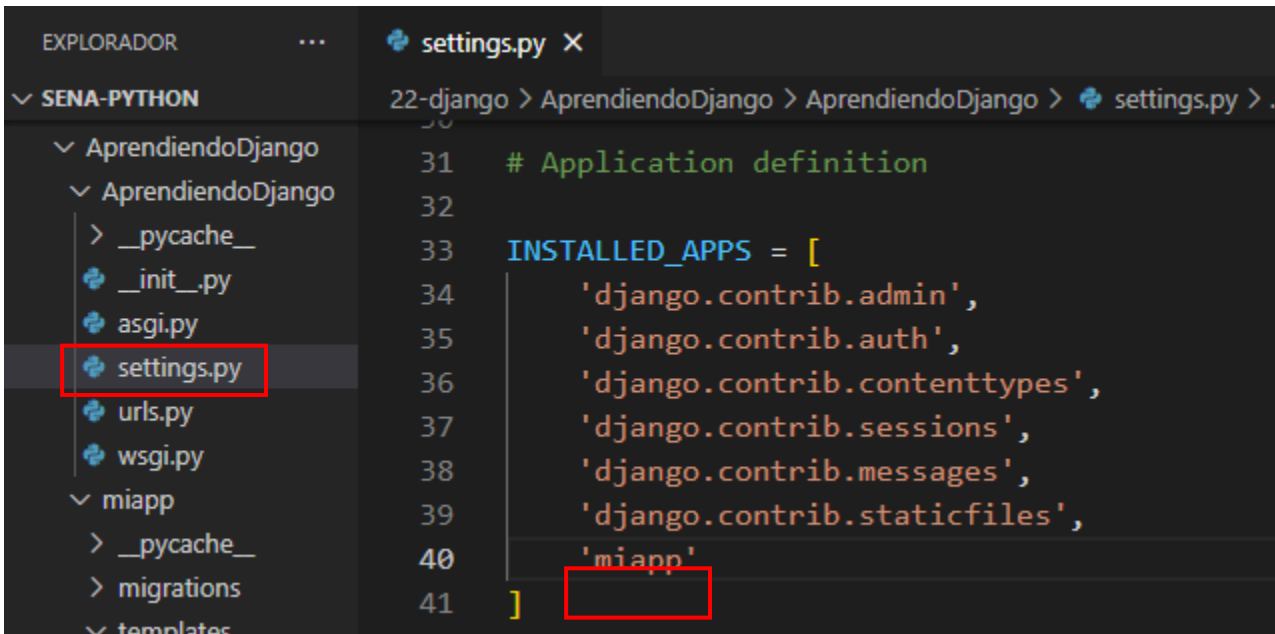


Vinculamos la vista al template en el archivo **views.py**

Verificamos que este importado el **render** en los **shortcuts**

```
22-django > AprendiendoDjango > miapp > views.py > ...  
1 from django.shortcuts import render, HttpResponseRedirect  
2
```

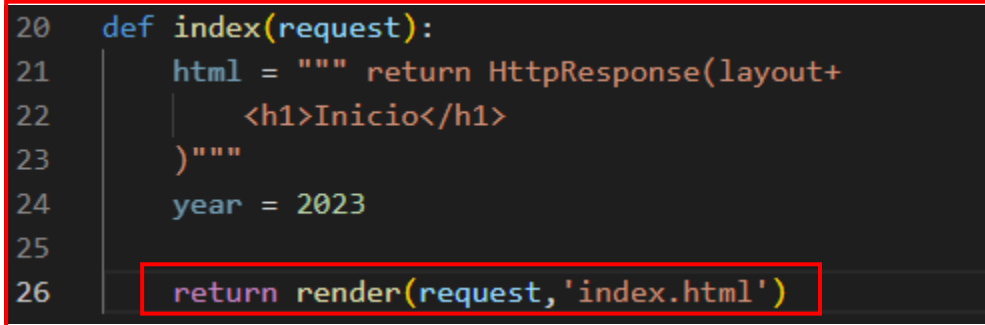
Luego **settings.py** en la línea de **INSTALLED_APPS**



The screenshot shows the VS Code interface with the **settings.py** file open. The **INSTALLED_APPS** list is visible, and the **miapp** entry is highlighted with a red box. The file explorer on the left shows the project structure with **settings.py** highlighted.

```
31 # Application definition  
32  
33 INSTALLED_APPS = [  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'miapp',  
41 ]
```

Después vamos **views.py** configuramos la siguiente línea para utilizar el **render** aquí es donde se separan las vistas del controlador



The screenshot shows the **views.py** file with the **index** function. The **return render(request, 'index.html')** line is highlighted with a red box.

```
20 def index(request):  
21     html = """ return HttpResponseRedirect(layout+  
22         <h1>Inicio</h1>  
23     )"""  
24     year = 2023  
25  
26     return render(request, 'index.html')
```

Ahora crearemos las plantillas para las diferentes funciones:

Creamos en la carpeta **templates** otro fichero llamado **hola_mundo.html** y **CORTAMOS** el siguiente código desde **views.py** y lo pegamos en el **hola_mundo.html**





```
views.py x hola_mundo.html settings.py urls.py
22-django > AprendiendoDjango > miapp > views.py > index
28 #request es un parametro que permite recibir datos de una URL
29 #Se le pasa a cada una de nuestras vistas
30 def hola_mundo(request):
31     #respuesta HTTP que se debe importarHttpResponse
32     return HttpResponse(layout+"""
33         <h1>Hola Mundo con Django!!!</h1>
34         <h3>Soy Aprendiz ADSO</h3>
35     """)
```

```
views.py x hola_mundo.html x settings.py urls.py
22-django > AprendiendoDjango > miapp > templates > hola_mundo.html
1 <h1>Hola Mundo con Django!!!</h1>
2 <h3>Soy Aprendiz ADSO</h3>
```

Luego utilizamos el método render para reemplazar en viwes.py

```
views.py x hola_mundo.html settings.py urls.py
22-django > AprendiendoDjango > miapp > views.py > hola_mundo
28 #request es un parametro que permite recibir datos de una URL
29 #Se le pasa a cada una de nuestras vistas
30 def hola_mundo(request):
31     #le pasamos el render, la respuesta request y el nombre de la template
32     return render(request, 'hola_mundo.html')
33
34 def pagina(request):
```

Probamos en el navegador:





Ahora creamos una para **pagina** de la misma manera que el anterior:

- Creamos el fichero en el templates que se llama pagina.html
- Corregimos el views.py en la función página.
- Correr en el navegador.

Layout, bloques y herencia de plantillas

Creación de Layout

En **templates** crear un fichero llamado **layout.html**, el cual servirá para cargar todas las templates después creamos una estructura html básica. Copiamos y pegamos en el body el menú que ya tenemos en views.py

```
<> layout.html X
22-django > AprendiendoDjango > miapp > templates > <> layout.html
1  <!DOCTYPE html>
2  <html>
3      <html lang="es">
4      <head>
5          <meta charset="utf-8">
6          <title>Sitio WEB con Django - ADSO SENA</title>
7      </head>
8      <body>
9          <h1>Sitio WEB con Django | Aprendices ADSO</h1>
10         <hr/>
11         <ul>
12             <li>
13                 <a href="/inicio">Inicio</a>
14             </li>
15             <li>
16                 <a href="/hola-mundo">Hola Mundo</a>
17             </li>
18             <li>
19                 <a href="/pagina-pruebas">Página de Pruebas</a>
20             </li>
21         </ul>
22         <hr/>
23         <div id="content">
24         </div>
25         <footer>
26             CURSO DE DJANGO ADSO 2023 &copy; Instructor Moisés García
27         </footer>
28     </body>
29 </html>
```



Ahora creamos diferentes bloques que es cuando sustituimos un trozo de pantalla por otro en las diferentes templates. Definimos un bloque dentro de **content** y en **title**

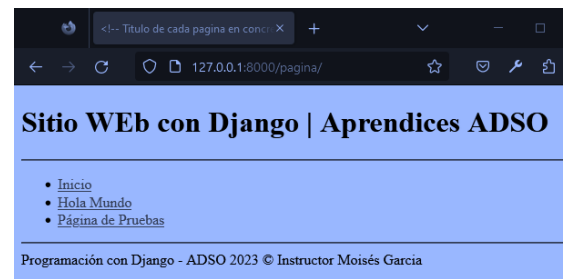
```
layout.html X  pagina.html  views.py  urls.py
22-django > AprendiendoDjango > miapp > templates > layout.html
1  <!DOCTYPE html>
2  <html>
3      <html lang="es">
4      <head>
5          <meta charset="utf-8">
6          <title>
7              {% block title %}
8              <!-- Titulo de cada pagina en concreto-->
9              {% endblock %}
10             |- ADSO SENA</title>
11      </head>
12      <body>
13          <h1>Sitio WEb con Django | Aprendices ADSO</h1>
14          <hr/>
15          <ul>
16              <li>
17                  <a href="/inicio">Inicio</a>
18              </li>
19              <li>
20                  <a href="/hola-mundo">Hola Mundo</a>
21              </li>
22              <li>
23                  <a href="/pagina-pruebas">Página de Pruebas</a>
24              </li>
25          </ul>
26          <hr/>
27          <div id="content">
28              {%block content %}
29              <!-- Viene codigo de templates, que sera diferente para cada caso-->
30              {% endblock %}
31          </div>
32          <footer>
33              Programación con Django - ADSO 2023 &copy; Instructor Moisés Garcia
34          </footer>
35      </body>
36  </html>
37
```

Ahora vamos a pagina.html y utilizamos el **extends** para heredar el layout principal en el template y correr pagina.html





```
layout.html | pagina.html x | views.py | urls.py
22-django > AprendiendoDjango > miapp > templates > < pagina.html
1  {% extends 'layout.html' %}
2
3  <h1>Pagina de mi WEB</h1>
4  <p>Creado por ADSO SENA - 2023 </p>
```



Como cargar el contenido de cada página en concreto. Ya que en este momento se pasa todo lo del layout.

```
layout.html | < pagina.html x | views.py | urls.py
22-django > AprendiendoDjango > miapp > templates > < pagina.html
1  {% extends 'layout.html' %}
2
3  {% block title %}
4  Pagina de prueba para heredar
5  {% endblock %}
6
7  {% block content %}
8  <h1>Pagina de mi WEB</h1>
9  <p>Creado por ADSO SENA - 2023 </p>
10 {% endblock %}
```



Realizar el mismo procedimiento de heredar en index, hola_mundo - para colocar el layout y titulo en cada página.

Ahora realizaremos la actualización de contenido por defecto en un bloque, para este caso vamos a **layout.html** y colocamos y luego aplicamos la herencia del padre en **hola_mundo.html**:

```
27 <div id="content">
28     {%block content %}
29     <strong>Contenido del bloque original!!</strong>
30     {% endblock %}
31 </div>
```

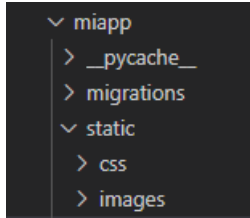
```
layout.html | pagina.html | index.html | < hola_mundo.html x
22-django > AprendiendoDjango > miapp > templates > < hola_mundo.html
1  {% extends 'layout.html' %}
2
3  {% block title %}
4  Hola Mundo!!!
5  {% endblock %}
6
7  {% block content %}
8  {{block.super}} <!--Heredamos del padre-->
9  <h1>Hola Mundo con Django!!!</h1>
10 <h3>Soy Aprendiz ADSO</h3>
11 {% endblock %}
```



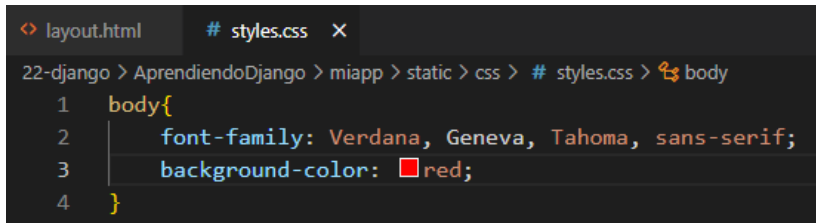


Vincular hojas de estilos CSS en Django

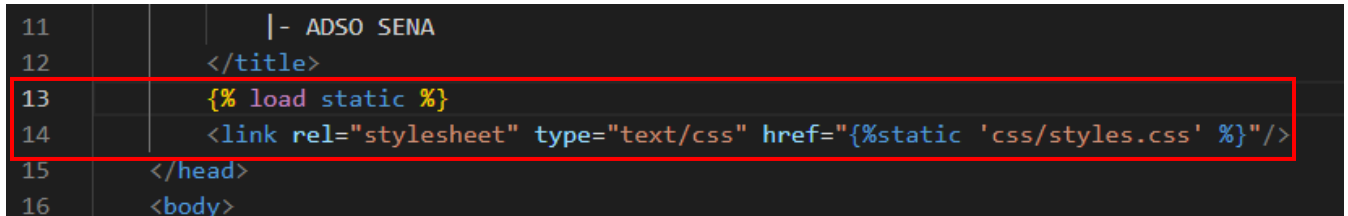
1. Creamos una carpeta **miapp** llamada **static** y dentro creamos otra carpeta llamada **css** y otra que se llama **images**



2. En **css** creamos un archivo llamado **styles.css**



3. Cargar los estilos estáticos en layout.html



Probamos el cambio





Maquetación estructura HTML5 mejorada:

Cambios en *layout.html*

```
layout.html x # styles.css
22-django > AprendiendoDjango > miapp > templates > layout.html
1 {% load static %}
2 <!DOCTYPE html>
3 <html>
4     <html lang="es">
5     <head>
6         <meta charset="utf-8">
7         <title>
8             {% block title %}
9             <!-- Titulo de cada pagina en concreto-->
10            {% endblock %}
11            |- ADSO SENA
12        </title>
13        {% load static %}
14        <link rel="stylesheet" type="text/css" href="{%static 'css/styles.css' %}" />
15    </head>
16    <body>
17        <!-- Cabecera de la Página -->
18        <header>
19            <div id="logotipo"> <!--se usa id porque sera algo que estara una vez en la pagina-->
20                <!--Logo Django-->
21                <h1> Sitio WEB con Django </h1>
22            </div>
23        </header>
24        <div id="nav">
25            <ul>
26                <li>
27                    <a href="/hola mundo">Hola Mundo</a>
28                </li>
29            </ul>
30        </div>
31        <div id="content">
32            <h2 class="title">Titulo Prueba</h2>
33            {%block content %}
34                <strong>Contenido del bloque original!!</strong>
35            {% endblock %}
36        </div>
37        <div id="footer">
38            Programación con Django - ADSO 2023 &copy; Instructor Moisés Garcia
39        </div>
40    </body>
41 </html>
```

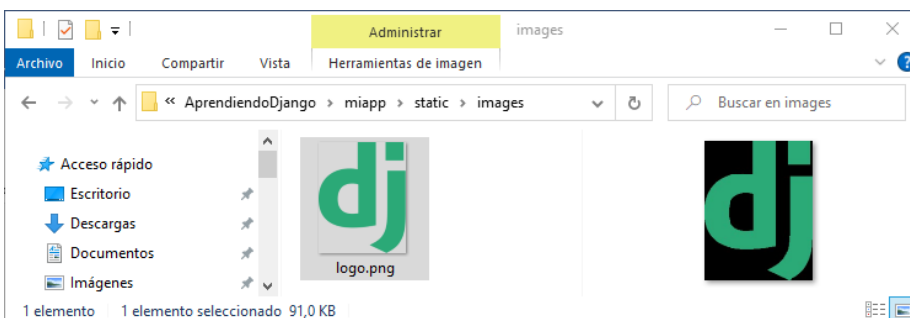


Estilos – css iniciales:

```
layout.html # styles.css X
22-django > AprendiendoDjango > miapp > static > css > # styles.css > #logotipo h1
1  /* Estilos Generales*/
2  *{
3      margin: 0px;
4      padding: 0px;
5      font-family: Arial, Helvetica, sans-serif;
6      text-decoration: none;
7  }
8
9  body{
10     background-color: #f2f2f2;
11 }
12
13 /*Estilos Cabecera*/
14 header{
15     width: 1212px;
16     height: 140px;
17     background-color: #23282b;
18     margin: 0px auto;
19 }
20 #logotipo h1{
21     display: block;
22     letter-spacing: 2px;
23     text-transform: uppercase;
24     font-weight: normal;
25     color: #42a096;
26 }
```

Cargar imagen en template de Django: en la carpeta **images** -> **mostrar en el explorador de archivos** y se pega la imagen con el logo.png

```
18 <header>
19     <div id="logotipo"> <!--se usa id porque sera algo que estara una vez en la pagina-->
20         <!--Logo Django-->
21         
22         <h1> Sitio WEB con Django </h1>
```



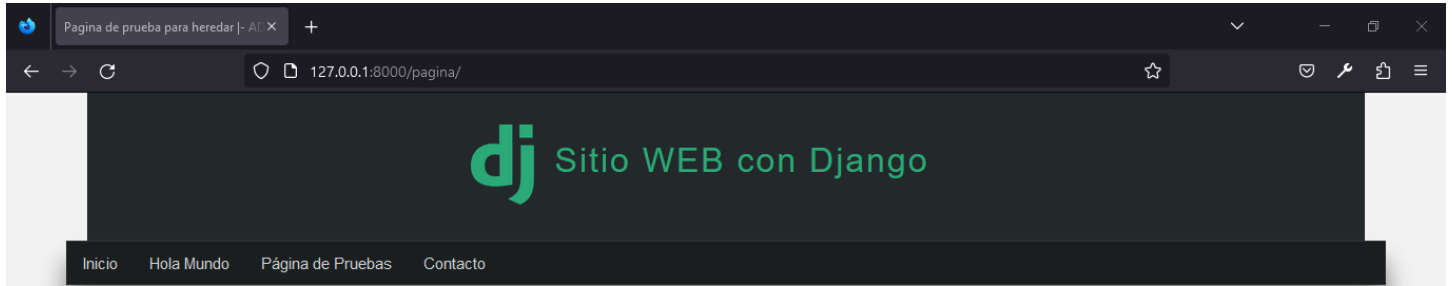


Estilos para la ubicación de logo y estilos: verificar los cambios realizados en el navegador

```
layout.html # styles.css X
22-django > AprendiendoDjango > miapp > static > css > # styles.css > #logotipo img
1  /* Estilos Generales*/
2  *{
3      margin: 0px;
4      padding: 0px;
5      font-family: Arial, Helvetica, sans-serif;
6      text-decoration: none;
7  }
8  body{
9      background-color: #f2f2f2;
10 }
11 /*Estilos Cabecera*/
12 header{
13     width: 1212px;
14     height: 140px;
15     background-color: #23282b;
16     margin: 0px auto;
17 }
18 /*Centrar texto y logo*/
19 #logotipo{
20     width: 40%;
21     height: 130px;
22     margin: 0px auto;
23     padding-top: 10px;
24 }
25 /*Tamaño y ubicacion de la imagen logo*/
26 #logotipo img{
27     display: block;
28     width: 60px;
29     float: left;
30     margin-top: 20px;
31 }
32 /*Tamaño y ubicacion del texto logo*/
33 #logotipo h1{
34     display: block;
35     float: left;
36     margin-top: 35px;
37     margin-left: 20px;
38     letter-spacing: 2px;
39     font-weight: lighter;
40     color: #2ba977;
41 }
```



Estilos barra de navegación:



Título Prueba

Pagina de mi WEB

Creado por ADSO SENA - 2023

Programación con Django - ADSO 2023 © Instructor Moisés García

Cambio en **style.css** para lograr el cambio de arriba

```
40     color: #2ba977;
41 }
42 /*Estilos barra de navegación*/
43 nav{
44     width: 1250px;
45     height: 40px;
46     background-color: #1b1e1f;
47     border: 1px solid #333333;
48     margin: 0 auto;
49     box-shadow: 0px 22px 22px gray;
50     font-size: 15px;
51     margin-bottom: 45px;
52 }
53 nav ul{
54     list-style: none;
55     text-decoration: 40px;
56 }
57 nav ul li{
58     line-height: 40px;
59     float: left;
60 }
61 /*enlaces*/
62 nav ul li a{
63     display: block;
64     padding-left: 15px;
65     padding-right: 15px;
66     color: #d1d4d6;
67 }
68
69 nav ul li a:hover{
70     background: #2ba977;
71     box-shadow: 0px 0px 5px #444444 inset;
72     color: white;
73     transition: all 300ms;
74 }
```





Estilos del contenido central:

```

73     transition: all 300ms;
74 }
75 /*Estilos contenido central*/
76 #content{
77     width: 1212px;
78     min-height: 930px;
79     margin: 0 auto;
80     margin-bottom: 30px;
81     margin-top: 30px;
82 }

```

Creamos una caja para el contenido central en *layout.html*

```

40     </nav>
41     <div id="content">
42         <div class="box">
43             <h2 class="title">Titulo Prueba</h2>
44             {%block content %}
45                 <strong>Contenido del bloque original!!</strong>
46             {% endblock %}
47         </div>
48     </div>
49 <footer>

```

Otros ajustes

```

74 }
75 /*Estilos contenido central*/
76 #content{ /*Posicionar*/
77     width: 1212px;
78     min-height: 930px;
79     margin: 0 auto;
80     margin-bottom: 30px;
81     margin-top: 40px;
82 }
83 .box{
84     background: white;
85     min-height: 930px;
86     width: 95%;
87     padding: 20px;
88     border: 1px solid #ddd;
89     border-radius: 2px;
90     margin: 0 auto;
91 }
92 /*Estilos para title*/
93 .title{
94     color: #444;
95     letter-spacing: 1px;
96     font-size: 30px;
97     margin-bottom: 10px;
98     margin-top: 5px;
99 }

```



Usamos la **class="title"** para nuestros títulos de cada página, realizarlos los cambios y probar en el navegador.

```

22-django > AprendiendoDjango > miapp > templates > < > pagina.html
1  {% extends 'layout.html' %}
2
3  {% block title %}
4  Pagina de prueba para heredar
5  {% endblock %}
6
7  {% block content %}
8  <h1 class="title">Pagina de mi WEB</h1>
9  <p>Creado por ADSO SENA - 2023 </p>
10 {% endblock %}

```

```

22-django > AprendiendoDjango > miapp > templates > < > index.html
1  {% extends 'layout.html' %}
2
3  {% block title %}
4  Inicio del Sistema
5  {% endblock %}
6
7  {% block content %}
8  <h1 class="title"> Inicio</h1>
9  <p>ADSO SENA 2023 - Regional CAUCA:</p>
10 {% endblock %}

```

Estilos para el footer: y probar en el navegador

```

99  }
100 /*Estilos para el footer*/
101 footer{
102     width: 1250px;
103     background-color: #1b1e1f;
104     border: 1px solid #333333;
105     color: #d1d4d6;
106     text-align: center;
107     padding-top: 20px;
108     margin: 0 auto;
109     padding-bottom: 20px;
110     box-shadow: 0px 0px 20px gray;
111 }

```

Lenguaje de plantillas y templates tags en Django

Comentarios en plantillas:

```

9  <p>ADSO SENA 2023 - Regional CAUCA:</p>
10 {% endblock %}
11
12 <!--Esto es un comentario para HTML-->
13
14 {% comment "Nota para el comentario" %}
15     <h1>lo que esta aqui tambien es un comentario</h1>
16 {% endcomment %}

```





Pasar datos desde la vista y mostrarlos en la plantilla:

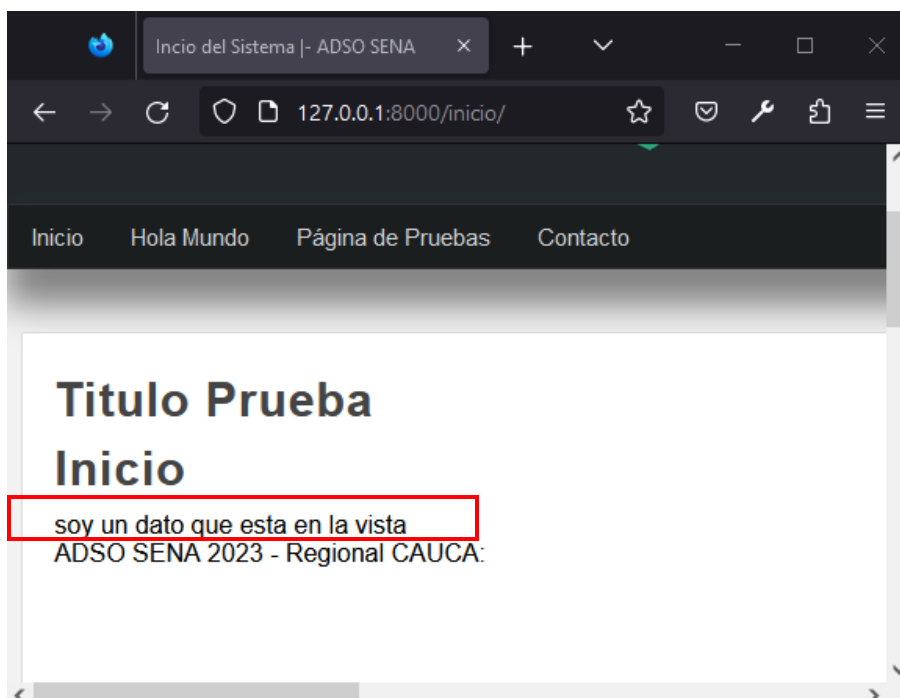
Desde el views.py se necesita pasar información al template para mostrarla y se necesita pasar desde el método o función

`return render(request, 'index.html', Aquí se pasa un diccionario con la información o las variables que quiero mostrar)`

```
30
31     return render(request, 'index.html', {
32         'mi_variable': 'soy un dato que esta en la vista'
33     })
34 # Create your views here.
35 #request es un parametro que permite recibir datos de una URL
```

Luego la llamamos en el index.html

```
7     {% block content %}
8     <h1 class="title"> Inicio</h1>
9     {% comment "interpolación" %}
10     mostrar los datos de una variable en un template
11     {% endcomment %}
12     {{mi_variable}}
13     <p>ADSO SENA 2023 - Regional CAUCA:</p>
14     {% endblock %}
```



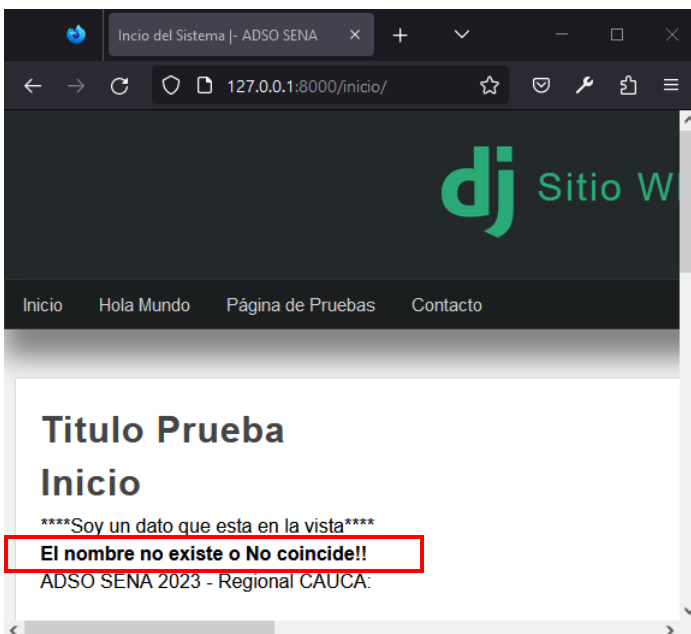


Condicionales – If templates Django: agregar en el `views.py`

```
28     year += 1
29     html += "</ul>"
30
31     nombre = 'Carlos Gomez'
32
33     return render(request, 'index.html', {
34         'mi_variable': '****Soy un dato que esta en la vista****',
35         'nombre': nombre
36     })
```

Y en `index.html` se genera el condicional para esa variable `nombre`

```
11 {% endcomment %}
12 <p>{{mi_variable}}</p>
13
14 <!--Condicionales-->
15 {% if nombre and nombre == 'Aprendiz ADSO' %}
16 <p>{{nombre}}</p>
17 {% else %}
18     <strong> El nombre no existe o No coincide!!</strong>
19 {% endif %}
20 <p>ADSO SENA 2023 - Regional CAUCA:</p>
```





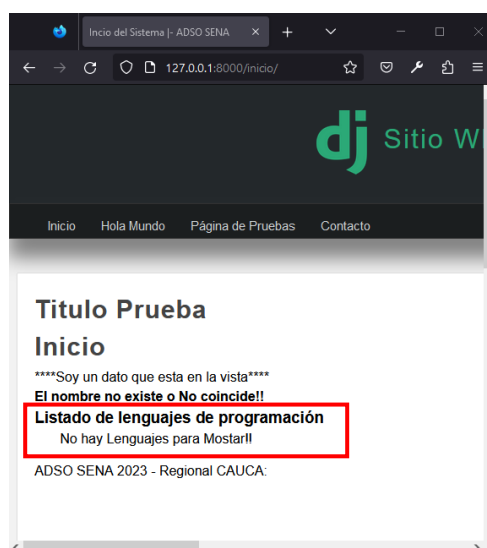
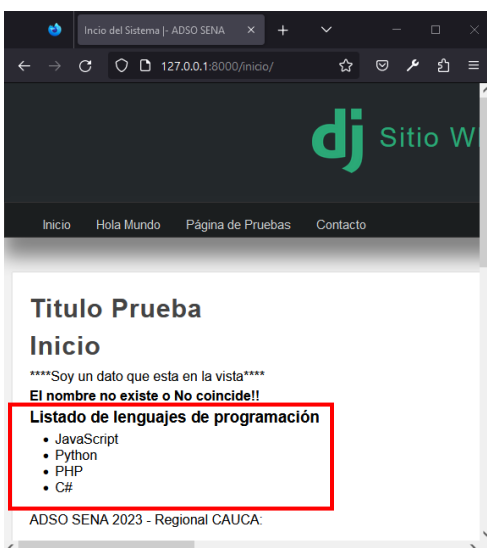
Blucles – For template Django

Las variables de deben definir en las vistas para pasarlas luego a las templates, en **views.py**:

```
31     nombre = 'Carlos Gomez'
32
33     #Variable para bucles For utilizando listas
34     lenguajes = ['JavaScript', 'Python', 'PHP', 'C#']
35
36     return render(request, 'index.html', {
37         'mi_variable': '****Soy un dato que esta en la vista****',
38         'nombre': nombre,
39         'lenguajes': lenguajes,
40     })
41 # Create your views here.
```

En **index.html**

```
18     <strong> El nombre no existe o No coincide!!</strong>
19     {% endif %}
20     <!--Bucle For y recorrer la lista -->
21     <h3>Listado de lenguajes de programación</h3>
22     <ul>
23     {% for lenguaje in lenguajes %}
24         <li>{{lenguaje}}</li>
25     {% empty %}
26         <p>No hay Lenguajes para Mostar!!</p>
27     {% endfor %}
28     </ul>
29
30     <p>ADSO SENA 2023 - Regional CAUCA:</p>
```





Includes dentro de templates Django:

Incluir una template dentro de otra template

Crear en templates un archivo **fecha-actual.html**

Y la incluyo en **index.html** y probar en el navegador

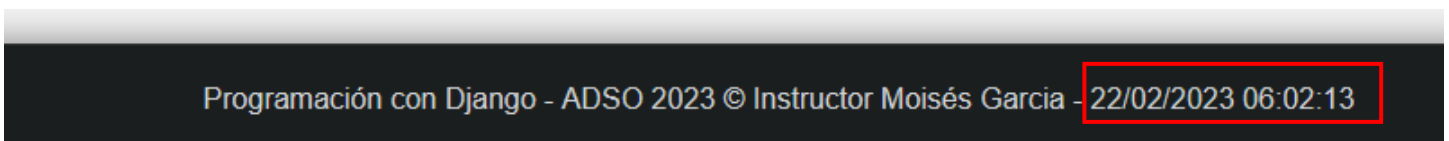
```
views.py  fecha-actual.html x  index.html
22-django > AprendiendoDjango > miapp > templates > fecha-actual.html
1  <br/>
2  <hr/>
3  <br/>
4  <h1> Bienvenido {{nombre}} </h1>
5  <h1> Estamos en el 2023 </h1>
6  <br/>
7  <hr/>
8  <br/>
9
```

```
views.py  fecha-actual.html  index.html x
22-django > AprendiendoDjango > miapp > templates > index.html
26  {% empty %}
27      <p>No hay Lenguajes para Mostar!!</p>
28  {% endfor %}
29  </ul>
30  <!--Esto es un include-->
31  {% include 'fecha-actual.html' %}
32
33  <p>ADSO SENA 2023 - Regional CAUCA:</p>
```

URLs en Templates: se abre el **layout.html** y se cambian las siguientes líneas de acuerdo a los **name** del archivo **urls.py**

```
layout.html x
22-django > AprendiendoDjango > miapp > templates > layout.html
21  
22  <h1> Sitio WEB con Django </h1>
23
24  </div>
25  </header>
26  <nav>
27      <ul>
28          <li>
29              <a href="{% url 'inicio' %}">Inicio</a>
30          </li>
31          <li>
32              <a href="{% url 'hola_mundo' %}">Hola Mundo</a>
33          </li>
34          <li>
35              <a href="{% url 'pagina' %}">Página de Pruebas</a>
36          </li>
37      </ul>
```

Fechas: colocar en el footer la fecha, abrimos la **layout.html** y colocamos la fecha que venga desde el servidor



```
layout.html x
22-django > AprendiendoDjango > miapp > templates > layout.html
46  </ul>
47  <footer>
48      Programación con Django - ADSO 2023 &copy; Instructor Moisés García - {% now "d/m/Y h:m:s"%}
49  </footer>
50  </body>
51  </html>
```



DESARROLLO CON PYTHON

Contenido de formación



Programación



Pyhton



Paradigma POO



Bases de datos SQL



Tkinter



Django



Flask

1. Primeros Instalación de SW
2. Variables y tipos de datos
3. Operadores (aritméticos, asignación)
4. Entrada y salida de datos
5. Estructuras de control
 - 5.1 Condicionales (operadores lógicos, operadores de comparación)
 - 5.2 Bucles (estructuras interactivas for, while)
6. Bloque de ejercicios de aplicación de conceptos
7. Funciones (parámetros, return, invocación, lambda)
 - 7.1 Variables locales y globales, funciones y métodos predefinidos
8. Lista y tuplas (creación, índices, recorrer y mostrar listas, listas multidimensionales)
9. Diccionarios y sets
10. Bloque de ejercicios aplicación de conceptos
11. Módulos y paquetes (creación y funcionalidad)
12. Sistemas de archivos y directorios
13. Manejo de errores (captura de excepciones, errores personalizados)
14. Programación orientada a objetos
15. Bases de datos SQLite
16. Bases de datos MySQL
17. Proyecto con Python
18. Interfaces graficas con Tkinter (aplicación de escritorio Python – Tkinter)
19. Desarrollo Web con Django
20. Interfaces graficas con Flask (aplicación de escritorio Python – Flask)





Formularios, crear vistas y urls

1. En el proceso de intentar definir una función, hay un parámetro llamado solicitud que se utiliza para recibir la información solicitada:

La información solicitada incluye: encabezado / cuerpo

Solicitud de clasificación:

get	Obtenga, desee que el servidor solicite el texto sin formato del recurso. obtener solicitud de? Inicio, clave es igual a valor, separados por &
post	Enviar, el método se utiliza para la transmisión física
head	Similar al método get. Solo no se devolverá el cuerpo de la respuesta, generalmente se usa para confirmar la validez de la URL y el momento de la actualización de recursos
put	Generalmente se usa para cargar archivos
delete	Especificar para eliminar un elemento
options	Método de soporte para consultar el recurso URL especificado
trace	El cliente puede rastrear la ruta de transmisión del mensaje de solicitud de esta manera
connect	Se requiere crear un túnel cuando se comunica con el servidor proxy para realizar la comunicación del protocolo TCP utilizando el protocolo de túnel.

2. En el desarrollo web, la mayoría de los datos se envían al servidor a través del formulario

Realizamos una copia de la vista crear_articulo

```
def crear_articulo(request, title, content, public):
    articulo = Article(
        title = title, #title(nombre del modelo) = title(nombre de la variable)
        content = content,
        public = public
    )
    #Para guardar este artículo en la BD
    articulo.save()
    return HttpResponseRedirect(f"Artículo creado: {articulo.title} - {articulo.content}")
```

y hacemos estos cambios a lo pegado

```
65     return HttpResponseRedirect(f"Artículo creado: {articulo.title} - {articulo.content}")
66
67 #Copia de crear_articulo y lo llamamos save_articulo
68 def save_article(request):# No pasamos parametros por url y lo vamos a pasar por formulario
69     if request.method == 'POST':
70         title = request.POST['title']
71         if len(title) < 5:
72             return HttpResponseRedirect(f"El titulo es mu pequeño")
73         content = request.POST['content']
74         public = request.POST['public']
75         articulo = Article(
76             title = title, #title(nombre del modelo) = title(nombre de la variable)
77             content = content,
78             public = public
79         )
80         #Para guardar este articulo en la BD
81         articulo.save()
82         return HttpResponseRedirect(f"Artículo creado: <strong> {articulo.title} - {articulo.content} </strong>")
83     else:
84         return HttpResponseRedirect(f"<h2> No se ha podido crear el articulo </h2>")
85
86
87 def create_article(request): #soporte para plantilla para visualizar el formulario
88     return render(request, 'create_article.html')
```



Ahora creamos la template para la vista lo llamamos create_articulo.html

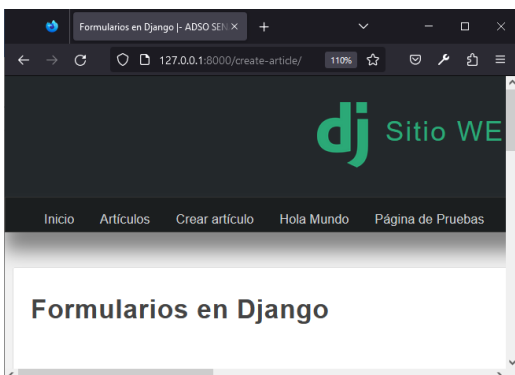
```
views.py layout.html create_articulo.html x articulos.html urls.py
22-django > AprendiendoDjango > miapp > templates > create_articulo.html
1 {% extends 'layout.html' %}
2 {% block title %} Formularios en Django {% endblock %}
3 {% block content %}
4 <h1 class="title">Formularios en Django</h1>
5
6
7 {% endblock %}
```

Creamos las url

```
40 path('borrar-articulo/<int:id>', views.borrar_articulo, name="borrar"),
41 #ruta para formularios
42 path('save-article/', views.save_article, name="save"),
43 path('create-article/', views.create_article, name="create"),
44 ]
```

Agregamos la vista a la barra de navegación layout.html

```
25 <nav>
26 <ul>
27 <li>
28 <a href="{% url 'inicio' %}">Inicio</a>
29 </li>
30 <li>
31 <a href="{% url 'articulos' %}">Artículos</a>
32 </li>
33 <li>
34 <a href="{% url 'create' %}">Crear artículo</a>
35 </li>
36 <li>
37 <a href="{% url 'hola_mundo' %}">Hola Mundo</a>
38 </li>
39 <li>
40 <a href="{% url 'pagina' %}">Página de Pruebas</a>
41 </li>
42 </ul>
43 </nav>
```





<> create_article.html X

22-django > AprendiendoDjango > miapp > templates > <> create_article.html

```
1  {% extends 'layout.html' %}
2  {% block title %} Formularios en Django {% endblock %}
3  {% block content %}
4  <h1 class="title">Formularios en Django</h1>
5
6  <form action="/save-article/" method="GET">
7      <!--Campo titulo-->
8      <label for="title">Titulo</label>
9      <input type="text" name="title" />
10     <!--Campo contenido-->
11     <label for="content">Contenido</label>
12     <textarea name="content"> </textarea>
13     <!--Campo publicado-->
14     <label for="public">Publicado</label>
15     <select name="public">
16         <option value="1">Si</option>
17         <option value="0">No</option>
18     </select>
19     <!--Botón-->
20     <input type="submit" value="Guardar" />
21 </form>
22 {% endblock %}
```





Estilos CSS

En nuestro proyecto tenemos ya un archivo en **static** → **css** → **styles.css**

Y vamos a **.box** donde tendremos el contenido del formulario

```
91 .box{
92     background: white;
93     min-height: 930px;
94     width: 95%;
95     padding: 20px;
96     border: 1px solid #ddd;
97     border-radius: 2px;
98     margin: 0 auto;
99 }
100 /*Estilos para formulario*/
101 .box form{
102     width: 40%;/*ancho del formulario*/
103 }
104 /*los controles ocupan una linea*/
105 .box form input,
106 .box form label{
107     display: block;
108     padding: 5px;
109     padding-left: 0px;
110 }
111 /*ancho de los controles*/
112 .box form input[type="text"],
113 .box form textarea,
114 .box form select{
115     width: 100%;
116     margin-bottom: 10px;
117 }
```

```
116     margin-bottom: 10px;
117 }
118 /*select*/
119 .box form select{
120     width: 70px;
121     padding: 5px;
122 }
123 /*Boton*/
124 .box form input[type="submit"],
125 .box form input[type="button"],
126 .box form button{
127     padding: 10px;
128     margin-top: 5px;
129     background: #2ba977;
130     border: 1px solid #444;
131     color: white;
132     transition: 300ms all;
133     border-radius: 7px;
134 }
135 /* Color del boton cuando pase por encima */
136 .box form input[type="submit"]:hover,
137 .box form input[type="button"]:hover,
138 .box form button:hover{
139     cursor: pointer;
140     background: #1f7e58;
141 }
142 /*Estilos para title*/
```





Recibir datos del formulario por GET

Vamos a **views.py** y vamos a la función **def save_article(request)** línea 95

Y rellenamos las variables y comprobamos si nos llegan datos por GET

```
66
67 #Copia de crear_articulo y lo llamamos save_articulo
68 def save_article(request):# No pasamos parametros por url y lo vamos a pasar por formulario
69     if request.method == 'GET':
70         title = request.GET['title']
71         content = request.GET['content']
72         public = request.GET['public']
73         articulo = Article(
74             title = title, #title(nombre del modelo) = title(nombre de la variable)
75             content = content,
76             public = public
77         )
78         #Para guardar este articulo en la BD
79         articulo.save()
80         return HttpResponse(f"Artículo creado: <strong> {articulo.title} - {articulo.content} </strong>")
81     else:
82         return HttpResponse(f"<h2> No se ha podido crear el articulo </h2>")
83
84
85 def create_article(request): #soporte para plantilla para visualizar el formulario
```

```
views.py ...\pages  views.py ...\miapp X  create_article.html  articulos.html
22-django > AprendiendoDjango > miapp > views.py > articulos
147 #Mostrar articulos
148 def articulos(request):#creacion de la vista articulos
149     #Creamos una template para listar varios articulos
150     #1. hacemos la peticion a la base de datos
151     #Creamos la variable articulos y Llamamos al modelo Article, usamos objects para poder hacer una consulta
152     articulos = Article.objects.all().order_by('-id') #En este caso no usamos el metodo get si no el metodo all()
153     #para sacar toda la información
154     #dentro de articulos tenemos una lista de objetos un array de objetos
155     """
156     #Consultas con condiciones filter para filtrar por un valor especifico
157     articulos = Article.objects.filter(title__contains = "Tiburón")
158
159     #Realizar una consulta que muestre solo los esten publicados y excluya bajo una condición
160     articulos = Article.objects.filter(title = "Tiburón").exclude(public=True)
161
162     #Consulta ejecutando SQL
163     articulos = Article.objects.raw("SELECT * FROM miapp_article WHERE title='Tiburón' AND public=1")
164
165     #Consulta utilizando el OR con ORM
166     articulos = Article.objects.filter(
167         Q(title__contains="John") | Q(title__contains="Señor")
168     )
169     """
170     return render(request, 'articulos.html', { #pasamos el request y la template articulos.html y como tercer parametro
171         'articulos': articulos #pasamos un diccionario con las variables que deseamos mostrar
172     })
```



```
articulos.html X
22-django > AprendiendoDjango > miapp > templates > <articulos.html
1  {% extends 'layout.html' %}
2  {% block title %} Listado de Artículos {% endblock %}
3  {% block content %}
4  <h1 class="title">Listado de Artículos</h1>
5  {% if messages %}
6      {% for message in messages %}
7          <div class="message">
8              {{message}}
9          </div>
10     {% endfor %}
11 {% endif %}
12     <ul>
13         {% for articulo in articulos %}
14             <li>
15                 <h4>{{articulo.id}} {{articulo.title}}</h4>
16                 <span>{{articulo.created_at}}</span>
17                 {% if articulo.public %}
18                     <strong>Publicado</strong>
19                 {% else %}
20                     <strong>Privado</strong>
21                 {% endif %}
22                 <p>
23                     {{articulo.content}}
24                     <br/>
25                     <a href = "{% url 'borrar' id=articulo.id %}">Eliminar</a>
26                 </p>
27                 <br/>
28             </li>
29         {% endfor %}
30     </ul>
31 {% endblock %}
```

Llenamos el formulario con crear artículo

Formularios en Django | ADSO SENA X

Inicio Artículos Crear artículo Hola Mundo Página de Pruebas

Formularios en Django

Titulo
El Escuadrón Suicida

Contenido
la Galaxia del UCM de Marvel para adentrarse en el universo de DC y otro grupo de inadaptados, los antihéroes del Escuadrón Suicida.

Publicado
Si

Guardar

Observamos el resultado de lista en la Artículos

Listado de Artículos | ADSO SENA X

Inicio Artículos Crear artículo Hola Mundo Página de Pruebas

Listado de Artículos

- **11 El Escuadrón Suicida**
9 de marzo de 2023 a las 02:21 **Publicado**
James Gunn abandonó temporalmente a sus Guardianes de la Galaxia del UCM de Marvel para adentrarse en los antihéroes del Escuadrón Suicida.
[Eliminar](#)
- **9 El bueno, el feo y el malo**
2 de marzo de 2023 a las 00:37 **Publicado**
Clint Eastwood se pone un poncho y se convierte en un icono americano en las llanuras españolas.
[Eliminar](#)
- **8 Déjame salir**
28 de febrero de 2023 a las 00:37 **Publicado**
La mejor película sobre la esclavitud americana jamás hecha, y encima sin sermones, todo bajo una original p...





Recoger datos del formulario por POST

create_article.html

```

views.py  create_article.html x  urls.py
22-django > AprendiendoDjango > miapp > templates > create_article.html
1  {% extends 'layout.html' %}
2  {% block title %} Formularios en Django {% endblock %}
3  {% block content %}
4  <h1 class="title">Formularios en Django</h1>
5
6  <form action="{% url 'save' %}" method="POST">
7  {% csrf_token %}
8  <!--Campo titulo-->
9  <label for="title">Titulo</label>
10 <input type="text" name="title" />

```

views.py

```

67 #Copia de crear_articulo y lo llamamos save_articulo
68 def save_article(request):# No pasamos parametros por u
69     if request.method == 'POST':
70         title = request.POST['title']
71         content = request.POST['content']
72         public = request.POST['public']
73         articulo = Article(
74             title = title, #title(nombre del modelo) =
75             content = content,

```

Agregamos información al formulario

Aquí se puede evidenciar que no se muestra la información enviada por la url

Formularios basados en clases:

Si se está construyendo una aplicación que gestiona una base de datos, lo más apropiado es usar los modelos ya declarados como formularios, y así evitar estar repitiendo las mismas reglas para procesar los datos.





Por esta razón, Django provee una clase de ayuda que permite crear un formulario a partir de un modelo, esta clase se llama `ModelForm`, y se emplea así (`forms.py`):

Creemos a nivel de **miapp** un fichero que se llamara **forms.py**. En este fichero importamos ***from django import forms***. Definimos las clases de la siguiente manera: recomendación que la clase se llame igual o similar a como se llama el modelo (***class Article(models.Model)***). Como concepto es importante conocer los diferentes field que existen para los diferentes controles o propiedades

<https://docs.djangoproject.com/en/4.1/ref/forms/fields/>

creación del formulario

```
forms.py x views.py create_article.html create_full_article.html
22-django > AprendiendoDjango > miapp > forms.py > FormArticle
1  from django import forms
2
3  class FormArticle(forms.Form): #Hereda de la clase forms
4
5      #creamos las propiedades
6      title = forms.CharField(
7          label= "Titulo"
8      )
9      content = forms.CharField(
10         label= "Contenido",
11         widget=forms.Textarea
12     )
```

En **views.py** para nuestro ejemplo vamos a crear un método ***def create_full_article***

Para crear este método debemos importar el formulario que hemos creado en `forms.py` para poder usar el objeto

```
forms.py x views.py x urls.py
22-django > AprendiendoDjango > miapp > views.py > ...
1  from django.shortcuts import render, HttpResponseRedirect, redirect
2  from miapp.models import Article
3  from django.db.models import Q
4  from miapp.forms import FormArticle #Importamos el formulario desde forms.py
5
```

Y mas abajo creamos el método

```
86  def create_article(request): #soporte para plantilla para visualizar el
87      return render(request, 'create_article.html')
88
89  def create_full_article(request):
90      #creamos una variable llamada formulario para instanciar el objeto
91      formulario = FormArticle()
92      #se carga la vista en html
93      return render(request, 'create_full_article.html', {
94          'form' : formulario
95      })
96
97  def articulo(request):
```

El siguiente paso será crear la vista html **create_full_article.html** en template y podemos copiar todo el código o la plantilla de **create_article.html** y se sustituye algunas cosas como se muestra a continuación:



```
forms.py | views.py | create_full_article.html x | urls.py
22-django > AprendiendoDjango > miapp > templates > create_full_article.html
1  {% extends 'layout.html' %}
2  {% block title %} Formularios en Django {% endblock %}
3  {% block content %}
4  <h1 class="title">Formularios en Django</h1>
5
6  <form action="" method="POST">
7      {% csrf_token %}
8
9      <!--aquí mostramos el Formulario mandando a llamar
10      la variable form que se creo en la vista-->
11      {{ form }}
12
13      <!--Botón-->
14      <input type="submit" value="Guardar" />
15  </form>
16  {% endblock %}
```

Ahora creamos la ruta

```
43 path('create-article/', views.create_article, name="create"),
44 path('create-full-article/', views.create_full_article, name="create_full")
45 ]
```

Y comprobamos los cambios en la ruta create-full-article/



Hay muchos otros tipos de campos de formulario, que reconocerá en gran medida por su similitud con las clases de campo de modelo





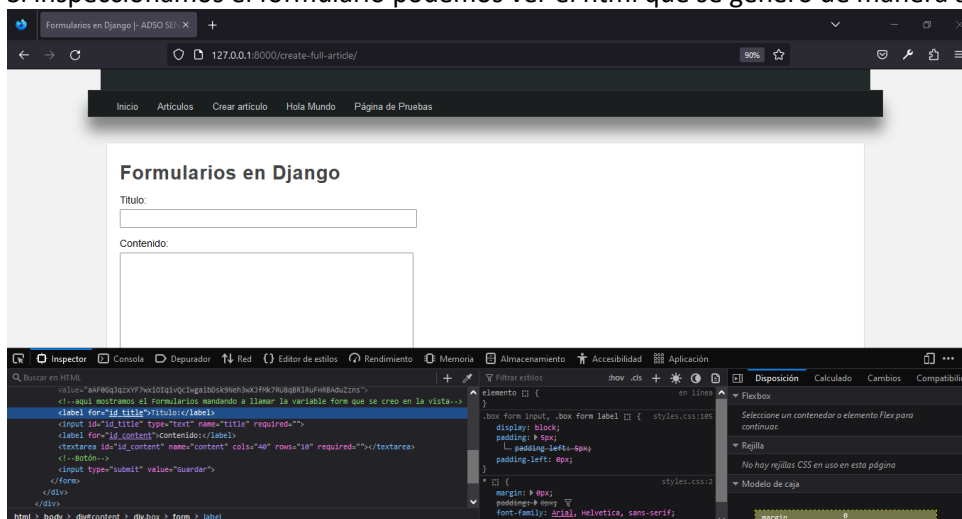
equivalentes: [BooleanField](#), [CharField](#), [ChoiceField](#), [TypedChoiceField](#), [DateField](#), [DateTimeField](#), [DecimalField](#), [DurationField](#), [EmailField](#), [FileField](#), [FilePathField](#), [FloatField](#), [ImageField](#), [IntegerField](#), [GenericIPAddressField](#), [MultipleChoiceField](#), [TypedMultipleChoiceField](#), [NullBooleanField](#), [RegexField](#), [SlugField](#), [TimeField](#), [URLField](#), [UUIDField](#), [ComboField](#), [MultiValueField](#), [SplitDateTimeField](#), [ModelMultipleChoiceField](#), [ModelChoiceField](#).

Los argumentos que son comunes a la mayoría de los campos se enumeran a continuación (estos tienen valores predeterminados sensibles):

- [required](#): Si es True, el campo no se puede dejar en blanco o dar un valor None. Los Campos son obligatorios por defecto, también puedes establecer `required=False` para permitir valores en blanco en el formulario.
- [label](#): label es usado cuando renderizamos el campo en HTML. Si [label](#) no es especificado entonces Django crearía uno a partir del nombre del campo al poner en mayúscula la primera letra y reemplazar los guiones bajos por espacios (por ejemplo. *Renewal date*).
- [label_suffix](#): Por defecto, se muestran dos puntos después de la etiqueta (ejemplo. *Renewal date:*). Este argumento le permite especificar como sufijo diferente que contiene otros caracteres.
- [initial](#): El valor inicial para el campo cuando es mostrado en el formulario.
- [widget](#): El widget de visualización para usar.
- [help_text](#) (como se ve en el ejemplo anterior): texto adicional que se puede mostrar en formularios para explicar cómo usar el campo.
- [error_messages](#): Una lista de mensajes de error para el campo. Puede reemplazarlos con sus propios mensajes si es necesario.
- [validators](#): Una lista de funciones que se invocarán en el campo cuando se valide.
- [localize](#): Permite la localización de la entrada de datos del formulario (consulte el enlace para obtener más información).
- [disabled](#): El campo se muestra, pero su valor no se puede editar si esto es True. Por defecto es False.

Ahora se puede generar el formulario de diferentes formas, por ejemplo:

Si inspeccionamos el formulario podemos ver el html que se genero de manera automatizada





Podemos mostrar el formulario y se quiere que todos los campos estén bajo una etiqueta en concreto lo que se puede hacer es:

```
6 <form action="" method="POST">
7     {% csrf_token %}
8
9     <!--aquí mostramos el Formulario mandando a llamar
10     la variable form que se creo en la vista-->
11     {{ form.as_p }} <!-- el as_p mete todos los campos dentro de una etiqueta de parrafo-->
12
13     <!--Botón-->
```

```
11 {{ form.as_ul }} <!-- el as_p mete todos los campos dentro de una etiqueta de lista-->
```

Formularios en Django

- Titulo:
- Contenido:

```
<li>
  ::marker
  <label for='id_content'>Contenido:</label>
  <textarea id='id_content' name='content' cols='40' rows='10' required=''></textarea>
</li>
```



Campo tipo select

Dejamos el form.as_p

Creamos el public que son las opciones y las integramos a un choices

forms.py

```
11     widget=forms.Textarea
12 )
13 #con choices se pueden pasar una serie de opciones
14 public_options = [
15     (1, 'Si'),
16     (0, 'No')
17 ]
18 public = forms.TypedChoiceField(#permite mostrar un campo select y pasarle las opciones anteriores
19     label = "¿Publicado?",
20     choices = public_options
21 )
```

Comprobamos el resultado

Formularios en Django

Inicio Artículos Crear artículo Hola Mundo Página de Pruebas

Formularios en Django

Título:

Contenido:

¿Publicado?

Si

Guardar



Recibir datos y guardar el formulario(Django Form API)

Para lo cual vamos a la vista **`def create_full_article`** y realizamos los siguientes cambios, comprobamos si nos envían datos por el formulario mediante que método (POST ó GET)

views.py

```

forms.py  views.py  x  create_full_article.html  urls.py
22-django > AprendiendoDjango > miapp > views.py > create_full_article
85
86 def create_article(request): #soporte para plantilla para visualizar el formulario
87     return render(request, 'create_article.html')
88
89 def create_full_article(request):
90     #Realizamos la comprobacion del metodo (POST-GET)
91     if request.method == 'POST':
92         #si llega datos por POST se debe:
93         formulario = FormArticle(request.POST)
94         #aquí podemos validar el formulario con un metodo is_valid
95         if formulario.is_valid():
96             #generamos una variable para recoger los datos del formulario
97             data_form = formulario.cleaned_data #que son los datos limpios que nos llegan
98             title = data_form.get('title') #lo puedo hacer asi ó
99             content = data_form['content'] # lo puedo hacer asi
100            public = data_form['public']
101            return HttpResponse(title + ' -- ' + content + ' -- ' + str(public))
102
103     else:
104         #si nos llegan datos por POST debemos generar un formulario vacio
105         formulario = FormArticle() #creamos una variable llamada formulario para instanciar el objeto
106         #se carga la vista en html
107         return render(request, 'create_full_article.html', {
108             'form' : formulario
109         })
110
111 def articulo(request):

```

Comprobamos el código enviando datos <http://127.0.0.1:8000/create-full-article/>

y aquí como se reciben





Ahora como se guardar esta información

Se puede utilizar por el momento el modelo de artículo y le paso las variables que se han recogido .

```

88
89 def create_full_article(request):
90     #Realizamos la comprobacion del metodo (POST-GET)
91     if request.method == 'POST':
92         #si llega datos por POST se debe:
93         formulario = FormArticle(request.POST)
94         #aqui podemos validar el formulario con un metodo is_valid
95         if formulario.is_valid():
96             #generamos una variable para recoger los datos del formulario
97             data_form = formulario.cleaned_data #que son los datos limpios que nos llegan
98             title = data_form.get('title') #lo puedo hacer asi ó
99             content = data_form['content'] # lo puedo hacer asi
100             public = data_form['public']
101
102             articulo = Article(
103                 title = title, #title(nombre del modelo) = title(nombre de la variable)
104                 content = content,
105                 public = public
106             )
107             #Para guardar este articulo en la BD
108             articulo.save()
109             #redireccionar la pagina a articulos despues de guardado
110             return redirect('articulos')
111             #return HttpResponse(articulo.title + ' -- ' + articulo.content + ' -- ' + str(articulo.public))
112
113     else:

```

Table: **miapp_artide**

id	content	public	created_at	updated_at	image	title
13	El argumento es lo más simple del mundo, pero a veces lo simple es lo más efectivo. Keanu Reeves y Sandra Bullock tratan de controlar un autobús a toda velocidad por la ciudad	1	2023-03-10 ...	2023-03-10 ...	null	Speed: Máxima potencia
12	Lettv Ortiz (Michelle Rodríguez) se ha nasado al lado oscuro. y forma parte de la saga de Fast & Furious	0	2023-03-09 ...	2023-03-09 ...	null	Fast & Furious 6





Se puede crear en layout otra nueva entrada para que aparezca en la barra de menú el nuevo tipo de formulario con clases.

```
<li>  
    <a href="{% url 'create_full' %}">Crear artículo Clases</a>  
</li>
```

Formularios en Django | - ADSO SEN X

127.0.0.1:8000/create-full-article/ 90%

Inicio Artículos Crear artículo **Crear artículo Clases** Hola Mundo Página de Prueb

Formularios en Django

Título:

Contenido:

¿Publicado?

Si

Guardar

127.0.0.1:8000/create-full-article/



Validación de formularios

Vamos a comprobar si el formulario trae un error → `create_full_article.html`

Importamos validators → `from django.core import validators`

Documentación <https://docs.djangoproject.com/en/4.1/ref/validators/>

Importar en forms.py la librería de validators

```
forms.py x views.py layout.html create_full_article.html # styl
22-django > AprendiendoDjango > miapp > forms.py > FormArticle
1 from django import forms
2 from django.core import validators
3
4 class FormArticle(forms.Form): #Hereda de la clase forms
5
```

```
forms.py views.py layout.html create_full_article.html x styles.css urls.py
22-django > AprendiendoDjango > miapp > templates > create_full_article.html
1 {% extends 'layout.html' %}
2 {% block title %} Formularios en Django {% endblock %}
3 {% block content %}
4 <h1 class="title">Formularios en Django</h1>
5 {% if form.errors %}
6     <strong class="rojo">
7         Hay errores en el formulario
8     </strong>
9 {% endif %}
10 <form action="" method="POST">
11     {% csrf_token %}
12
13     <!--aquí mostramos el Formularios mandando a llamar
14     la variable form que se creo en la vista-->
15     {{ form.as_p }} <!-- el as_p mete todos los campos dentro de una etiqueta de parrafo-->
16
17     <!--Botón-->
18     <input type="submit" value="Guardar" />
19 </form>
20 {% endblock %}
```

Estilos de clase rojo

```
140     background: #1f7e58;
141 }
142 /* estilos para clase rojo de validacion */
143 .rojo{
144     color: red;
145     box-shadow: 0px 0px 4px black;
146 }
147 /*Estilos para title*/
148 .title{
```



Validaciones en **forms.py** aquí se valida el primer campo del título **'^[A-Za-z-9ñÑáéíóúÁÉÍÓÚ]*\$'**
Todas las validaciones de acuerdo al alfabeto español que no las posee el alfabeto inglés

```
6 #creamos las propiedades
7 title = forms.CharField(
8     label= "Titulo",
9     max_length=20,
10    required=True,
11    widget=forms.TextInput(
12        attrs={
13            'placeholder': 'Digite el titulo',
14            'class': 'titulo_form_article'
15        }
16    ),
17    validators=[
18        validators.MinLengthValidator(4, 'El titulo es demasiado corto'),
19        validators.RegexValidator('^[A-Za-z-9ñÑáéíóúÁÉÍÓÚ ]*$', 'El titulo esta mal formado', 'invalid_title')#expresion regular
20    ]
21 )
22
23 content = forms.CharField(
```

Formularios en Django

Hay errores en el formulario

- El titulo es demasiado corto

Titulo:
ff

Contenido:
hfg

¿Publicado?
Si

Guardar

Validación del content

```
19     validators.RegexValidator('^[A-Za-z-9 ]*$', 'El titulo esta mal for
20
21 ]
22 )
23 content = forms.CharField(
24     label= "Contenido",
25     widget=forms.Textarea,
26     validators=[
27         validators.MaxLengthValidator(20, 'El texto es demasiado largo')
28
29 ]
30
31 )
32 #con choices se pueden pasar una serie de opciones
33 public_options = [
```

Formularios en Django - ADS X

127.0.0.1:8000/crea

Inicio Artículos Crear artículo Crear artículo Clases Hola Mu

Formularios en Django

Hay errores en el formulario

- El titulo es demasiado corto

Título:

- El texto es demasiado largo

Contenido:

Las diferentes viñetas que ilustran este Montmartre y la tierna, conmovedora historia de amor entre Amélie y Nino (Mathieu Kassovitz) son la prueba de que Jeunet reivindica la realidad a través de los sueños. O lo que es lo mismo, nos dice que solo seremos felices si luchamos por convertirlos en realidad. Eso sí, por encima de cualquier moraleja, "Amelie" es, simplemente, un prodigio de inventiva, una absoluta delicia, un regalo para los ojos, una canción de Charles Trénet hecha película



Mensajes flash / sesiones flash

Vamos **views.py** e importamos la librería *from django.contrib import messages*

```
views.py X
22-django > AprendiendoDjango > miapp > views.py > ...
1  from django.shortcuts import render, HttpResponseRedirect, redirect
2  from miapp.models import Article
3  from django.db.models import Q
4  from miapp.forms import FormArticle #Importamos el formulario desde f
5  from django.contrib import messages #libreria mensajes flash
6  layout = """
```

Todo eso se hace en la redirección después de la validación

```
106         )
107         #Para guardar este articulo en la BD
108         articulo.save()
109         #Crear mensaje flash(sesión que solo se muestra 1 vez)
110         messages.success(request, f'Ha creado correctamente el articulo {articulo.id}')
111
112         #redireccionar la pagina a ariculos despues de guardado
113         return redirect('articulos')
```

Y para mostrar el mensaje se hace en la vista **articulos.html**

```
views.py X  articulos.html X
22-django > AprendiendoDjango > miapp > templates > articulos.html
1  {% extends 'layout.html' %}
2  {% block title %} Listado de Artículos {% endblock %}
3  {% block content %}
4  <h1 class="title">Listado de Artículos</h1>
5  {% if messages %}
6      {% for message in messages %}
7          <div class="messages">
8              {{message}}
9          </div>
10     {% endfor %}
11 {% endif %}
12 <ul>
13     {% for articulo in articulos %}
14         <li>
15             <h4>{{articulo.id}} {{articulo.title}}</h4>
16             <span>{{articulo.created_at}}</span>
17             {% if articulo.public %}
18                 <strong>Publicado</strong>
19             {% else %}
20                 <strong>Privado</strong>
21             {% endif %}
22         <p>
23             {{articulo.content}}
24             <br/>
25             <a href = "{% url 'borrar' id=articulo.id %}">Eliminar
```



Probamos el código: recordemos que las sesiones flash solo duran una recarga de pantalla y desaparecen.

Formularios en Django | - ADSO SEN X +

← → ↻ 127.0.0.1:8000/create-full-article/

Formularios en Django

Título:

Contenido:

Uno de los productos más refinados entre los que dirigió Hitchcock, en el que el azar se convierte en motor de una historia elaborada con tanta coherencia como rigor.

¿Publicado?

Si ▾

Guardar

Listado de Artículos

Ha creado correctamente el artículo 16

16 Con la muerte en los talones

14 de marzo de 2023 a las 13:51 Publicado

Uno de los productos más refinados entre los que dirigió Hit como rigor.

Eliminar

Colocar estilos a la clase messages