

Templates en Django

En el contexto de desarrollo web, un template es un archivo o conjunto de archivos que se utilizan para definir la estructura y presentación de una página web. Los templates se utilizan para separar la lógica de presentación de la lógica de negocio en una aplicación web.

En el caso específico de Django, los templates son archivos HTML que contienen marcadores y etiquetas especiales, conocidos como etiquetas de template, que permiten la inserción de datos dinámicos y la lógica de programación. Django utiliza su propio motor de templates para renderizar estos archivos HTML y generar el contenido final que se envía al navegador.

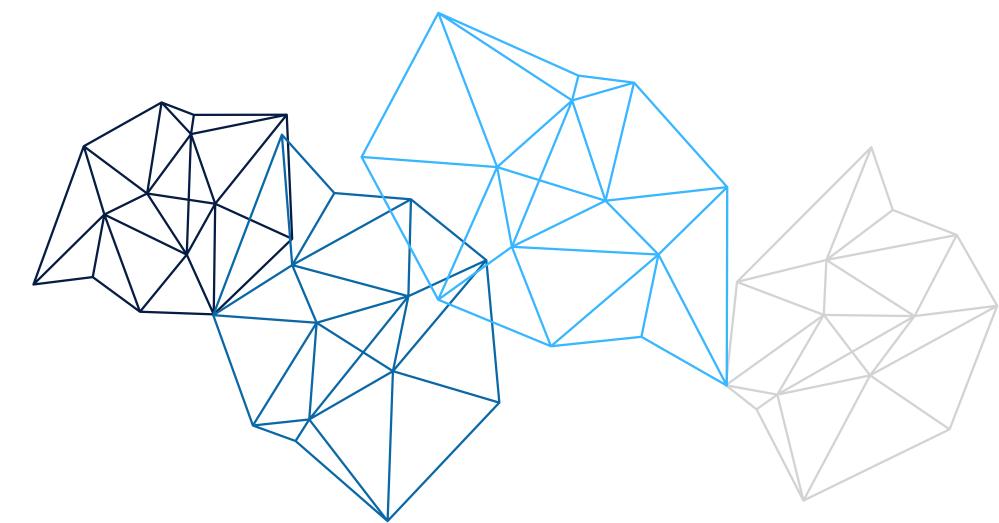
Los templates en Django tienen las siguientes características:

- Sintaxis: Django utiliza una sintaxis especial para los marcadores y etiquetas de template dentro de los archivos HTML. Estos marcadores se encierran entre llaves dobles { % % } y se utilizan para realizar operaciones lógicas, bucles, condicionales y más.
- Herencia: Los templates en Django pueden heredar de otros templates, lo que permite crear una jerarquía de plantillas. Un template base puede definir la estructura común de todas las páginas, mientras que los templates secundarios pueden extender y personalizar ese diseño base.



- Variables y filtros: Los templates permiten el uso de variables que contienen datos dinámicos y se pueden mostrar en la página final. Además, Django proporciona una amplia gama de filtros que se pueden aplicar a estas variables para modificar su formato o realizar cálculos.
- Iteración y condicionales: Los templates de Django incluyen estructuras de control como bucles for y condicionales if, que permiten iterar sobre listas de datos y mostrar contenido condicionalmente en función de ciertas condiciones.
- Contexto: Django permite pasar un contexto de datos desde las vistas a los templates. El contexto es un diccionario de Python que contiene las variables y valores que se utilizarán en el template. Estos datos pueden provenir de la base de datos, formularios u otras fuentes.

Proporcionan una forma flexible y poderosa de separar la lógica de presentación de la lógica de negocio, lo que facilita el desarrollo y mantenimiento de aplicaciones web en Django.



Vamos a crear unos ejemplos básicos de lo que podemos hacer con ellos:

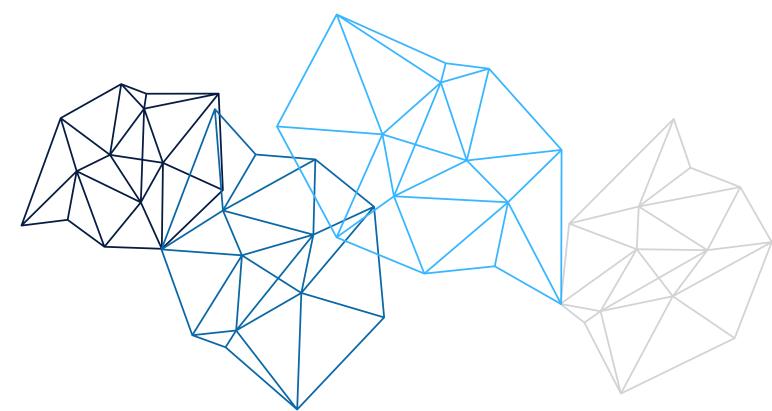
- Crear un template simple: Supongamos que quieres crear un archivo HTML básico utilizando Django. En tu directorio de aplicaciones, crea una carpeta llamada "templates". Dentro de ella, crea un archivo HTML llamado "base.html" y agrega el siguiente contenido

```
<!DOCTYPE html>
<html>

    <head>
        <title>Mi sitio web</title>
    </head>

    <body>
        <h1>Bienvenido a mi sitio web</h1>
        {% block content %}{% endblock %}
    </body>

</html>
```

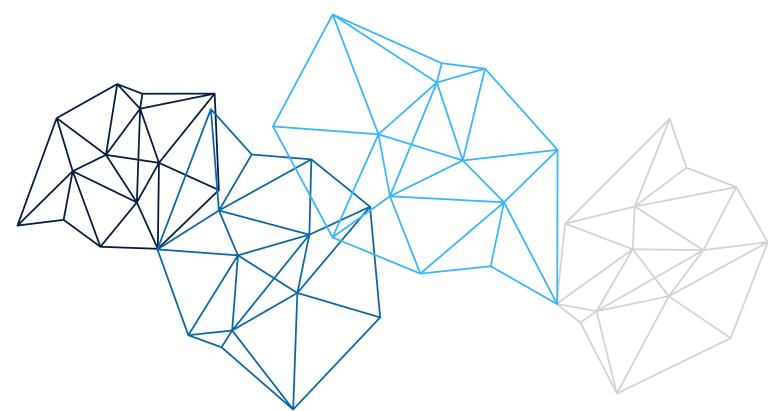


- Extender un template: Puedes extender el template base anterior para crear una nueva página. Por ejemplo, creamos un archivo "home.html" dentro de la carpeta "templates" con el siguiente contenido:

```
{% extends 'base.html' %}

{% block content %}
    <p>Esta es la página de inicio de mi sitio web.</p>
{% endblock %}
```

INFORMATORIO
Hacia una mejor industria informática



- Pasar datos al template: Puedes pasar datos desde tu vista a un template para mostrarlos dinámicamente. Por ejemplo, supongamos que tienes una vista que pasa una lista de nombres de usuarios al template. Aquí tienes un ejemplo de cómo puedes mostrarlos:

En tu vista (archivo views.py):

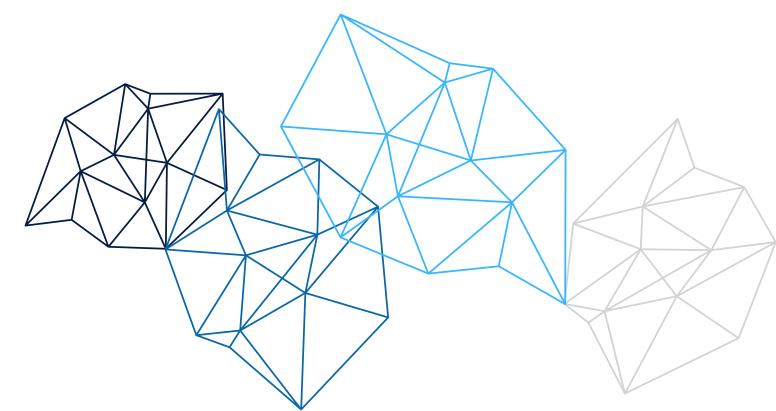
```
from django.shortcuts import render

def usuarios(request):
    nombres = ['Juan', 'María', 'Pedro']
    return render(request, 'usuarios.html', {'nombres': nombres})
```

En el archivo de template "usuarios.html":

```
{% extends 'base.html' %}

{% block content %}
    <h2>Lista de usuarios:</h2>
    <ul>
        {% for nombre in nombres %}
            <li>{{ nombre }}</li>
        {% endfor %}
    </ul>
{% endblock %}
```



- Uso de inclusión de templates:

Template de navegación – navigation.html:

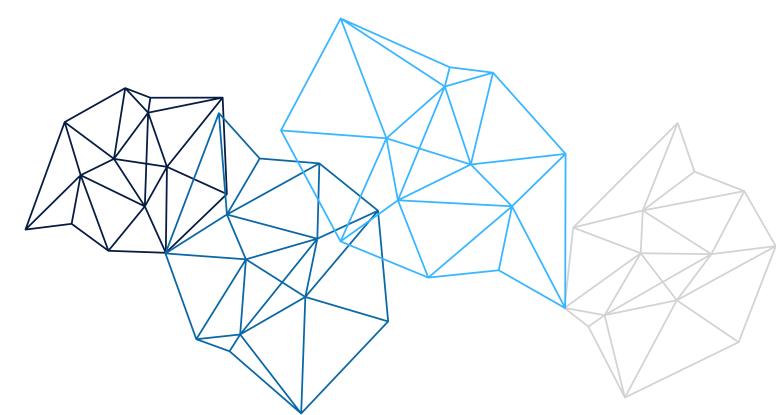
```
<nav>
  <ul>
    <li><a href="/">Inicio</a></li>
    <li><a href="/productos/">Productos</a></li>
    <li><a href="/contacto/">Contacto</a></li>
  </ul>
</nav>
```

Template principal – base.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Mi sitio web</title>
</head>
<body>
  {% include 'navigation.html' %}

  <main>
    {% block content %}{% endblock %}
  </main>

  <footer>
    <p>Derechos de autor © 2023. Todos los derechos reservados.</p>
  </footer>
</body>
</html>
```



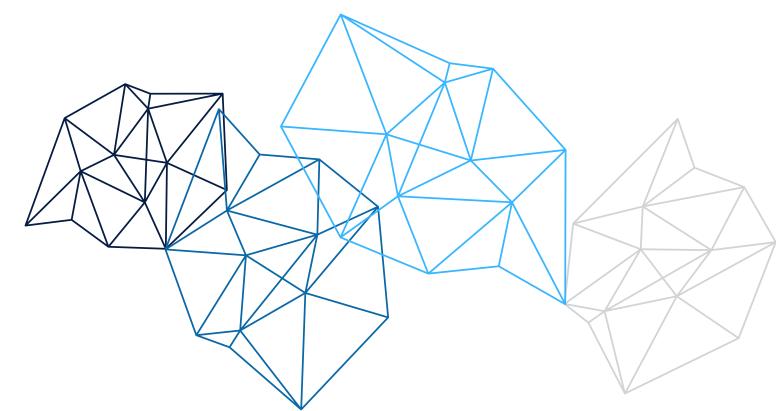
- Uso de condiciones

```
{% if variable %}  
    <p>El valor de la variable es verdadero.</p>  
{% else %}  
    <p>El valor de la variable es falso.</p>  
{% endif %}
```

- Uso de ciclos

```
<ul>  
    {% for item in lista %}  
        <li>{{ item }}</li>  
    {% endfor %}  
</ul>
```

ORIO
informática



Notaciones para distinguirlas:

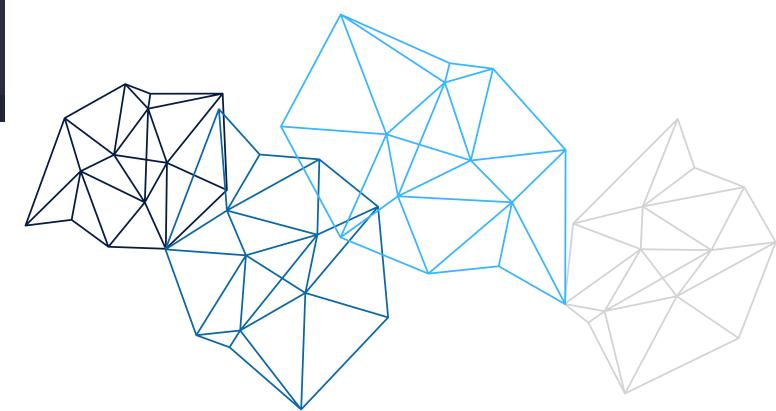
- Variables {{ variable }}
- Instrucciones % instrucción %
- Comentarios #{ comentarios #}

En Django, puedes utilizar la función {% url %} en tus templates para llamar a una URL definida en tu archivo de configuración de URLs (urls.py). La función {% url %} se encarga de generar dinámicamente la URL correspondiente a una vista específica.

Supongamos que tienes una URL definida en tu archivo urls.py de la siguiente manera:

```
from django.urls import path
from . import views

urlpatterns = [
    path('contacto/', views.contacto, name='contacto'),
]
```



Luego, en tu template, puedes llamar a la URL utilizando la función `{% url %}` de la siguiente manera:

```
<a href="{% url 'contacto' %}">Ir a la página de contacto</a>
```

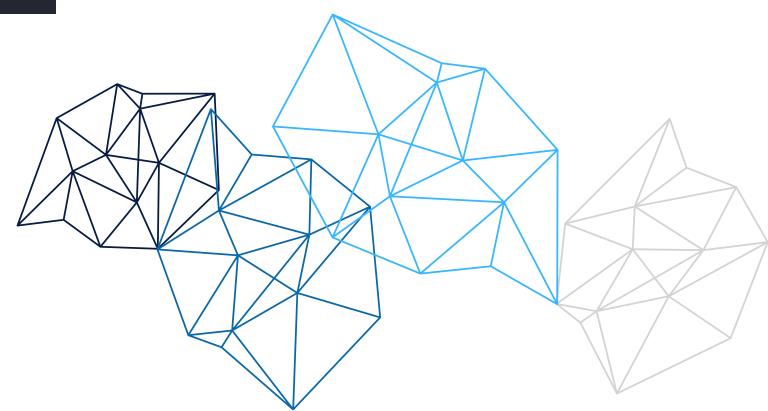
En este caso, 'contacto' es el nombre asignado a la URL mediante el parámetro name en tu archivo urls.py. La función `{% url %}` buscará el nombre proporcionado y generará la URL correspondiente a la vista contacto.

**Si la URL definida en tu archivo urls.py requiere argumentos, también puedes pasarlos a la función `{% url %}`. Por ejemplo, si tienes una URL que recibe un parámetro id:

```
urlpatterns = [
    path('producto/<int:id>', views.detalle_producto, name='detalle_producto'),
]
```

En el template, puedes llamar a la URL y pasar el argumento de la siguiente manera:

```
<a href="{% url 'detalle_producto' id=1 %}">Ver detalles del producto</a>
```



Uso de filtros:

Formateo de fechas:

```
<p>Fecha actual: {{ fecha_actual|date:"d/m/Y" }}</p>
```

Ordenamiento de listas:

```
<ul>
    {% for item in lista|sort %}
        <li>{{ item }}</li>
    {% endfor %}
</ul>
```

Conversión a mayúsculas o minúsculas:

```
<p>Texto en mayúsculas: {{ texto|upper }}</p>
<p>Texto en minúsculas: {{ texto|lower }}</p>
```

Existen muchos más, te invitamos a investigar y probar todos estos códigos!

