EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: CREANDO PLANTILLAS CON BUCLES Y CONDICIONES.
- EXERCISE 2: PLANTILLAS CON FILTROS Y CARGADORES.
- EXERCISE 3: PLANTILLAS INCRUSTADAS, HERENCIA DE PLANTILLAS Y AGREGANDO UN NAVBAR DE BOOTSTRAP A VARIAS PLANTILLAS HIJAS.

EXERCISE 1: CREANDO PLANTILLAS CON BUCLES Y CONDICIONES

El objetivo del presente ejercicio es plantear una guía paso a paso para entender las variables, propiedades, bucles, condiciones, filtros, cargadores, plantillas incrustadas y herencias de plantillas.

Requisitos previos:

- Tener conocimiento de una terminal o línea de comando e instalación de paquetes de software en el sistema operativo, tanto en Windows 10, como en Linux (en este caso, Linux Ubuntu).
- Tener previamente instalada la versión de Python 3 y el virtualenv.
- Hacer uso de la herramienta Visual Studio Code.

CREANDO PLANTILLAS CON BUCLES Y CONDICIONES

Previamente creamos el entorno virtual `project_django`, y dentro el proyecto `site_web_django` y una aplicación boards. Para abrir el proyecto nos ubicamos en el directorio `site_web_django`, y lo hacemos con VSC:

```
$ cd site_web_django

$ code .
```

Activamos el proyecto con `workon`:

```
$ workon projects_django
```

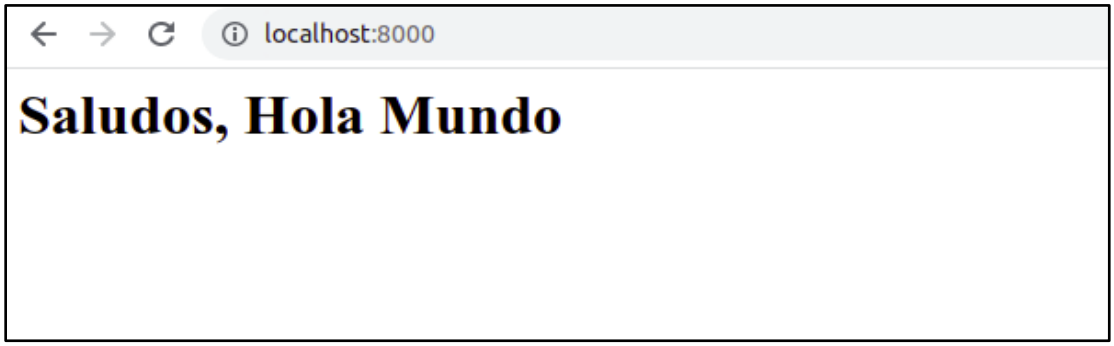
Ejecutamos la aplicación:

```
$ python manage.py runserver
```

Observamos en la terminal que se está ejecutando el servidor en: http://127.0.0.1:8000/

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 10, 2022 - 20:35:21
Django version 4.0.5, using settings 'site_web_django.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Tenemos un contenido estático que muestra lo siguiente:



Para efectos prácticos, procedemos a crear un objeto tipo persona en la vista, en el cual podemos acceder a cada uno de los atributos del objeto y mostrarlo en la plantilla. Para ello, procedemos a adecuar en la vista creando el objeto de la siguiente manera:

BOARDS/VIEWS.PY

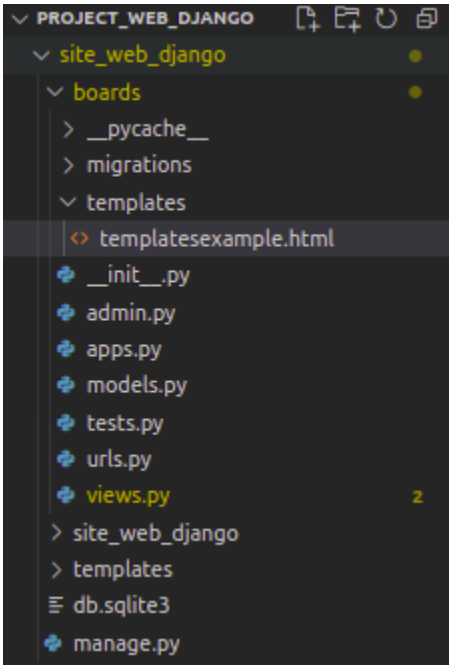
```
from django.views.generic import TemplateView
from django.shortcuts import render
import datetime

class Persona(object):
    def __init__ (self, nombre, apellido):
        self.nombre=nombre
        self.apellido=apellido


class IndexPageView(TemplateView):
    template_name = "index.html"

def obtenerFecha(request, name):
    fechaActual = datetime.datetime.now()
    context = { 'fecha' : fechaActual, 'name' : name}
    return  render(request, 'fecha.html', context)

def menuView(request):
    template_name = 'menu.html'
    return render(request, template_name)
```

Procedemos a crear una nueva vista llamada mostrar, agregando el siguiente método al final del archivo:

BOARDS/VIEWS.PY

```
def mostrar(request):
    persona = Persona("Juan", "Peréz")
    context = {'nombre' : persona.nombre, "apellido" : persona.apellido}
    return render(request, "templatesexample.html", context)
```

Luego, procedemos a crear nuestra plantilla `templatesexample.html`. Previamente creamos un directorio templates dentro de boards, quedando la siguiente estructura:



Agregamos al archivo `templatesexample.html` lo siguiente, que mostrará solo el nombre y apellido:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

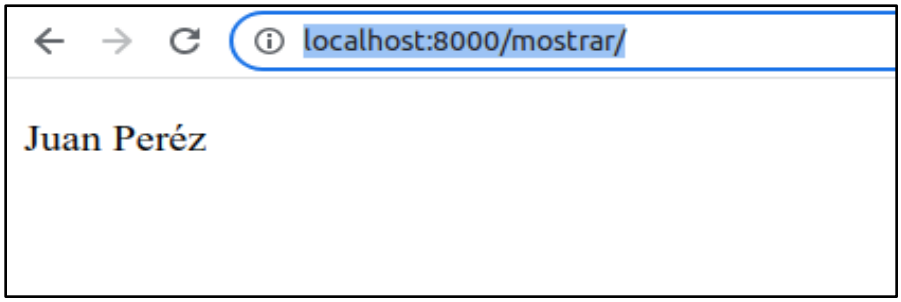
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de Plantillas</title>
</head>
<body>
    <p>{{ nombre }} {{ apellido }}</p>

</body>
</html>
```

Actualizamos la URL, agregando la nueva vista creada:

BOARDS/URLS.PY

```
from django.urls import path

from .views import IndexPageView, obtenerFecha, menuView, mostrar

urlpatterns = [
    path('', IndexPageView.as_view(), name='index'),
    path('fecha/<name>',  obtenerFecha, name='index'),
    path('menu/',  menuView, name='menu'),
    path('mostrar/',  mostrar, name='mostrar'),
```

Ejecutamos el servidor de Django, y observamos en: http://localhost:8000/mostrar/



Procedemos a crear una lista de elementos en la vista, para mostrarla en la plantilla:

BOARDS/VIEWS.PY

```
def mostrar(request):
    persona = Persona("Juan", "Peréz")
    items=["Primero", "Segundo", "Tercero", "Cuarto"]
    context = {'nombre' : persona.nombre, "apellido" : persona.apellido,
"items" : items}
    return render(request, "templatesexample.html", context)
```

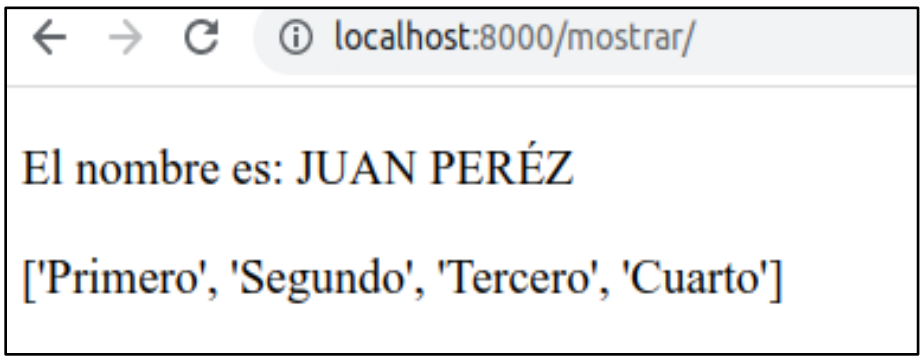
Y actualizamos la plantilla para mostrar la lista de elementos:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>
    <p>El nombre es: {{ nombre.upper }} {{ apellido.upper }}</p>
    <p>{{ items }}</p>


</body>
</html>

```

Observamos lo siguiente:



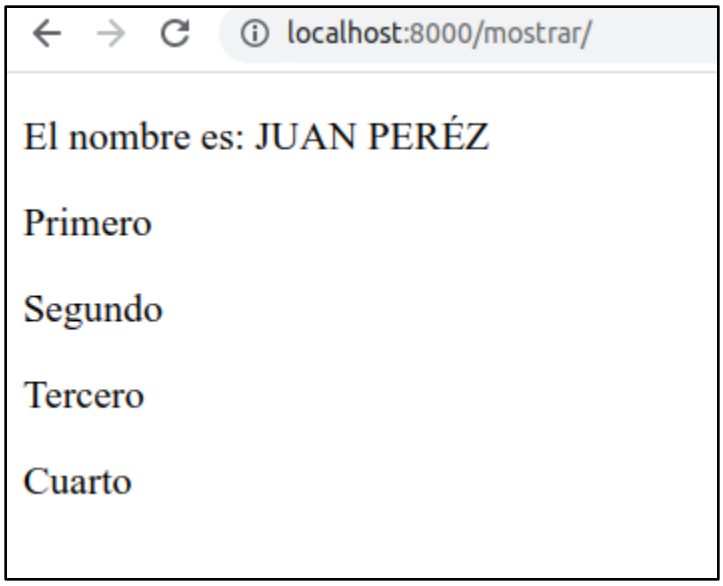
Para realizar una lista de elementos podemos adecuar el código de la siguiente manera:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>
    <p>El nombre es: {{ nombre.upper }} {{ apellido.upper }}</p>
```

```
    <p>{{ items.0 }}</p>
    <p>{{ items.1 }}</p>
    <p>{{ items.2 }}</p>
    <p>{{ items.3 }}</p>

</body>
</html>
```

Obteniendo:



Para mejorar el código podemos hacer uso del bucle **FOR**, es:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>
```

```
    <p>El nombre es: {{ nombre.upper }} {{ apellido.upper }}</p>

    {% for item in items %}
        <li>{{ item }}</li>
    {% endfor%}

</body>
</html>

```

Teniendo como resultado:

Para hacer uso del condicional **IF**, vamos a validar que la lista no esté vacía. Si lo está, muestra al usuario un mensaje especificando que la misma no contiene ítems, adecuando el template de la siguiente manera:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>

    <p>El nombre es: {{ nombre.upper }} {{ apellido.upper }}</p>

    {% if items %}

```

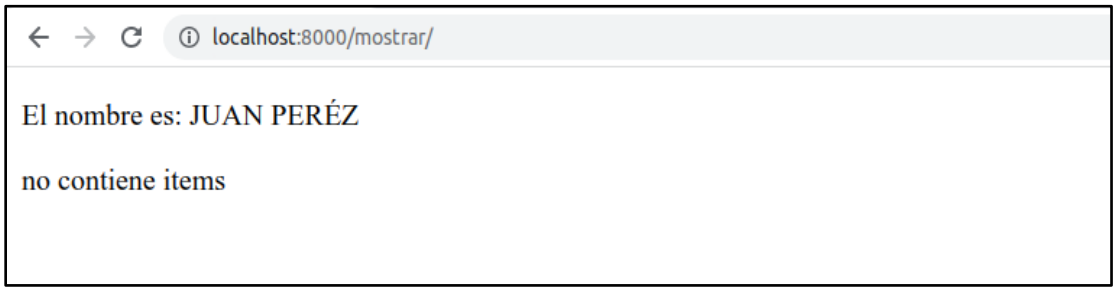
```
15          {% for item in items %}
16
17              <li>{{ item }}</li>
18
19          {% endfor%}
20
21      {% else %}
22
23          <p>no contiene items</p>
24
25      {% endif%}
26
27
28  </body>
29  </html>
```

Adecuamos que la lista esté vacía en la vista:

BOARDS/VIEWS.PY

```
1  def mostrar(request):
2      persona = Persona("Juan", "Peréz")
3      items=[]
4      context = {'nombre' : persona.nombre, "apellido" : persona.apellido,
5  "items" : items}
6      return render(request, "templatesexample.html", context)
```

Obtenemos como resultado:



EXERCISE 2: PLANTILLAS CON FILTROS Y CARGADORES.

Vamos a agregar un nuevo atributo al objeto persono, el cual contiene un campo login, que es de tipo booleano y nos indicará si está logeado o no el usuario. Si está logeado, el login es verdadero; en caso contrario, no se encuentra logeado, y para esto modificamos la clase persona.

BOARDS/VIEWS.PY

```
from django.views.generic import TemplateView
from django.shortcuts import render
import datetime

class Persona(object):
    def __init__ (self, nombre, apellido, login):
        self.nombre=nombre
        self.apellido=apellido
        self.login=login


class IndexPageView(TemplateView):
    template_name = "index.html"

def obtenerFecha(request, name):
    fechaActual = datetime.datetime.now()
    context = { 'fecha' : fechaActual, 'name' : name}
    return  render(request, 'fecha.html', context)

def menuView(request):
    template_name = 'menu.html'
    return render(request, template_name)

def mostrar(request):
    persona = Persona("Juan", "Peréz", False)
    # items=["Primero", "Segundo", "Tercero", "Cuarto"]
    items=[]
context = {'nombre' : persona.nombre, "apellido" : persona.apellido,
"login" : persona.login, "items" : items}
    return render(request, "templatesexample.html", context)
```

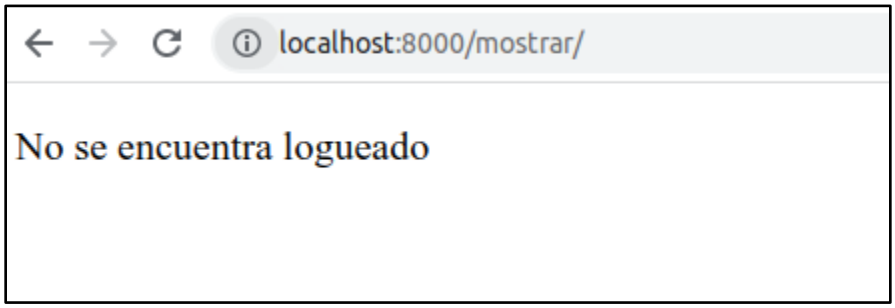
Adecuando el template con la finalidad de que muestre el contenido si el atributo es login:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

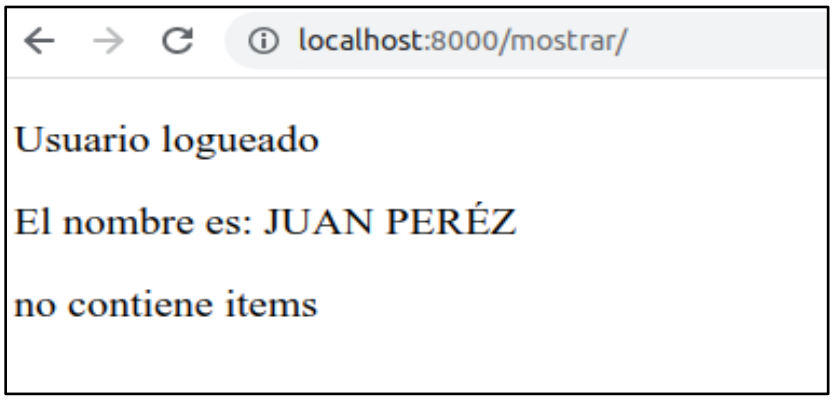
```
<!DOCTYPE html>
```

```
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>

    {% if login %}
        <p>Usuario logueado</p>
        <p>El nombre es: {{ nombre.upper }} {{ apellido.upper }}</p>

        {% if items %}
            {% for item in items %}
                <li>{{ item }}</li>
            {% endfor%}
        {% else %}
            <p>no contiene items</p>
        {% endif%}

    {% else %}
       <p>No se encuentra logueado</p>
    {% endif %}
</body>
</html>
```

Observamos:



Si cambiamos en la vista el valor de la variable login por `True`, tenemos:



Procedemos a trabajar con los filtros. Para más referencia se pueden revisar los Hints, o en:

https://www.w3schools.com/django/django_tags_filter.php

Vamos a colocar el nombre con el filtro de title, y apellidos en letra mayúscula con el filtro `upper`, esto es en el template:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>

    {% if login %}
        <p>Usuario logueado</p>
        <p>El nombre es: {{  nombre|title }} {{ apellido.upper }}</p>

        {% if items %}
            {% for item in items %}
                <li>{{ item }}</li>
            {% endfor%}
        {% else %}
            <p>no contiene items</p>
        {% endif%}

```

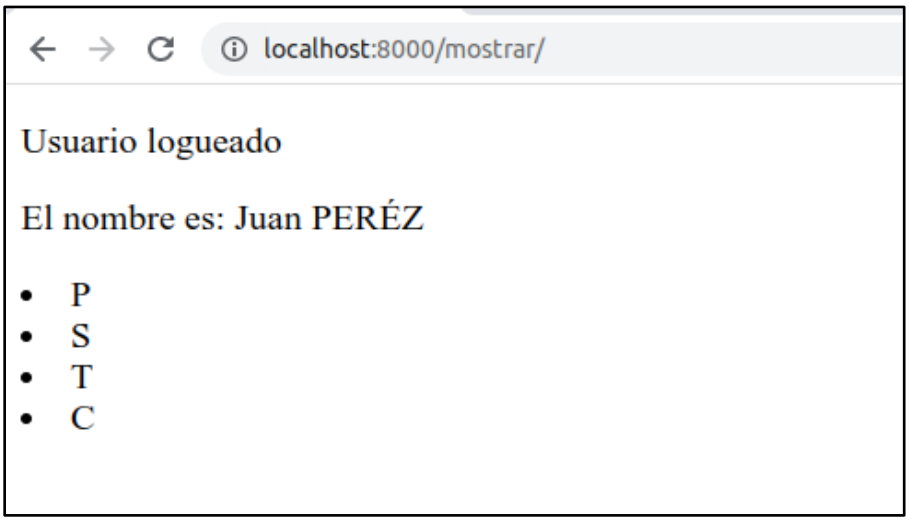
```
    {% else %}
       <p>No se encuentra logueado</p>
    {% endif %}
</body>
</html>
```

Adicionalmente, si queremos observar que se imprima el primer carácter de los ítems en mayúsculas, tenemos:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>

    {% if login %}
        <p>Usuario logueado</p>
        <p>El nombre es: {{ nombre|title }} {{ apellido|upper }}</p>

        {% if items %}
            {% for item in items %}
                <li>{{ item|first|upper }}</li>
            {% endfor%}
        {% else %}
            <p>no contiene items</p>
        {% endif%}

    {% else %}
       <p>No se encuentra logueado</p>

    {% endif %}


</body>
</html>
```

Tenemos como resultado:



Podemos agregar una nueva variable de tiempo, con la finalidad de observar la hora actual. Para ello agregamos en la vista lo siguiente:

BOARDS/VIEWS.PY

```
def mostrar(request):
    persona = Persona("Juan", "Peréz", True)
    items=["Primero", "Segundo", "Tercero", "Cuarto"]
    hrs= datetime.datetime.now()
    #items=[]
    context = {'nombre' : persona.nombre, "apellido" : persona.apellido,
"login" : persona.login, "items" : items, "hora" : hrs}
    return render(request, "templatesexample.html", context)
```

Y en la plantilla:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
```

```
</head>
<body>

    {% if login %}
        <p>Usuario logueado</p>
        <p>El nombre es: {{ nombre|title }} {{ apellido|upper }}</p>
        <p>La hora es: {{ hora|time  }}</p>

        {% if items %}
            {% for item in items %}
                <li>{{ item|first|upper }}</li>
            {% endfor%}
        {% else %}
            <p>no contiene items</p>
        {% endif%}

    {% else %}
       <p>No se encuentra logueado</p>

    {% endif %}


</body>
</html>
```

EXERCISE 3: PLANTILLAS INCRUSTADAS, HERENCIA DE PLANTILLAS Y AGREGANDO UN NAVBAR DE BOOTSTRAP A VARIAS PLANTILLAS HIJAS.

Partiendo de que hemos creado previamente, y creando un archivo de plantilla base con Bootstrap, el cual contiene lo siguiente:

/TEMPLATES/BASE.HTML

```
<!DOCTYPE html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>

```

```
    {% if login %}
        <p>Usuario logueado</p>
        <p>El nombre es: {{ nombre|title }} {{ apellido|upper }}</p>
        <p>La hora es: {{ hora|time  }}</p>

        {% if items %}
            {% for item in items %}
                <li>{{ item|first|upper }}</li>
            {% endfor%}
        {% else %}
            <p>no contiene items</p>
        {% endif%}

    {% else %}
       <p>No se encuentra logueado</p><!DOCTYPE html>
<html lang="en">
<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">
    {% load bootstrap5 %}

    {% bootstrap_css %}

    {% bootstrap_javascript %}

    {% bootstrap_messages %}
    <title>Bootstrap Navbar</title>
</head>
<body>
    {% block content %}

    {% endblock content %}

</body>
</html>
```

Creamos una plantilla menú, que hereda de la plantilla base y lee Bootstrap, el cual contiene lo siguiente:

/TEMPLATES/MENU.HTML

```
{% extends 'base.html' %}
{% load bootstrap5 %}

{% block content %}
```

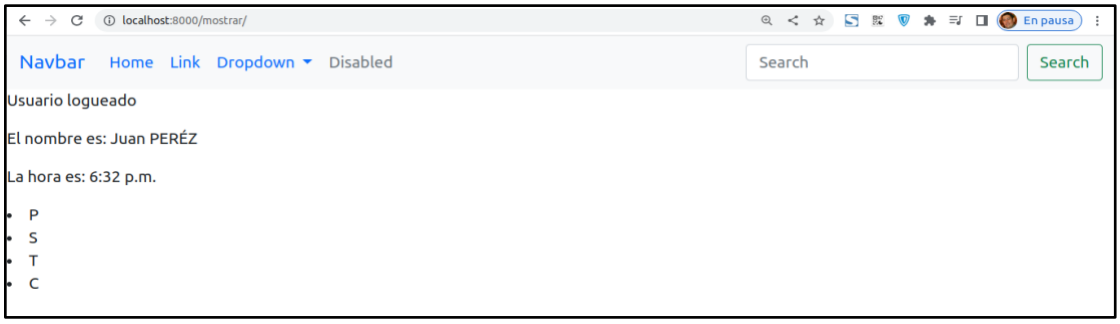
```
<nav class="navbar navbar-expand-lg bg-light">
    <div class="container-fluid">
      <a class="navbar-brand" href="#">Navbar</a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page"
href="#">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Link</a>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#"
id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-
expanded="false">
              Dropdown
            </a>
            <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
              <li><a class="dropdown-item" href="#">Action</a></li>
              <li><a class="dropdown-item" href="#">Another
action</a></li>
              <li><hr class="dropdown-divider"></li>
              <li><a class="dropdown-item" href="#">Something else
here</a></li>
            </ul>
          </li>
          <li class="nav-item">
            <a class="nav-link disabled">Disabled</a>
          </li>
        </ul>
        <form class="d-flex" role="search">
          <input class="form-control me-2" type="search"
placeholder="Search" aria-label="Search">
          <button class="btn btn-outline-success"
type="submit">Search</button>
        </form>
      </div>
    </div>
  </nav>
  {% endblock %}
```

Procedemos a incluir en nuestra plantilla templatesexample.html la barra de menú que corresponde a la plantilla menu.html, es decir, tenemos nuestra plantilla de la siguiente manera:

BOARDS/TEMPLATES/TEMPLATESEXAMPLE.HTML

```
{% extends 'base.html>
<html lang="en">
<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>Ejemplo de Plantillas</title>
</head>
<body>
    {% include "menu.html" %}

    {% if login %}
        <p>Usuario logueado</p>
        <p>El nombre es: {{ nombre|title }} {{ apellido|upper }}</p>
        <p>La hora es: {{ hora|time  }}</p>

        {% if items %}
            {% for item in items %}
                <li>{{ item|first|upper }}</li>
            {% endfor%}
        {% else %}
            <p>no contiene items</p>
        {% endif%}

    {% else %}
       <p>No se encuentra logueado</p>

    {% endif %}


</body>
</html>
```

Y observamos:


localhost:8000/mostrar/
En pausa
Navbar
Home
Link
Dropdown
Disabled
Search
Search
Usuario logueado
El nombre es: Juan PERÉZ
La hora es: 6:32 p.m.
P
S
T
C