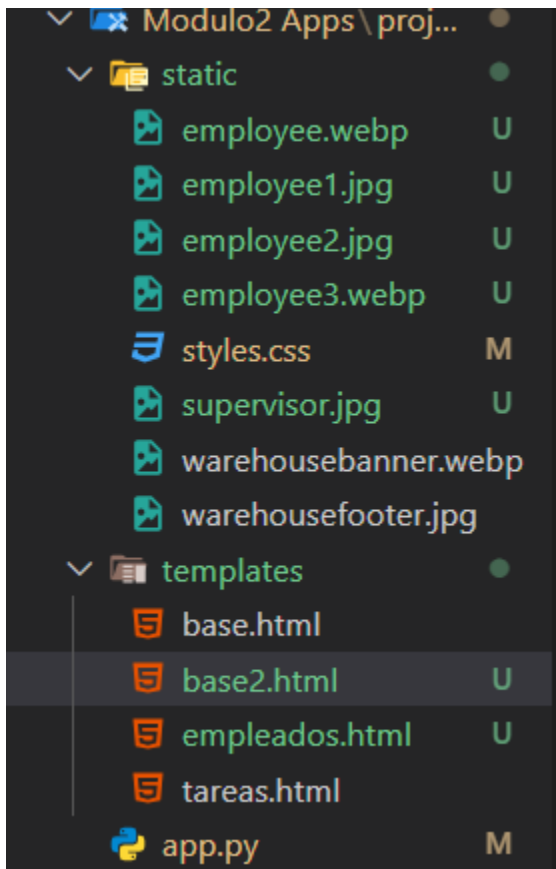


## Actividad de la Lección

La actividad de esta lección está diseñada para que apliques los conceptos aprendidos sobre sistemas de plantillas Front-End con Flask. Crearás una aplicación básica que utiliza plantillas HTML dinámicas para mostrar datos provenientes del Back-End. Esta práctica te permitirá entender la importancia de la separación entre la capa lógica y la capa de presentación, y cómo estructurar plantillas utilizando la herencia de Jinja2. Al completar la actividad, habrás dado los primeros pasos en el diseño de plantillas personalizadas para proyectos más avanzados, como tu proyecto capstone.

1. **Configura un proyecto Flask** de tu creatividad en tu máquina siguiendo la estructura mostrada en clase y la lección.
2. Utiliza los ejemplos que el profesor ha provisto en clase o el repositorio de la clase (<https://github.com/javierdastas/comp2052>)
3. **Crea una plantilla base** para una aplicación de tu creación.
4. Desarrolla al menos **dos plantillas específicas** utilizando datos dinámicos, como listas o tablas.
5. Usa **herencia de plantillas** para organizar el diseño, incluye una página principal y desde ella incluye el contenido de dos páginas adicionales.
6. Reflexiona sobre cómo la separación entre Back-End y Front-End mejora la claridad y escalabilidad de tu proyecto.
7. Sube tu trabajo en GITHUB.
8. Desarrolla un documento en formato PDF documentando tu aplicación.
9. Incluye en el documento capturas de pantalla de pruebas.
10. Explica brevemente cada pantalla.
11. Incluye el enlace de tu Github en el documento.

Estructura basada en los discutido en clase:



Archivo de estilos para ambas aplicaciones:

```
base2.html U empleados.html U styles.css M X
Modulo2 Apps > project > static > styles.css > .header-image
You, 1 second ago | 1 author (You)
1 body {
2     background-color: lightsalmon;
3     margin: 0;
4     padding: 15;
5 }
6 .titlefont {
7     font-weight: bolder;
8     color: white;
9     text-align: center;
10    font-family: 'Times New Roman', Times, serif;
11 }
12 .letter {
13     color: white;
14     font-family: 'Times New Roman', Times, serif;
15     font-weight: bold;
16     font-size: large;
17 }
18 .header-image {
19     width: 100%;
20     height: 150px;
21     object-fit: cover;
22 }
23 .footer-image {
24     width: 100%;
25     height: 150;
26     object-fit: cover;
27 }
28 .employeePic {
29     width: 500px;
30     height: auto;
31     border-radius: 10%;
32 }
```

```
33 table {
34     width: 100%;
35     border-collapse: collapse;
36     margin-top: 20px;
37     background-color: lightpink;
38 }
39
40 table, th, td {
41     border: 1px solid #333; /* Color y grosor del borde */
42 }
43
44 th, td {
45     padding: 10px;
46     text-align: center;
47 }
48
49
```

Código del app.py:

- El código en app.py se encarga de permitir que el usuario vaya a las rutas de empleados y tareas mediante funciones. Mis plantillas base contienen elementos html como header, body y footer y Jinja nos permite heredarlos a nuestras páginas mediante “{% extends 'base.html' %}”, en el caso de tareas.html.

```

You, 28 minutes ago | 1 author (You)
1  from flask import Flask, render_template
2
3  app = Flask(__name__)
4
5  @app.route('/tareas')
6  def tareas():
7      tareas = [
8          "Limpiar muestras",
9          "Verificar inventario",
10         "Entregar mercancía",
11         "Organizar almacén",
12         "Recibir mercancía",
13         "Enviar mercancía"
14     ]
15     return render_template('tareas.html', tareas=tareas)
16
17 @app.route('/empleados')
18 def empleados():
19     empleados_info = [
20         {"nombre": "Carlos Pérez", "rol": "Carpintero", "foto": "employee1.jpg"},
21         {"nombre": "Ana López", "rol": "Supervisora", "foto": "supervisor.jpg"},
22         {"nombre": "Luis Ramírez", "rol": "Carpintero", "foto": "employee2.jpg"},
23         {"nombre": "James Díaz", "rol": "Mantenimiento", "foto": "employee3.webp"}
24     ]
25     return render_template('empleados.html', empleados=empleados_info)
26
27 if __name__ == '__main__':
28     app.run(debug=True)

```

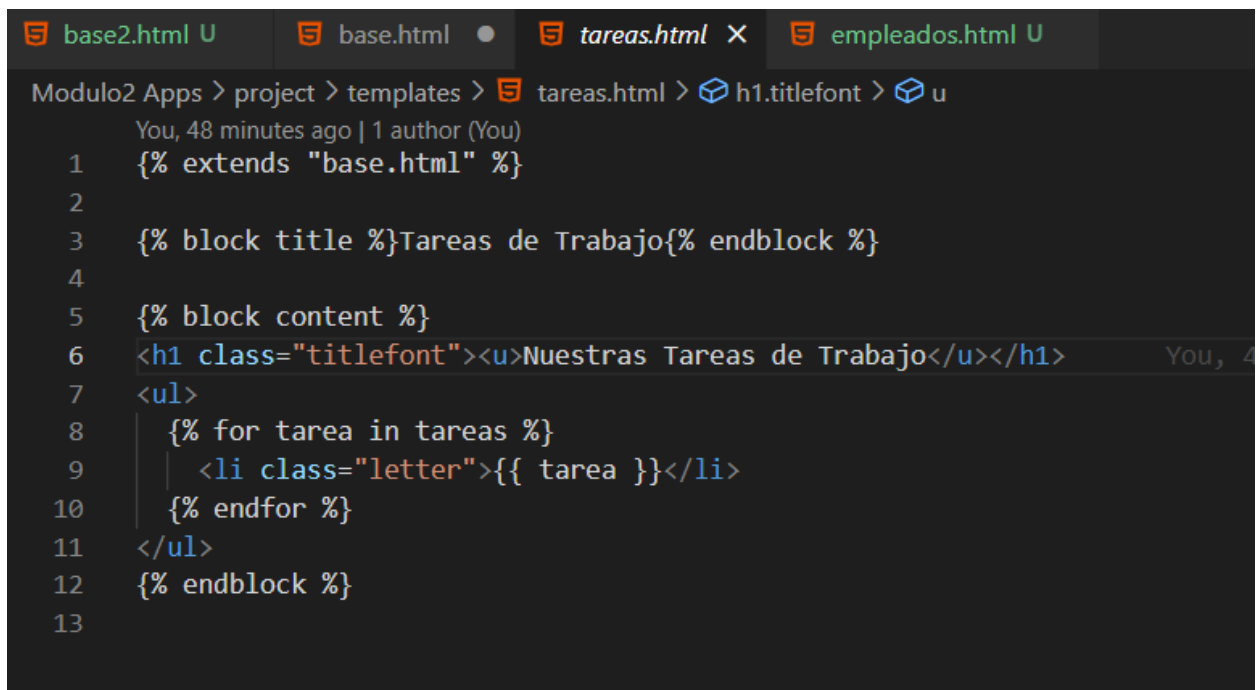
Plantilla para página de tareas:

```

base2.html U  base.html  empleados.html U
Modulo2 Apps > project > templates > base.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>{% block title %}Mi Aplicación{% endblock %}</title>
5      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles.css') }}">
6  </head>
7  <body>
8      <header class="header-image"> </header>
10     <main>
11         {% block content %}{% endblock %}
12     </main>
13     <footer></footer>
14 </body>
15 </html>

```

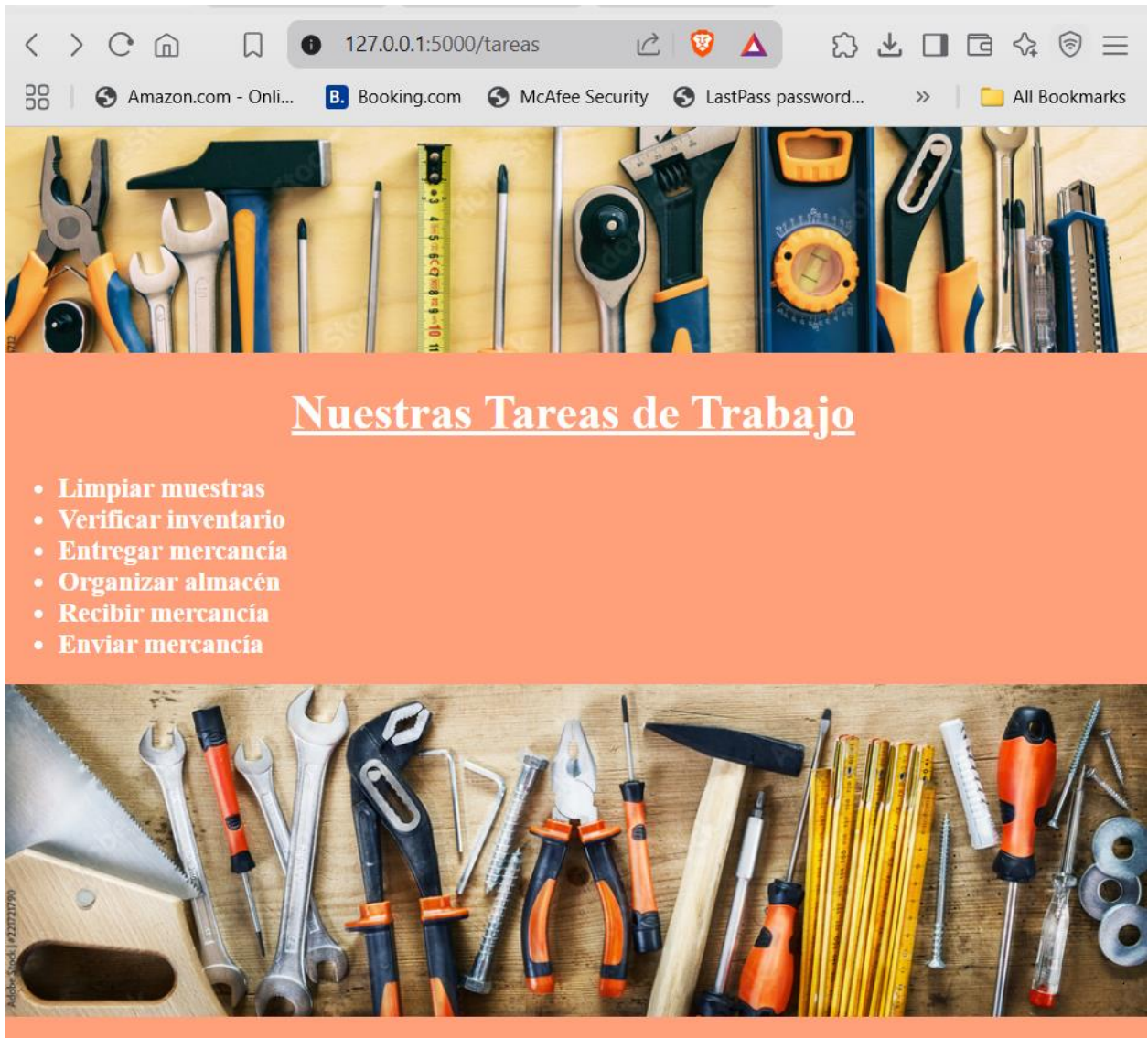
Página de tareas:



The screenshot shows a web browser with four tabs: 'base2.html U', 'base.html', 'tareas.html X', and 'empleados.html U'. The address bar shows the path 'Modulo2 Apps > project > templates > tareas.html > h1.titlefont > u'. The page content is a Jinja2 template for a task page. It extends 'base.html' and includes a title 'Tareas de Trabajo' and a content block. The content block contains an h1 tag with class 'titlefont' and text 'Nuestras Tareas de Trabajo', followed by a list of tasks. The list is generated by a loop over 'tareas'.

```
1 {% extends "base.html" %}
2
3 {% block title %}Tareas de Trabajo{% endblock %}
4
5 {% block content %}
6 <h1 class="titlefont"><u>Nuestras Tareas de Trabajo</u></h1>
7 <ul>
8     {% for tarea in tareas %}
9         <li class="letter">{{ tarea }}</li>
10     {% endfor %}
11 </ul>
12 {% endblock %}
13
```

Output de la página de tareas:



127.0.0.1:5000/tareas

Amazon.com - Onli... Booking.com McAfee Security LastPass password... All Bookmarks

## Nuestras Tareas de Trabajo

- Limpiar muestras
- Verificar inventario
- Entregar mercancía
- Organizar almacén
- Recibir mercancía
- Enviar mercancía

## Plantilla de empleados:


```
base2.html U x base.html app.py M empleados.html U
Modulo2 Apps > project > templates > base2.html > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>{% block title %}Mi Aplicación{% endblock %}</title>
5 <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='styles.css') }}">
6 </head>
7 <body>
8 <header class="header-image">
9 
10 </header>
11
12 <main>
13 {% block content %}{% endblock %}
14 </main>
15
16 <footer>
17 
18 </footer>
19 </body>
20 </html>
```


## Página de empleados:



```
base2.html U base.html app.py M empleados.html U x
Modulo2 Apps > project > templates > empleados.html > table > tbody.letter > tr > td > img.employeePic
1 {% extends 'base2.html' %}
2
3 {% block title %}Empleados - Carpintería{% endblock %}
4
5 {% block content %}
6 <h2 class="titlefont">Lista de Empleados</h2>
7 <table>
8 <thead>
9 <tr class="letter">
10 <th>Nombre</th>
11 <th>Rol</th>
12 <th>Foto</th>
13 </tr>
14 </thead>
15 <tbody class="letter">
16 {% for empleado in empleados %}
17 <tr>
18 <td>{{ empleado.nombre }}</td>
19 <td>{{ empleado.rol }}</td>
20 <td></td>
21 </tr>
22 {% endfor %}
23 </tbody>
24 </table>
25 {% endblock %}
26
```

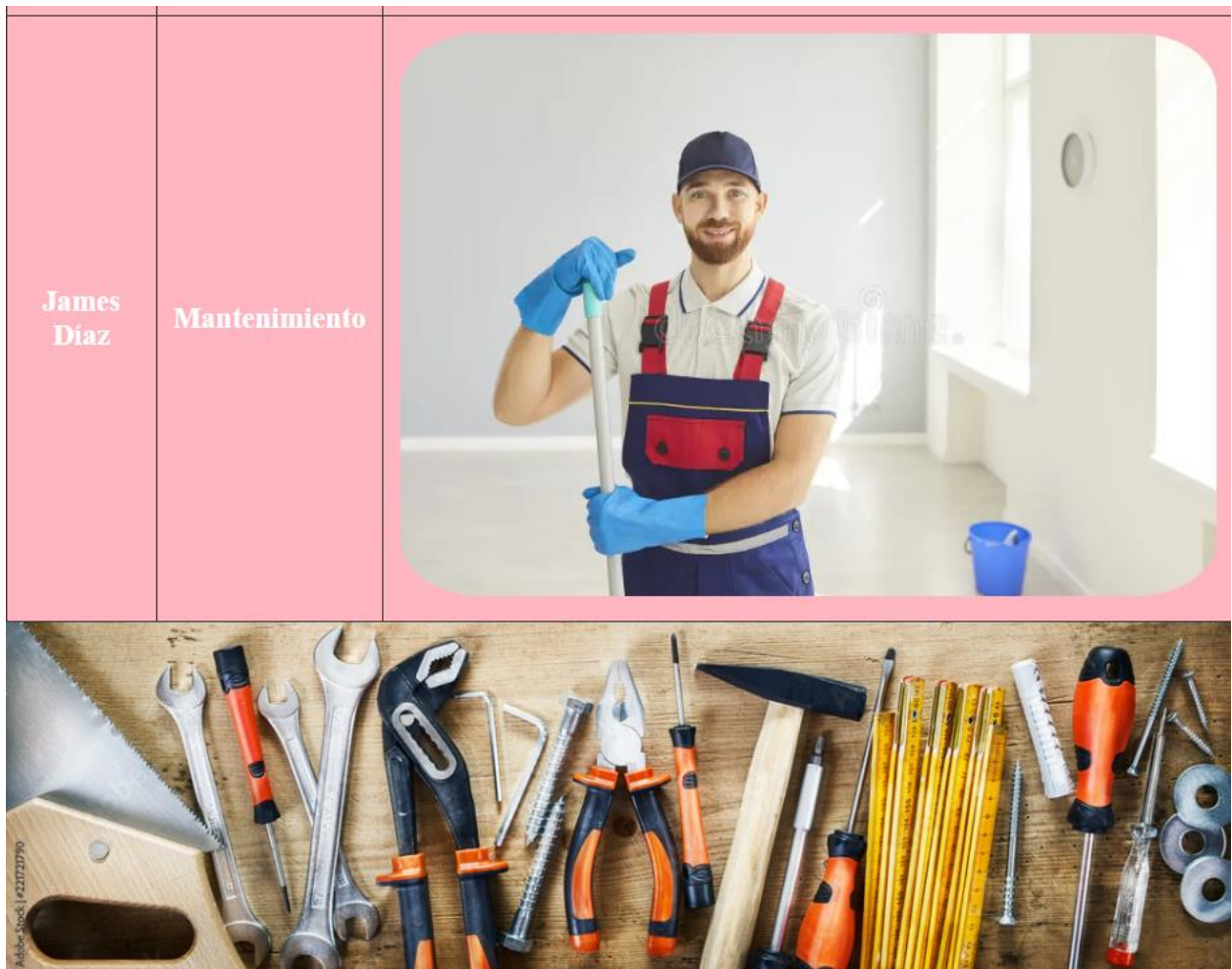


Output de la página de empleados:



Lista de Empleados		
Nombre	Rol	Foto
Carlos Pérez	Carpintero	

Ana López	Supervisora	
Luis Ramírez	Carpintero	



### Reflexión:

Separar el Front-End del Back-End puede ser muy productivo en este tipo de proyectos de aplicación. He trabajado en este formato anteriormente y con los vistos en este tipo de actividades estoy seguro de que es lo mas efectivo. Esta separación permite que los archivos necesarios estén organizados y bien estructurados, además de que el hecho de que todo este separado permite que los errores sean más fáciles de encontrar y, por lo tanto, arreglar. Este formato también permite que trabajar en grupo sea mas sencillo debido a que puede haber diferentes personas trabajando diferentes secciones de código sin interferir con los demás. Definitivamente es un formato productivo y asertivo.

### Link GitHub:

<https://github.com/Drewster64/microserver.git>