# UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE SEDE SANTO DOMINGO

## DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS

## CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

| | | |
|---|---|---|
| **PERIODO** | : | Noviembre 2023 – Marzo 2024 |
| **ASIGNATURA** | : | Programación Integrativa de componentes WEB |
| **TEMA** | : | Implementación de Custom Elements, Shadow DOM y HTML Templates |
| **NOMBRES** | : | Hernández Stefanny |
| **NIVEL-PARALELO** | : | Sexto |
| **DOCENTE** | : | Ing. Pablo Puente MSc. |
| **FECHA DE ENTREGA** | : | 13 de diciembre del 2023 |

**SANTO DOMINGO - ECUADOR**

**2023**

**Índice de contenido**

## 1. Introducción

En la evolución constante del desarrollo web, la creación de interfaces de usuario flexibles y eficientes ha sido un desafío primordial. En este contexto, la introducción de tecnologías como Custom Elements, Shadow DOM y HTML Templates ha marcado un hito significativo al proporcionar a los desarrolladores herramientas poderosas para la construcción de componentes web reutilizables y altamente encapsulados.

El objetivo fundamental de este informe es profundizar en la comprensión y aplicación de estas tecnologías, explorando cómo Custom Elements permite la creación de componentes personalizados, cómo el Shadow DOM facilita la encapsulación efectiva de estilos y lógica, y

cómo HTML Templates contribuye a la construcción de estructuras de componentes dinámicas.

## 2. Objetivos

### 2.1.Objetivo General

.Comprender a fondo la implementación de Custom Elements, Shadow DOM y HTML Templates en el desarrollo web, evaluando su impacto en la eficiencia del código, la reutilización de componentes y la seguridad de la aplicación.

### 2.2.Objetivos Específicos

2.2.1. Analizar los fundamentos de Custom Elements y cómo permiten la creación de componentes personalizados, facilitando el desarrollo modular y la reutilización de código.

2.2.2. Explorar el Shadow DOM y su capacidad para encapsular estilos y lógica de manera efectiva, mejorando la coherencia y evitando posibles conflictos en la interfaz de usuario.

2.2.3. Investigar la utilidad de HTML Templates para la creación de estructuras de componentes flexibles y dinámicas, facilitando la manipulación del DOM de manera eficiente.

## 3. Desarrollo

Iniciando desde la estructura planteada por el mockup propuesto en clase del profesor, asegúrese de seguir los siguientes pasos:

**PASO 1. Construir el template para el web component.**

El cuerpo de la página (**<body>**) contiene un elemento **<template>** con el ID "menu". Dentro de este template, se definen estilos específicos utilizando etiquetas **<style>** para el cuerpo de la página, tarjetas, pie de página, etc. A continuación, se presenta una barra de navegación (navbar) que incluye un logotipo, elementos de menú y otros componentes interactivos como

íconos de campana, mensajes y perfil de usuario.

components.js | Index.html

Lab2 > Index.html > ...

```
7     <title>customElements</title>
8
9     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
10    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
11        integrity="sha512-2eeGf6tjftPZBA9Q3HLxDn6Lq6MOdCOHeP8PVOVpDQ+2bcO+Wynpwh7l2zvAhYT5Qq+GOlA4ZMjQxSbRsI1FSQ=="
12        crossorigin="anonymous" referrerpolicy="no-referrer" />
13
14  </head>
15
16  <body>
17      <template id="menu">
18          <style>···
52          </style>
53
54          <nav class="navbar navbar-expand-lg navbar-light bg-white fixed-top" style="font-size: 18px;">···
97          </nav>
98
99          <div class="main">···
101         </div>
102
103         <footer class="mt-5">···
187         </footer>
188
189         <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
190         <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></script>
191         <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
192     </template>
193
194     <script src="components.js"></script>
195
196     <x-menu></x-menu>
197 </body>
198
199 </html>
```

Este código en JavaScript define la clase `MenuElement`, que representa un elemento personalizado llamado `x-menu`. Al extender la clase `HTMLElement`, se establece el comportamiento específico para este nuevo elemento. En el constructor, se asigna a la propiedad `template` una referencia al elemento HTML con el ID "menu". Cuando el elemento personalizado se conecta al DOM, el método `connectedCallback` clona el contenido del elemento con ID "menu" y lo añade al propio elemento. Aunque se declara el método `attributeChangedCallback`, actualmente no contiene ninguna lógica específica. Además, se establece la propiedad estática `observedAttributes` como vacía, indicando que no se observan cambios en atributos específicos. Finalmente, el elemento personalizado se registra con el nombre "x-menu" mediante `window.customElements.define`, permitiendo su uso y reconocimiento en documentos HTML.

components.js × Index.html

Lab2 > components.js > MenuElement

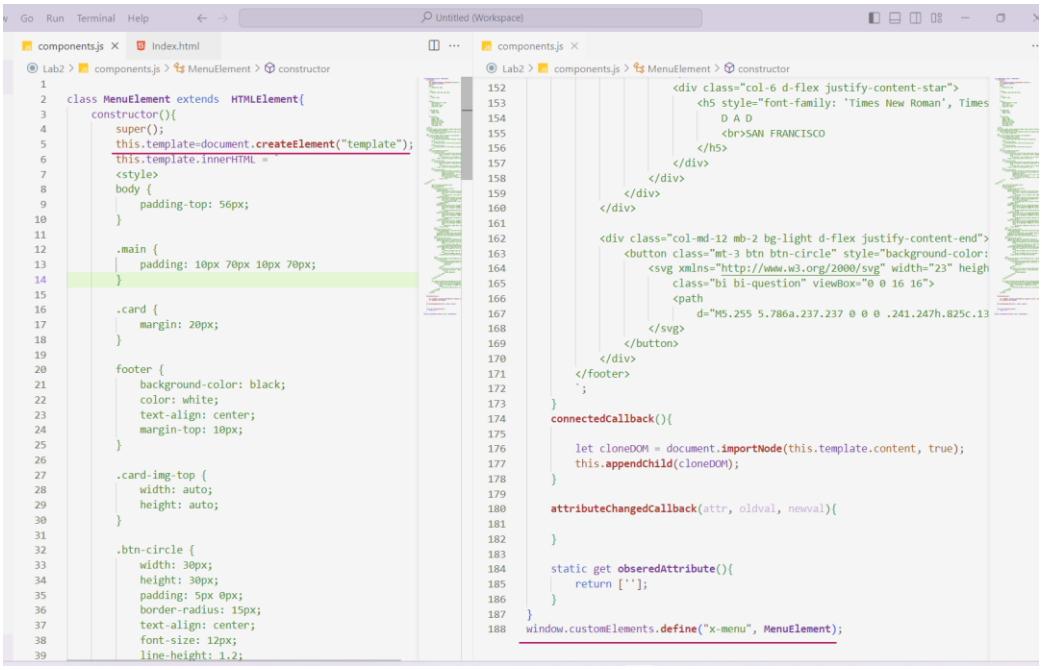
```
1
2   class MenuElement extends  HTMLElement{
3       constructor(){
4           super();
5           this.template=document.getElementById("menu");
6       }
7       connectedCallback(){
8
9           let cloneDOM = document.importNode(this.template.content, true);
10          this.appendChild(cloneDOM);
11      }
12
13      attributeChangedCallback(attr, oldval, newval){
14
15      }
16
17      static get obseredAttribute(){
18          return [''];
19      }
20  }
21  window.customElements.define("x-menu", MenuElement);
```

Resultado


**PASO 2. Ponerla disponible para su reutilización.**

Para hacer que el código del componente **MenuElement** sea reutilizable, puedes seguir estos pasos:

- **Encapsular estilos y plantilla del menú en el componente:** Mueve la sección de estilos y la plantilla del menú directamente al componente **MenuElement** en lugar de tenerlos en el documento principal.



```
class MenuElement extends  HTMLElement{
    constructor(){
        super();
        this.template=document.createElement("template");
        this.template.innerHTML = `
        <style>
        body {
            padding-top: 56px;
        }

        .main {
            padding: 10px 70px 10px 70px;
        }

        .card {
            margin: 20px;
        }

        footer {
            background-color: black;
            color: white;
            text-align: center;
            margin-top: 10px;
        }

        .card-img-top {
            width: auto;
            height: auto;
        }

        .btn-circle {
            width: 30px;
            height: 30px;
            padding: 5px 0px;
            border-radius: 15px;
            text-align: center;
            font-size: 12px;
            line-height: 1.2;
```

```
                        <div class="col-6 d-flex justify-content-star">
                            <h5 style="font-family: 'Times New Roman', Times
                                D A D
                                <br>SAN FRANCISCO
                            </h5>
                        </div>
                    </div>
                </div>

                <div class="col-md-12 mb-2 bg-light d-flex justify-content-end">
                    <button class="mt-3 btn btn-circle" style="background-color:
                        <svg xmlns="http://www.w3.org/2000/svg" width="23" heigh
                            class="bi bi-question" viewBox="0 0 16 16">
                            <path
                                d="M5.255 5.786a.237.237 0 0 0 .241.247h.825c.13
                            </svg>
                    </button>
                </div>
            </footer>
            `;
    }
    connectedCallback(){

        let cloneDOM = document.importNode(this.template.content, true);
        this.appendChild(cloneDOM);
    }

    attributeChangedCallback(attr, oldval, newval){

    }

    static get observedAttribute(){
        return [''];
    }
}
window.customElements.define("x-menu", MenuElement);
```

- **Modificar el documento principal:** En el documento HTML principal, elimina la sección de estilos y la plantilla del menú, ya que ahora están encapsuladas en el componente **MenuElement**. Solo deja la referencia al archivo **components.js** y el uso del elemento **<x-menu>**.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>customElements</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
        integrity="sha512-2eeGf6tjftPZBA9Q3HLxDn6Lq6MOdCOHeP8PVOVpDQ+2bcO+Wynpwh7l2zvAhYT5Qq+GOlA4ZMjQxSbRsI1FSQ=="
        crossorigin="anonymous" referrerpolicy="no-referrer" />

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>

<body>
    <script src="components.js"></script>
    <x-menu></x-menu>
</body>

</html>
```

Ahora ya podemos reutilizar el componente **MenuElement** en cualquier documento HTML sin tener que repetir la estructura y estilos del menú. Solo necesitas asegurarte de incluir el archivo **components.js** y utilizar el elemento **<x-menu>**.

area_Personal.html

Lab2 > area_Personal.html > ...

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>customElements</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
        integrity="sha512-2eeGf6tjftPZBA9Q3HLxDn6Lq6MOdCOHeP8PVOVpDQ+2bcO+Wynpwh7l2zvAhYT5Qq+GOlA4ZMjQxSbRsI1FSQ=="
        crossorigin="anonymous" referrerpolicy="no-referrer" />

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>

<body>
    <script src="components.js"></script>
    <x-menu></x-menu>
</body>
```

Mis_Cursos.html

Lab2 > Mis_Cursos.html > ...

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>customElements</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
        integrity="sha512-2eeGf6tjftPZBA9Q3HLxDn6Lq6MOdCOHeP8PVOVpDQ+2bcO+Wynpwh7l2zvAhYT5Qq+GOlA4ZMjQxSbRsI1FSQ=="
        crossorigin="anonymous" referrerpolicy="no-referrer" />

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</head>

<body>
    <script src="components.js"></script>
```

**PASO 3. Aplicar instancias de la web component.**

Para aplicar instancias del componente web **MenuElement**, simplemente necesitas crear instancias del elemento personalizado **<x-menu>** en tu código JavaScript y luego añadirlas al DOM donde desees que aparezcan.

Crear una nueva clase de componente web llamada **MenuElement2 y MenuElement3**.

Ahora, vamos a utilizar instancias de este componente web en tu código HTML.

components.js

Lab2 > components.js > ...

```
1   class MenuElement extends HTMLElement {···
67  }
68  window.customElements.define("x-menu", MenuElement);
69
70  class MenuElement2 extends HTMLElement {···
177 }
178 window.customElements.define("x-menu2", MenuElement2);
179
180 class MenuElement3 extends HTMLElement {···
310 }
311 window.customElements.define("x-menu3", MenuElement3);
```

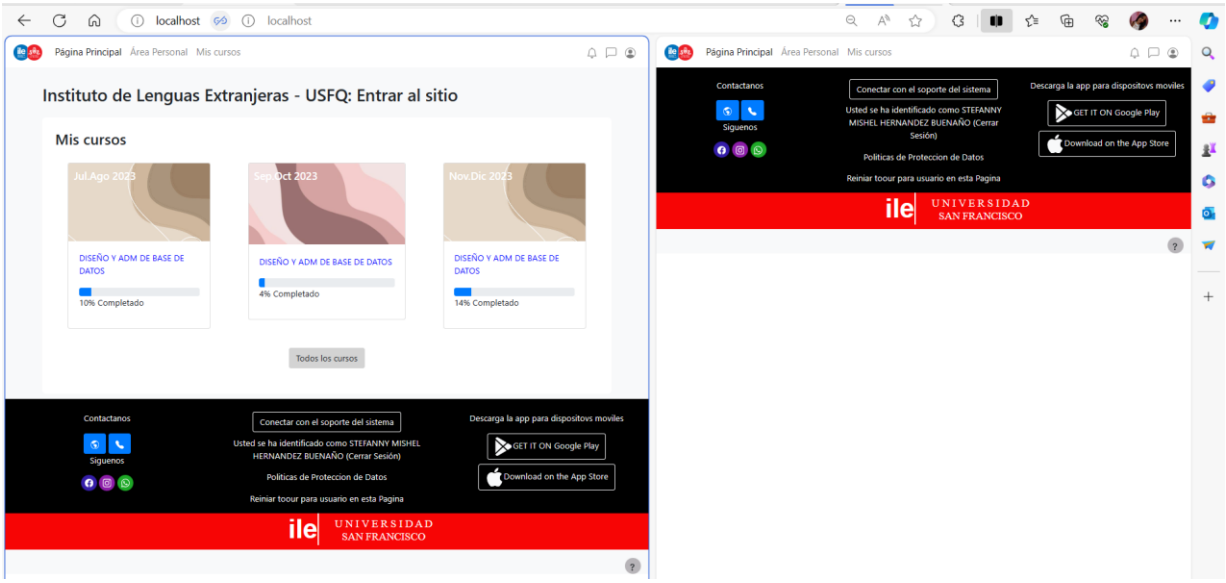
Actualiza tu código HTML para incluir instancias tanto de los componentes **x-menu** como de **x-menu2 y x-menu3**.

Lab2 > Index.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7
8      <title>customElements</title>
9
10     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
11     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"
12         integrity="sha512-2eeGf6tjftPZBA9Q3HLxDn6Lq6MOdCOHeP8PVOVpDQ+2bcO+Wynpwh7l2zvAhYT5Qq+GOlA4ZMjQxSbRsI1FSQ=="
13         crossorigin="anonymous" referrerpolicy="no-referrer" />
14
15     <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
16     <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"></script>
17     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
18 </head>
19
20 <body>
21     <script src="components.js"></script>
22     <x-menu></x-menu>
23     <x-menu2></x-menu2>
24     <x-menu3></x-menu3>
25 </body>
26
27 </html>
```

El resultado de los dos archivos HTML dependerá de cómo estén definidos los componentes **x-menu**, **x-menu2** y **x-menu3** en tu archivo **components.js**.

El resultado visual dependerá de la implementación específica de cada componente y de cómo interactúan con el contenido de la página. Además, ten en cuenta que si hay algún error en las definiciones de los componentes o en la carga de scripts, podrías experimentar problemas en la renderización de la página.




**PASO 4. Establecer cambios a través de parámetros en las etiquetas.**

Para permitir que los textos de los elementos **h2** y **h3** se cambien a través de parámetros, puedes hacer lo siguiente:

- Agregar atributos personalizados para los textos que deseas cambiar en **h2** y **h3**.

```
97          <div class="main bg-light">
98          <h2 class="mt-4 mb-4" id="mainTitle"></h2>
99          <div class="container2 mt-2" style="background-color: white; border-radius: 6px;">
100             <div class="row">
101                 <h3 class="col-12 ml-4 mt-4 mb-2" id="subTitle"></h3>
```

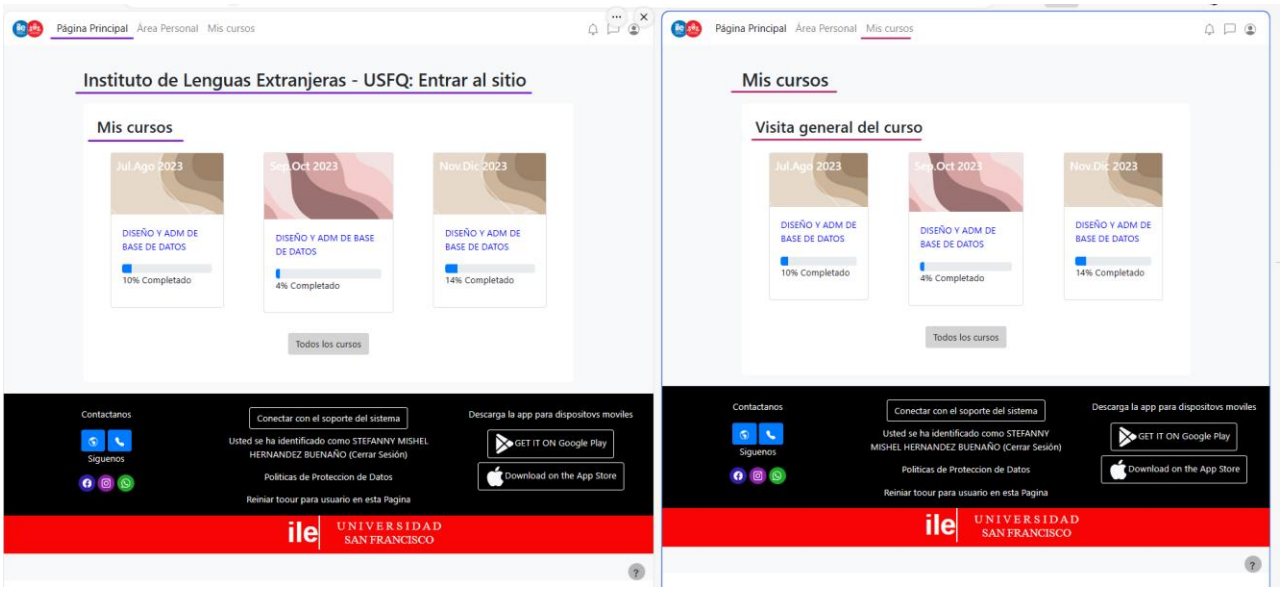
- Utilizar los valores de esos atributos en el contenido del **template**.

```
70  class MenuElement2 extends HTMLElement {
71      constructor() {
72          super();
73          this.template = document.createElement("template");
74  >       this.template.innerHTML = `···
163         `;
164         // Añadir atributos observados
165         this.mainTitle = this.getAttribute('main-title') || 'Instituto de Lenguas Extranjeras - USFQ: Entrar al sitio';
166         this.subTitle = this.getAttribute('sub-title') || 'Mis cursos';
167     }
168     connectedCallback() {
169
170         let cloneDOM = document.importNode(this.template.content, true);
171         cloneDOM.getElementById('mainTitle').innerText = this.mainTitle;
172         cloneDOM.getElementById('subTitle').innerText = this.subTitle;
173         this.appendChild(cloneDOM);
174     }
175
176     attributeChangedCallback(attr, oldval, newval) {
177         // Manejar cambios en los atributos observados si es necesario
178         if (attr === 'main-title') {
179             this.mainTitle = newval;
180         } else if (attr === 'sub-title') {
181             this.subTitle = newval;
182         }
183     }
184
185     static get observedAttributes() {
186         return ['main-title', 'sub-title'];
187     }
188 }
189 window.customElements.define("x-menu2", MenuElement2);
```

Con estos cambios, ahora puedes establecer los textos de **h2** y **h3** utilizando los atributos **main-title** y **sub-title** respectivamente cuando usas el componente en tu HTML:

```
<body>
    <script src="components.js"></script>
    <x-menu></x-menu>
    <x-menu2 main-title="Mis cursos" sub-title="Visita general del curso"></x-menu2>
    <x-menu3></x-menu3>
</body>
```

De esta manera, puedes cambiar el texto del **h2** y **h3** al proporcionar valores diferentes para los atributos **main-title** y **sub-title** en la instancia del componente **<x-menu2>**. Asegúrate de que estos atributos coincidan con los que has definido como observados en tu componente personalizado.




**PASO 5. Aplicar variaciones para ir integrando en cada ejemplo, cada uno de los estándares aplicables a los "web components".**

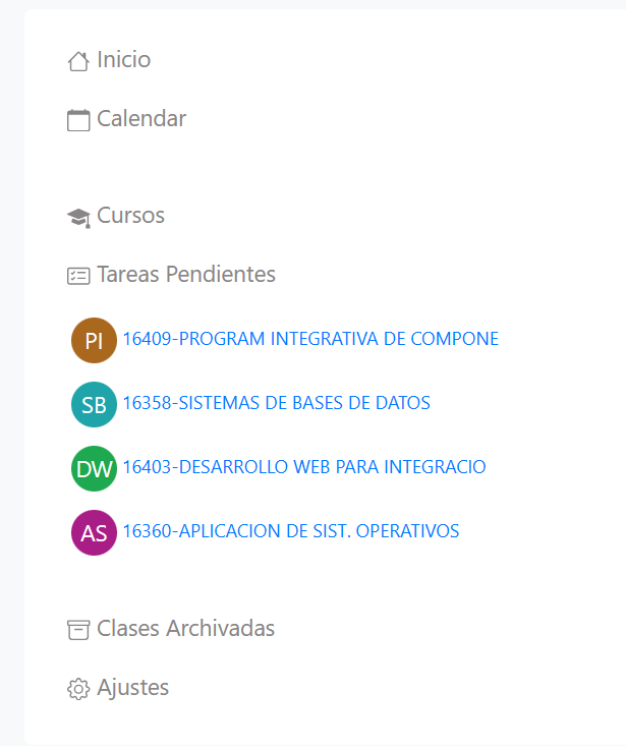
```
class MenuElement4 extends HTMLElement {
    constructor() {
        super();
        this.template = document.createElement("template");
        this.template.innerHTML = `
        <style>

        :host {
            display: block;
            padding-top: 56px;
            padding: 10px 70px 10px 70px;
        }
```

**PASO 6. Demostrar con la visualización lo realizado**

Este código define una clase llamada **MenuElement4** que extiende la clase **HTMLElement**, creando así un nuevo elemento personalizado llamado "x-menu4". Este elemento encapsula una interfaz de usuario para un menú, con dos columnas divididas en las secciones izquierda y derecha.

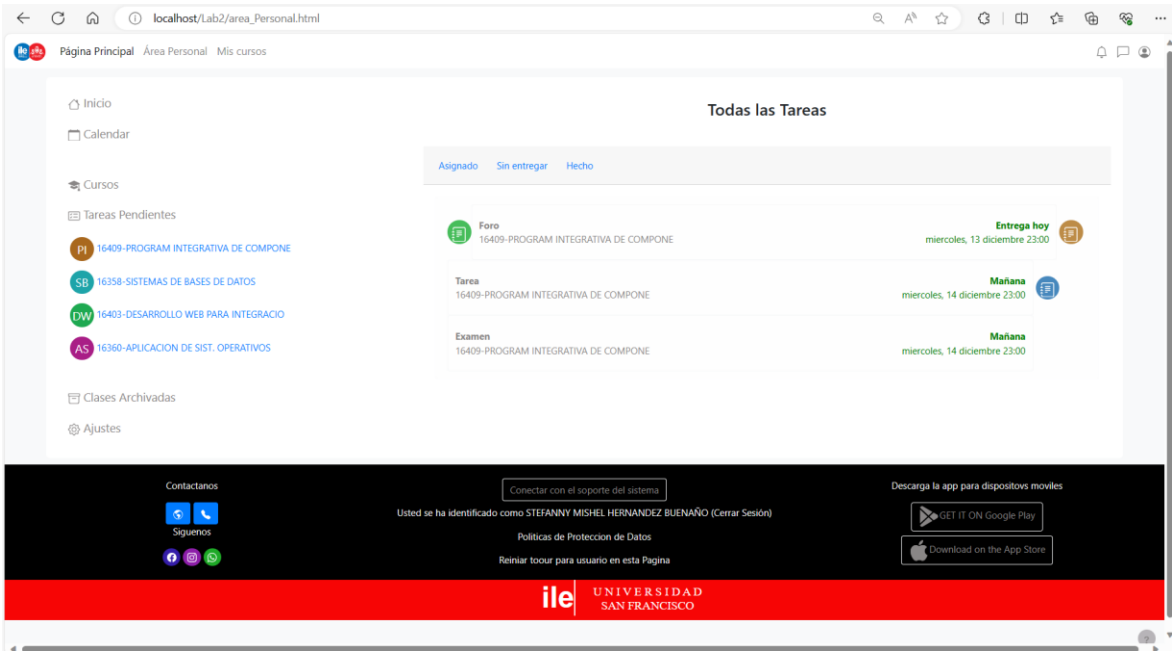
```
323  class MenuElement4 extends HTMLElement {
324      constructor() {
325          super();
326          this.template = document.createElement("template");
327  >       this.template.innerHTML = `···
534          `;
535      }
536      connectedCallback() {
537
538          let cloneDOM = document.importNode(this.template.content, true);
539          this.appendChild(cloneDOM);
540      }
541
542      attributeChangedCallback(attr, oldval, newval) {
543
544      }
545
546      static get obseredAttribute() {
547          return [''];
548      }
549  }
550  window.customElements.define("x-menu4", MenuElement4);
```

En la sección izquierda, se encuentran botones de navegación para distintas funcionalidades, como "Inicio", "Calendario", "Cursos", y "Tareas Pendientes". Además, se presenta una lista de cursos y sus respectivos códigos, cada uno representado por un botón circular de colores distintos.



En la sección derecha, se muestra un conjunto de tarjetas que representan distintas categorías de tareas, como "Asignado", "Sin entregar", y "Hecho". Cada tarjeta contiene elementos visuales como botones circulares con íconos y etiquetas para identificar y describir tareas específicas. Estas tareas incluyen elementos como foros, tareas y exámenes, con detalles como el nombre de la tarea, el curso asociado y la fecha límite de entrega.


Resultado visual completa, el código utiliza HTML y CSS para la estructura y estilo, respectivamente, y hace uso de las capacidades de Web Components para encapsular y reutilizar la funcionalidad del menú en otras partes de una aplicación web. Además, el código define un método **connectedCallback** que se llama cuando el elemento personalizado se agrega al DOM, permitiendo que el contenido del menú se renderice y sea visible en la interfaz de usuario.



## 4. Conclusiones

- La implementación de Custom Elements, Shadow DOM y HTML Templates ofrece una solución poderosa para la creación de componentes web reutilizables y bien encapsulados, lo que conduce a un código más limpio, mantenible y escalable.
- La adopción de estas tecnologías no solo mejora la eficiencia del desarrollo, sino que también contribuye a una experiencia de usuario más consistente al prevenir la interferencia no deseada entre componentes y estilos.

## 5. Recomendaciones

- Se recomienda que los desarrolladores web adquieran un conocimiento profundo de las mejores prácticas asociadas con Custom Elements, Shadow DOM y HTML Templates para aprovechar al máximo estas tecnologías y evitar posibles problemas de rendimiento y seguridad.
- La implementación de herramientas de construcción y marcos de trabajo que admitan nativamente Custom Elements, Shadow DOM y HTML Templates puede agilizar el proceso de desarrollo y garantizar la coherencia en la aplicación de estas tecnologías en proyectos más grandes.