ESPE UNIVERSIDAD DE LAS FUERZAS ARMADAS INNOVACIÓN PARA LA EXCELENCIA	INFORME DE PRÁCTICA DE LABORATORIO

DEPARTAMENTO:	**Ciencias de la computación**	**CARRERA:**	**Tecnologías de la información (TICS)**		
ASIGNATURA:	Programación integrativa de Componentes Web	**NIVEL:**	Sexto	**FECHA:**	11/05/2025
DOCENTE:	Ing. Paulo Cesar Galarza Sánchez	**TAREA N°:**	1-U1	**CALIFICACIÓN:**	

Creación de un Componente Personalizado con Sintaxis y Funcionalidad Básica

Giovanny Francisco Durán Sánchez

1. Introducción

En el presente informe se desarrolla un componente personalizado utilizando la tecnología Web Components haciendo uso de Custom Elements, Shadow DOM y slots, se trata de encapsular la estructura, estilo y la funcionalidad de un componente HTML reutilizable que represente una tarjeta informativa del usuario esto promueve la reutilización de código.

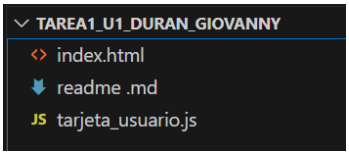
2. Objetivos

- Diseñar e implementar un componente web personalizado <tarjeta-usuario>.
- Aplicar los conceptos de Shadow DOM para encapsular el estilo y estructura del componente.

3. Desarrollo

3.1 Directorio

Ilustración 1 Directorio del Proyecto



3.2 Index

Ilustración 2 Desarrollo del Código html.

```
index.html > html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Tarjeta Personal</title>
  <script type="module" src="./tarjeta_usuario.js"></script>
  <style>
    body {
      background-color: #badaf9;
      font-family: Arial, sans-serif;
      padding: 2rem;
    }
  </style>
</head>
<body>

  <!-- Uso del componente con slots -->
  <tarjeta-usuario tema="#ffffff">
    <h2 slot="titulo">Giovanny Durán</h2>
    <p slot="contenido">
      Estudiante Militar de planta en la Universidad de Fuerzas Armadas ESPE sede Santo Domingo,
      de la carrera Ingeniería en Tecnologías de la Información, cursando el 6to nivel.
    </p>
  </tarjeta-usuario>
```

3.3 **JavaScript**

Ilustración 3 Código JS

```
tarjeta_usuario.js > ...
// Definimos un nuevo componente personalizado llamado <tarjeta-usuario>
class TarjetaUsuario extends HTMLElement {
  constructor() {
    super();

    // Activamos Shadow DOM para encapsular estilos y estructura
    this.attachShadow({ mode: 'open' });

    // Insertamos el contenido del componente en el Shadow DOM
    this.shadowRoot.appendChild(this.plantilla().content.cloneNode(true));
  }

  // Escuchamos cambios en el atributo "tema" para cambiar el color de fondo
  static get observedAttributes() {
    return ['tema'];
  }

  attributeChangedCallback(nombre, valorAntiguo, nuevoValor) {
    if (nombre === 'tema') {
      this.shadowRoot.querySelector('.tarjeta').style.setProperty('--color-fondo', nuevoValor);
    }
  }

  // Creamos la plantilla del componente con estilos y slots
  plantilla() {
    const plantilla = document.createElement('template');
    plantilla.innerHTML = `
```

```
      .descripcion {
        font-size: 1em;
        color: #555;
        line-height: 1.6;
      }
    </style>

    <section class="tarjeta">
      <!-- Slot para el nombre o título -->
      <div class="encabezado">
        <slot name="titulo">[Nombre por defecto]</slot>
      </div>

      <!-- Slot para la descripción o contenido -->
      <div class="descripcion">
        <slot name="contenido">[Descripción por defecto]</slot>
      </div>
    </section>
  `;
  return plantilla;
  }
}

// Registramos el componente para que pueda usarse como <tarjeta-usuario>
customElements.define('tarjeta-usuario', TarjetaUsuario);
```

3.4 Ejecución

Ilustración 4 Ejecución de la pagina

4. Preguntas

4.1 ¿Cómo se logra la encapsulación con Shadow DOM?

La encapsulación con Shadow DOM se logra al crear un árbol DOM separado e independiente del documento principal usando el método attachShadow({ mode: 'open' }) y todo el contenido HTML y CSS insertado dentro del Shadow DOM queda aislado, lo que significa que:

- Los estilos del componente no afectan al resto de la página.
- Los estilos externos tampoco afectan al interior del componente. Esto permite construir elementos reutilizables con estructura y estilos propios sin riesgo de colisión o conflictos con otros elementos del sitio.

4.2 ¿Diferencias entre slots con nombre y slots por defecto?

- Slot por defecto (<slot>): inserta cualquier contenido que no tenga un atributo slot. Solo se puede usar una vez por componente.
- Slots con nombre (<slot name="...">): permiten definir varias posiciones dinámicas dentro del componente, identificando el contenido por el atributo slot="nombre" desde el HTML.

4.3 ¿Limitaciones de los Web Components en aplicaciones reales?

- No todos los frameworks como React, Vue o Angular los integran de forma nativa y pueden requerir adaptaciones.
- A diferencia de librerías como React o Vue, el ecosistema de herramientas, testing y documentación es más limitado.
- Renderizar Web Components del lado del servidor (Server Side Rendering) puede ser más complejo.
- Crear demasiados componentes mal estructurados puede impactar el rendimiento, especialmente si contienen Shadow DOM complejos.

5. Conclusiones

- El desarrollo del componente <tarjeta-usuario> permitió aplicar de forma práctica los fundamentos de los Web Components, logrando encapsular estilos y estructura de manera efectiva mediante el uso del Shadow DOM.
- La incorporación de slots con nombre facilita la reutilización del componente con diferentes contenidos, sin alterar su estructura interna y además la implementación del atributo tema demuestra cómo es posible personalizar el comportamiento visual desde el HTML externo.

6. Recomendaciones

- Se recomienda continuar aprendiendo el potencial de los Web Components en proyectos y especialmente en aplicaciones donde la reutilización y el aislamiento de estilos son fundamentales.
- También es recomendable complementar los componentes con pruebas unitarias y documentaciones más detalladas también considerar su integración en frameworks para facilitar tareas más complejas.

7. Referencias

https://github.com/GiovannyGuso/Tarea1DuranGiovannyWebComps.git