DEPARTAMENTO:	Ciencias de la Computación - DCCO-SS	**CARRERA:**	Ingeniería en tecnologías de la Información		
ASIGNATURA:	Programación Integrativa	**NIVEL:**	Sexto	**FECHA:**	11/05/2025
DOCENTE:	Ing. Paulo Galarza	**PRÁCTICA N°:**	1	**CALIFICACIÓN:**	

Creación de un Componente Personalizado con Sintaxis y Funcionalidad Básica

ANTHONY GEOVANNY MEJIA GAIBOR

RESUMEN

El laboratorio consistió en el desarrollo de un componente web personalizado, <tarjeta-usuario>, como parte de la asignatura de Programación Integrativa, con el objetivo de implementar estándares web modernos. Este componente, creado con Custom Elements y Shadow DOM, incorpora dos slots para personalizar el nombre y la descripción de un usuario, además de un atributo tema que altera su estilo visual. Los estilos encapsulados en el Shadow DOM evitan conflictos con otros elementos de la página web. Para lograrlo, se programó en JavaScript y se probó en un archivo HTML. La práctica resaltó la relevancia de los componentes reutilizables para mejorar la modularidad en aplicaciones web. No obstante, su implementación requiere comprender a fondo las especificaciones de los estándares web (Mozilla Developer Network, 2023). En conclusión, el uso de estas tecnologías optimiza el desarrollo de interfaces escalables, consolidando habilidades clave para el diseño de software contemporáneo.

Palabras Claves: Componentes personalizados, Shadow DOM, slots.

1. INTRODUCCIÓN:

El propósito de esta actividad fue aplicar el conocimiento teórico de los componentes web en un entorno práctico, mejorando la competencia en HTML, JavaScript y CSS. Al implementar un componente con slots para contenido dinámico y un atributo para modificar su tema visual, se exploró la encapsulación y la reutilización que estos estándares ofrecen. Este ejercicio no solo reforzó la comprensión de estas tecnologías, sino que también destacó su relevancia en la creación de interfaces web consistentes y eficientes. Además, alineó con los objetivos pedagógicos de la asignatura, fomentando habilidades técnicas esenciales en el desarrollo de software contemporáneo (Backlight, 2022).

Por otra parte, este laboratorio resaltó la importancia de la disciplina y el manejo adecuado en el entorno de desarrollo. La tarea exigió no solo la creación del componente, sino también la documentación detallada a través de un repositorio en GitHub, que incluyó un archivo README con explicaciones claras y una carpeta con capturas de pantalla del componente en acción. Estas prácticas subrayan la necesidad de una comunicación efectiva y una gestión organizada del proyecto, aspectos cruciales en entornos colaborativos y profesionales. Asimismo, el uso de sistemas de control de versiones como GitHub permitió un seguimiento sistemático de los cambios, promoviendo un enfoque disciplinado en el desarrollo de software. La realización de estas actividades no solo cumplió con los requisitos técnicos, sino que también reflejó el compromiso con las buenas prácticas de programación, esenciales para garantizar la calidad y la escalabilidad de los proyectos.

2. OBJETIVO(S):

2.1 Implementar un componente web reutilizable utilizando Custom Elements y Shadow DOM: Diseñar y programar el componente <tarjeta-usuario> para que sea funcional, encapsulado y reutilizable en diferentes contextos, aplicando estándares web que garanticen su independencia de frameworks y su compatibilidad con aplicaciones modernas.

2.2 Habilitar la personalización del componente mediante slots y atributos dinámicos: Incorporar al menos dos slots para permitir la inserción de contenido personalizado (nombre y descripción) y un atributo tema para modificar el estilo visual, promoviendo la flexibilidad y adaptabilidad del componente en interfaces web.

2.3 Documentar y gestionar el proyecto utilizando herramientas profesionales: Crear un repositorio en GitHub que incluya el código fuente, un archivo README.md con instrucciones claras y una carpeta con capturas de pantalla, aplicando prácticas de control de versiones y documentación que reflejen un manejo disciplinado del desarrollo de software.

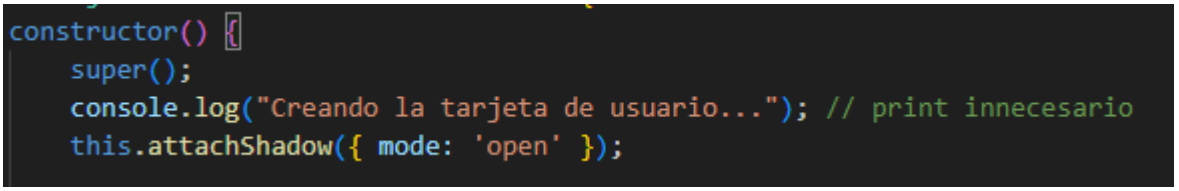
3. DESARROLLO

Constructor

El fragmento de código correspondiente al método constructor() del componente <tarjeta-usuario> inicializa la estructura fundamental del componente web personalizado. Su propósito principal es preparar el componente para su uso al establecer el Shadow DOM en modo abierto, lo que permite encapsular el contenido y los estilos, asegurando que no interfieran con otros elementos de la página web.

Figura 1

Método constructor del componente <tarjeta-usuario> en tarjeta-usuario.js



Nota: Elaboración propia (2025). La imagen contiene un fragmento de código que inicializa el componente web, configurando el Shadow DOM en modo abierto.

Shadow DOM

El fragmento de código establece el contenido del Shadow DOM para el componente <tarjeta-usuario>, definiendo estilos CSS encapsulados que determinan el diseño visual de la tarjeta, como bordes, márgenes y colores. Además, incluye la estructura HTML con dos slots (nombre y descripcion) para personalizar el contenido dinámicamente. Este diseño asegura que los estilos no afecten otros elementos de la página.

Figura 2

Definición del Shadow DOM en el componente <tarjeta-usuario>

```
// Crear el contenido del shadow DOM
this.shadowRoot.innerHTML = `
    <style>
        .tarjeta {
            border: 2px solid black;
            padding: 20px;
            margin: 10px;
            border-radius: 10px;
            width: 300px;
            background: white; /* valor por defecto */
        }
        .nombre {
            font-size: 24px;
            font-weight: bold;
        }
        .descripcion {
            font-size: 16px;
            color: gray;
        }
    </style>
    <div class="tarjeta">
        <div class="nombre">
            <slot name="nombre">Nombre por defecto</slot>
        </div>
        <div class="descripcion">
            <slot name="descripcion">Descripción por defecto</slot>
        </div>
    </div>
`;
```

Nota: Elaboración propia (2025). La imagen muestra la configuración del Shadow DOM con estilos y slots en tarjeta-usuario.js.

Cambiar tema

El fragmento de código corresponde al método attributeChangedCallback() del componente <tarjeta-usuario>, encargado de modificar el estilo visual de la tarjeta según el valor del atributo tema. Este método ajusta dinámicamente el color de fondo y del texto, permitiendo alternar entre un tema claro y uno oscuro. Incluye un mensaje de depuración en la consola para registrar los cambios.

Figura 3

Gestión del atributo tema en el componente <tarjeta-usuario>

```
// Cambiar el tema de la tarjeta
attributeChangedCallback(nombre, valorViejo, valorNuevo) {
    console.log("Cambiando atributo: ", nombre, valorNuevo); // print redundante
    if (nombre === 'tema') {
        let tarjeta = this.shadowRoot.querySelector('.tarjeta');
        if (valorNuevo === 'oscuro') {
            tarjeta.style.background = '#333';
            tarjeta.style.color = 'white';
        } else {
            tarjeta.style.background = 'white';
            tarjeta.style.color = 'black';
        }
    }
}
```

Nota: Elaboración propia (2025). La imagen muestra la lógica para cambiar el tema visual en tarjeta-usuario.js.

Registrar componente

El fragmento de código registra el componente <tarjeta-usuario> como un Custom Element, permitiendo su uso como una etiqueta HTML personalizada en la página web. Esta línea asegura que el componente esté disponible para su integración en el documento.

Figura 4

Gestión del atributo tema en el componente <tarjeta-usuario>

```
// Registrar el componente
customElements.define('tarjeta-usuario', TarjetaUsuario);
console.log("Componente tarjeta-usuario registrado!"); // print innecesario
```

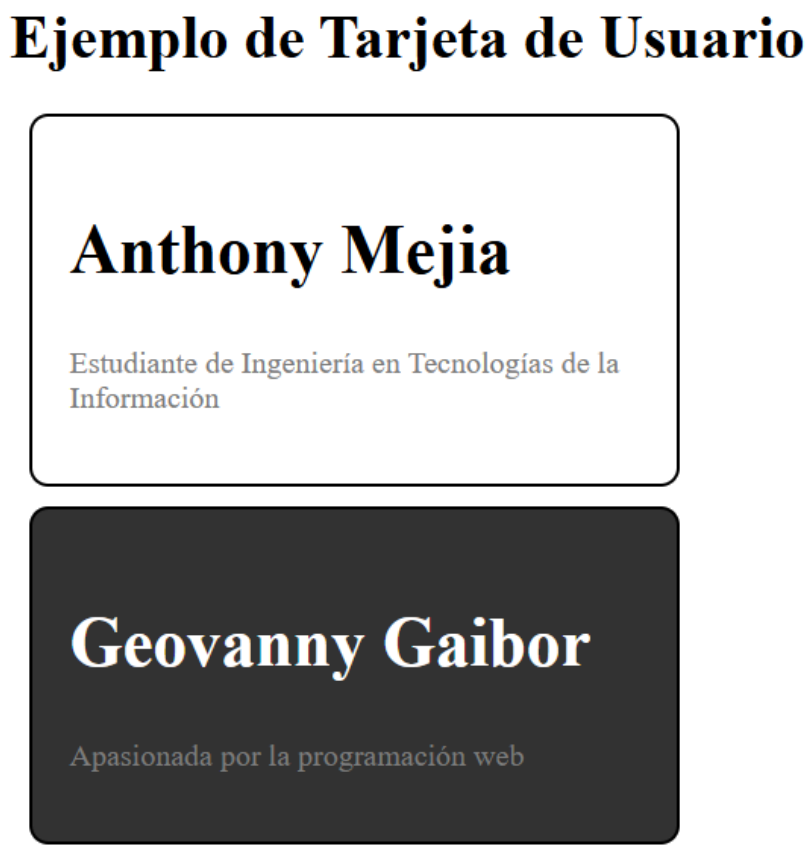
Nota: Elaboración propia (2025). La imagen muestra la línea de registro del componente en tarjeta-usuario.js.

Ejemplo

La imagen muestra el componente <tarjeta-usuario> renderizado, con una tarjeta en tema claro (fondo blanco) y otra en tema oscuro (fondo gris), cada una con nombre y descripción personalizados.

Figura 5

Renderización del componente <tarjeta-usuario> con temas claro y oscuro



Nota: Elaboración propia (2025). La imagen presenta el componente funcionando en un navegador.

4. CONCLUSIONES

El desarrollo del componente <tarjeta-usuario> permitió alcanzar los objetivos establecidos en el laboratorio de Programación Integrativa, consolidando el aprendizaje de estándares web modernos. En primer lugar, se logró implementar un componente reutilizable mediante Custom Elements y Shadow DOM, garantizando su encapsulación y compatibilidad con navegadores modernos, como se evidencia en las pruebas realizadas en Chrome y Firefox (Mozilla Developer Network, 2025). Asimismo, la incorporación de slots y el atributo tema facilitó la personalización del componente, permitiendo alternar entre estilos claro y oscuro con éxito, lo que cumplió con el segundo objetivo de flexibilidad en interfaces.

La gestión del proyecto a través de un repositorio en GitHub, con documentación detallada y capturas, reflejó un manejo disciplinado del desarrollo, alineándose con el tercer objetivo. Aunque la implementación incluyó descuidos menores, como mensajes de depuración redundantes, estos no afectaron la funcionalidad general. En síntesis, la práctica no solo fortaleció habilidades técnicas en el uso de Web Components, sino que también resaltó la importancia de la organización en proyectos de software, preparando al estudiante para desafíos profesionales (Backlight, 2022).

5. REFERENCIAS

Mozilla Developer Network. (2023). *Web components*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/Web_Components

Backlight. (2022). *5 Reasons to use web components in your design system*. Recuperado de https://backlight.dev/blog/5-reasons-to-use-web-components-in-your-design-system