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

DEPARTAMENTO:	**Ciencias de la Computación**	**CARRERA:**	**Ingeniería en Tecnologías de la Información**		
ASIGNATURA:	PROGRAMACION INTEGRATIVA DE COMPONENTES	**NIVEL:**	Sexto	**FECHA:**	23/05/2025
DOCENTE:	Ing. Paulo Galarza	**PRÁCTICA N°:**	1	**CALIFICACIÓN:**	

Desarrollar una Tarjeta de Producto para un E-commerce con Composición de Componentes

Toapanta Paez Darwin Andres

RESUMEN

En esta práctica se desarrolló un componente web personalizado utilizando HTML, CSS y JavaScript, con el objetivo de crear tarjetas de producto reutilizables para una página de compras. Se empleó Shadow DOM para encapsular estilos y comportamiento, y se implementaron eventos personalizados para simular la funcionalidad de un carrito de compras. Además, se utilizó una plantilla HTML y estilos globales para organizar la estructura del sitio. La práctica permitió aplicar conceptos de encapsulamiento, eventos y diseño responsivo. Como resultado, se obtuvo una página web funcional con tarjetas interactivas que responden a las acciones del usuario y actualizan dinámicamente la cantidad de productos en el carrito

Palabras Claves: Web Components, Shadow DOM, Tarjeta de producto.

1. INTRODUCCIÓN:

El desarrollo de componentes reutilizables en la web moderna mejora la escalabilidad, mantenibilidad y eficiencia del código. En esta práctica se construyó una tarjeta de producto mediante la API de Web Components, integrando estilos internos, contenido dinámico y comunicación mediante eventos. El trabajo se enfocó en la organización del código, el respeto por los estándares de desarrollo y el uso disciplinado de las herramientas de desarrollo web.

2. OBJETIVO(S):

2.1 Crear un Custom Element para una tarjeta de producto.

2.2 Utilizar Shadow DOM para encapsular estilos y estructura.

2.3 Implementar composición mediante <slot> y <template>.

2.4 Integrar datos dinámicos mediante propiedades y atributos.

3. MARCO TEÓRICO:

Paso 1: Configuración Inicial

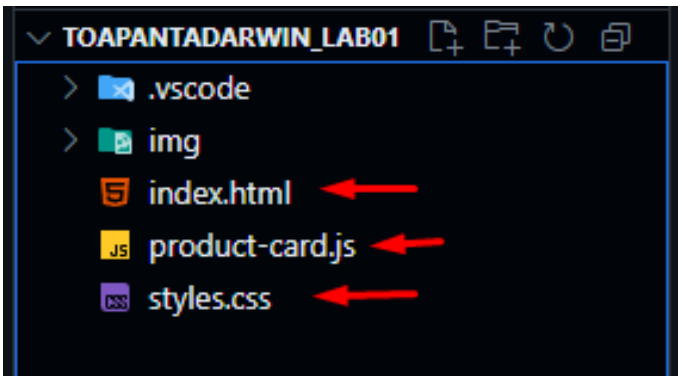
1. Crea una carpeta para el proyecto con la estructura:

product-card/

├── index.html

├── product-card.js

└── styles.css



2. En index.html, define un <template> para la estructura base de la tarjeta:

<template id="product-card-template">

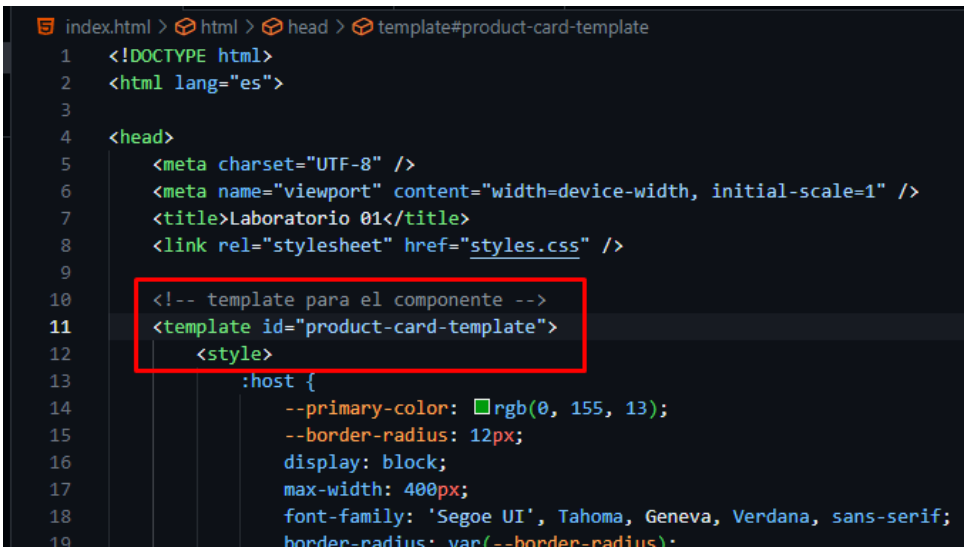
<style>

/* Estilos encapsulados con Shadow DOM */

:host { display: block; font-family: Arial; }

.card { border: 1px solid #ccc; padding: 20px; border-radius: 8px; }

</style>



<slot name="image">Imagen no disponible</slot>

<h3><slot name="title">Título del producto</slot></h3>

<slot name="price">Precio no disponible</slot>

</template>

```
        <div class="container">
            <img part="image" src="" alt="Imagen del producto" />
            <h2 part="title">
                <slot name="title">Producto</slot>
            </h2>
            <p part="price">
                <slot name="price">Precio no disponible</slot>
            </p>
            <p part="description">
                <slot name="description">Descripción del producto</slot>
            </p>
            <p part="collection">
                <slot name="collection"></slot>
            </p>
            <button part="button">Añadir al carrito</button>
        </div>
    </template>
```

Paso 2: Crear el Custom Element

1. En product-card.js, define la clase del componente:

class ProductCard extends HTMLElement {

constructor() {

super();

const template = document.getElementById('product-card-template').content;

this.attachShadow({ mode: 'open' }).appendChild(template.cloneNode(true));

}

}

customElements.define('product-card', ProductCard);

```
class ProductCard extends HTMLElement {
    constructor() {
        super();
        this.attachShadow({ mode: "open" });

        const template = document.getElementById("product-card-template");
        this.shadowRoot.appendChild(template.content.cloneNode(true));

        // agrego estilos al shadow DOM
        const style = document.createElement("style");
        style.textContent = this.getStyles();
        this.shadowRoot.appendChild(style);
    }
```

```
131
132   customElements.define("product-card", ProductCard);
133
```

2. En index.html, usa el componente:

```
<product-card>
 <img slot="image" src="producto.jpg" alt="Producto">
 <span slot="title">Camiseta Moderna</span>
 <span slot="price">$25.99</span>
</product-card>
```

```
<product-card img="./img/nike-verde.png" title="Zapato Nike Verde" price="$100.00"
    description="Zapato Nike de alta calidad" collection="Colección Running"
    style="--primary-color: rgb(0, 155, 13); --border-radius: 16px;"></product-card>
```

Paso 3: Añadir Datos Dinámicos

1. Modifica ProductCard para aceptar atributos:

```
class ProductCard extends HTMLElement {
 static get observedAttributes() { return ['title', 'price', 'image']; }
 attributeChangedCallback(name, oldValue, newValue) {
 if (this.shadowRoot) {
 this.shadowRoot.querySelector(`[slot="${name}"]`).textContent = newValue;
 }
 }
}
```

```
15      static get observedAttributes() {
16          return ["img", "title", "price", "description", "collection"];
17      }
18
19      attributeChangedCallback(name, oldValue, newValue) {
20          if (!this.shadowRoot) return;
21
22          switch (name) {
23              case "img":
24                  const imgElem = this.shadowRoot.querySelector("img");
25                  if (imgElem) {
26                      imgElem.src = newValue;
27                      imgElem.alt = this.getAttribute("title") || "Imagen del producto";
28                  }
29                  break;
30
31              case "title":
32                  this._updateSlotContent("title", newValue);
33                  const img = this.shadowRoot.querySelector("img");
34                  if (img) img.alt = newValue;
35                  break;
36
37              case "price":
38                  this._updateSlotContent("price", newValue);
39                  break;
40
41              case "description":
42                  this._updateSlotContent("description", newValue);
43                  break;
```

2. Actualiza el HTML para usar atributos:

```
<product-card title="Zapatos Deportivos" price="$89.99" image="zapatos.jpg"></product-card>
```

```
<main>
    <product-card img="./img/nike-verde.png" title="Zapato Nike Verde" price="$100.00"
        description="Zapato Nike de alta calidad" collection="Colección Running"
        style="--primary-color: rgb(0, 155, 13); --border-radius: 16px;"></product-card>

    <product-card img="./img/nike-negro.png" title="Zapato Nike Negro" price="$95.00"
        description="Zapato Nike elegante en color negro" collection="Colección Urbana"
        style="--primary-color: rgb(0, 0, 0); --border-radius: 16px;"></product-card>

    <product-card img="./img/nike-azul.png" title="Zapato Nike Azul" price="$110.00"
        description="Zapato Nike deportivo en color azul" collection="Colección Running"
        style="--primary-color: rgb(0, 0, 255); --border-radius: 16px;"></product-card>
</main>
```

Paso 4: Estilos Personalizados con CSS Variables

1. En el <style> del template, define variables CSS:

```
:host {
 --primary-color: #007bff;
 --border-radius: 8px;
}
.card {
 border-radius: var(--border-radius);
 background: var(--primary-color, #007bff);
}
```

```
11      <template id="product-card-template">
12          <style>
13              :host {
14                  --primary-color: rgb(0, 155, 13);
15                  --border-radius: 12px;
16                  display: block;
17                  max-width: 400px;
18                  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
19                  border-radius: var(--border-radius);
20                  border: 1px solid var(--primary-color);
21                  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
22                  overflow: hidden;
23                  background-color: white;
24              }
25
26              .container {
27                  display: flex;
28                  flex-direction: column;
29                  align-items: center;
30                  padding: 16px;
31              }
```

SE DEFINE LOS ESTILOS DENTRO DE LA ETIQUETA TEMPLATE

2. Personaliza el componente desde el HTML:

```
<product-card style="--primary-color: #ff5722;"></product-card>
```

```
<main>
    <product-card img="./img/nike-verde.png" title="Zapato Nike Verde" price="$100.00"
        description="Zapato Nike de alta calidad" collection="Colección Running"
        style="--primary-color: rgb(0, 155, 13); --border-radius: 16px;"></product-card>

    <product-card img="./img/nike-negro.png" title="Zapato Nike Negro" price="$95.00"
        description="Zapato Nike elegante en color negro" collection="Colección Urbana"
        style="--primary-color: rgb(0, 0, 0); --border-radius: 16px;"></product-card>

    <product-card img="./img/nike-azul.png" title="Zapato Nike Azul" price="$110.00"
        description="Zapato Nike deportivo en color azul" collection="Colección Running"
        style="--primary-color: rgb(0, 0, 255); --border-radius: 16px;"></product-card>
</main>
```

Personalizo cada uno de los componentes desde el HTML

Procedemos a subir el proyceto desde git

4. CONCLUSIONES:

Se cumplió con éxito el objetivo de crear un Web Component reutilizable e interactivo, además el uso de Shadow DOM permitió separar responsabilidades y evitar conflictos de estilos.

La implementación de eventos personalizados hizo posible comunicar acciones del componente con el resto de la página de forma limpia.

Esta práctica refuerza la importancia de aplicar principios modernos de desarrollo web como la modularidad, encapsulamiento y reutilización.

5. BIBLIOGRAFÍA:

MDN Web Docs. (2023). Using custom elements. Mozilla. https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_custom_elements

Google Developers. (2022). Introduction to Web Components. https://developers.google.com/web/fundamentals/web-components

W3C. (2023). Shadow DOM v1. https://www.w3.org/TR/shadow-dom/ (Consulta realizada en mayo de 2025)