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:	Programación Integrativa de Componentes	**NIVEL:**	6	**FECHA:**	26/05/2025
DOCENTE:	Ing. Paulo Galarza Mgs.	**PRÁCTICA N°:**	1	**CALIFICACIÓN:**	

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

Jordan Alexander Guevara Chalial

RESUMEN

Durante este laboratorio se desarrolló un Web Component personalizado que representa visualmente un producto, utilizando una plantilla fotográfica que simula una tarjeta de presentación como las que se encuentran en tiendas online. Se aplicó el uso de attachShadow() para encapsular los estilos y estructuras del componente dentro del Shadow DOM, garantizando su independencia del resto de la interfaz. Se implementan funcionalidades interactivas como el cambio de imagen según el color seleccionado, el ajuste de cantidad con botones de incremento y decremento, y una valoración por estrellas mediante clic. Esto permitió demostrar cómo se puede lograr un diseño modular, reutilizable y visualmente atractivo, cumpliendo con los principios de la programación moderna en JavaScript.

Palabras Claves: Plantilla, attachShadow, DOM

1. INTRODUCCIÓN:

En el presente laboratorio se demuestra la creación de un Web Component replicando el cómo se presenta un producto en una página web, viendo el uso de los colores, el cómo se puede implementar con el Shadow DOM y el poder reutilizar el código sin la necesidad de crear otro código.

2. OBJETIVO(S):

2.1 Recrear una plantilla de un producto.
2.2 Implementar los conocimientos sobre el uso de un web component.
2.3 Demostrar el conocimiento visto en clase con el uso de las funcionalidades del componente.

3. MARCO TEÓRICO:

Se fundamenta en los principios del desarrollo web moderno, específicamente en el uso de Web Components para la creación de interfaces modulares, reutilizables y encapsuladas.

Un Web Component es una tecnología estandarizada por el W3C que permite definir nuevos elementos HTML personalizados, con su propia estructura, estilo y comportamiento. Este enfoque mejora la mantenibilidad del código y promueve la reutilización de componentes en diferentes partes de una aplicación o entre proyectos distintos.

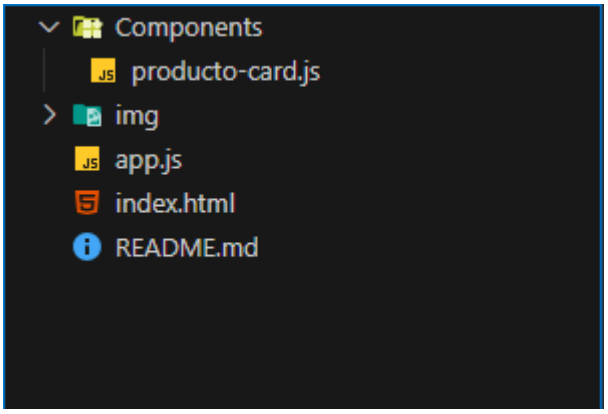
El uso de Shadow DOM es clave dentro de los Web Components, ya que permite encapsular los estilos y el contenido del componente, evitando conflictos con otros elementos del DOM principal. Asimismo, el uso de templates facilita la separación entre estructura y lógica de programación. Además, se emplean técnicas de manipulación del DOM, eventos personalizados y estilos dinámicos para gestionar la interacción del usuario con el componente. Estas técnicas permiten una experiencia de usuario más rica, dinámica e intuitiva.

En el diseño de interfaces se aplican principios básicos de diseño de producto y usabilidad, como la claridad visual, la organización del contenido y la facilidad de interacción. También se consideran prácticas de responsive design y accesibilidad, aunque estas pueden ser ampliadas según las necesidades del proyecto, el uso de clases, funciones flecha, manejo de eventos, y la manipulación de atributos personalizados (data-*) para lograr una mayor flexibilidad e interactividad del componente.

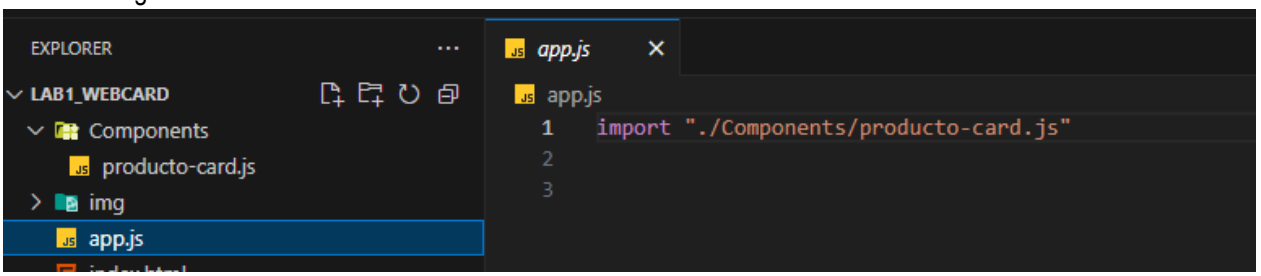
4. DESCRIPCIÓN DEL PROCEDIMIENTO:

La creación del proyecto se basa en lo que es el desarrollo y creación de los archivos para la demostración de lo visto en clase. Para ello dentro del mismo se creó los siguientes archivos que son producto-card.js, index.html, y app.js.

De las cuales el app.js será la conexión que dará paso a lo que son nuestros componentes hacia el index en sí. En el cual solo dentro del index podremos mostrar solo la plantilla formada a través de nuestro componente.



Como antes se mencionó lo que el app.js hará sera la importación del componente en sí de lo cual solo será una linea de codigo.



Dentro del js de nuestro producto-card construiremos lo que es nuestra clase que tendrá lo que es las siguientes funciones que son lo la clase con el constructor, los estilos, el template, render y el connectedCallback. De la cual se define un nuevo componente personalizado llamado productCard.

producto-card.js

Components > producto-card.js > productCard

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

    getStyles() {
      return `
        :host {
          font-family: 'Arial', sans-serif;
          display: block;
          width: 415px;
          height: auto;
          border-radius: 20px;
          overflow: hidden;
          background: #fff;
          box-shadow: 0 5px 20px rgba(0,0,0,0.1);
        }
        .card-container{
          display:inline-block;
          gap: 20px;
        }
        .card-2{
          margin: 100px;
          padding-right: 100px;
        }
        .card-header {
          background: #3ff275;
          height: 80px;
          border-bottom-left-radius: 50% 20px;
          border-bottom-right-radius: 50% 20px;
```

Creamos lo que es nuestro attachShadow que nos ayudará a activar el Shadow DOM para encapsular estilos y estructura.

```
constructor() {
  super();
  this.attachShadow({ mode: "open" });
}
```

Creamos nuestra plantilla basada en la imagen que se demostró en la clase, con la ayuda de la IA y se cambiaron algunas de las configuraciones que se nos muestra dentro del template generado dentro del getTemplate .

```
getTemplate() {
  const template = document.createElement("template");
  template.innerHTML = `
    <style>${this.getStyles()}</style>
    <section class="card-container">
      <div class="card">
        <div class="card-header"></div>
        <div class="card-body">
          <h2>Camisa Elegante</h2>
          <p>Es el artículo con la mejor relación calidad-precio</p>
          <div class="image">
              <img id="productImage" class="product-img" src="./img/Camisa.jpg" alt="Children Shirt">
          </div>
          <div class="price-rating">
              <span>$45</span>
              <div class="stars" id="stars">
                <span data-value="1">★</span>
                <span data-value="2">★</span>
                <span data-value="3">★</span>
                <span data-value="4">★</span>
                <span data-value="5">★</span>
              </div>
            </div>
          <div class="sizes">
            <strong>SIZE</strong><br>
            <span>XS</span><span>S</span><span>M</span><span>L</span><span>XL</span>
          </div>
          <div class="colors">
              <strong>COLORS</strong><br>
              <span class="color blue" data-img="./img/Camisa.jpg"></span>
              <span class="color fur" data-img="./img/Camisa2.webp"></span>
          </div>
          <div class="quantity">
              <button id="decrease">-</button>
              <span id="qty">1</span>
              <button id="increase">+</button>
            </div>
        </div>
        <div class="add-to-cart">ADD TO CART</div>
      </div>


    </section>

  `;
  return template;
}
```

Dentro de nuestro getStyles definimos en variables para el uso de los colores que tendrá nuestra interfaz gráfica que por medio de la misma se podrá observar.

```
getStyles() {
  return `
    :host {
      font-family: 'Arial', sans-serif;
      display: block;
      width: 415px;
      height: auto;
      border-radius: 20px;
      overflow: hidden;
      background: #fff;
      box-shadow: 0 5px 20px rgba(0,0,0,0.1);
    }
    .card-container{
      display:inline-block;
      gap: 20px;
    }
    .card-2{
      margin: 100px;
      padding-right: 100px;
    }
    .card-header {
      background: #3ff275;
      height: 80px;
      border-bottom-left-radius: 50% 20px;
      border-bottom-right-radius: 50% 20px;
    }

    .card-body {
      padding: 40px;
      text-align: center;

    }
    h2 {
      color: #7e3ff2;
      margin: 10px 0 5px;
      font-size: 1.2rem;
    }
    p {
      font-size: 0.85rem;
      color: #888;
    }
    .product-img {
```

Para una mayor interacción se implementa el cambio de que ciertas cosas se pueden modificar dentro de nuestra plantilla, que es la cantidad del producto y de el cambio de imagen dentro del producto mismo.

```
setRatingEvents() {
    const stars = this.shadowRoot.querySelectorAll("#stars span");
    stars.forEach(star => {
      star.addEventListener("click", () => {
        const rating = parseInt(star.dataset.value);
        stars.forEach((s, index) => {
          s.textContent = index < rating ? "★" : "☆";
        });
      });
    });
  }

  setQuantityEvents() {
    const decreaseBtn = this.shadowRoot.getElementById("decrease");
    const increaseBtn = this.shadowRoot.getElementById("increase");
    const qtyDisplay = this.shadowRoot.getElementById("qty");
    let quantity = 1;

    decreaseBtn.addEventListener("click", () => {
      if (quantity > 1) {
        quantity--;
        qtyDisplay.textContent = quantity;
      }
    });

    increaseBtn.addEventListener("click", () => {
      quantity++;
      qtyDisplay.textContent = quantity;
    });
  }
  setColorEvents() {
    const colors = this.shadowRoot.querySelectorAll(".color");
    const img = this.shadowRoot.getElementById("productImage");

    colors.forEach(color => {
      color.addEventListener("click", () => {
        // Remueve selección actual
        colors.forEach(c => c.classList.remove("selected"));
        color.classList.add("selected");

        // Cambia la imagen
        const newImg = color.getAttribute("data-img");
        img.setAttribute("src", newImg);
      });
    });
  }
```

Una vez creado las funciones para la interacción de nuestro card, hacemos la función de render que nos inserta la plantilla en el Shadow DOM del componente.

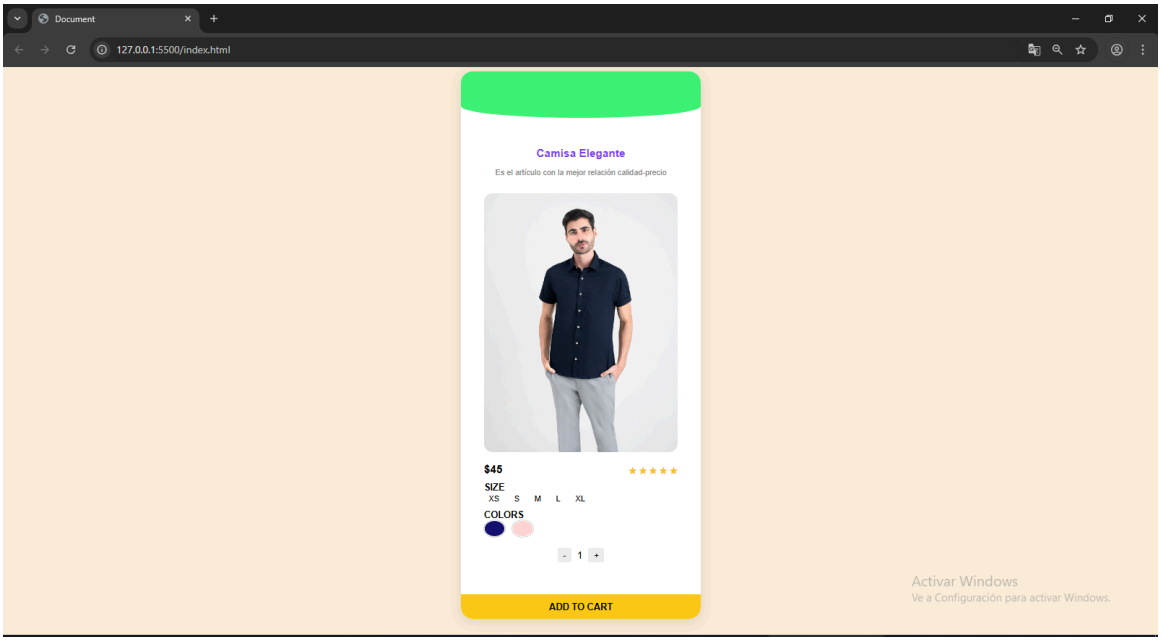
```
render() {
  this.shadowRoot.appendChild(this.getTemplate().content.cloneNode(true));
}
```

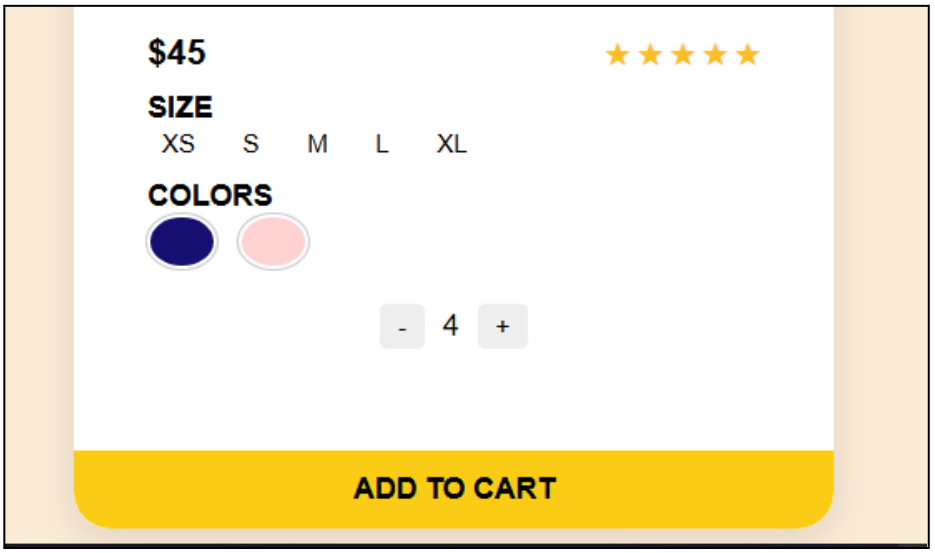
Creamos nuestro connedctedCallback que se ejecuta cuando el elemento se inserta en el DOM. Renderiza el HTML y activa los eventos: estrellas, cantidad, y colores. Y por último creamos el complemento en el cual registra el componente personalizado <product-card> para que pueda usarse en HTML como <product-card></product-card>.

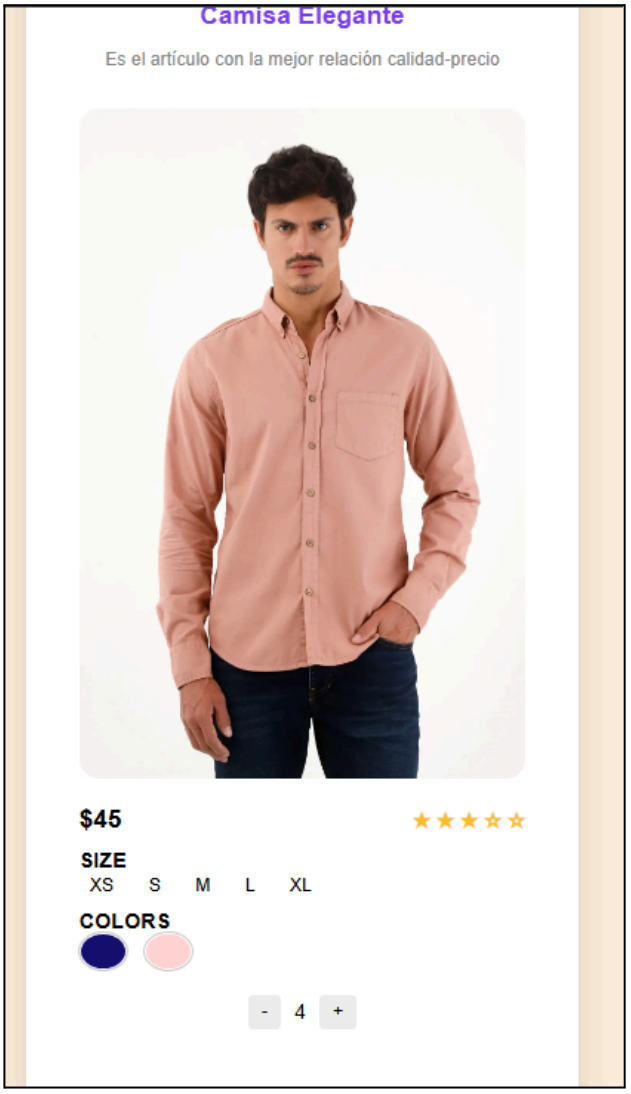
```
  connectedCallback() {
    this.render();
    this.setRatingEvents();
    this.setQuantityEvents();
    this.setColorEvents();
  }


}

customElements.define("product-card", productCard);
```

5. ANÁLISIS DE RESULTADOS:

El resultado final sería la demostración de nuestro producto-card en el index principal con los cambios correspondientes que es el aumento de la cantidad del producto, el cambio de imagen y las estrellas.







6. CONCLUSIONES:

- Se logró recrear una plantilla de un producto utilizando HTML, CSS y JavaScript, representando fielmente cómo se mostraría un artículo en la fotografía, y aplicando el diseño visual y estructural.

- Se implementan correctamente los conocimientos sobre Web Components, especialmente el uso del Shadow DOM y la encapsulación de estilos, lo cual permitió que el componente fuera independiente y fácil de reutilizar en otros contextos.

- Se observa las vistas en clase, integrando eventos, manipulación del DOM y lógica dinámica dentro del componente, lo cual refleja un dominio básico de los conceptos fundamentales en desarrollo frontend moderno.

7. BIBLIOGRAFÍA:

Wicher, J. (2021). Introduction to Web Components. MDN Web Docs, Mozilla. Disponible en: https://developer.mozilla.org/en-US/docs/Web/Web_Components

Bosworth, A., & Donnelly, J. (2018). Web Components: The Future of Web Development. Journal of Modern Web Engineering