

<b>DEPARTAMENTO:</b>	Ciencias de la Computación - DCCO-SS	<b>CARRERA:</b>	Ingeniería en tecnologías de la Información		
<b>ASIGNATURA:</b>	Programación Integrativa	<b>NIVEL:</b>	Sexto	<b>FECHA:</b>	25/05/2025
<b>DOCENTE:</b>	Ing. Paulo Galarza	<b>PRÁCTICA N°:</b>	3	<b>CALIFICACIÓN:</b>	

## Implementar Especificaciones Avanzadas en Web Components

ANTHONY GEOVANNY MEJIA GAIBOR

### RESUMEN

En esta práctica de laboratorio, se desarrolló un componente web modular utilizando tecnologías modernas como ES Modules, <template>, Shadow DOM y eventos personalizados. El propósito fue explorar y aplicar estas herramientas para crear componentes reutilizables y encapsulados, enfocándose en la implementación de un modal (<espe-modal>). Se logró estructurar el código de manera modular, aislar estilos y estructura mediante Shadow DOM, y establecer comunicación entre componentes con eventos personalizados. Los resultados muestran un componente funcional que cumple con los requisitos establecidos, destacando la importancia de estas tecnologías en el desarrollo web moderno. Este ejercicio también promueve la disciplina en el uso de herramientas de programación.

**Palabras Clave:** ES Modules, Shadow DOM, eventos personalizados.

## **1. INTRODUCCIÓN:**

La programación web ha evolucionado hacia un enfoque modular que prioriza la reutilización y el encapsulamiento de componentes. Tecnologías como ES Modules facilitan la organización del código, mientras que `<template>` y Shadow DOM permiten crear componentes independientes del DOM global. Los eventos personalizados, por su parte, mejoran la interacción entre elementos de una aplicación. Esta práctica de laboratorio tuvo como objetivo aplicar estos conceptos en el desarrollo de un componente web, resaltando el manejo disciplinado de las herramientas y la adherencia a buenas prácticas. A través de este ejercicio, se buscó comprender y dominar estas tecnologías en un entorno controlado.

## **2. OBJETIVO(S):**

- 2.1 Implementar un componente web utilizando ES Modules para lograr una modularización efectiva del código.
- 2.2 Diferenciar y aplicar correctamente `<template>` y Shadow DOM en la creación de componentes web encapsulados.
- 2.3 Utilizar eventos personalizados para permitir la comunicación entre el componente y la página principal, demostrando su utilidad en escenarios reales.

### 3. DESARROLLO

#### 4.1 Cómo se logra la modularización con ES Modules

La modularización se consigue al definir el componente en un archivo JavaScript independiente, como `espe-modal.js`. En este archivo, se crea una clase (`EspeModal`) que se exporta con `export { EspeModal };`. Posteriormente, en el archivo principal (`index.html`), se importa mediante `<script type="module" src="espe-modal.js"></script>`. Este enfoque aísla la lógica del componente, permitiendo su reutilización y manteniendo el código organizado y escalable.

#### 4.2 Diferencias entre `<template>` y Shadow DOM

`<template>` se emplea para definir la estructura HTML y los estilos del componente sin renderizarlos de inmediato. Por ejemplo, se crea un elemento `<template>` con el contenido del modal, que luego se clona y usa en el componente. En contraste, Shadow DOM encapsula esta estructura y estilos al adjuntarse al componente con `this.attachShadow({ mode: 'open' })`. Esto aísla el DOM del componente del resto de la página, evitando conflictos de estilos y mejorando la mantenibilidad.

#### 4.3 Casos de uso de eventos personalizados en aplicaciones reales

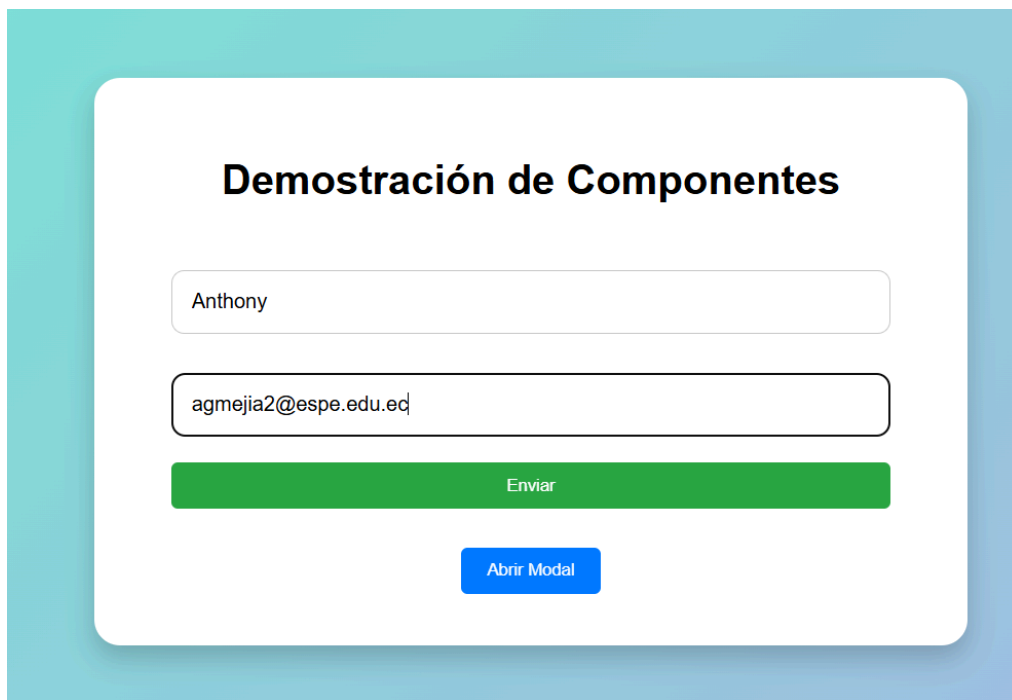
Los eventos personalizados facilitan la comunicación entre componentes. En el modal `<espe-modal>`, al hacer clic en "Cerrar", se emite un evento `'cerrar-modal'` con `this.dispatchEvent(new CustomEvent('cerrar-modal', { bubbles: true, composed: true }));`. La página principal escucha este evento y responde, por ejemplo, ocultando el modal. Este patrón es útil en aplicaciones complejas donde múltiples componentes deben interactuar, como en sistemas de notificaciones o interfaces dinámicas.

### Código en ejecución

La imagen presenta la interfaz principal de la aplicación web desarrollada para la Tarea 3. Se observa un formulario con campos para ingresar el nombre y el correo, un botón verde "Enviar" y un botón azul "Abrir Modal". La interfaz está contenida en un cuadro blanco con bordes redondeados sobre un fondo degradado, diseñada para ser visualmente atractiva y funcional.

### Figura 1

*Interfaz de la Aplicación Web con Componentes Modulares*

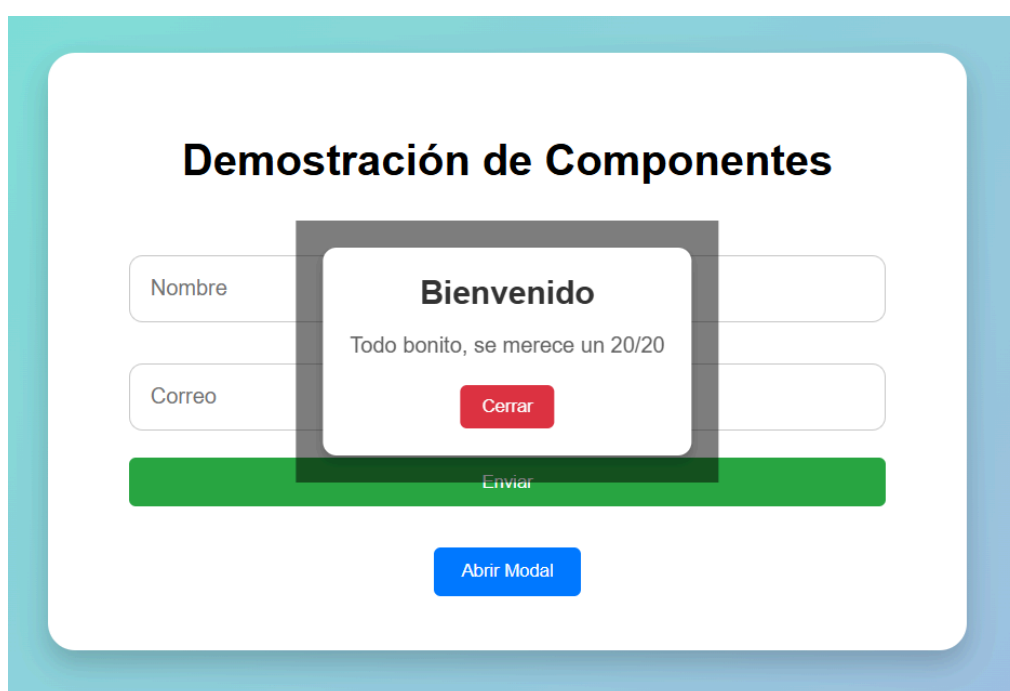


*Nota: Elaboración propia (2025). La imagen contiene una captura de pantalla de la aplicación web, mostrando el componente <espe-formulario> con campos personalizados y botones interactivos para abrir un modal.*

La imagen muestra el modal abierto tras hacer clic en el botón "Abrir Modal" de la aplicación web. El modal presenta un título "Bienvenido" y un mensaje "Todo bonito, se merece un 20/20", junto con un botón rojo "Cerrar". El modal se superpone sobre la interfaz principal, destacando su diseño encapsulado y funcional.

### **Figura 2**

#### *Modal Abierto en la Aplicación Web*

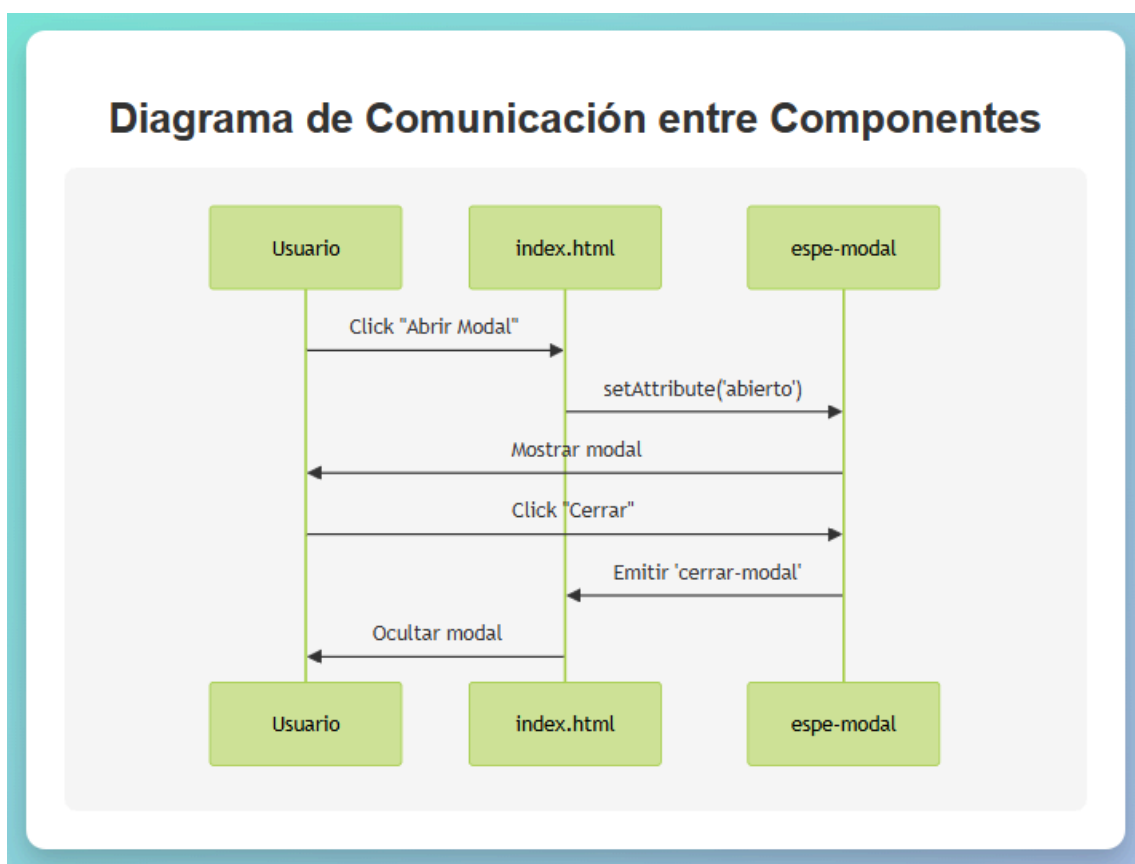


*Nota: Elaboración propia (2025). La imagen contiene una captura de pantalla del componente <espe-modal> en acción, mostrando su contenido personalizado y el botón de cierre que emite un evento personalizado.*

La imagen muestra un diagrama de secuencia que ilustra las interacciones entre el usuario, index.html, y el componente <espe-modal>. Se observa cómo el usuario activa el modal mediante un clic, el evento cerrar-modal emitido por el componente, y la respuesta de la página principal para ocultar el modal.

**Figura 3**

*Diagrama de Comunicación entre Componentes*



*Nota: Elaboración propia (2025). La imagen contiene un diagrama de secuencia generado a partir de la sintaxis*

*Mermaid, representando el flujo de comunicación en la aplicación.*

#### **4. CONCLUSIONES**

La práctica permitió implementar un componente modular con ES Modules, logrando un código reutilizable y bien estructurado. Se comprendió la diferencia entre `<template>`, que define estructuras inertes, y Shadow DOM, que las encapsula, cumpliendo el segundo objetivo. Finalmente, los eventos personalizados demostraron ser efectivos para la comunicación entre componentes, validando su aplicación en escenarios reales. Estos resultados refuerzan la utilidad de estas tecnologías en el desarrollo de aplicaciones web modernas.