Fase 1-De libreta a pantalla: nuestra primera web de la biblioteca musical

Objetivos

* Muestra un título principal y un párrafo de presentación.
* Tiene un formulario con un input de texto y un botón.
* Cuando el usuario pulsa el botón, aparece un mensaje personalizado con su nombre.
* Estéticamente debe tener estilo básico: fondo claro, texto legible, botones visibles.
* Si el usuario no escribe nada y pulsa el botón, debe aparecer un mensaje de error suave

Resultado

Campo/s faltante/s:

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Todos los campos correctos:

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

Fase 2 – Tu primer sistema digital de registro

Objetivos

* Reusa el formulario anterior.
* Validar que no falte nada importante.
* Crear un objeto con esos datos.
* Guardarlo dentro de un array que representa tu base de datos temporal.
* Mostrar las canciones registradas en una lista en pantalla, con su información ordenada.

Resultado

En mi caso, se pueden agregar portadas, y si no se agregan, se añade una por defecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Preguntas

##### ¿Qué ventajas tiene mostrar un mensaje dinámico respecto a una presentación estática?

Esto tiene grandes ventajas utilitarias, la principal es que no se tiene que cambiar el contenido en el código manualmente. Si se quieren hacer modificaciones se modifica o genera el archivo de nuevo.

##### ¿Qué es el textContent en JavaScript y por qué lo usaste para mostrar el mensaje?

El textContent es un método de JS que permite interactuar con el texto de una etiqueta.

En mi caso, lo he usado para añadir el texto interior del mensaje. Primero lo identifico con un ID:

<div *id*="msg" *class*="alert d-none position-fixed top-0 end-0 mt-3 me-3 text- start shadow" *role*="alert">

      <i *id*="alert-icon" *class*="me-2"></i>

<p *id*="alert-msg" *class*="m-0"></p>

</div>

Y luego le añado contenido

const alertMsg = document.getElementById("alert-msg");

alertMsg.textContent = `🎵 "${titulo}" de ${artista} añadida correctamente!`;

##### ¿Qué problema evita la función event.preventDefault() en los formularios?

Esta función es muy importante, ya que impide que se reinicie la página web al pulsar el botón de “submit”. Esto es vital para poder tratar la información a la hora de comprobar el contenido.

##### ¿Qué comprobación hiciste para evitar que el usuario dejara el campo vacío?

Compruebo si el valor de las variables existe. Si no existe, se le añade el contenido correspondiente al error. Si error tiene contenido, se añade la alerta.

const titulo = document.getElementById('titulo').value.trim();

const artista = document.getElementById('artista').value.trim();

const genero = document.getElementById('genero').value.trim();

let error = '';

if (!titulo) error = "El título es requerido";

else if (!artista) error = "El artista es requerido";

else if (!genero) error = "Debes seleccionar un género";

if (error) {

    alertIcon.className = "fas fa-exclamation-circle text-danger";

    alertMsg.textContent = error;

    msg.className = "alert alert-danger position-fixed top-0 end-0 mt-3 me-3 text-start shadow";

    msg.classList.remove("d-none");

    setTimeout(() => msg.classList.add("d-none"), 3000);

    return false;

}

##### Muestra una captura del formulario básico y del mensaje personalizado funcionando.

Este es el formulario. Como se observa, si lo mando vacío, sale el mensaje de error correspondiente.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

En cambio, si lo envío con contenido, se puede ver que se registra.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

##### ¿Qué diferencias encuentras entre escribir un nombre en papel y hacerlo en un formulario digital?

##### Encuentro las siguientes diferencias:

##### Fiabilidad: En un formulario digital puedes controlar que el nombre esté completo, sin caracteres inválidos y sin errores de ortografía mediante validaciones automáticas. En papel, esto depende 100% del usuario

* Velocidad: Se puede llegar a escribir más rápido digitalmente si se usan funciones de autocompletado, sugerencias, etc. Además, la modificación del texto ya escrito es algo que no se encuentra en el formato analógico.
* Legibilidad: Uno digital asegura que todo se encuentre escrito con el mismo formato de letra. Mientras que, en el papel, no se garantiza que tenga una consistencia.

##### ¿Qué pasos del proceso tradicional (en papel) se han digitalizado en esta fase?

Estos son los siguientes pasos del proceso tradicional que he logrado identificar, junto con sus respectivos ejemplos:

* Entrada de datos manual (escribir) ➜ Entrada por teclado digital:

<input *type*="text" *id*="titulo" *name*="titulo" *class*="form-control" *required*>

* Validación manual (revisar visualmente) ➜ Validación automática

if (!titulo) error = "El título es requerido";

* Almacenamiento físico (libreta, documento) ➜ Almacenamiento digital (arrayList o Json)

songsArray.push(newSong);

##### ¿Cómo afecta la digitalización a la accesibilidad de la información?

Creo que mejora muchísimo la accesibilidad. Esto se refleja en que mejoraría mucho la función de la biblioteca musical: Puedes consultar, modificar o borrar datos desde cualquier dispositivo con conexión, sin necesidad de buscar entre carpetas o papeles. También puedes hacer copias de seguridad o compartir los datos con otras personas en segundos.

##### ¿Qué función visual o técnica implementaste para mejorar la experiencia del usuario?

He añadido diversas funciones visuales que ayudan al usuario a encontrarse más cómodo. Por ejemplo, el mensaje de error o de éxito al introducir los campos no es invasivo y desaparece a los segundos.

Otra función visual es el efecto de zoom en la portada de la canción al hacer hover.

##### Si otra persona quisiera reutilizar tu formulario, ¿qué partes del código necesitaría modificar?

Todos los fragmentos de código relacionados con un formulario, con validarlo o con gestionar la información que se introduce, podrían ser fácilmente reusados.

Fase 3 - Buscando canciones reales con una API

Objetivos:

1. Introducir el nombre de un Pokémon en un campo de texto.
2. Al pulsar un botón, realizar una petición HTTP usando fetch() a la dirección: https://pokeapi.co/api/v2/pokemon/{nombre} API
3. Si el Pokémon existe, mostrar en pantalla:
   1. Imagen frontal
   2. Nombre oficial
   3. Tipo o tipos
   4. Lista de habilidades
   5. Peso (en kilogramos)
4. Si no existe o hay un error, mostrar un mensaje claro y amigable