Universidad de Guadalajara

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Ingeniería Informática.

Actividad Integradora 1: Agregación de eventos de forma dinámica

Desarrollo de Front End.

Yosef Sánchez Gutiérrez
Marco Alejandro González Mireles
Mirella Stephania Palomera Gómez
Roberto Carlos Martinez Aviña

Mtra. Rosalia Iñiguez .

Introducción

En esta actividad se desarrolló un formulario web interactivo utilizando HTML, CSS y JavaScript, con el objetivo de aplicar y reforzar los conocimientos adquiridos sobre el DOM (Document Object Model), selectores, eventos y la manipulación dinámica de estilos en el navegador, así como conocer los eventos que se disparan de forma dinámica en una página Web. El formulario simula un sistema de registro de futbolistas, permitiendo validar los datos introducidos en tiempo real, proporcionando retroalimentación visual inmediata al usuario, así como borrar un elemento insertado, o bien, el listado de elementos completo.

El objetico principal fue implementar la gestión dinámica de elementos en el DOM mediante el uso de eventos con addEventListener, permitiendo agregar y eliminar elementos de forma interactiva, y mantener sincronizados los cambios con el almacenamiento local sin recargar la página.

Desarrollo de la Actividad

Estructura del Formulario HTML

El formulario permite ingresar los siguientes datos de un futbolista:

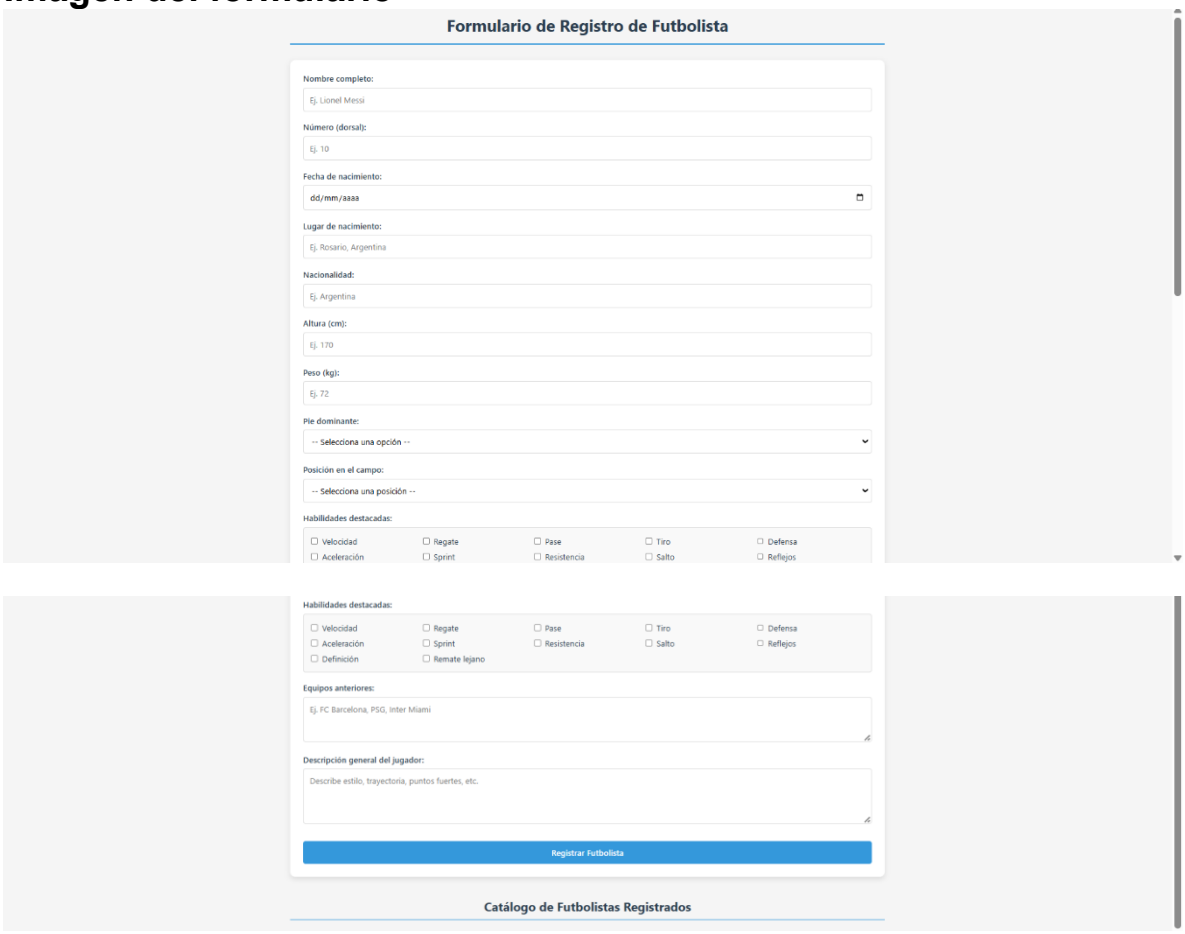
- Nombre completo
- Nacionalidad
- Edad
- Descripción
- Posición en el campo (desplegable)
- Habilidades (casillas de verificación)

Se incluyó un botón para activar la validación sin enviar el formulario, lo que permite controlar completamente la experiencia del usuario desde JavaScript.

También se incluyó el botón para eliminar un registro, y otro botón para eliminar el listado completo.

El diseño de la estructura fue enfocado en la accesibilidad, claridad y facilidad de uso, permitiendo una interacción fluida y amigable.

Imagen del formulario



Registro

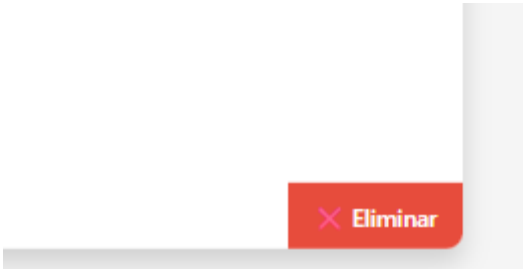
Código CSS

En el archivo styles.css se mantuvo el enfoque en un diseño limpio, moderno y profesional, con mejoras que optimizan la experiencia visual y la usabilidad del formulario web.
Se conservaron las bases del diseño anterior con bordes redondeados, fondo blanco y sombras suaves para dar profundidad, pero se agregaron detalles que mejoran la organización y la adaptabilidad en distintos dispositivos mediante el uso de CSS Grid para el manejo de las casillas de verificación de habilidades.
La tipografía sigue siendo clara y legible, usando fuentes sans-serif modernas para facilitar la lectura.
La paleta de colores se mantiene consistente, con:
Tonos de azul brillante para botones y elementos destacados que aportan identidad visual y contraste.
Grises suaves para fondos y campos de entrada que generan un ambiente neutro y profesional.
Detalles en amarillo para acentos secundarios en encabezados.
Se añadieron transiciones suaves en los focos y en los botones para mejorar la respuesta visual a la interacción del usuario.
Para la validación visual, se mantienen y refuerzan las clases .error y .success, que aplican bordes y sombras de colores rojo y verde respectivamente, brindando retroalimentación clara e inmediata.
Además, se definieron estilos específicos para las tarjetas dinámicas que representan a cada futbolista, incluyendo efectos hover para hacerlas más interactivas, distribución en flexbox para ordenar imagen y detalles, y estilos responsivos para adaptar la visualización en dispositivos móviles.
Se diseñaron botones específicos para la interacción de eliminación:

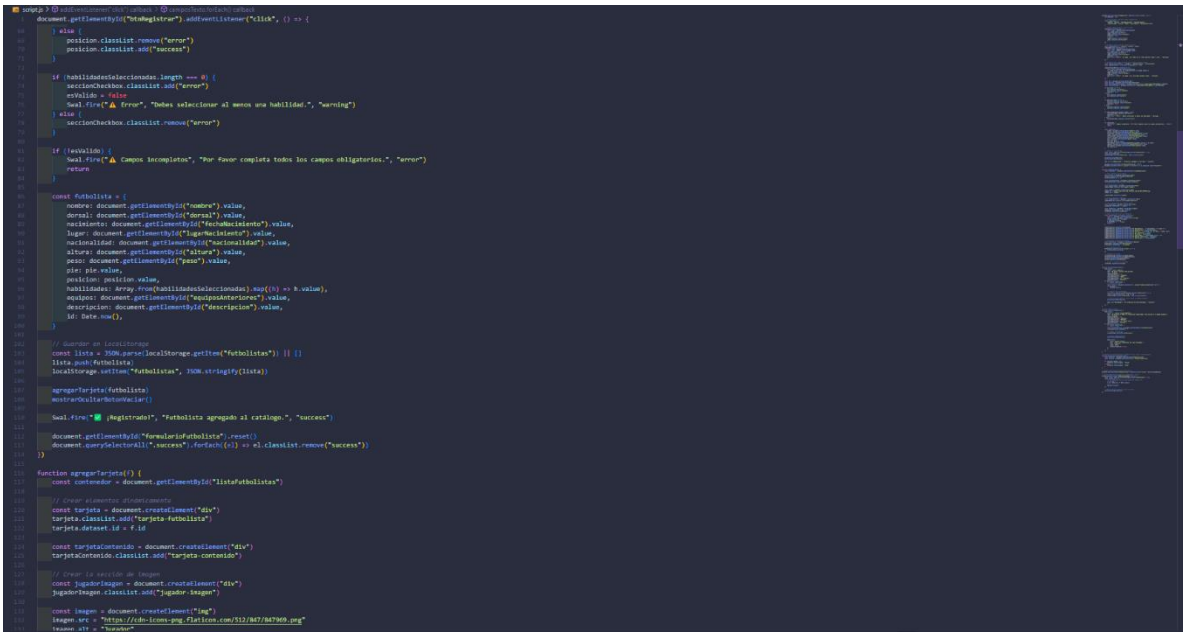
- El botón de borrar registro individual posee un estilo destacado en rojo, ubicado en la esquina inferior derecha de cada tarjeta, con efecto hover que intensifica el color para indicar acción de borrado.

- El botón para vaciar todo el listado se integra al inicio del listado (aunque en el CSS se puede usar estilos comunes de botón) y mantiene la coherencia visual con el resto de los controles, asegurando que su función sea clara y accesible para el usuario.

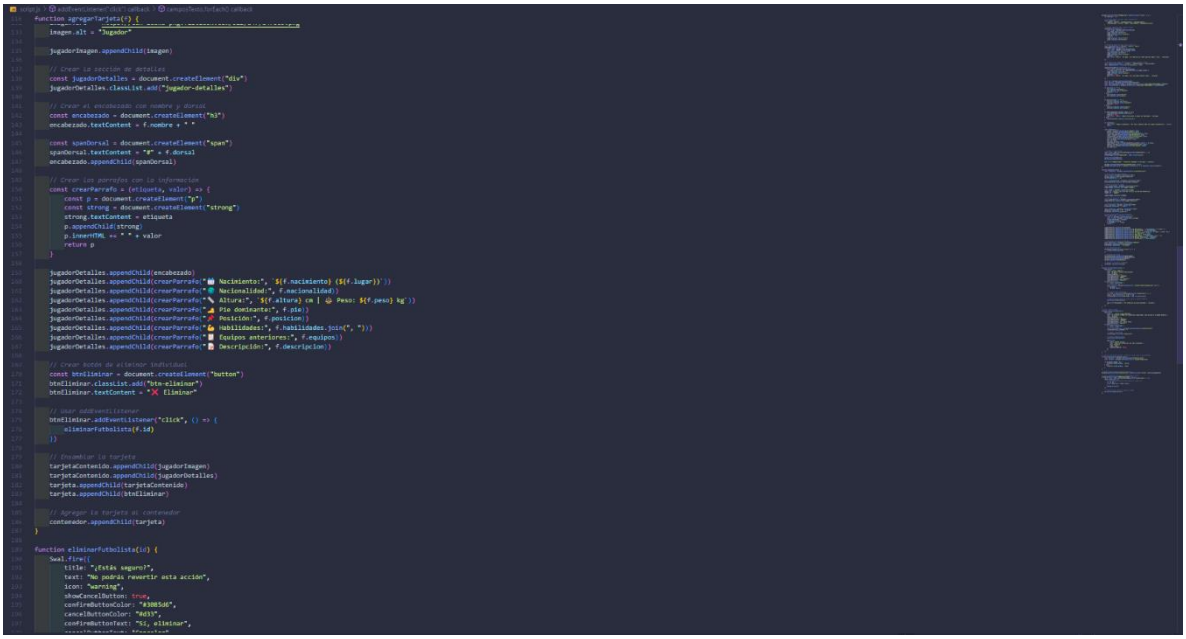
Estas mejoras visuales y estructurales hacen que la interfaz sea más amigable, accesible y moderna, facilitando la interacción y lectura tanto en escritorio como en pantallas pequeñas, además de ofrecer un control intuitivo y rápido para la gestión de los registros visualizados.



Código Javascript



```
script.js > addEventListener("click") callback > camposTexto.forEach() callback
document.getElementById("btnRegistrar").addEventListener("click", () => {
    } else {
        posicion.classList.remove("error")
        posicion.classList.add("success")
    }

    if (habilidadesSeleccionadas.length === 0) {
        seccionCheckbox.classList.add("error")
        esValido = false
        Swal.fire("⚠️ Error", "Debes seleccionar al menos una habilidad.", "warning")
    } else {
        seccionCheckbox.classList.remove("error")
    }

    if (!esValido) {
        Swal.fire("⚠️ Campos incompletos", "Por favor completa todos los campos obligatorios.", "error")
        return
    }

    const futbolista = {
        nombre: document.getElementById("nombre").value,
        dorsal: document.getElementById("dorsal").value,
        nacimiento: document.getElementById("fechaNacimiento").value,
        lugar: document.getElementById("lugarNacimiento").value,
        nacionalidad: document.getElementById("nacionalidad").value,
        altura: document.getElementById("altura").value,
        peso: document.getElementById("peso").value,
        pie: pie.value,
        posicion: posicion.value,
        habilidades: Array.from(habilidadesSeleccionadas).map((h) => h.value),
        equipos: document.getElementById("equiposAnteriores").value,
        descripcion: document.getElementById("descripcion").value,
        id: Date.now(),
    }

    // Guardar en LocalStorage
    const lista = JSON.parse(localStorage.getItem("futbolistas")) || []
    lista.push(futbolista)
    localStorage.setItem("futbolistas", JSON.stringify(lista))

    agregarTarjeta(futbolista)
    mostrarOcultarBotonVaciar()

    Swal.fire("✅ ¡Registrado!", "Futbolista agregado al catálogo.", "success")

    document.getElementById("formularioFutbolista").reset()
    document.querySelectorAll(".success").forEach((el) => el.classList.remove("success"))
})

function agregarTarjeta(f) {
    const contenedor = document.getElementById("listaFutbolistas")

    // Crear elementos dinámicamente
    const tarjeta = document.createElement("div")
    tarjeta.classList.add("tarjeta-futbolista")
    tarjeta.dataset.id = f.id

    const tarjetaContenido = document.createElement("div")
    tarjetaContenido.classList.add("tarjeta-contenido")

    // Crear la sección de imagen
    const jugadorImagen = document.createElement("div")
    jugadorImagen.classList.add("jugador-imagen")

    const imagen = document.createElement("img")
    imagen.src = "https://cdn-icons-png.flaticon.com/512/847/847969.png"
    imagen.alt = "Jugador"

    jugadorImagen.appendChild(imagen)

    // Crear la sección de detalles
    const jugadorDetalles = document.createElement("div")
    jugadorDetalles.classList.add("jugador-detalles")

    // Crear el encabezado con nombre y dorsal
    const encabezado = document.createElement("h3")
    encabezado.textContent = f.nombre + " "

    const spanDorsal = document.createElement("span")
    spanDorsal.textContent = "#" + f.dorsal
    encabezado.appendChild(spanDorsal)

    // Crear los párrafos con la información
    const crearParrafo = (etiqueta, valor) => {
        const p = document.createElement("p")
        const strong = document.createElement("strong")
        strong.textContent = etiqueta
        p.appendChild(strong)
        p.innerHTML += " " + valor
        return p
    }

    jugadorDetalles.appendChild(encabezado)
    jugadorDetalles.appendChild(crearParrafo("📅 Nacimiento:", `${f.nacimiento} (${f.lugar})`))
    jugadorDetalles.appendChild(crearParrafo("🌍 Nacionalidad:", f.nacionalidad))
    jugadorDetalles.appendChild(crearParrafo("📏 Altura:", `${f.altura} cm | ⚖️ Peso: ${f.peso} kg`))
    jugadorDetalles.appendChild(crearParrafo("🦶 Pie dominante:", f.pie))
    jugadorDetalles.appendChild(crearParrafo("📌 Posición:", f.posicion))
    jugadorDetalles.appendChild(crearParrafo("💪 Habilidades:", f.habilidades.join(", ")))
    jugadorDetalles.appendChild(crearParrafo("🏟️ Equipos anteriores:", f.equipos))
    jugadorDetalles.appendChild(crearParrafo("📝 Descripción:", f.descripcion))

    // Crear botón de eliminar individual
    const btnEliminar = document.createElement("button")
    btnEliminar.classList.add("btn-eliminar")
    btnEliminar.textContent = "❌ Eliminar"

    // Evento para eliminar
    btnEliminar.addEventListener("click", () => {
        eliminarFutbolista(f.id)
    })

    // Ensamblar la tarjeta
    tarjetaContenido.appendChild(jugadorImagen)
    tarjetaContenido.appendChild(jugadorDetalles)
    tarjeta.appendChild(tarjetaContenido)
    tarjeta.appendChild(btnEliminar)

    // Agregar la tarjeta al contenedor
    contenedor.appendChild(tarjeta)
}

function eliminarFutbolista(id) {
    Swal.fire({
        title: "¿Estás seguro?",
        text: "No podrás revertir esta acción",
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: "Sí, eliminar",
```



```
script.js > addEventListener("click") callback > camposTexto.forEach() callback
function agregarTarjeta(f) {
    imagen.alt = "Jugador"

    jugadorImagen.appendChild(imagen)

    // Crear la sección de detalles
    const jugadorDetalles = document.createElement("div")
    jugadorDetalles.classList.add("jugador-detalles")

    // Crear el encabezado con nombre y dorsal
    const encabezado = document.createElement("h3")
    encabezado.textContent = f.nombre + " "

    const spanDorsal = document.createElement("span")
    spanDorsal.textContent = "#" + f.dorsal
    encabezado.appendChild(spanDorsal)

    // Crear los párrafos con la información
    const crearParrafo = (etiqueta, valor) => {
        const p = document.createElement("p")
        const strong = document.createElement("strong")
        strong.textContent = etiqueta
        p.appendChild(strong)
        p.innerHTML += " " + valor
        return p
    }

    jugadorDetalles.appendChild(encabezado)
    jugadorDetalles.appendChild(crearParrafo("📅 Nacimiento:", `${f.nacimiento} (${f.lugar})`))
    jugadorDetalles.appendChild(crearParrafo("🌍 Nacionalidad:", f.nacionalidad))
    jugadorDetalles.appendChild(crearParrafo("📏 Altura:", `${f.altura} cm | ⚖️ Peso: ${f.peso} kg`))
    jugadorDetalles.appendChild(crearParrafo("🦶 Pie dominante:", f.pie))
    jugadorDetalles.appendChild(crearParrafo("📌 Posición:", f.posicion))
    jugadorDetalles.appendChild(crearParrafo("💪 Habilidades:", f.habilidades.join(", ")))
    jugadorDetalles.appendChild(crearParrafo("🏟️ Equipos anteriores:", f.equipos))
    jugadorDetalles.appendChild(crearParrafo("📝 Descripción:", f.descripcion))

    // Crear botón de eliminar individual
    const btnEliminar = document.createElement("button")
    btnEliminar.classList.add("btn-eliminar")
    btnEliminar.textContent = "❌ Eliminar"

    // Evento para eliminar
    btnEliminar.addEventListener("click", () => {
        eliminarFutbolista(f.id)
    })

    // Ensamblar la tarjeta
    tarjetaContenido.appendChild(jugadorImagen)
    tarjetaContenido.appendChild(jugadorDetalles)
    tarjeta.appendChild(tarjetaContenido)
    tarjeta.appendChild(btnEliminar)

    // Agregar la tarjeta al contenedor
    contenedor.appendChild(tarjeta)
}

function eliminarFutbolista(id) {
    Swal.fire({
        title: "¿Estás seguro?",
        text: "No podrás revertir esta acción",
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: "Sí, eliminar",
```

Código para eliminar un registro

```
    // Crear botón de eliminar individual
    const btnEliminar = document.createElement("button")
    btnEliminar.classList.add("btn-eliminar")
    btnEliminar.textContent = "❌ Eliminar"

    // Usar addEventListener
    btnEliminar.addEventListener("click", () => {
        eliminarFutbolista(f.id)
    })

    // Ensamblar la tarjeta
    tarjetaContenido.appendChild(jugadorImagen)
    tarjetaContenido.appendChild(jugadorDetalles)
    tarjeta.appendChild(tarjetaContenido)
    tarjeta.appendChild(btnEliminar)

    // Agregar la tarjeta al contenedor
    contenedor.appendChild(tarjeta)
}

function eliminarFutbolista(id) {
    Swal.fire({
        title: "¿Estás seguro?",
        text: "No podrás revertir esta acción",
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#3085d6",
        cancelButtonColor: "#d33",
        confirmButtonText: "Sí, eliminar",
        cancelButtonText: "Cancelar",
    }).then((result) => {
        if (result.isConfirmed) {
        // Eliminar del DOM
        const tarjeta = document.querySelector(`.tarjeta-futbolista[data-id="${id}"]`)
        if (tarjeta) {
            tarjeta.remove()
        }

        // Eliminar del localStorage
        let lista = JSON.parse(localStorage.getItem("futbolistas")) || []
        lista = lista.filter((f) => f.id !== id)
        localStorage.setItem("futbolistas", JSON.stringify(lista))

        // Mostrar/ocultar botón de vaciar según si quedan elementos
        mostrarOcultarBotonVaciar()

        Swal.fire("¡Eliminado!", "El futbolista ha sido eliminado.", "success")
        }
    })
}
```

Eliminar un registro individual (eliminarFutbolista)

- Esta función recibe el id del futbolista que se quiere eliminar.
- Muestra un cuadro de confirmación usando Swal.fire para que el usuario confirme la acción.

- Si el usuario confirma:
    1. Busca en el DOM la tarjeta del futbolista con ese id y la elimina visualmente (tarjeta.remove()).
    2. Actualiza el localStorage eliminando ese futbolista del arreglo guardado.
    3. Actualiza la visibilidad del botón para vaciar la lista, en caso de que ya no queden elementos.
    4. Muestra una alerta de éxito indicando que el futbolista fue eliminado.

Código para vaciar el listado

```
// Vaciar toda la lista
function vaciarListaCompleta() {
    Swal.fire({
        title: "⚠️ ¿Vaciar lista completa?",
        text: "Se eliminarán TODOS los futbolistas registrados. Esta acción no se puede deshacer.",
        icon: "warning",
        showCancelButton: true,
        confirmButtonColor: "#e74c3c",
        cancelButtonColor: "#95a5a6",
        confirmButtonText: "Sí, vaciar todo",
        cancelButtonText: "Cancelar",
    }).then((result) => {
        if (result.isConfirmed) {
        // Limpiar el DOM
        const listaFutbolistas = document.getElementById("listaFutbolistas")
        listaFutbolistas.innerHTML = ""

        // Limpiar localStorage
        localStorage.removeItem("futbolistas")

        // Ocultar botón de vaciar
        mostrarOcultarBotonVaciar()

        Swal.fire({
            title: "¡Lista vacía!",
            text: "Todos los futbolistas han sido eliminados.",
            icon: "success",
            timer: 2000,
            showConfirmButton: false,
        })
        }
    })
}
```

Vaciar toda la lista (vaciarListaCompleta)

- Muestra una alerta de confirmación para vaciar todos los futbolistas.
- Si el usuario confirma:
    1. Limpia todo el contenido del contenedor donde se muestran las tarjetas (listaFutbolistas.innerHTML = "").
    2. Borra completamente la lista guardada en localStorage.
    3. Oculta el botón de vaciar la lista, porque ya no hay elementos.
    4. Muestra un mensaje confirmando que la lista fue vaciada exitosamente.

Conclusión

Podemos concluir que en esta actividad se fortaleció la interacción dinámica con el DOM al agregar a la interfaz la funcionalidad para eliminar elementos individuales mediante un botón de borrar que acompaña a cada registro, así como un botón general que permite vaciar completamente el listado de elementos. Se implementó el uso de addEventListener para asignar eventos de forma dinámica a los botones de eliminar durante la creación de cada elemento, asegurando una gestión eficiente y flexible del listado. Esto permitió que la interfaz respondiera a las acciones del usuario en tiempo real, mejorando la usabilidad y el control sobre los datos mostrados.

Además, la funcionalidad para vaciar todo el listado facilita la gestión masiva de datos, dejando solo el formulario disponible para nuevas entradas, lo que contribuye a una experiencia de usuario más fluida y organizada.

Esta práctica demuestra cómo la manipulación dinámica del DOM y el manejo de eventos en JavaScript son herramientas claves para construir aplicaciones web interactivas y adaptativas.

Archivos fuente entregados

- index.html
- styles.css
- script.js