UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

UNIDAD DE EDUCACIÓN A DISTANCIA

ASIGNATURA:

Aplicación de Tecnologías Web

ING.

Angel Geovanny Cudco Pomagually

TEMA:

Actividad de Aprendizaje N°2

NOMBRES:

Carlos Ramiro Yánez Yazán

NRC:

1382

Quito…8…diciembre de 2024

Creación de una aplicación web interactiva con funcionalidades básicas

Objetivos

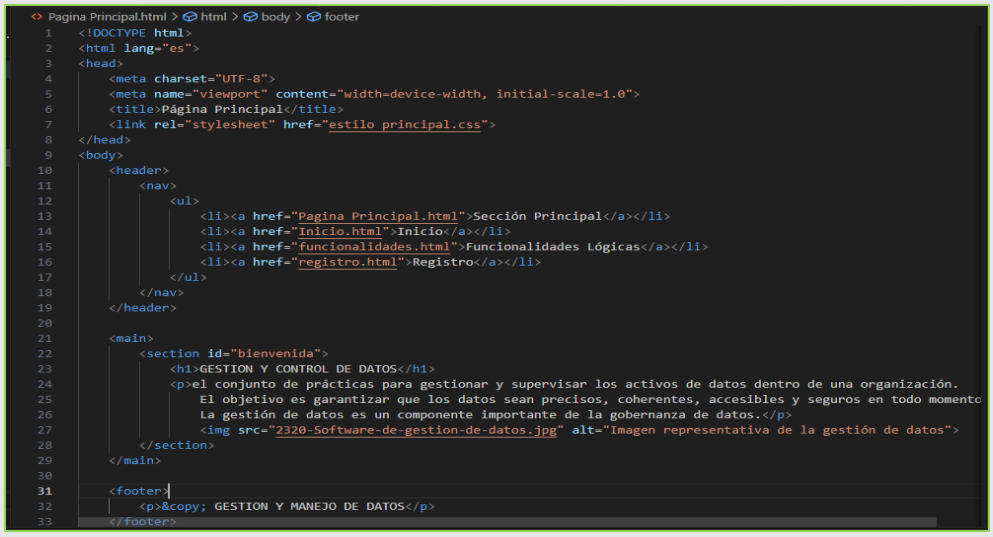
1. Gestionar una aplicación web interactiva con contenido, menús, párrafos, imágenes, etc que direccione a diferentes páginas que tendrán funcionalidades básicas “arrowfunctions” y estilos a través del uso de VScode.
2. Permitir que la aplicación responda de manera inteligente a las acciones del usuario, tomando decisiones basadas en condiciones específicas donde ciertas estructuras permitirán manejar flujos de control, como la validación de entradas de usuario, el procesamiento de datos o la visualización dinámica de contenido según las selecciones realizadas.
3. Crear un formulario dentro de la aplicación que permita la recopilación de datos del usuario de manera organizada, como información de registro. Mediante el uso de JavaScript, el formulario podrá validar las entradas antes de enviarlas, mostrar mensajes de error o confirmación, y realizar acciones adicionales.

Descripción y Comprobación de funcionalidades

Para iniciar con la comprobación de las funcionalidades partiremos a través de los requisitos que se nos solicitaron dentro de la actividad.

Pagina como sección principal de la aplicación web interactiva

En este punto la estructura que se uso fue primero definir una sección principal donde se tendrá todas las opciones dentro de un menú para opciones como Sección principal, Inicio, funcionalidades lógicas y Registro.



Se busco definir párrafos como imágenes que le den una mejor presentación al aplicativo web dándonos como resultado el siguiente esquema dentro del navegador.

A partir de aquí vamos a proceder a comprobar las diferentes funcionalidades que le di a mi página web.

PAGINA DE INICIO

- **Mostrar los datos personales del estudiante (nombre, correo y curso) en un diseño organizado**.

Para mostrar un diseño organizado se estableció la siguiente estructura en el código HTML.

Inicio.html > html > body > main

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DATOS DEL ESTUDIANTE</title>
    <link rel="stylesheet" href="estilo_principal.css">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="Pagina_Principal.html">Sección Principal</a></li>
                <li><a href="Inicio.html">Inicio</a></li>
                <li><a href="funcionalidades.html">Funcionalidades Lógicas</a></li>
                <li><a href="registro.html">Registro</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="datos-personales">
            <h1>Datos Personales del Estudiante</h1>
            <div class="info">
                <p><strong>Nombre:</strong> CARLOS RAMIRO YANEZ YAZAN</p>
                <p><strong>Correo:</strong> Ramiro3456_@outlook.com</p>
                <p><strong>Curso:</strong> Aplicacion de Tecnologias web</p>
            </div>
        </section>
    </main>

    <footer>
        <p>&copy; GESTION Y MANEJO DE DATOS</p>
```

En el código se puede observar cómo se estableció título para la página, diferentes enlaces externos que direcciona a otras páginas y lo esencial de este punto que es un cuadro con los **datos personales del estudiante.**

<h1>Datos Personales del Estudiante</h1>:

- Un encabezado de nivel 1 que describe el tema principal de la sección. Este es el título de la sección donde se mostrarán los datos personales.

** **:

- Un contenedor div con la clase info. Este es un contenedor de bloque que agrupa la información relacionada con los datos personales del estudiante. Usar clases permite aplicar estilos de manera específica desde el CSS.

** Nombre: CARLOS RAMIRO YANEZ YAZAN **:

- Un párrafo que contiene información textual. La etiqueta resalta el texto "Nombre:" en negrita para darle énfasis.

Los otros comandos son aspectos ya conocidos dentro de lo que es el manejo de VScode ya que se busco solo resaltar los más relevantes.

Resultado de aplicar dicha funcionalidad para el aplicativo web

PAGINA DE FUNCIONALIDADES LOGICAS

Para esta sección se mostrará las configuraciones para cada estructura solicitada de forma continua.

En esta sección se tomo en cuenta el orden para cada una de las estructuras **(if, swicth y array)** como también un modelo que conlleva la misma estructura de la pagina de inicio para opciones de enlaces externos por parte del menú.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Funcionalidades Lógicas</title>
    <link rel="stylesheet" href="estilo principal.css">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="Pagina Principal.html">Sección Principal</a></li>
                <li><a href="Inicio.html">Inicio</a></li>
                <li><a href="funcionalidades.html">Funcionalidades Lógicas</a></li>
                <li><a href="registro.html">Registro</a></li>
            </ul>
        </nav>
    </header>
```

Estructura if

Cree un div con dos campos de entrada (inputs) donde el usuario pueda ingresar dos números. Incluya un botón que compare ambos números utilizando una estructura condicional if y, al presionar el botón, despliegue el mayor de los dos números o un mensaje personalizado cuando sean iguales en un div de salida.

```html
    <!-- Estructura if -->
    <section id="if">
        <h1>Comparación de Números (if)</h1>
        <input type="number" id="numero1" placeholder="Ingresa un número">
        <input type="number" id="numero2" placeholder="Ingresa otro número">
        <button onclick="compararNumeros()">Comparar</button>
        <div id="resultado"></div>
    </section>
```

Estructura switch

Diseñe un div con un campo de entrada (input) para que el usuario ingrese un número entero del 1 al 12. Al presionar un botón, utilice una estructura switch para mostrar mediante una alerta el mes correspondiente al número ingresado.

```html
    <!-- Estructura switch -->
    <section id="switch">
        <h1>Meses del Año (switch)</h1>
        <input type="number" id="mes" placeholder="Ingresa un número entre 1 y 12">
        <button onclick="mostrarMes()">Mostrar Mes</button>
    </section>
```

Estructura y gestión de arrays

Implemente un div con un input que permita al usuario ingresar nombres de personas. Incluya un botón "Agregar", que almacene los nombres en un array y muestre dinámicamente la lista de nombres ingresados en otro div.

```
    <!-- Gestión de arrays -->
    <section id="array">
        <h1>Lista de Nombres</h1>
        <input type="text" id="nombre" placeholder="Ingresa un nombre">
        <button onclick="agregarNombre()">Agregar</button>
        <div id="nombres"></div>
    </section>
</main>
```

Aquí se observa como se manejo cada estructura en base a las condiciones establecidas, pero también se manejo un archivo java script del cual engloba toda la funcionalidad de cada una de las estructuras de la siguiente manera

```
    <footer>
        <p>&copy; GESTION Y MANEJO DE DATOS</p>
    </footer>

    <script src="script.js"></script>
</body>
</html>
```

Existe un pie de página, pero lo más importante aquí es que este <script> se usa para enlazar un archivo JavaScript externo (en este caso, "script.js"). El archivo contiene código JavaScript que puede añadir interactividad o funcionalidad adicional a la página web. Esta línea debe colocarse generalmente al final del documento HTML para garantizar que todo el contenido de la página se cargue antes de que el script se ejecute.

Ahora para determinar que las estructuras estén bien establecidas nos vamos a dirigir a lo que es el archivo java script y vemos como la estructura funcionara en base a las condiciones que tendremos.

Determinando la función para comparar dos números tenemos que:

```
// Función para comparar dos números utilizando if
const compararNumeros = () => {
    const num1 = parseInt(document.getElementById("numero1").value);
    const num2 = parseInt(document.getElementById("numero2").value);
    const resultado = document.getElementById("resultado");

    if (num1 > num2) {
        resultado.innerHTML = `El mayor número es: ${num1}`;
    } else if (num2 > num1) {
        resultado.innerHTML = `El mayor número es: ${num2}`;
    } else {
        resultado.innerHTML = "Ambos números son iguales.";
    }
};
```

Función para mostrar el mes consultado:

```
// Función para mostrar el mes utilizando switch
const mostrarMes = () => {
    const mes = parseInt(document.getElementById("mes").value);
    switch (mes) {
        case 1: alert("Enero"); break;
        case 2: alert("Febrero"); break;
        case 3: alert("Marzo"); break;
        case 4: alert("Abril"); break;
        case 5: alert("Mayo"); break;
        case 6: alert("Junio"); break;
        case 7: alert("Julio"); break;
        case 8: alert("Agosto"); break;
        case 9: alert("Septiembre"); break;
        case 10: alert("Octubre"); break;
        case 11: alert("Noviembre"); break;
        case 12: alert("Diciembre"); break;
        default: alert("Número inválido. Ingresa un número entre 1 y 12.");
    }
};
```

Y por último la función para gestionar un array de nombres:

```
// Función para gestionar un array de nombres
let nombresArray = [];
const agregarNombre = () => {
    const nombre = document.getElementById("nombre").value;
    if (nombre) {
        nombresArray.push(nombre);
        document.getElementById("nombres").innerHTML = `<p><strong>Lista de Nombres:</strong></p><ul>${nombresArray.map(n => `<li>${n}</li>`).joi
    }
};
```

Aquí se observó cómo se establecen cada una de las estructuras para darles un accionar en el aplicativo web con el uso de JavaScript.

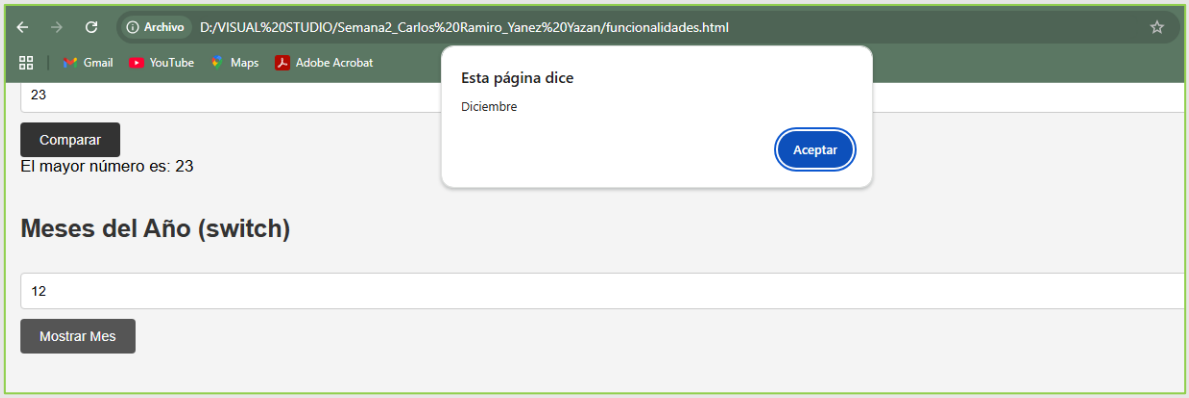
Como resultado tendremos las siguientes condiciones establecidas en la pagina del navegador en la sección **“funcionalidades lógicas”**

Para la estructura if:

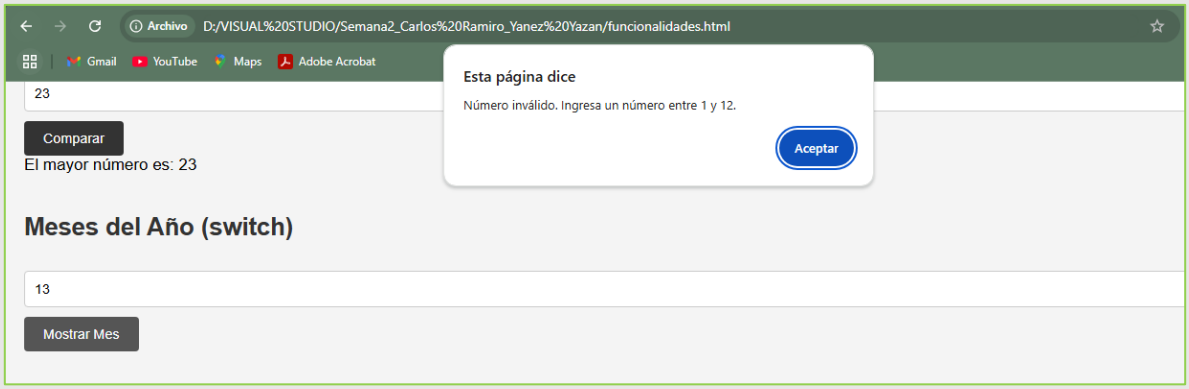
Demostrando que número será mayor.

Para cuando ambos números son iguales.

Para la estructura switch:

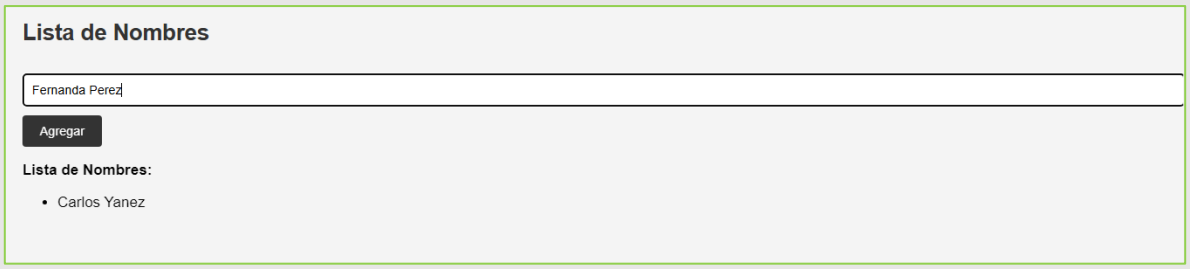


Aquí un dato importante es que si el numero no esta establecido entre los números del 1 al 12 nos surgirá un mensaje de error remitiéndonos que dicho numero no se encuentra establecido, esto se da gracias a las configuraciones que se establecieron dentro del archivo java script.

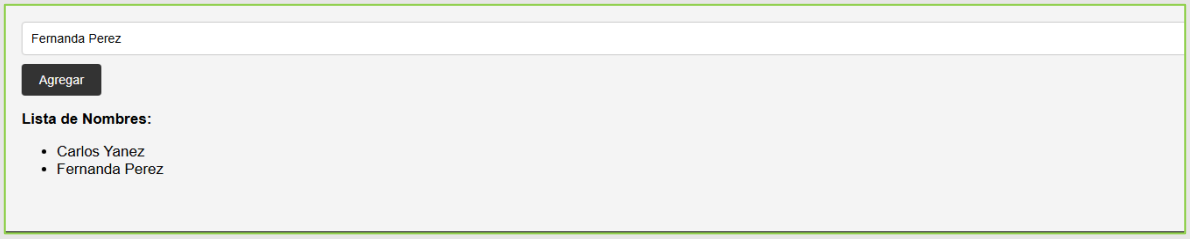


Para la estructura arrays

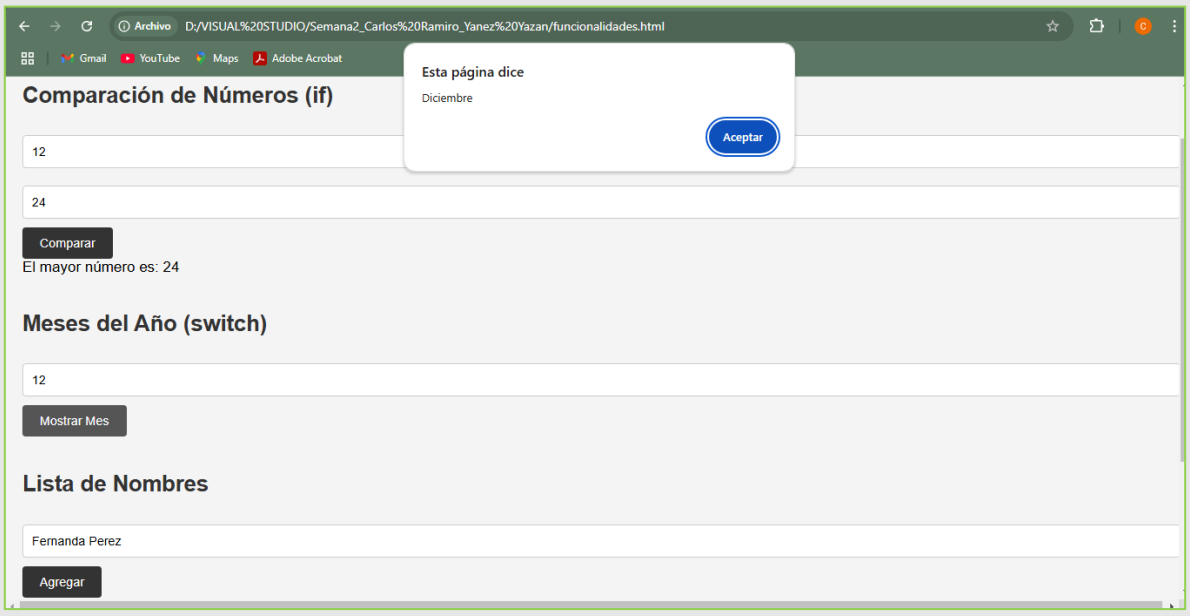
En este punto una manera más factible de comprobar es que vamos a establecer otro nombre y se deberá agregar oprimiendo un clic en el botón que tenemos para eso usaremos como ejemplo el nombre de “Fernanda Pérez”



Damos clic en agregar y a través del uso de la estructura array el nombre se desplazará hacia abajo.



Así queda demostrado la funcionalidad por parte de la sección de las funcionalidades lógicas dándonos como resultado general la siguiente imagen.



PAGINA DE REGISTRO DE USUARIOS

Cree un formulario para registrar usuarios con los siguientes campos:

- ID
- Cédula de identidad
- Nombres y apellidos
- Fecha de nacimiento
- Ciudad de procedencia (utilice un menú desplegable select con al menos cinco opciones).

Incluya un botón de "Registrar". Al presionarlo, redirija a una nueva página que muestre en formato tabular los datos ingresados.

Para esta sección se planto el archivo de la siguiente manera:

Se aplico una estructura general con enlaces externos para el menú del aplicativo web

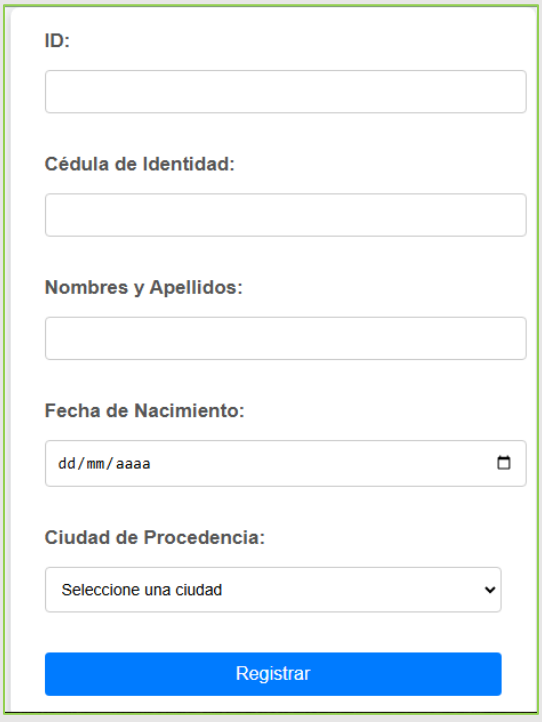
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registro de Usuarios</title>
    <link rel="stylesheet" href="estilo principal.css">
    <link rel="stylesheet" href="Formulario.css">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="Pagina Principal.html">Sección Principal</a></li>
                <li><a href="inicio.html">Inicio</a></li>
                <li><a href="funcionalidades.html">Funcionalidades Lógicas</a></li>
                <li><a href="registro.html">Registro</a></li>
            </ul>
        </nav>
```

A partir del siguiente apartado se establece lo que es el apartado para la función del formulario.

```html
<h1>Formulario de Registro</h1>
<form id="registroForm" target="_blank">
    <label for="id">ID:</label>
    <input type="text" id="id" required>
    <br><br>
    <label for="cedula">Cédula de Identidad:</label>
    <input type="text" id="cedula" required>
    <br><br>
    <label for="nombres">Nombres y Apellidos:</label>
    <input type="text" id="nombres" required>
    <br><br>
    <label for="fechaNacimiento">Fecha de Nacimiento:</label>
    <input type="date" id="fechaNacimiento" required>
    <br><br>
    <label for="ciudad">Ciudad de Procedencia:</label>
    <select id="ciudad" required>
        <option value="" disabled selected>Seleccione una ciudad</option>
        <option value="Quito">Esmeraldas</option>
        <option value="Guayaquil">Quito</option>
        <option value="Cuenca">Otavalo</option>
        <option value="Manta">Manta</option>
        <option value="Ambato">Ibarra</option>
    </select>
    <br><br>
    <button type="submit">Registrar</button>
</form>
<script src="script.js"></script>
```

Aquí se estable los requerimientos solicitados como **(ID, cedula, nombres fecha de nacimiento y ciudad).** Se definió también a través del archivo java script las funcionalidades que se le dar al formulario una vez establecido.

Una imagen general del mismo es la siguiente



Ahora al momento de nosotros registrar todos los datos nos pedía que la misma nos redirigiera a otra pagina por lo cual la estructura de la misma es la siguiente.

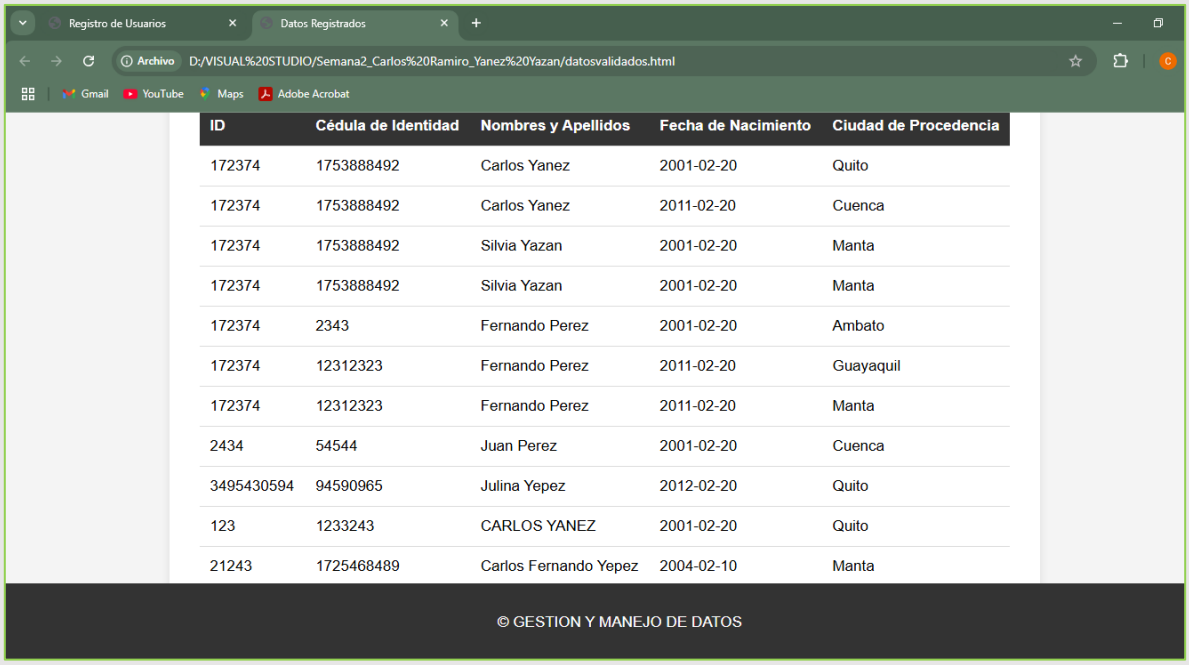
```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Datos Registrados</title>
    <link rel="stylesheet" href="estilo principal.css">
    <link rel="stylesheet" href="datos.css">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="Pagina Principal.html">Sección Principal</a></li>
                <li><a href="inicio.html">Inicio</a></li>
                <li><a href="funcionalidades.html">Funcionalidades Lógicas</a></li>
                <li><a href="registro.html">Registro</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="datos-registrados">
            <h1>Datos Registrados</h1>
            <table>
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Cédula de Identidad</th>
                        <th>Nombres y Apellidos</th>
                        <th>Fecha de Nacimiento</th>
                        <th>Ciudad de Procedencia</th>
                    </tr>
                </thead>
                <tbody id="tabla-datos">
                    <!-- Los datos serán insertados aquí con JavaScript -->
                </tbody>
            </table>
        </section>
    </main>

    <footer>
        <p>&copy; GESTION Y MANEJO DE DATOS</p>
    </footer>

    <script src="script.js"></script>
</body>
</html>
```

Estableciendo ya dichos comandos la funcionalidad de java también es importante la cual se configuro de la siguiente manera para lo que es mostrar los datos insertados en la tabla.

```javascript
// Verificar si el formulario de registro existe
const registroForm = document.getElementById("registroForm");

if (registroForm) {
    // Manejar el evento submit del formulario
    registroForm.onsubmit = (e) => {
        e.preventDefault(); // Prevenir el comportamiento predeterminado

        // Obtener los datos ingresados en el formulario
        const datos = {
            id: document.getElementById("id").value,
            cedula: document.getElementById("cedula").value,
            nombres: document.getElementById("nombres").value,
            fechaNacimiento: document.getElementById("fechaNacimiento").value,
            ciudad: document.getElementById("ciudad").value
        };

        // Recuperar datos previos del localStorage
        const registros = JSON.parse(localStorage.getItem("registros")) || [];

        // Agregar los nuevos datos al arreglo
        registros.push(datos);

        // Guardar el arreglo actualizado en localStorage
        localStorage.setItem("registros", JSON.stringify(registros));
        console.log("Registros actualizados:", registros); // Verificar que los datos se guardan correctamente

        // Abrir la página de datos en una nueva pestaña
        window.open("datosvalidados.html", "_blank");
    };
}
```

```
// Mostrar los datos en la página datosvalidados.html
if (window.location.pathname.includes("datosvalidados.html")) {
    // Obtener los datos del localStorage
    const registrosGuardados = JSON.parse(localStorage.getItem("registros")) || [];

    if (registrosGuardados.length > 0) {
        // Obtener el contenedor de la tabla
        const tablaDatos = document.getElementById("tabla-datos");

        // Recorrer los registros y agregar cada uno a la tabla
        registrosGuardados.forEach((registro) => {
            const row = document.createElement("tr");
            row.innerHTML = `
                <td>${registro.id}</td>
                <td>${registro.cedula}</td>
                <td>${registro.nombres}</td>
                <td>${registro.fechaNacimiento}</td>
                <td>${registro.ciudad}</td>
            `;
            tablaDatos.appendChild(row);
        });
    } else {
        alert("No se encontraron datos registrados.");
    }
}
```

Ya aplicando dichas configuraciones tendremos que como resultado al momento de ingresar los datos serán los siguientes.



Se observa como se cumplió con todos los requisitos tanto de desplegarse en una nueva pantalla como que los datos se vayan guardando.

(NOTA: Todos los procedimientos están explicados de mejor manera dentro del archivo readme.md por secciones)

Conclusiones

1. Una aplicación web interactiva con menús y contenido dinámico mejora la experiencia del usuario, facilitando el acceso a múltiples funcionalidades y páginas interconectadas.
2. Integrar lógica mediante **arrow functions** y estructuras de control permite que la aplicación reaccione de forma adecuada a las acciones del usuario, ofreciendo una experiencia más personalizada y funcional.
3. Incorporar formularios con validación en tiempo real mediante JavaScript asegura la calidad de los datos recopilados, mejora la usabilidad y reduce errores antes de procesar la información.