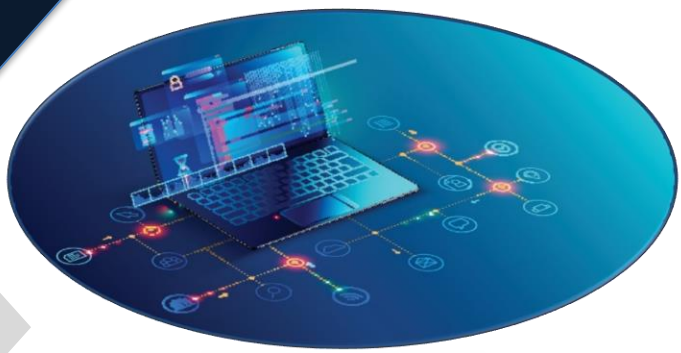
Desarrollo web en entorno cliente



1. Contenido

1.1 HTML

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>MedacDent - Gestión de Citas</title>
    <link rel="stylesheet" href="estilos.css">
    <link rel="icon" href="img/icono.png" type="image/x-icon">
</head>
<body>
    <header>
        <img src="img/icono.png" alt="Logo de MedacDent">
        <h1>MedacDent</h1>
    </header>

    <section>
        <form id="formCita">
            <h2>Programar Cita</h2>

            <!-- Campo oculto para almacenar id de la cita -->
            <input type="hidden" id="idCita">

            <!-- Datos de la cita -->
            <label for="fecha">Fecha:</label>
            <input type="date" id="fecha" required>

            <label for="hora">Hora:</label>
            <input type="time" id="hora" required>

            <!-- Datos del paciente -->
            <label for="nombre">Nombre:</label>
            <input type="text" id="nombre" pattern="[A-Za-z ]+" title="Solo letras y espacios permitidos" required>

            <label for="apellidos">Apellidos:</label>
            <input type="text" id="apellidos" pattern="[A-Za-z ]+" title="Solo letras y espacios permitidos" required>

            <label for="dni">DNI:</label>
            <input type="text" id="dni" pattern="[0-9]{8}[A-Za-z]" title="8 números seguidos de una letra" required>

            <label for="telefono">Teléfono:</label>
            <input type="tel" id="telefono" pattern="[0-9]{9}" title="Solo 9 números permitidos" required>

            <label for="fechaNacimiento">Fecha de Nacimiento:</label>
            <input type="date" id="fechaNacimiento" required>

            <!-- Observaciones -->
            <label for="observaciones">Observaciones:</label>
            <textarea id="observaciones"></textarea>

            <!-- Guardar cita -->
            <button type="submit">Guardar Cita</button>
        </form>
    </section>

    <h2 id="Citasprogramadas"> <b>Citas programadas</b></h2>

    <section>
        <table id="tablaCitas">
            <thead>
                <tr>
                    <th>Orden</th>
                    <th>Fecha</th>
                    <th>Hora</th>
                    <th>Nombre</th>
                    <th>Apellidos</th>
                    <th>DNI</th>
                    <th>Teléfono</th>
                    <th>Fecha de Nacimiento</th>
                    <th>Observaciones</th>
                    <th>Acciones</th>
                </tr>
            </thead>
            <tbody>
                <!-- Las citas se añadirán aquí -->
            </tbody>
        </table>
    </section>

    <footer>
        <p>Contacto: +34 123 456 789</p>
        <p>Dirección: C. de Albalá, 5, San Blas-Canillejas, 28037 Madrid</p>
        <p><a href="https://www.google.com/maps?q=C. de Albalá, 5, San Blas-Canillejas, 28037 Madrid"
                target="_blank">Ver en Google Maps</a></p>
    </footer>
    <script src="scripts.js"></script>
</body>
</html>
```

1.2 JavaScript

```javascript
// Esta sección de código se ejecuta cuando se carga completamente el documento HTML.
document.addEventListener('DOMContentLoaded', (event) => {
    establecerFechaMinima(); //Establece condiciones en las fechas y el horario
    cargarCitas(); // Carga y muestra las citas existentes.

    // Agrega un evento al enviar el formulario.
    const form = document.getElementById('formCita');
    form.addEventListener('submit', function (e) {
        e.preventDefault(); // Evita el comportamiento por defecto de envío del formulario.
        if (validarFormulario()) { // Verifica si el formulario es válido.
            guardarCita(); // Guarda la cita si el formulario es válido.
        }
    });
});

//Establece condiciones en las fechas y el horario
function establecerFechaMinima() {
    var today = new Date(); // Obtiene la fecha y hora actual.
    var dd = String(today.getDate()).padStart(2, '0'); // Obtiene el día actual y lo formatea con dos dígitos.
    var mm = String(today.getMonth() + 1).padStart(2, '0'); // Obtiene el mes actual (0-11) y lo ajusta a formato (1-12) con dos dígitos.
    var yyyy = today.getFullYear(); // Obtiene el año actual.

    var fechaActual = yyyy + '-' + mm + '-' + dd;

    document.getElementById('fecha').setAttribute('min', fechaActual); // Establece la fecha mínima para la fecha
    document.getElementById('fechaNacimiento').setAttribute('max', fechaActual); // Establece la fecha máxima para la fecha de nacimiento

    // Restringe el rango de horas seleccionables
    document.getElementById('hora').setAttribute('min', '08:00');
    document.getElementById('hora').setAttribute('max', '20:00');
}

// Función para cargar y mostrar todas las citas guardadas.
function cargarCitas() {
    let citas = JSON.parse(localStorage.getItem('citas')) || [];
    let cuerpoTabla = document.querySelector('#tablaCitas tbody');
    cuerpoTabla.innerHTML = '';

    if (citas.length === 0) {
        // Si no hay citas, muestra una fila con el mensaje "Dato vacío"
        let fila = cuerpoTabla.insertRow();
        let celda = fila.insertCell();
        celda.colSpan = 10; //Asume un total de 10 columnas
        celda.textContent = "Dato vacío";
        celda.className = 'dato-vacio'; // Se agrega una clase para poder identificar esta fila
    } else {
        // Si hay citas, muestra cada una en la tabla
        citas.forEach((cita, index) => {
            let fila = cuerpoTabla.insertRow();

            // Añade la celda para el número de orden
            fila.insertCell().textContent = index + 1;

            // Añade las celdas para los demás datos de la cita
            fila.insertCell().textContent = cita.fecha;
            fila.insertCell().textContent = cita.hora;
            fila.insertCell().textContent = cita.nombre;
            fila.insertCell().textContent = cita.apellidos;
            fila.insertCell().textContent = cita.dni;
            fila.insertCell().textContent = cita.telefono;
            fila.insertCell().textContent = cita.fechaNacimiento;
            fila.insertCell().textContent = cita.observaciones;

            // Añade las celdas para las acciones (editar/eliminar)
            let tdAcciones = fila.insertCell();

            //Botón editar
            let btnEditar = document.createElement('button');
            btnEditar.textContent = 'Editar';
            btnEditar.className = 'editar';
            btnEditar.onclick = function () { editarCita(cita.id); };
            tdAcciones.appendChild(btnEditar);

            //Botón eliminar
            let btnEliminar = document.createElement('button');
            btnEliminar.textContent = 'Eliminar';
            btnEliminar.className = 'eliminar';
            btnEliminar.onclick = function () { eliminarCita(cita.id); };
            tdAcciones.appendChild(btnEliminar);
        });
    }
}


// Esta función valida los campos del formulario antes de guardar una cita.
function validarFormulario() {
    let valido = true;
```

```
function validarFormulario() {
    let valido = true;

    // Aquí se realizan múltiples validaciones para cada campo:
    // Validación para el nombre
    const nombre = document.getElementById('nombre').value;
    if (nombre.trim() === '') {
        mostrarError('nombre', 'El nombre es obligatorio');
        valido = false;
    } else {
        limpiarError('nombre');
    }

    // Validación para el DNI
    const dni = document.getElementById('dni').value;
    const regexDNI = /^[0-9]{8}[A-Z]$/;
    if (!regexDNI.test(dni)) {
        mostrarError('dni', 'El DNI no es válido (debe tener 8 números seguidos de una letra mayúscula)');
        valido = false;
    } else {
        limpiarError('dni');
    }

    // Validación para los apellidos
    const apellidos = document.getElementById('apellidos').value;
    if (apellidos.trim() === '') {
        mostrarError('apellidos', 'Los apellidos son obligatorios');
        valido = false;
    } else {
        limpiarError('apellidos');
    }

    // Validación para el teléfono
    const telefono = document.getElementById('telefono').value;
    const regexTelefono = /^[0-9]{9}$/;
    if (!regexTelefono.test(telefono)) {
        mostrarError('telefono', 'El teléfono no es válido (debe tener 9 dígitos)');
        valido = false;
    } else {
        limpiarError('telefono');
    }

```

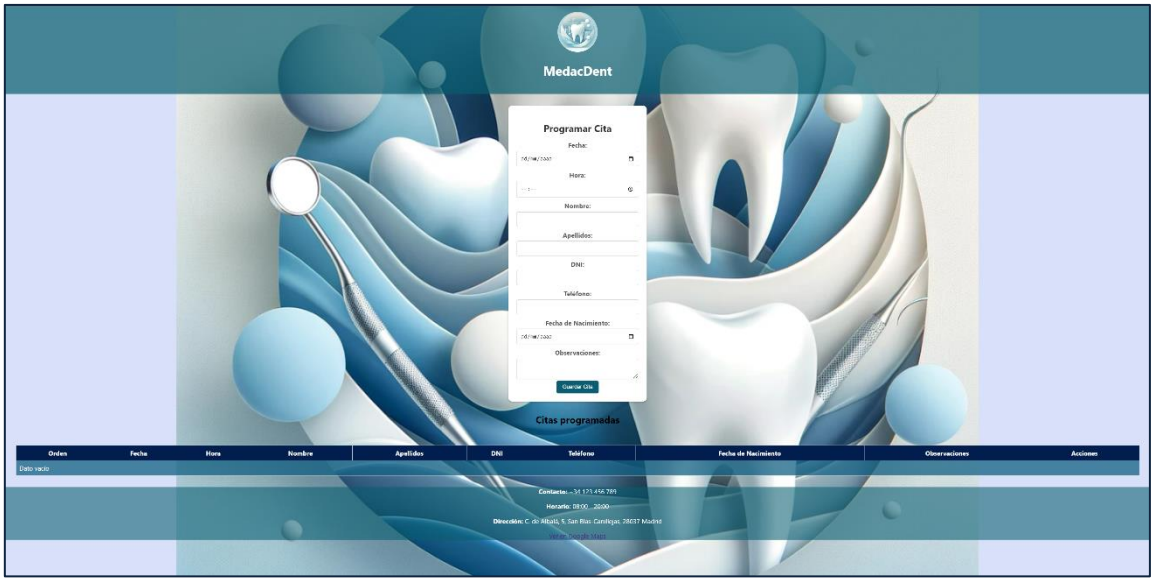
```
    // Validación para la fecha: asegurándose de que no sea en el pasado
    const fechaCita = document.getElementById('fecha').value;
    const fechaActual = new Date();
    // Establecer tiempo a 00:00:00 para comparar solo la fecha
    fechaActual.setHours(0, 0, 0, 0);
    // Convertir la fecha del input a un objeto Date para comparación
    const fechaSeleccionada = new Date(fechaCita);

    if (!fechaCita) {
        mostrarError('fecha', 'La fecha es obligatoria');
        valido = false;
    } else if (fechaSeleccionada < fechaActual) {
        mostrarError('fecha', 'La fecha de la cita no puede ser en el pasado');
        valido = false;
    } else {
        limpiarError('fecha');
    }

    // Validación para la hora
    const hora = document.getElementById('hora').value;
    if (!hora) {
        mostrarError('hora', 'La hora es obligatoria');
        valido = false;
    } else {
        limpiarError('hora');
    }

    // Validación para la fecha de nacimiento
    const fechaNacimiento = document.getElementById('fechaNacimiento').value;
    if (!fechaNacimiento) {
        mostrarError('fechaNacimiento', 'La fecha de nacimiento es obligatoria');
        valido = false;
    } else {
        limpiarError('fechaNacimiento');
    }

    return valido;
}

// Función para mostrar un mensaje de error cerca de un campo del formulario.
function mostrarError(campo, mensaje) {
    const elemento = document.getElementById(campo);
    const divError = document.createElement('div');
    divError.textContent = mensaje;
    divError.className = 'error';
    if (!elemento.nextSibling) {
        elemento.parentNode.insertBefore(divError, elemento.nextSibling);
```

```
    }
}

// Función para eliminar un mensaje de error existente.
function limpiarError(campo) {
    const elemento = document.getElementById(campo);
    if (elemento.nextSibling && elemento.nextSibling.className === 'error') {
        elemento.parentNode.removeChild(elemento.nextSibling);
    }
}

// función que limpie el formulario al enviar la cita
function limpiarFormulario() {
    document.getElementById('idCita').value = '';
    document.getElementById('fecha').value = '';
    document.getElementById('hora').value = '';
    document.getElementById('nombre').value = '';
    document.getElementById('apellidos').value = '';
    document.getElementById('dni').value = '';
    document.getElementById('telefono').value = '';
    document.getElementById('fechaNacimiento').value = '';
    document.getElementById('observaciones').value = '';
}

// Función para guardar una cita en el almacenamiento local del navegador.
function guardarCita() {
    const idCita = document.getElementById('idCita').value;
    const cita = {
        id: idCita || new Date().getTime(),
        fecha: document.getElementById('fecha').value,
        hora: document.getElementById('hora').value,
        nombre: document.getElementById('nombre').value,
        apellidos: document.getElementById('apellidos').value,
        dni: document.getElementById('dni').value,
        telefono: document.getElementById('telefono').value,
        fechaNacimiento: document.getElementById('fechaNacimiento').value,
        observaciones: document.getElementById('observaciones').value
    };

    let citas = JSON.parse(localStorage.getItem('citas')) || [];
    const indice = idCita ? citas.findIndex(c => c.id === idCita) : -1;

    if (indice > -1) {
        citas[indice] = cita; // Actualiza la cita existente
        alert("Cita modificada correctamente"); // Mensaje de confirmación
    } else {
        citas.push(cita); // Agrega una nueva cita
        alert("Cita guardada correctamente"); // Mensaje de confirmación
    }

    localStorage.setItem('citas', JSON.stringify(citas));
    cargarCitas();
    limpiarFormulario(); // Limpia el formulario después de guardar la cita
}

// Función para cargar los datos de una cita en el formulario para su edición.
function editarCita(idCita) {
    const citas = JSON.parse(localStorage.getItem('citas')) || [];
    const citaAEditar = citas.find(cita => cita.id === idCita);

    if (citaAEditar) {
        document.getElementById('fecha').value = citaAEditar.fecha;
        document.getElementById('hora').value = citaAEditar.hora;
        document.getElementById('nombre').value = citaAEditar.nombre;
        document.getElementById('apellidos').value = citaAEditar.apellidos;
        document.getElementById('dni').value = citaAEditar.dni;
        document.getElementById('telefono').value = citaAEditar.telefono;
        document.getElementById('fechaNacimiento').value = citaAEditar.fechaNacimiento;
        document.getElementById('observaciones').value = citaAEditar.observaciones;

        // Guardar el ID de la cita en un campo oculto
        document.getElementById('idCita').value = idCita;
    }
}


// Función para eliminar una cita del almacenamiento local.
function eliminarCita(idCita) {
    let citas = JSON.parse(localStorage.getItem('citas')) || [];
    citas = citas.filter(cita => cita.id !== idCita);
    localStorage.setItem('citas', JSON.stringify(citas));
    cargarCitas();
    alert("Cita eliminada correctamente"); // Mensaje de confirmación
}
```

2. Ejecución

A continuación, explicaré el proceso de ejecución de la aplicación web detallando las funciones que la definen:

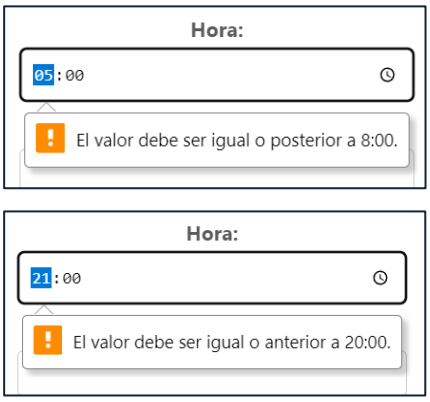
1) **Interfaz web**: Aquí se muestra el diseño de la interfaz web aproximadamente, ya que para realizar la captura he tenido que ampliar el zoom del navegador, dañando así el diseño.



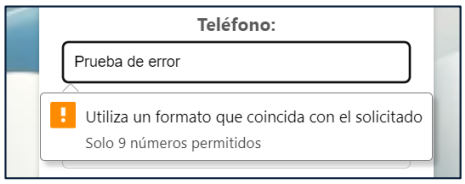
2) **Restricciones en el formulario**: El formulario presenta diferentes restricciones:

- **Restricción fecha**: No se puede elegir una fecha anterior a la actual:

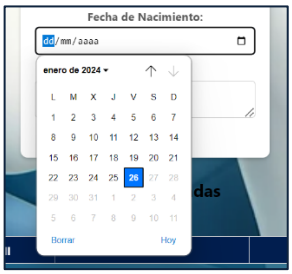
- **Restricción hora**: Ya que he establecido un horario de 08:00-20:00, no se puede elegir una hora de cita fuera de ese margen:



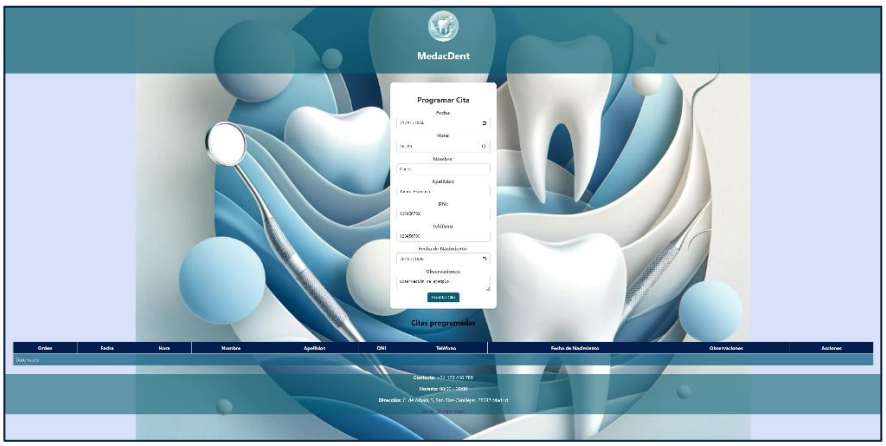
- **Restricción de campos**: Realicé una prueba aleatoria de error de los campos alfanuméricos en el campo "Teléfono".



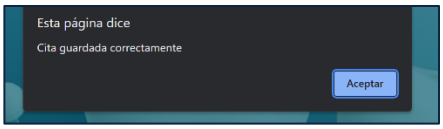
- **Restricción fecha de nacimiento**: No se puede seleccionar una posterior a la actual

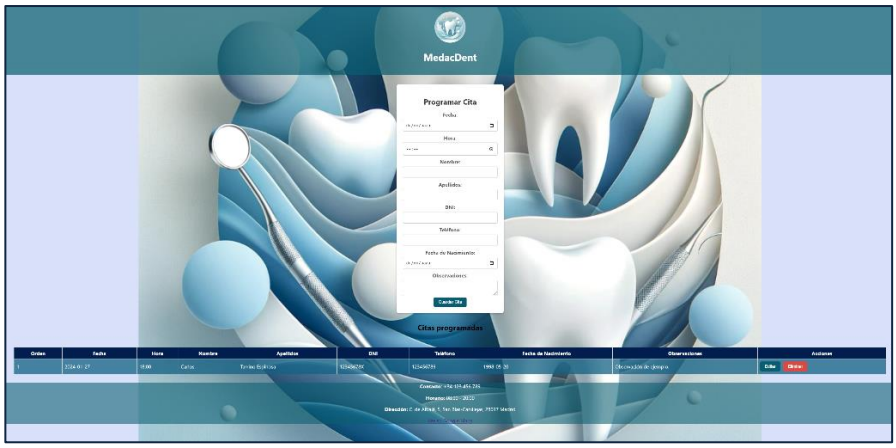


3) **Cita antes de guardar**: Se muestra el formulario rellenado de la cita antes de guardar:



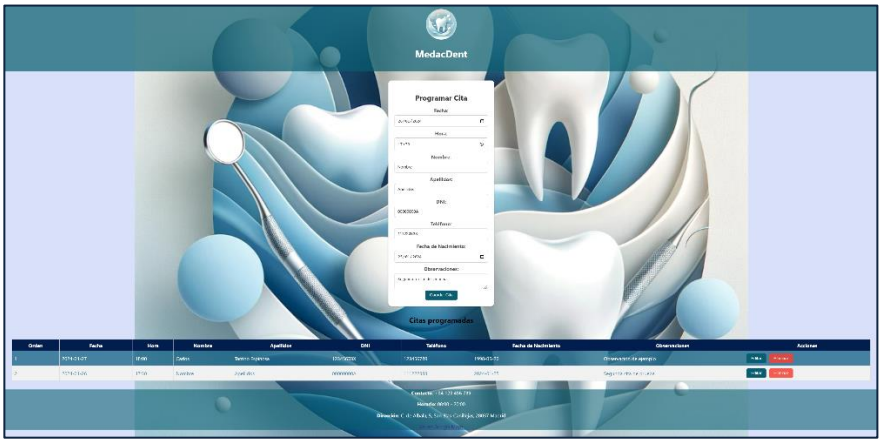
4) **Cita guardada:** Al pulsar en el botón "Guardar cita" se lanza un aviso de confirmación tras validar los campos, se guarda la cita en la tabla y se limpia el formulario para poder añadir más citas:



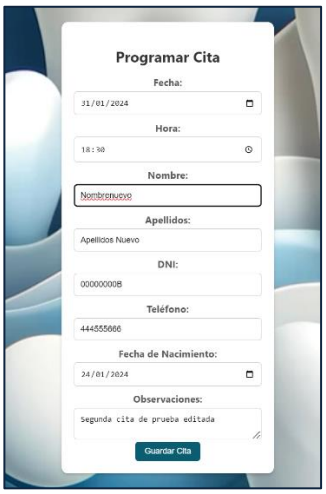


5) **Creación de segunda cita:** Creo una segunda cita para lucir el diseño de la tabla y mostrar el correcto funcionamiento del campo de orden:

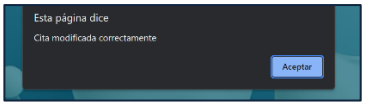
6) **Modificación de la segunda cita:** Al pulsar en el botón editar de la columna acciones de la segunda cita, se muestran de nuevo los datos en el formulario para poder ser modificados:



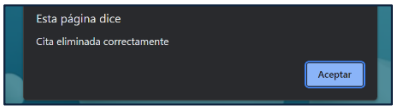
7) **Cambio de valores de la segunda cita:** Cambio los valores de la segunda cita:



8) **Edición exitosa de la segunda cita:** Al guardar con los nuevos valores, aparece un aviso de confirmación, se cambian los valores de la cita en la tabla y se limpia el formulario:



9) **Eliminación de citas:** Si pulsamos en el botón de eliminar de ambas citas, aparece un aviso y estas se eliminan, volviendo al estado inicial de la tabla:

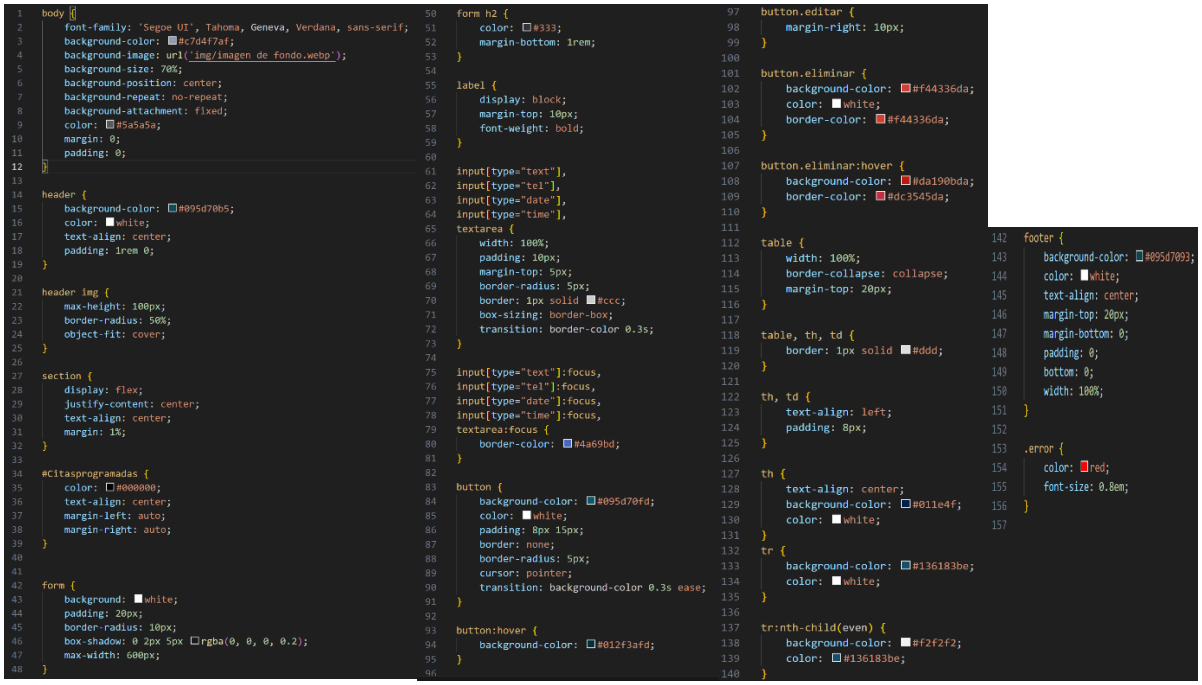


3. Problemas

Aunque no haya sido un desarrollo sencillo y he tenido que dedicar mucho tiempo y esfuerzo para solventar algunos errores en la funcionalidad de la aplicación, no ha habido ninguna funcionalidad que no haya podido realizar.

4. Anexos

A continuación, agrego también la hoja de estilos externa:



```css
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #c7d4f7af;
    background-image: url('img/imagen de fondo.webp');
    background-size: 70%;
    background-position: center;
    background-repeat: no-repeat;
    background-attachment: fixed;
    color: #5a5a5a;
    margin: 0;
    padding: 0;
}

header {
    background-color: #095d70b5;
    color: white;
    text-align: center;
    padding: 1rem 0;
}

header img {
    max-height: 100px;
    border-radius: 50%;
    object-fit: cover;
}

section {
    display: flex;
    justify-content: center;
    text-align: center;
    margin: 1%;
}

#Citasprogramadas {
    color: #000000;
    text-align: center;
    margin-left: auto;
    margin-right: auto;
}


form {
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    max-width: 600px;
}

form h2 {
    color: #333;
    margin-bottom: 1rem;
}

label {
    display: block;
    margin-top: 10px;
    font-weight: bold;
}

input[type="text"],
input[type="tel"],
input[type="date"],
input[type="time"],
textarea {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border-radius: 5px;
    border: 1px solid #ccc;
    box-sizing: border-box;
    transition: border-color 0.3s;
}

input[type="text"]:focus,
input[type="tel"]:focus,
input[type="date"]:focus,
input[type="time"]:focus,
textarea:focus {
    border-color: #4a69bd;
}

button {
    background-color: #095d70fd;
    color: white;
    padding: 8px 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #012f3afd;
}

button.editar {
    margin-right: 10px;
}

button.eliminar {
    background-color: #f44336da;
    color: white;
    border-color: #f44336da;
}

button.eliminar:hover {
    background-color: #da190bda;
    border-color: #dc3545da;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    text-align: left;
    padding: 8px;
}

th {
    text-align: center;
    background-color: #011e4f;
    color: white;
}
tr {
    background-color: #136183be;
    color: white;
}

tr:nth-child(even) {
    background-color: #f2f2f2;
    color: #136183be;
}

footer {
    background-color: #095d7093;
    color: white;
    text-align: center;
    margin-top: 20px;
    margin-bottom: 0;
    padding: 0;
    bottom: 0;
    width: 100%;
}

.error {
    color: red;
    font-size: 0.8em;
}
```