Portafolio de Evidencias enero-mayo 2025

Tópicos Avanzados de Programación

RAMON ORTEGA LOPEZ

Contenido

[Semana 1 – Lista de Software 4](#_Toc191536962)

[Primer Proyecto 4](#_Toc191536963)

[Sesión 2 Configuraciones: 5](#_Toc191536964)

[Agenda primera Unidad: 6](#_Toc191536965)

[**Sintaxis PHP** 6](#_Toc191536966)

[Sensibilidad a mayúsculas y minúsculas en PHP 6](#_Toc191536967)

[Comentarios en PHP 9](#_Toc191536968)

[Variables PHP 10](#_Toc191536969)

[Semana 2 Operación CRUD usando PHP y MySQLi 11](#_Toc191536970)

[**Paso 1 – Crear una base de datos** 11](#_Toc191536971)

[**Archivo Index.php** 15](#_Toc191536972)

[**Archivo conexión.php** 17](#_Toc191536973)

[**Archivo Mostrar** 18](#_Toc191536974)

[**Archivo editar** 23](#_Toc191536975)

[**Archivo eliminar** 29](#_Toc191536976)

[Semana 3 Registro y sesión 36](#_Toc191536977)

[**Archivo sesión** 36](#_Toc191536978)

[**Archivo registro** 38](#_Toc191536979)

[**Semana 4 index.php y salud.php s conectados al Servidor** 41](#_Toc191536980)

[**Archivo index.php** 41](#_Toc191536981)

Evaluación Diagnostica

# Semana 1 – Lista de Software

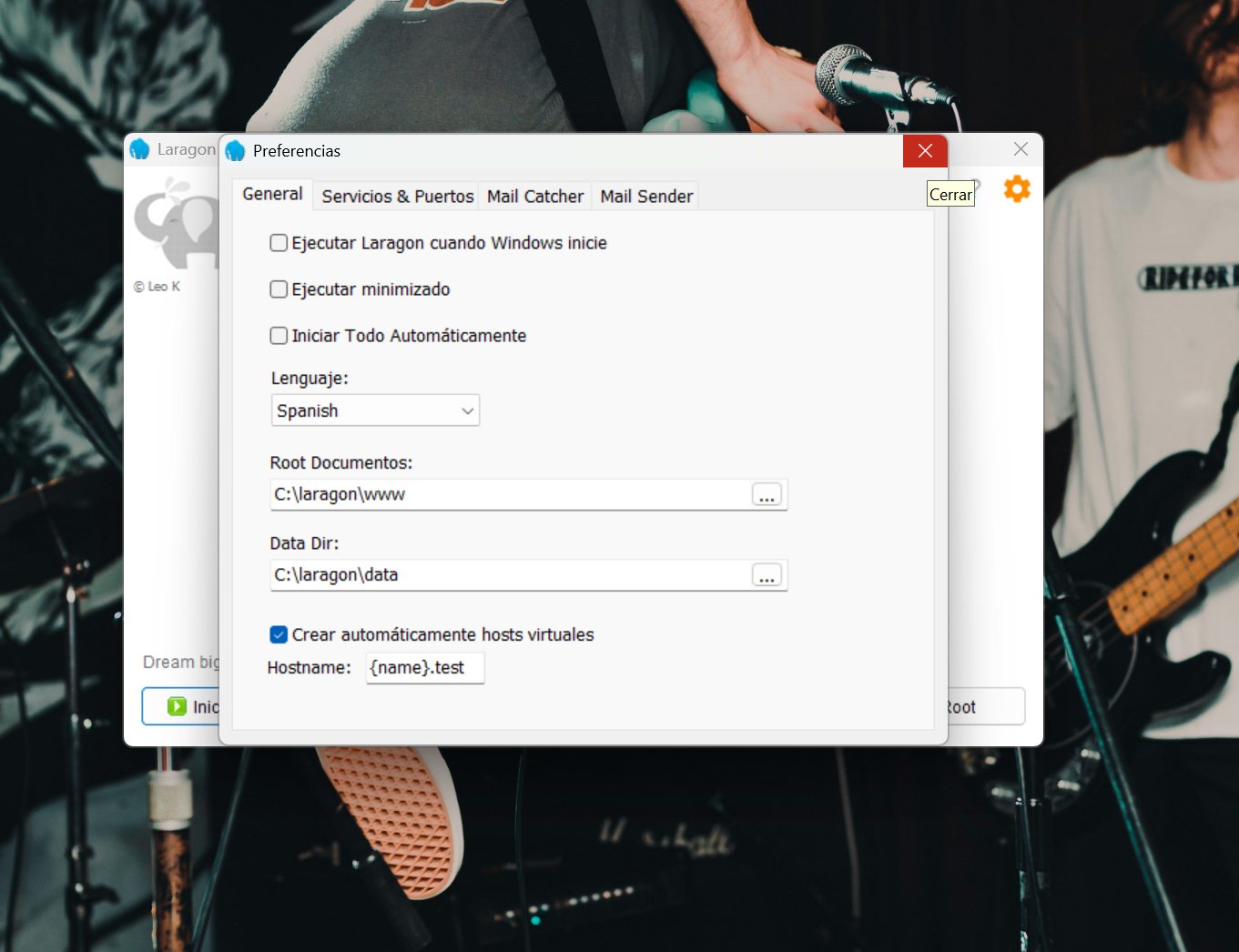
1. Laragon: Laragon es un entorno de desarrollo local (local development environment) que permite a los desarrolladores crear, gestionar y probar aplicaciones web en sus computadoras personales. Es una herramienta que simplifica la configuración de servidores web, bases de datos y otros componentes necesarios para el desarrollo de aplicaciones.
2. Extensiones de VS Code: **Code Runner, Live Server, PHP Debug y PHP IntelliSense**

## Primer Proyecto

Que es PHP: PHP (Hypertext Preprocessor) es un lenguaje de programación de código abierto ampliamente utilizado para el desarrollo web. Es especialmente adecuado para la creación de sitios web dinámicos y aplicaciones web, ya que puede ser incrustado directamente en el código HTML. PHP se ejecuta en el servidor, lo que significa que el código PHP se procesa en el servidor y el resultado (generalmente HTML) se envía al navegador del cliente.

Ejemplo:

## Sesión 2 Configuraciones:



La URL de nuestras aplicaciones siempre será: localhost/tópicos/semanaCorriente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

## Agenda primera Unidad:

¿Qué puede hacer PHP?

* PHP puede generar contenido de página dinámico
* PHP puede crear, abrir, leer, escribir, eliminar y cerrar archivos en el servidor.
* PHP puede recopilar datos de formularios
* PHP puede enviar y recibir cookies
* PHP puede agregar, eliminar y modificar datos en su base de datos
* PHP se puede utilizar para controlar el acceso de los usuarios
* PHP puede cifrar datos

# **Sintaxis PHP**

La extensión de archivo predeterminada para los archivos PHP es " .php".

Un archivo PHP normalmente contiene etiquetas HTML y algún código de script PHP.

A continuación, tenemos un ejemplo de un archivo PHP simple, con un script PHP que utiliza una función PHP incorporada " echo" para generar el texto "¡Hola mundo!" en una página web:

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

**<?php**

echo "Hello World!";

**?>**

</body>

</html>

## Sensibilidad a mayúsculas y minúsculas en PHP

En PHP, las palabras clave (por ejemplo if, else, while, echo, etc.), las clases, funciones y funciones definidas por el usuario no distinguen entre mayúsculas y minúsculas.

Ejemplo

**<?php**

ECHO "Hello World!<br>";

echo "Hello World!<br>";

EcHo "Hello World!<br>";

**?>**

**Ejemplo 2 Sensibilidad con variables**

**<?**php  
$color = "red";

echo "My car is " . $color . "<br>";

echo "My house is " . $COLOR . "<br>";

echo "My boat is " . $coLOR . "<br>";

**?>**

**Ejercicios**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Sitio web

Descripción generada automáticamente

## Comentarios en PHP

Un comentario en el código PHP es una línea que no se ejecuta como parte del programa. Su único propósito es que la lea alguien que esté mirando el código.

Los comentarios se pueden utilizar para:

* Deja que otros entiendan tu código
* Recuerda lo que hiciste: la mayoría de los programadores han tenido que volver a su trabajo un año o dos después y tener que volver a pensar en lo que hicieron. Los comentarios pueden recordarte lo que estabas pensando cuando escribiste el código.
* Omite algunas partes de tu código
* // This is a single-line comment
* # This is also a single-line comment
* /\* This is a
* multi-line comment \*/
* <h2>Comentarios en PHP</h2>
* <ol>
* <li>Comentario de una sola linea</li>
* <?php
* //Este es un comentario de una sola linea
* echo "Este es un comentario de una sola linea";
* ?>
* <li>Comentario de una sola linea</li>
* <?php
* # Este es un comentario de una sola linea
* echo "Este es un comentario de una sola linea";
* ?>
* <li>Comentario de varias lineas</li>
* <?php
* /\*Este es un comentario
* de varias lineas\*/
* echo "Este es un comentario de varias lineas";
* ?>
* </ol>

## Variables PHP

Las variables son "contenedores" para almacenar información.

Reglas para las variables PHP:

* Una variable comienza con el $signo, seguido del nombre de la variable.
* Un nombre de variable debe comenzar con una letra o el carácter de guión bajo.
* Un nombre de variable no puede comenzar con un número
* Un nombre de variable solo puede contener caracteres alfanuméricos y guiones bajos (Az, 0-9 y \_)
* Los nombres de las variables distinguen entre mayúsculas y minúsculas ( $agey $AGEson dos variables diferentes)

# Semana 2 Operación CRUD usando PHP y MySQLi

## **Paso 1 – Crear una base de datos**

Abra el navegador, escriba http://localhost/phpmyadmin y cree una base de datos llamada "phpcrud". Después de crear la base de datos, ejecute el script SQL o importe el archivo SQL.

Estructura de la tabla **datos**[de MySQL](https://phpgurukul.com/how-to-create-database-and-tables-in-mysql/)

**USE** demo1;

**CREATE** **TABLE** datos (

id **BIGINT** **AUTO\_INCREMENT** **PRIMARY** **KEY**,

nombre **VARCHAR**(100),

edad **INT**,

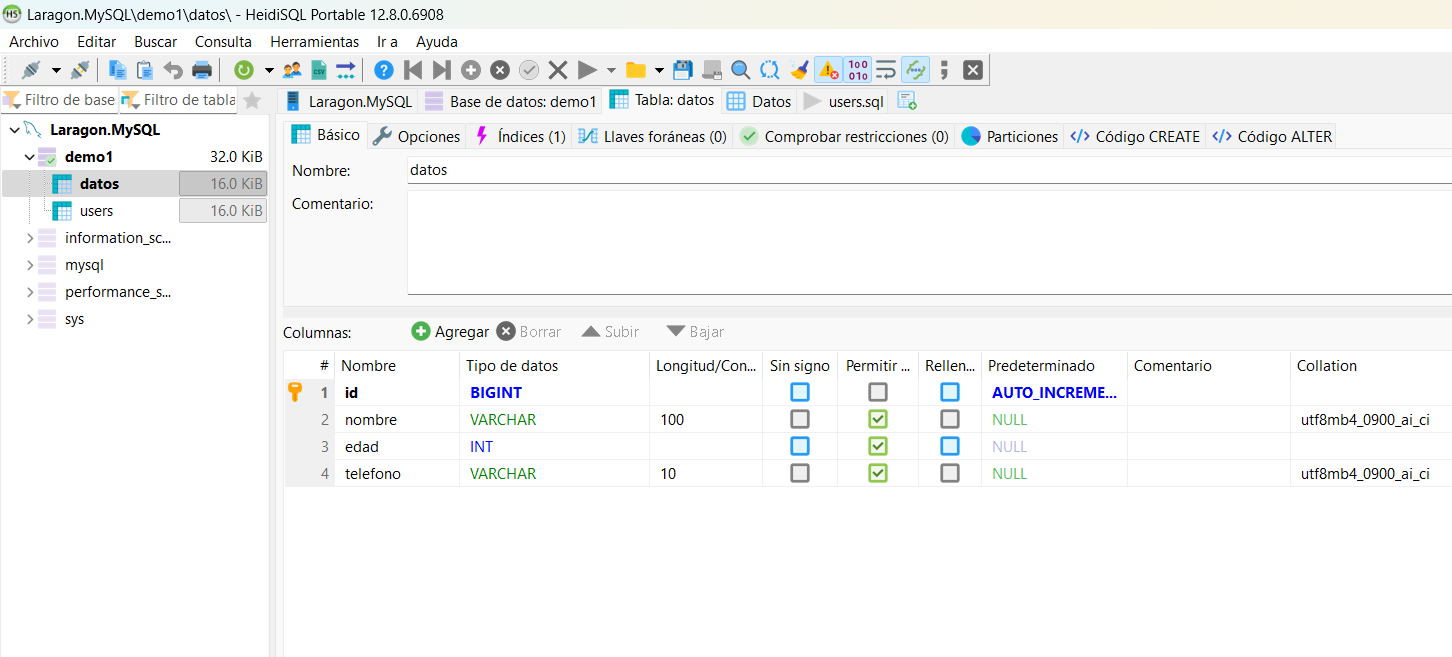
telefono **VARCHAR**(10)

);

Este código SQL tiene dos partes principales:

1. **USE demo1;** Selecciona la base de datos llamada demo1 para trabajar en ella. Si la base de datos no existe, primero debe crearse manualmente en phpMyAdmin o mediante un comando SQL (CREATE DATABASE demo1;).
2. **CREATE TABLE datos (...);** Crea una tabla llamada datos dentro de la base de datos demo1. La tabla tiene cuatro columnas:
   * **id**: Es un campo de tipo BIGINT que se autoincrementa (AUTO\_INCREMENT) y actúa como clave primaria (PRIMARY KEY). Esto significa que cada registro tendrá un identificador único generado automáticamente.
   * **nombre**: Almacena el nombre de una persona con un máximo de 100 caracteres (VARCHAR (100)).
   * **edad**: Almacena la edad como un número entero (INT).
   * **telefono**: Almacena un número de teléfono como una cadena de texto con un máximo de 10 caracteres (VARCHAR (10)).

Esta estructura de tabla es esencial para almacenar los datos que se ingresarán a través del formulario HTML.

****

**Paso 2: Crear un archivo de conexión a la base de datos (connection.php)**

<?php

//Parametros del servidor local

$localhost = "localhost";

$user = "root";

$password = "";

$db = "demo";

//Variable de conexion al servidor

$conexion = mysqli\_connect($localhost, $user, $password, $db);

?>

Este archivo PHP es fundamental para establecer una conexión entre la aplicación web y la base de datos MySQL. Aquí se definen los parámetros de conexión:

* **$localhost**: Especifica la dirección del servidor de la base de datos. En este caso, es localhost, lo que significa que la base de datos está en el mismo servidor donde se ejecuta el código PHP.
* **$user**: Define el nombre de usuario para acceder a la base de datos. En este caso, es root, que es el usuario predeterminado en muchos entornos de desarrollo local.
* **$password**: Es la contraseña del usuario. En este ejemplo, está vacía (""), lo que es común en entornos de desarrollo local, pero en producción siempre debe usarse una contraseña segura.
* **$db**: Especifica el nombre de la base de datos a la que se conectará (demo).

La función **mysqli\_connect** se utiliza para crear una conexión a la base de datos. Si la conexión es exitosa, la variable $conexion almacena el objeto de conexión, que se utilizará en otros archivos PHP para realizar consultas a la base de datos.

**Paso 3 – Crear un**[**formulario HTML**](https://phpgurukul.com/html-forms/)**para la inserción de datos**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Semana 2 Topicos Avanzados de Programación</title>

</head>

<body>

    <h1>Diseño de Operaciones CRUD en PHP y MySQli</h1>

    <h2>Modulo 1: Insertar Datos</h2>

    <h3>Formulario HTML a PHP</h3>

    <form method = "post">

        <p>Nombre: <input type="text" name="nombre"></p>

        <p>Edad: <input type="text" name="edad"></p>

        <p>Telefono: <input type="text" name="tel"></p>

        <p><input type="submit" value="Enviar"></p>

    </form>

<!--Importando Conexion a base de datos-->

    <?php

        Include 'conexion.php';

        if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

            $name = $\_POST['nombre'];

           $age = $\_POST['edad'];

           $phone = $\_POST['tel'];

           //Metodo Insert

           $query = mysqli\_query($conexion, "INSERT INTO datos (nombre, edad, telefono) VALUES ('$name', '$age', '$phone')");

        }

    ?>

</body>

</html>

Este archivo combina HTML y PHP para crear un formulario que permite a los usuarios ingresar datos y enviarlos a la base de datos. Aquí se detalla cada parte:

1. **Formulario HTML**:

* El formulario tiene tres campos de entrada: nombre, edad y tel (teléfono).
* El método de envío es POST, lo que significa que los datos se enviarán al servidor de manera segura (no visible en la URL).
* Cuando el usuario hace clic en el botón "Enviar", los datos se envían al servidor.

1. **Procesamiento en PHP**:

* Se incluye el archivo conexion.php para utilizar la conexión a la base de datos.
* Si el método de solicitud es POST (es decir, si el formulario se ha enviado), los datos se capturan en variables PHP ($name, $age, $phone).
* Luego, se ejecuta una consulta SQL (INSERT INTO) para insertar los datos en la tabla datos.

Este archivo es la base para la operación **Crear (Create)** del CRUD.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

## **Archivo Index.php**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Semana 2 Topicos Avanzados de Programación</title>

</head>

<body>

    <h1>Diseño de Operaciones CRUD en PHP y MySQli</h1>

    <h2>Modulo 1: Insertar Datos</h2>

    <h3>Formulario HTML a PHP</h3>

    <form method = "post">

        <p>Nombre: <input type="text" name="nombre"></p>

        <p>Edad: <input type="text" name="edad"></p>

        <p>Telefono: <input type="text" name="tel"></p>

        <p><input type="submit" value="Enviar"></p>

    </form>

<!--Importando Conexion a base de datos-->

    <?php

        Include 'conexion.php';

        if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

            $name = $\_POST['nombre'];

           $age = $\_POST['edad'];

           $phone = $\_POST['tel'];

           //Metodo Insert

           $query = mysqli\_query($conexion, "INSERT INTO datos (nombre, edad, telefono) VALUES ('$name', '$age', '$phone')");

        }

    ?>

    <h3>Modulo Despliegue de datos <a href="mostrar.php">Aqui</a></h3>

</body>

Este archivo es similar al formulario HTML del Paso 3, pero incluye un enlace adicional para redirigir al usuario a otra página (mostrar.php), donde se visualizarán los datos almacenados.

**Propósito:**

* Además de permitir la inserción de datos, este archivo actúa como la página principal de la aplicación.
* El enlace **"Modulo Despliegue de datos"** redirige al usuario a mostrar.php, donde se listan todos los registros de la base de datos.
* Esto facilita la navegación entre las diferentes funcionalidades de la aplicación (insertar y ver datos).

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

## **Archivo conexión.php**

<?php

//Parametros del servidor local

$localhost = "localhost";

$user = "root";

$password = "";

$db = "demo1";

//Variable de conexion al servidor

$conexion = mysqli\_connect($localhost, $user, $password, $db);

?>

## **Archivo Mostrar**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

    <title>Topicos Avanzados de Programación - Modulo Mostrar</title>

</head>

<body class="container mt-4">

    <h2 class="text-center mb-4">Lista de Datos</h2>

    <div class="table-responsive">

        <table class="table table-bordered table-hover table-striped">

            <thead class="table-dark">

                <tr>

                    <th>ID</th>

                    <th>Nombre</th>

                    <th>Edad</th>

                    <th>Teléfono</th>

                    <th>Editar</th>

                    <th>Eliminar</th>

                </tr>

            </thead>

            <tbody>

                <?php

                include 'conexion.php';

                $query = "SELECT \* FROM datos";

                $resultado = mysqli\_query($conexion, $query);

                if (mysqli\_num\_rows($resultado) > 0) {

                    while ($fila = mysqli\_fetch\_assoc($resultado)) {

                        echo "<tr>";

                        echo "<td>" . $fila['id'] . "</td>";

                        echo "<td>" . $fila['nombre'] . "</td>";

                        echo "<td>" . $fila['edad'] . "</td>";

                        echo "<td>" . $fila['telefono'] . "</td>";

                        echo "<td class='text-center'>

                                <a href='editar.php?id=" . $fila['id'] . "' class='btn btn-warning btn-sm'>

                                    <i class='bi bi-pencil-square'></i> Editar

                                </a>

                              </td>";

                        echo "<td class='text-center'>

                                <a href='eliminar.php?id=" . $fila['id'] . "' class='btn btn-danger btn-sm' onclick='return confirm(\"¿Estás seguro de eliminar este registro?\");'>

                                    <i class='bi bi-trash'></i> Eliminar

                                </a>

                              </td>";

                        echo "</tr>";

                    }

                } else {

                    echo "<tr><td colspan='6' class='text-center'>No hay datos disponibles</td></tr>";

                }

                ?>

            </tbody>

        </table>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css" rel="stylesheet">

</body>

</html>

Este archivo es responsable de la operación **Leer (Read)** del CRUD. Su función principal es mostrar todos los registros almacenados en la tabla datos de la base de datos. Aquí se detalla su funcionamiento:

1. **Conexión a la base de datos**:
   * Se incluye el archivo conexion.php para establecer la conexión con la base de datos.
2. **Consulta SQL**:
   * Se ejecuta una consulta SQL (SELECT \* FROM datos) para obtener todos los registros de la tabla datos.
3. **Mostrar los datos en una tabla**:
   * Si hay registros (mysqli\_num\_rows($resultado) > 0), se recorre cada fila con un bucle while.
   * Los datos de cada registro se muestran en una tabla HTML, con columnas para ID, Nombre, Edad, Teléfono, y opciones para **Editar** y **Eliminar**.
4. **Botones de acción**:
   * **Editar**: Redirige al usuario a editar.php con el id del registro que se desea modificar.
   * **Eliminar**: Redirige al usuario a eliminar.php con el id del registro que se desea eliminar. Se muestra una confirmación antes de proceder con la eliminación.
5. **Mensaje si no hay datos**:
   * Si no hay registros en la base de datos, se muestra un mensaje en la tabla: "No hay datos disponibles".
6. **Estilos con Bootstrap**:
   * Se utiliza Bootstrap para dar estilo a la tabla, los botones y otros elementos de la interfaz.
   * La clase table-responsive hace que la tabla sea adaptable a pantallas pequeñas.
   * Los botones de editar y eliminar tienen estilos específicos (btn-warning, btn-danger) para diferenciar sus acciones.

**Integración en la aplicación CRUD**

El archivo mostrar.php es una parte clave de la aplicación, ya que permite:

1. **Visualizar los datos**: Los usuarios pueden ver todos los registros almacenados en la base de datos.
2. **Navegar a otras funcionalidades**: Desde esta página, los usuarios pueden acceder a las opciones de **Editar** y **Eliminar** registros.
3. **Confirmar acciones**: Antes de eliminar un registro, se solicita confirmación al usuario para evitar errores.

**Ejemplo de flujo de uso**

1. El usuario ingresa datos a través del formulario en index.php.
2. Los datos se almacenan en la base de datos y el usuario es redirigido a mostrar.php.
3. En mostrar.php, el usuario ve todos los registros almacenados.
4. Si el usuario desea modificar un registro, hace clic en el botón **Editar**, lo que lo lleva a editar.php.
5. Si el usuario desea eliminar un registro, hace clic en el botón **Eliminar**, lo que lo lleva a eliminar.php (previo a una confirmación).

**Mejoras posibles**

1. **Paginación**: Si hay muchos registros, se puede implementar paginación para mejorar la experiencia del usuario.
2. **Búsqueda**: Agregar un campo de búsqueda para filtrar registros por nombre, edad o teléfono.
3. **Validación de datos**: Asegurarse de que los datos ingresados cumplan con ciertos criterios (por ejemplo, que la edad sea un número válido).
4. **Seguridad**: Implementar medidas de seguridad adicionales, como la validación de entradas y la protección contra inyecciones SQL.

El archivo mostrar.php es esencial para la operación **Leer (Read)** del CRUD. Proporciona una interfaz clara y funcional para visualizar los datos almacenados en la base de datos, así como opciones para editar y eliminar registros. Junto con los otros archivos (index.php, editar.php, eliminar.php), forma una aplicación CRUD completa y eficiente.

Tabla

El contenido generado por IA puede ser incorrecto.

## **Archivo editar**

<?php

include 'conexion.php'; // Conexión a la base de datos

// Verifica si se recibió un ID válido

if (isset($\_GET['id']) && is\_numeric($\_GET['id'])) {

    //variablre local

    $id = $\_GET['id'];

    // Consulta para obtener los datos del usuario

    $query = "SELECT \* FROM datos WHERE id = $id";

    $resultado = mysqli\_query($conexion, $query);

    // Verifica si el registro existe

    if ($fila = mysqli\_fetch\_assoc($resultado)) {

        $nombre = $fila['nombre'];

        $edad = $fila['edad'];

        $telefono = $fila['telefono'];

    } else {

        //echo "<script>alert('Registro no encontrado.'); window.location='mostrar.php';</script>";

        exit();

    }

} else {

    echo "<script>alert('ID inválido.'); window.location='error.html';</script>";

    exit();

}

// Procesar la actualización cuando se envía el formulario

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    $name = $\_POST['nombre'];

    $age = $\_POST['edad'];

    $phone = $\_POST['tel'];

    // Actualizar el registro

    $query\_update = "UPDATE datos SET nombre='$name', edad=$age, telefono='$phone' WHERE id=$id";

    if (mysqli\_query($conexion, $query\_update)) {

echo '

<div class="modal fade" id="alertModal" tabindex="-1" aria-hidden="true">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header bg-success text-white">

                <h5 class="modal-title">¡Éxito!</h5>

            </div>

            <div class="modal-body">

                <p>Registro actualizado correctamente.</p>

            </div>

            <div class="modal-footer">

                <button id="redirectBtn" class="btn btn-success">Aceptar</button>

            </div>

        </div>

    </div>

</div>

<script>

    // Esperar a que el DOM cargue completamente

    document.addEventListener("DOMContentLoaded", function() {

        var alertModal = new bootstrap.Modal(document.getElementById("alertModal"));

        alertModal.show();

        // Redirigir al cerrar el modal

        document.getElementById("redirectBtn").addEventListener("click", function() {

            window.location.href = "mostrar.php";

        });

    });

</script>

';

    } else {

        echo '

<div class="modal fade" id="alertModal" tabindex="-1" aria-hidden="true">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header bg-warning text-white">

                <h5 class="modal-title">¡Error!</h5>

            </div>

            <div class="modal-body">

                <p>Error al Actualizar Registro. Entrada no Valida.</p>

            </div>

            <div class="modal-footer">

                <button id="redirectBtn" class="btn btn-danger">Terminar</button>

            </div>

        </div>

    </div>

</div>

<script>

    // Esperar a que el DOM cargue completamente

    document.addEventListener("DOMContentLoaded", function() {

        var alertModal = new bootstrap.Modal(document.getElementById("alertModal"));

        alertModal.show();

        // Redirigir al cerrar el modal

        document.getElementById("redirectBtn").addEventListener("click", function() {

            window.location.href = "index.php";

        });

    });

</script>';

    }

}

?>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

    <title>Editar Registro</title>

</head>

<body class="container mt-4">

    <h2 class="text-center mb-4">Editar Registro</h2>

    <form method="POST" class="card p-4">

        <div class="mb-3">

            <label class="form-label">Nombre:</label>

            <input type="text" name="nombre" class="form-control" value="<?php echo htmlspecialchars($nombre); ?>" required>

        </div>

        <div class="mb-3">

            <label class="form-label">Edad:</label>

            <input type="number" name="edad" class="form-control" value="<?php echo $edad; ?>" required>

        </div>

        <div class="mb-3">

            <label class="form-label">Teléfono:</label>

            <input type="text" name="tel" class="form-control" value="<?php echo htmlspecialchars($telefono); ?>" required>

        </div>

        <div class="text-center">

            <button type="submit" class="btn btn-success">Actualizar</button>

            <a href="mostrar.php" class="btn btn-secondary">Cancelar</a>

        </div>

    </form>

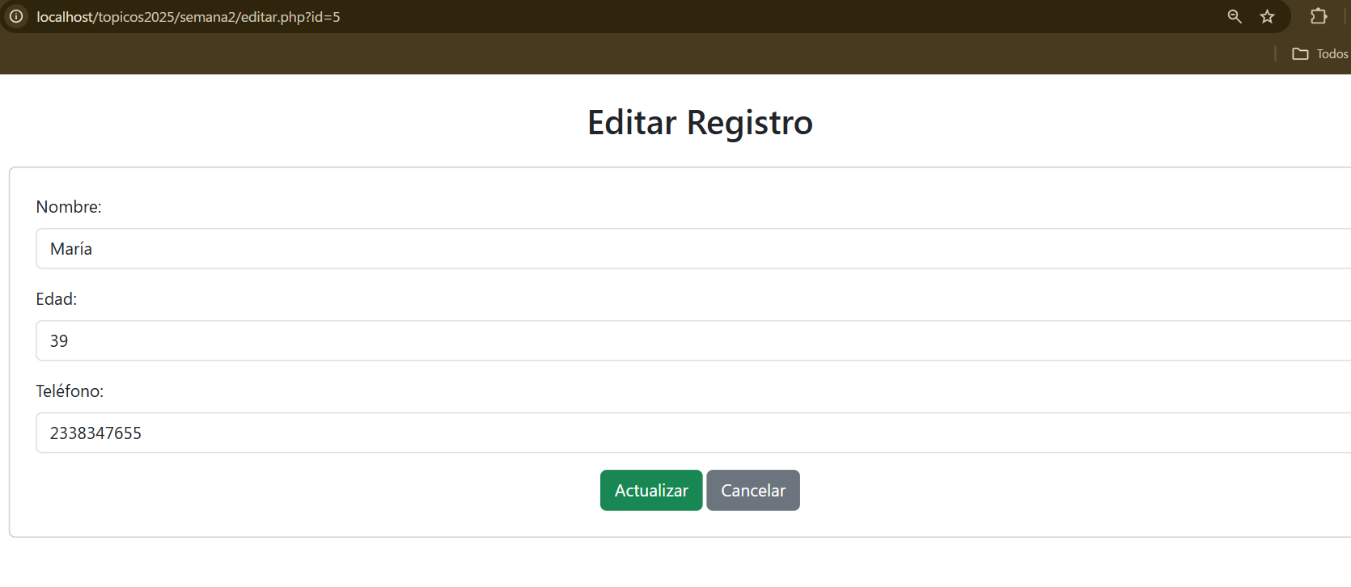
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

Este archivo permite la operación **Actualizar (Update)** del CRUD. Aquí se detalla su funcionamiento:

1. **Obtener el ID del registro**:
   * El ID del registro a editar se pasa a través de la URL (editar.php?id=1).
   * Se verifica que el ID sea válido y numérico.
2. **Consulta para obtener los datos**:
   * Se realiza una consulta SQL (SELECT \* FROM datos WHERE id = $id) para obtener los datos del registro específico.
   * Los datos se almacenan en variables PHP ($nombre, $edad, $telefono).
3. **Formulario de edición**:
   * Se muestra un formulario prellenado con los datos actuales del registro.
   * El usuario puede modificar los campos y enviar el formulario.
4. **Procesamiento de la actualización**:
   * Cuando el formulario se envía (método POST), los nuevos datos se capturan en variables PHP.
   * Se ejecuta una consulta SQL (UPDATE datos SET ... WHERE id=$id) para actualizar el registro en la base de datos.
   * Si la actualización es exitosa, se muestra un mensaje de éxito y se redirige al usuario a mostrar.php.
   * Si hay un error, se muestra un mensaje de error y se redirige al usuario a index.php.



## **Archivo eliminar**

<?php

include 'conexion.php'; // Conexión a la base de datos

// Verifica si se recibió un ID válido

if (isset($\_GET['id']) && is\_numeric($\_GET['id'])) {

    //variablre local

    $id = $\_GET['id'];

    // Consulta para obtener los datos del usuario

    $query = "SELECT \* FROM datos WHERE id = $id";

    $resultado = mysqli\_query($conexion, $query);

    // Verifica si el registro existe

    if ($fila = mysqli\_fetch\_assoc($resultado)) {

        $nombre = $fila['nombre'];

        $edad = $fila['edad'];

        $telefono = $fila['telefono'];

    } else {

        //echo "<script>alert('Registro no encontrado.'); window.location='mostrar.php';</script>";

        exit();

    }

} else {

    echo "<script>alert('ID inválido.'); window.location='error.html';</script>";

    exit();

}

// Procesar la actualización cuando se envía el formulario

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

    // Actualizar el registro

    $query\_delete = "DELETE FROM datos WHERE id=$id";

    if (mysqli\_query($conexion, $query\_delete)) {

echo '

<div class="modal fade" id="alertModal" tabindex="-1" aria-hidden="true">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header bg-success text-white">

                <h5 class="modal-title">¡Éxito!</h5>

            </div>

            <div class="modal-body">

                <p>Registro eliminado correctamente.</p>

            </div>

            <div class="modal-footer">

                <button id="redirectBtn" class="btn btn-success">Aceptar</button>

            </div>

        </div>

    </div>

</div>

<script>

    // Esperar a que el DOM cargue completamente

    document.addEventListener("DOMContentLoaded", function() {

        var alertModal = new bootstrap.Modal(document.getElementById("alertModal"));

        alertModal.show();

        // Redirigir al cerrar el modal

        document.getElementById("redirectBtn").addEventListener("click", function() {

            window.location.href = "mostrar.php";

        });

    });

</script>

';

    } else {

        echo '

<div class="modal fade" id="alertModal" tabindex="-1" aria-hidden="true">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header bg-warning text-white">

                <h5 class="modal-title">¡Error!</h5>

            </div>

            <div class="modal-body">

                <p>Error al eliminar registro. Entrada no Valida.</p>

            </div>

            <div class="modal-footer">

                <button id="redirectBtn" class="btn btn-danger">Terminar</button>

            </div>

        </div>

    </div>

</div>

<script>

    // Esperar a que el DOM cargue completamente

    document.addEventListener("DOMContentLoaded", function() {

        var alertModal = new bootstrap.Modal(document.getElementById("alertModal"));

        alertModal.show();

        // Redirigir al cerrar el modal

        document.getElementById("redirectBtn").addEventListener("click", function() {

            window.location.href = "index.php";

        });

    });

</script>';

    }

}

?>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

    <title>Eliminar Registro</title>

</head>

<body class="container mt-4">

    <h2 class="text-center mb-4">Editar Registro</h2>

    <form method="POST" class="card p-4">

        <div class="mb-3">

            <label class="form-label">Nombre:</label><?php echo htmlspecialchars($nombre); ?>

        </div>

        <div class="mb-3">

            <label class="form-label">Edad:</label><?php echo htmlspecialchars($edad); ?>

        </div>

        <div class="mb-3">

            <label class="form-label">Teléfono:</label><?php echo htmlspecialchars($telefono); ?>

        </div>

        <div class="text-center">

            <button type="button" class="btn btn-danger">Eliminar</button>

            <a href="mostrar.php" class="btn btn-secondary">Cancelar</a>

        </div>

    </form>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

**Propósito**

Incluye el archivo **conexion.php**, que contiene la configuración para conectarse a la base de datos MySQL. Esto permite reutilizar la conexión en lugar de repetir el código.

* Verifica si se ha recibido un ID válido a través de la URL (eliminar.php?id=1).
* Si el ID es válido y numérico, se realiza una consulta SQL (SELECT \* FROM datos WHERE id = $id) para obtener los datos del registro que se desea eliminar.
* Si el registro existe, los datos se almacenan en variables PHP ($nombre, $edad, $telefono) para mostrarlos en el formulario.
* Si el ID no es válido o el registro no existe, se muestra un mensaje de error y se redirige al usuario
* Si el formulario se envía (método POST), se ejecuta una consulta SQL (DELETE FROM datos WHERE id=$id) para eliminar el registro de la base de datos.
* Si la eliminación es exitosa, se muestra un mensaje de éxito en un modal de Bootstrap y se redirige al usuario a mostrar.php.
* Si ocurre un error durante la eliminación, se muestra un mensaje de error y se redirige al usuario a index.php.
* Muestra un formulario con los datos del registro que se desea eliminar.
* El usuario puede confirmar la eliminación haciendo clic en el botón **Eliminar**.
* Si el usuario decide no eliminar el registro, puede hacer clic en **Cancelar** para volver a mostrar.php.

**Flujo de uso**

* El usuario hace clic en el botón **Eliminar** en mostrar.php.
* Se redirige al usuario a eliminar.php con el ID del registro que se desea eliminar.
* El usuario ve los datos del registro y confirma la eliminación.
* Si la eliminación es exitosa, se muestra un mensaje de éxito y el usuario es redirigido a mostrar.php.
* Si ocurre un error, se muestra un mensaje de error y el usuario es redirigido a index.php.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

# Semana 3 Registro y sesión

## **Archivo sesión**

<?php

session\_start();

?>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login</title>

</head>

<body>

    <h2>Iniciar Sesión</h2>

    <form action="login.php" method="POST">

        <label for="usuario">Usuario:</label>

        <input type="text" name="usuario" required>

        <br>

        <label for="password">Contraseña:</label>

        <input type="password" name="password" required>

        <br>

        <button type="submit">Ingresar</button>

        <a href="registro.php">Registrarse</a>

    </form>

    <?php

    if (isset($\_SESSION['error'])) {

        echo "<p style='color: red'>" . $\_SESSION['error'] . "</p>";

        unset($\_SESSION['error']);

    }

    ?>

</body>

</html>

Este archivo es un **formulario de inicio de sesión** que permite a los usuarios autenticarse en la aplicación web. Su función principal es verificar las credenciales del usuario (nombre de usuario y contraseña) y, si son correctas, permitirle acceder a las funcionalidades restringidas de la aplicación. Aquí se detalla su funcionamiento:

1. **Formulario de inicio de sesión**:
   * El formulario HTML solicita dos datos: el **nombre de usuario** y la **contraseña**.
   * Los campos son obligatorios (required), lo que garantiza que el usuario no pueda enviar el formulario vacío.
   * El formulario utiliza el método **POST** para enviar los datos de manera segura al servidor.
   * Incluye un enlace a registro.php para que los usuarios que no tienen una cuenta puedan registrarse.
2. **Manejo de errores**:
   * Si hay un error durante el proceso de autenticación (por ejemplo, credenciales incorrectas), se muestra un mensaje de error en rojo.
   * El mensaje de error se almacena en la variable de sesión $\_SESSION['error'] y se elimina después de mostrarse para evitar que persista en futuras cargas de la página.
3. **Integración con sesiones**:
   * El archivo utiliza **session\_start()** para iniciar o reanudar una sesión, lo que permite almacenar y recuperar datos del usuario durante su visita a la aplicación.
   * Las sesiones son esenciales para mantener el estado de autenticación del usuario mientras navega por la aplicación.

**Integración en la aplicación**

Este archivo es una parte clave del sistema de autenticación de la aplicación. Permite a los usuarios que ya tienen una cuenta iniciar sesión y acceder a funcionalidades restringidas. Junto con otros archivos como registro.php (para el registro de nuevos usuarios) y conexion.php (para la conexión a la base de datos), forma un sistema completo de gestión de usuarios.

## **Archivo registro**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Formulario de Registro Usuarios</title>

</head>

<body>

    <h1>Formulario de Registro Usuarios</h1>

    <form method="post">

        <label for="nombre">Nombre de Usuario:</label>

        <input type="text" name="nombre" id="nombre" required>

        <br>

        <label for="password">Contraseña:</label>

        <input type="password" name="password" id="password" required>

        <br>

        <input type="submit" value="Registrar">

        <a href="sesion.php">Iniciar Sesión</a>

    </form>

</body>

<?php

    include 'conexion.php';

    if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

        $nombre = $\_POST['nombre'];

        $password = $\_POST['password'];

        $pass = password\_hash($password, PASSWORD\_DEFAULT);

        echo $pass;

        $query = "INSERT INTO users (username, password) VALUES ('$nombre', '$pass')";

        mysqli\_query($conexion, $query);

        header('Location: index.php');

    }

?>

</html>

Este archivo es un **formulario de registro de usuarios** que permite a los usuarios crear una cuenta en la aplicación web. Su función principal es capturar el nombre de usuario y la contraseña, almacenarlos de manera segura en la base de datos y redirigir al usuario a la página principal (index.php) después del registro. Aquí se detalla su funcionamiento:

1. **Formulario de registro**:
   * El formulario HTML solicita dos datos: el **nombre de usuario** y la **contraseña**.
   * Los campos son obligatorios (required), lo que garantiza que el usuario no pueda enviar el formulario vacío.
   * El formulario utiliza el método **POST** para enviar los datos de manera segura al servidor.
2. **Procesamiento del registro**:o
   * Cuando el formulario se envía, el código PHP captura los datos ingresados (nombre y password).
   * La contraseña se cifra utilizando la función **password\_hash**, que genera un hash seguro para almacenar en la base de datos. Esto es fundamental para proteger la información sensible del usuario.
   * Luego, se ejecuta una consulta SQL (INSERT INTO) para guardar el nombre de usuario y la contraseña cifrada en la tabla users de la base de datos.
3. **Redirección**:
   * Después de registrar al usuario, se redirige automáticamente a la página principal (index.php) usando la función **header**.
4. **Enlace para iniciar sesión**:
   * El formulario incluye un enlace a sesion.php, que permite a los usuarios que ya tienen una cuenta iniciar sesión en lugar de registrarse.

**Integración**

Este archivo es parte esencial del sistema de autenticación del sitio. Permite a los nuevos usuarios crear una cuenta, lo que es el primer paso para acceder a funcionalidades restringidas o personalizadas. Junto con otros archivos como sesion.php (para iniciar sesión) y conexion.php (para la conexión a la base de datos), forma un sistema completo de gestión de usuarios.

# **Semana 4 index.php y salud.php s conectados al Servidor**

## **Archivo index.php**

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Ingresar Nombre</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

    <style>

        body {

*background-color*: #343541;

*color*: #fff;

*display*: flex;

*justify-content*: center;

*align-items*: center;

*height*: 100vh;

        }

        .container {

*background-color*: #40414F;

*padding*: 2rem;

*border-radius*: 12px;

*box-shadow*: 0px 4px 10px rgba(0, 0, 0, 0.2);

*text-align*: center;

        }

        .form-control {

*background-color*: #565869;

*color*: #fff;

*border*: none;

        }

        .btn-custom {

*background-color*: #10A37F;

*color*: #fff;

*border-radius*: 8px;

*padding*: 10px 20px;

        }

        .btn-custom:hover {

*background-color*: #0d8a6b;

        }

    </style>

</head>

<body>

    <div class="container">

        <h2>Bienvenido</h2>

        <p>Introduce tu nombre para continuar</p>

        <form action="saludo.php" method="POST">

            <div class="mb-3">

                <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Tu nombre aquí" required>

            </div>

            <button type="submit" class="btn btn-custom">Aceptar</button>

        </form>

    </div>

</body>

</html>

Este archivo es una **página de bienvenida** que solicita al usuario que ingrese su nombre para continuar. Su función principal es capturar el nombre del usuario y enviarlo a otra página (saludo.php) para procesarlo y mostrar un mensaje personalizado. Aquí se detalla su funcionamiento:

1. **Interfaz de usuario**:
   * La página tiene un diseño moderno y atractivo, con un fondo oscuro y colores personalizados que mejoran la experiencia del usuario.
   * El formulario está centrado en la pantalla y tiene un estilo minimalista, lo que lo hace fácil de usar.
   * El campo de entrada (input) solicita al usuario que ingrese su nombre, y el botón "Aceptar" envía el formulario.
2. **Estilos personalizados**:
   * Se utiliza **Bootstrap** para dar estilo a los elementos del formulario, como el campo de entrada y el botón.
   * Los estilos personalizados en el <style> definen un tema oscuro con colores específicos para el fondo, el texto y los botones, lo que le da a la página un aspecto profesional y coherente.
3. **Envío de datos**:
   * El formulario utiliza el método **POST** para enviar el nombre del usuario a la página saludo.php.
   * Esto permite que la página saludo.php procese el nombre y muestre un mensaje personalizado, como un saludo.
4. **Integración con el servidor**:
   * La página está diseñada para funcionar en un servidor web, donde el formulario puede enviar datos y recibir respuestas dinámicas.
   * La conexión entre esta página y saludo.php permite crear una experiencia interactiva para el usuario.