

Práctica 2: Formularios

Objetivo:

- Crear un formulario HTML para capturar datos de usuarios.
- Procesar los datos del formulario usando PHP.
- Almacenar los datos en una base de datos MySQL.
- Mostrar los datos almacenados en una página web.

1. Configuración del Entorno

1. Ejecutar XAMPP:

Inicia los servicios de Apache y MySQL desde el panel de control de XAMPP.

```
juanjoseguerrero@fedora:~$ sudo /opt/lampp/lampp status
[sudo] contraseña para juanjoseguerrero:
egrep: warning: egrep is obsolescent; using grep -E
egrep: warning: egrep is obsolescent; using grep -E
egrep: warning: egrep is obsolescent; using grep -E
egrep: warning: egrep is obsolescent; using grep -E
egrep: warning: egrep is obsolescent; using grep -E
Version: XAMPP for Linux 8.2.12-0
egrep: warning: egrep is obsolescent; using grep -E
Apache is running.
egrep: warning: egrep is obsolescent; using grep -E
MySQL is running.
ProFTPD is deactivated.
juanjoseguerrero@fedora:~$
```

2. Crea una base de datos:

- Abre phpMyAdmin (<http://localhost/phpmyadmin>).
- Crea una nueva base de datos llamada formulario_db.
- Dentro de la base de datos, crea una tabla llamada usuarios con las siguientes columnas:
 - id (INT, AUTO_INCREMENT, PRIMARY KEY)
 - nombre (VARCHAR(100))
 - email (VARCHAR(100))
 - edad (INT)

← T →	▼ id	nombre	email	edad
-------	------	--------	-------	------

2. Crear el Formulario HTML

Crea un archivo llamado formulario.html en una nueva carpeta llamada lab_2 dentro de htdocs de XAMPP

(por ejemplo, C:\xampp\htdocs\lab_2\formulario.html).

```
📁 Fedora Linux / opt / lampp / htdocs / lab_2
```

3. Procesar el Formulario con PHP

Crea un archivo llamado procesar_formulario.php en la misma carpeta (lab_2).

```

procesar_formulario.php
1  <?php
2  header('Content-Type: application/json');
3
4  $servername = "localhost";
5  $username = "root";
6  $password = "";
7  $dbname = "formulario_db";
8
9  $conn = new mysqli($servername, $username, $password, $dbname);
10 if ($conn->connect_error) {
11     echo json_encode(["status"=>"error","message"=>"Conexión fallida"]);
12     exit;
13 }
14
15 $accion = isset($_POST['accion']) ? $_POST['accion'] : 'insert';
16 $nombre = isset($_POST['nombre']) ? trim($_POST['nombre']) : '';
17 $email = isset($_POST['email']) ? trim($_POST['email']) : '';
18 $edad = isset($_POST['edad']) ? (int)$_POST['edad'] : 0;
19 $id = isset($_POST['id']) ? (int)$_POST['id'] : 0;
20
21 // Validación
22 if($accion=="insert" || $accion=="update"){
23     if(empty($nombre) || empty($email) || $edad <= 0){
24         echo json_encode(["status"=>"error","message"=>"Todos los campos son obligatorios y la edad debe
25         exit;
26     }

```

4. Mostrar los Datos Almacenados

Crea un archivo llamado mostrar_datos.php para mostrar los datos almacenados en la base de datos.

```

mostrar_datos.php
1  <?php
2  header('Content-Type: application/json');
3
4  $servername = "localhost";
5  $username = "root";
6  $password = "";
7  $dbname = "formulario_db";
8
9  $conn = new mysqli($servername, $username, $password, $dbname);
10 if ($conn->connect_error) {
11     echo json_encode([]);
12     exit;
13 }
14
15 $sql = "SELECT id, nombre, email, edad FROM usuarios ORDER BY id DESC";
16 $result = $conn->query($sql);
17
18 $datos = [];
19 if ($result->num_rows > 0){
20     while($row = $result->fetch_assoc()){
21         $datos[] = $row;
22     }
23 }
24
25 echo json_encode($datos);
26 $conn->close();

```

5. Ejecución de la Práctica

- Abre tu navegador y visita http://localhost/lab_2/formulario.php.

Registro de Usuario

Nombre *

Email *

Edad *

Registro de Usuario

Nombre *

Email *

Edad *

- Completa el formulario y envía los datos.
- Verifica que los datos se hayan guardado en la base de datos.

http://localhost/phpmyadmin/index.php?route=/sql&db=formulario_db&table=usuarios&pos=0

Base de datos: formulario_db » Tabla: usuarios

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Seguimiento Disparadores

Mostrando filas 0 - 0 (total de 1, La consulta tardó 0.0008 segundos.)

SELECT * FROM `usuarios`

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla

Opciones extra

id	nombre	email	edad
8	Juanete	jujugp@gmail.com	19

- Visita http://localhost/lab_2/tabla.php para ver los datos almacenados.

Usuarios Registrados

ID	Nombre	Email	Edad	Acciones
8	<input type="text" value="Juanete"/>	<input type="text" value="jujugp@gmail.com"/>	<input type="text" value="19"/>	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>

6. Ejercicios Propuestos

- Editar y Eliminar Registros: Crea funcionalidades para editar y eliminar registros de la base de datos.
- Estilos CSS:
- Mejora la apariencia del formulario y la tabla usando CSS.
- Paginación: Implementa paginación en la página mostrar_datos.php si hay muchos registros.
- Validación de Datos: Agrega validaciones de lado del servidor para asegurarte de que el email tenga un formato válido y que la edad sea un número positivo. Modificar procesar_formulario.php para verificar que los campos no estén vacíos: Este código valida que los campos no estén vacíos y que el correo tenga un formato válido.

MOSTRE LAS MEJORAS EN LOS INCISOS INTERIORES AL 6

Coloque una coma a proposito para validar que hay un (.com, .mx, o alguna variante)

Registro de Usuario

A user registration form with three input fields and two buttons. The first field is labeled 'Nombre *' and contains the text 'Pepe'. The second field is labeled 'Email *' and contains the text 'ppjjgp@gmail.com', which is highlighted with a red border. The third field is labeled 'Edad *' and contains the text '-19'. Below the fields are two buttons: a blue 'Registrar' button and a light blue 'Borrar' button.

Registro exitoso!

7. Preguntas de Reflexión

¿Qué ventajas tiene usar PHP para procesar formularios?

- PHP se ejecuta en el servidor, lo que permite procesar datos de manera segura antes de mostrarlos al usuario.
- Facilita la interacción con bases de datos como MySQL para almacenar, modificar y consultar información.
- Permite validar y sanitizar los datos enviados por el usuario, mejorando la seguridad y la integridad.
- Es ampliamente soportado, con mucha documentación y ejemplos disponibles, ideal para proyectos web dinámicos.

¿Cómo se puede prevenir la inyección SQL en este proyecto?

- Usando consultas preparadas (prepared statements) con parámetros, evitando concatenar directamente los datos del usuario.
- Sanitizando y validando los datos de entrada antes de enviarlos a la base de datos.
- Escapando caracteres especiales en cadenas cuando se usan queries dinámicas.
- Limitando los permisos del usuario de la base de datos para que no pueda ejecutar operaciones peligrosas.

¿Qué otros métodos HTTP (GET, PUT, DELETE) podrían ser útiles en este contexto?

- GET: Para obtener y mostrar los datos de los usuarios sin modificar la base de datos.
- PUT: Para actualizar completamente un registro existente, útil en APIs RESTful.
- DELETE: Para eliminar registros específicos de manera explícita y segura.
- Estos métodos son especialmente útiles si se decide exponer la funcionalidad mediante una API REST en lugar de usar solo formularios HTML.