Práctica 2: Formularios
Objetivo:

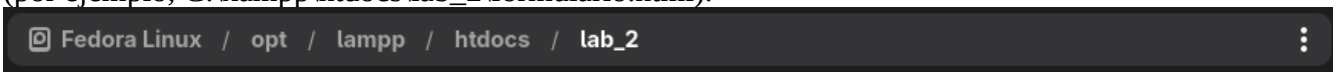
Crear un formulario HTML para capturar datos de usuarios.
Procesar los datos del formulario usando PHP.
Almacenar los datos en una base de datos MySQL.
Mostrar los datos almacenados en una página web.

1. Configuración del Entorno
1. Ejecutar XAMPP:
Inicia los servicios de Apache y MySQL desde el panel de control de XAMPP.

```
juanjoseguerreroperez@fedora:~$ sudo /opt/lampp/lampp status
[sudo] contraseña para juanjoseguerreroperez:
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.
juanjoseguerreroperez@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)

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).



3. Procesar el Formulario con PHP
Crea un archivo llamado procesar_formulario.php en la misma carpeta (lab_2).

procesar_formulario.php

```php
<?php
header('Content-Type: application/json');

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "formulario_db";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    echo json_encode(["status"=>"error","message"=>"Conexión fallida"]);
    exit;
}

$accion = isset($_POST['accion']) ? $_POST['accion'] : 'insert';
$nombre = isset($_POST['nombre']) ? trim($_POST['nombre']) : '';
$email  = isset($_POST['email']) ? trim($_POST['email']) : '';
$edad   = isset($_POST['edad']) ? (int)$_POST['edad'] : 0;
$id     = isset($_POST['id']) ? (int)$_POST['id'] : 0;

// Validación
if($accion=="insert" || $accion=="update"){
    if(empty($nombre) || empty($email) || $edad <= 0){
        echo json_encode(["status"=>"error","message"=>"Todos los campos son obligatorios y la edad debe
        exit;
    }
```

4. Mostrar los Datos Almacenados

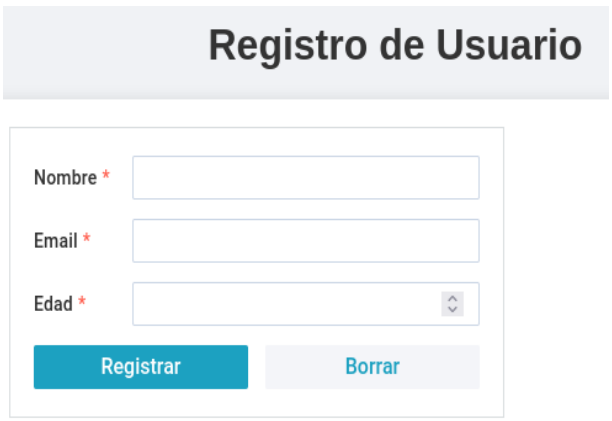
Crea un archivo llamado mostrar_datos.php para mostrar los datos almacenados en la base de datos.

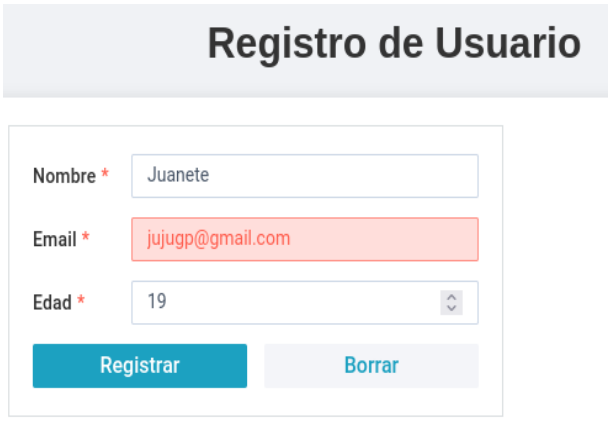
mostrar_datos.php

```php
<?php
header('Content-Type: application/json');

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "formulario_db";

$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    echo json_encode([]);
    exit;
}

$sql = "SELECT id, nombre, email, edad FROM usuarios ORDER BY id DESC";
$result = $conn->query($sql);

$datos = [];
if ($result->num_rows > 0){
    while($row = $result->fetch_assoc()){
        $datos[] = $row;
    }
}

echo json_encode($datos);
$conn->close();
```

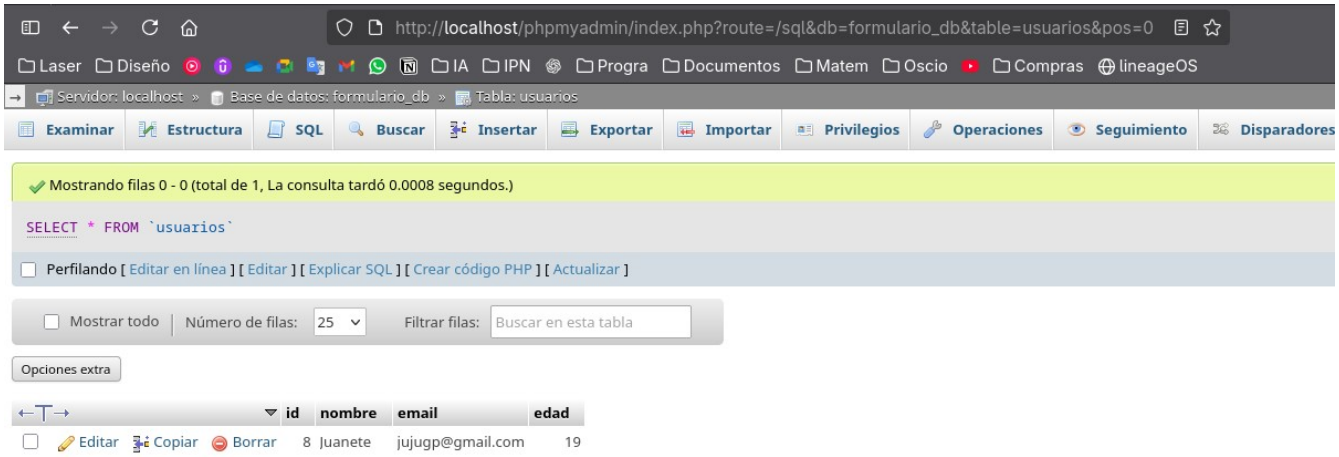
5. Ejecución de la Práctica

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

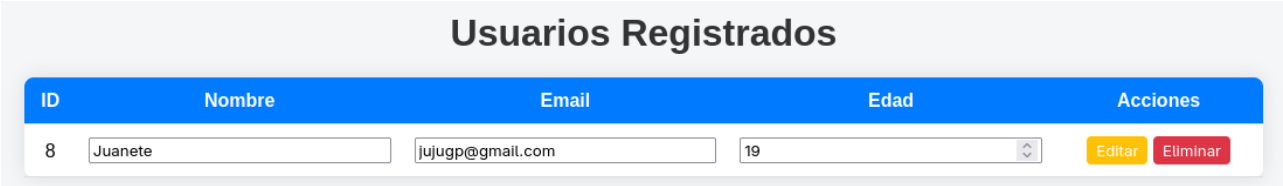




- Completa el formulario y envía los datos.
- Verifica que los datos se hayan guardado en la base de datos.



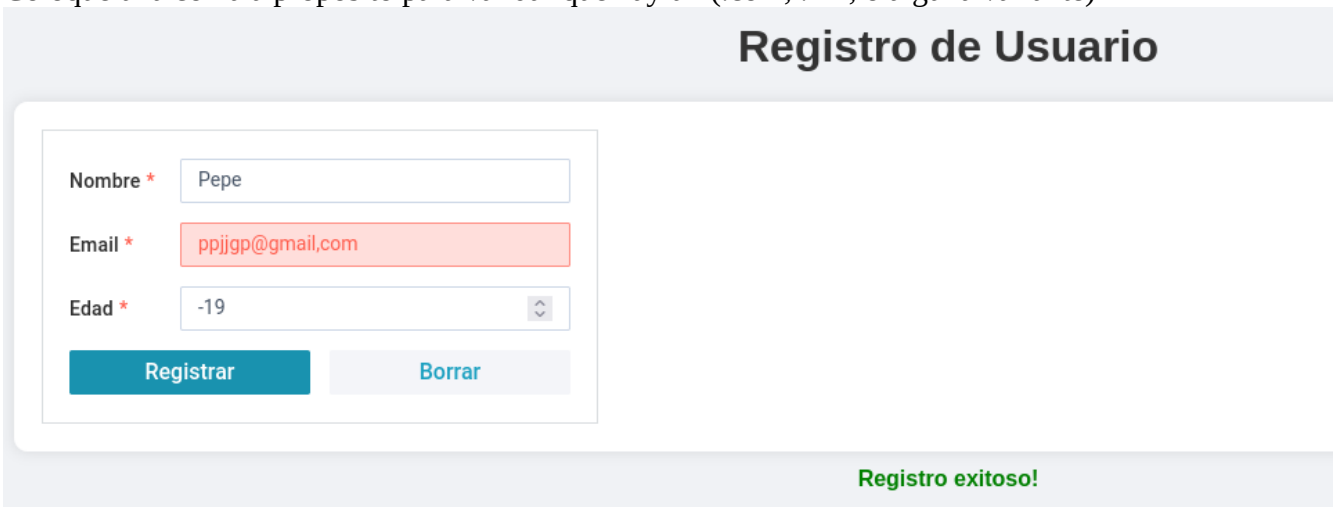
- Visita http://localhost/lab_2/tabla.php para ver los datos almacenados.



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)



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.