Blog fake: un formulario con tres controles: uno de texto (etiqueta: Nuevo mensaje) un botón de enviar y un área de texto de solo lectura. Cada pulsación a Enviar provocará que el área de texto acumule el texto enviado del campo Nuevo mensaje al final de los mensajes anteriores.

Avanzado: que el texto vaya apareciendo en un div como párrafos en vez de en un textarea.

En la primera línea de este bloque, utilizo `isset()` para comprobar si hay un parámetro `m` en la URL. Si existe, lo asigno a la variable `$m`, de lo contrario, la inicializo como una cadena vacía. Esto me permite acumular mensajes en la variable.

```php
$m = isset($_GET['m']) ? $_GET['m'] : '';
```

A continuación, verifico si se ha enviado un nuevo mensaje al servidor mediante el formulario. Si es así, utilizo `htmlspecialchars()` para asegurarme de que el texto ingresado por el usuario no contenga caracteres especiales que puedan afectar la estructura del HTML. Esto es importante para prevenir ataques de inyección de código. Finalmente, agrego el nuevo mensaje a la variable `$m`, envuelto en etiquetas `<p>` para que se muestre como un párrafo en la página.

```php
if (isset($_GET['new_message'])) {
    $Nmensaje = htmlspecialchars($_GET['new_message']);
    $m .= '<p>' . $Nmensaje . '</p>';
}
```

Utilizo un `<div>` para mostrar los mensajes acumulados. La variable `$m` se imprime directamente en este `<div>`, lo que permite que cada nuevo mensaje aparezca como un nuevo párrafo.

```php
<div id="M">
    <?php echo $m; ?>
</div>
```

Conclusiones

Hacer este proyecto ha sido una buena forma de practicar cómo manejar formularios en PHP y almacenar datos temporalmente. Aprendí que usar `htmlspecialchars()` es súper importante para evitar problemas de seguridad, como inyecciones de código. Además, descubrí cómo usar parámetros GET para mantener los mensajes que el usuario envía, lo que me permitió acumular información sin necesidad de usar una base de datos.

Dificultades :

Al principio, no sabía bien cómo hacer para que los mensajes se quedaran en la página después de enviar el formulario. Era un poco confuso entender cómo funcionaban los parámetros GET, pero después de investigar un poco, logré entenderlo.

El ejercicio de adivinar un número fijado previamente por el ordenador, dando retroalimentación sobre si es mayor o menor en cada intento (envío del formulario) y mostrando al adivinarlo el número de intentos consumidos. El número aleatorio debe calcularse una sola vez, en el primer acceso (acceso sin ningún parámetro enviado)

Comienzo con `session_start()` para poder guardar el número aleatorio y el conteo de intentos en la sesión del usuario. Esto es clave para que el número permanezca entre las diferentes solicitudes al servidor.

Si el número no ha sido generado aún (es la primera vez que se accede a la página), se genera un número aleatorio entre 1 y 100. También inicializo el contador de intentos a 0.

```php
session_start();

if (!isset($_SESSION['numero'])) {
    $_SESSION['numero'] = rand(1, 100);
    $_SESSION['intentos'] = 0;
}
```

Defino una variable `$mensaje` que usaré para mostrar la retroalimentación al usuario.

```php
$mensaje = '';
```

Si el usuario envía un intento, incremento el contador de intentos y convierto el intento a un número entero para asegurarme de que estoy trabajando con el tipo de dato correcto.

```php
$_SESSION['intentos']++;
$intentoUsuario = (int)$_GET['intento'];
```

Luego, comparo el intento del usuario con el número que he guardado en la sesión. Según el resultado, le doy retroalimentación al usuario: si su número es menor, le digo que el número es mayor, y viceversa. Si adivina el número, le felicito y muestro cuántos intentos le tomó. Finalmente, destruyo la sesión para que el juego pueda empezar de nuevo si el usuario lo desea.

```php
if ($intentoUsuario < $_SESSION['numero']) {
    $mensaje = 'El número es mayor.';
} elseif ($intentoUsuario > $_SESSION['numero']) {
    $mensaje = 'El número es menor.';
} else {
    $mensaje = '¡Felicidades! Has adivinado el número ' . $_SESSION['numer
    session_destroy();
}
```

Conclusiones

Este proyecto ha sido muy divertido y me ha ayudado a aprender cómo manejar sesiones en PHP y trabajar con formularios. Ahora tengo una mejor idea de cómo almacenar datos temporales y cómo darle retroalimentación al usuario de una manera más interactiva

Dificultades :

Al final, la solución fue sencilla: utilicé `session_start()` y almacené el número y los intentos en variables de sesión. Así, todo quedó guardado hasta que el usuario adivinara el número.

Los ejercicios de validación de formularios no estoy entendiendo bien cómo hacerlos y para poner algo copiado y no sacar nada prefiero poder entregarlos con retraso al menos aprendiendo algo .