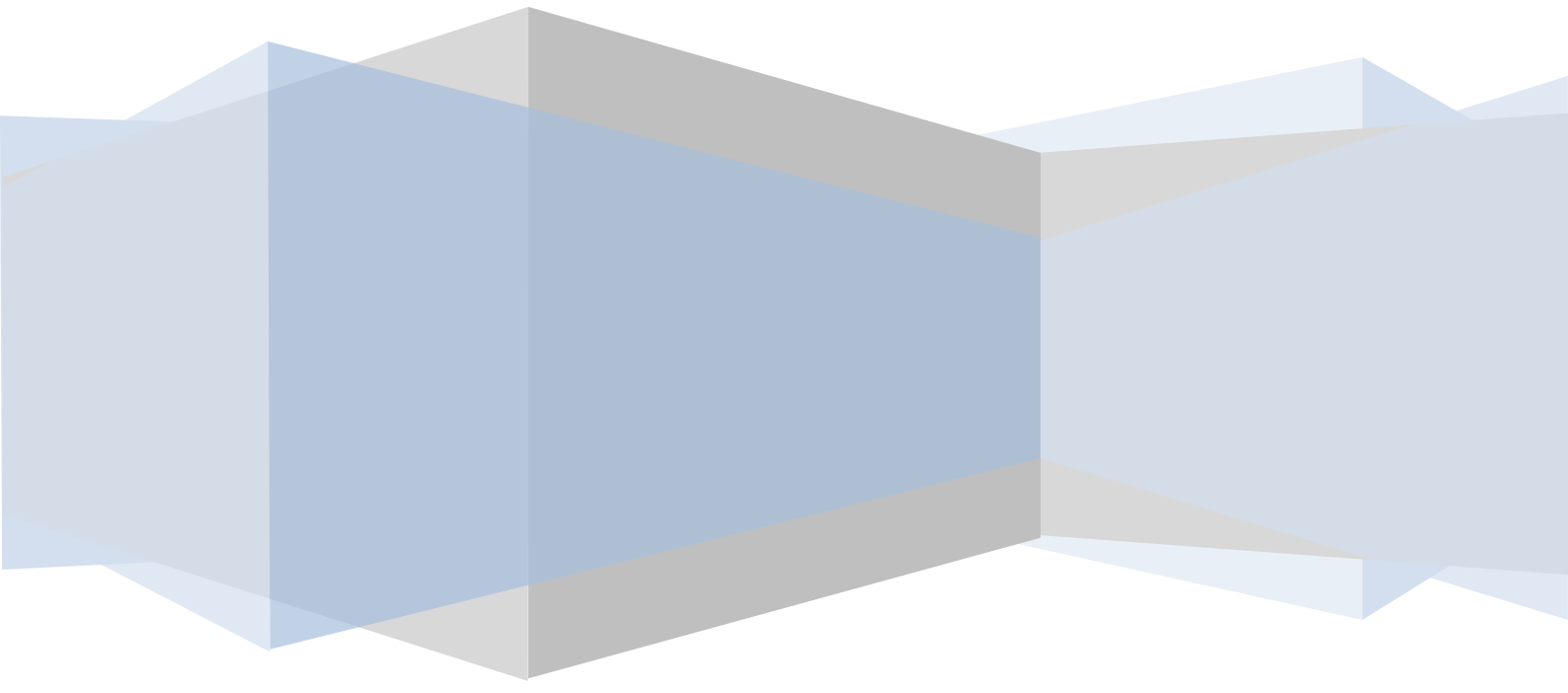

APLICACIÓN ADIVINAR NÚMERO

El objetivo era 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)

Pasos Realizados

1. Creación formulario HTML

Primero se hace un diseño inicial con la creación de una página HTML básica con un formulario, que definimos con form, con un campo de entrada numérica para que el usuario ingrese el número a adivinar y un botón de envío, tipo submit.

En el form indicamos el atributo action con **$_SERVER['PHP_SELF'],** lo que significa que cuando se envía el formulario, los datos se enviarán de vuelta al mismo script PHP en el que se encuentra actualmente.

```
<body>
    <h1>Adivina un número del 1 al 100</h1>
    <!-- indicamos en action el propio php (PHP_SELF) para que el formulario se dirija
    <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="post">
        <label for="nombre">Numero: </label>
        <!-- Se crea un campo de entrada de tipo número y obligatorio que permite
        al usuario ingresar un número del 1 al 100.
        Y un botón de tipo submit para enviar el formulario -->
        <input type="number" name="num" min="1" max="100" required >
        <input type="submit" name="adivinar" value="Adivinar" />
```

2. Número Aleatorio y Contador de intentos y campos ocultos.

Después de los input del formulario, sin cerrar este, abrimos etiqueta PHP.

Antes de generar un número aleatorio, se verifica si existe la variable con **isset $_POST["numMaquina"]**. Si no existe (lo que significa que es la primera vez que se carga la página o el juego se reinicia), se genera un número aleatorio entre 1 y 100 y se almacena en la variable **$numMaquina**.

Y con el contador de intentos hacemos lo mismo: Se verifica si existe la variable con **isset $_POST["contador"]**. Si no existe (en la primera carga de la página o al iniciar el juego), se inicializa **$contador** en 0.

```php
<?php
    // Comprueba si ya hay un número aleatorio para adivinar, si no,
    //se genera uno del 1 al 100.
    if (isset($_POST["numMaquina"])) {
        $numMaquina = $_POST["numMaquina"];
    } else {
        $numMaquina = rand(1, 100);
    }

    // Compueba si ya hay contador, si no, se inicializa en 0.
    if (isset($_POST["contador"])) {
        $contador = $_POST["contador"];
    } else {
        $contador = 0;
    }
```

Muy importante, estas dos variables las tenemos que almacenar dentro del formulario HTML pero a la vez que estén ocultas para el usuario, por lo que para ello creamos imput de tipo hidden, para que almacenen estos valores. Permiten mantener el mismo número aleatorio y el registro de intentos en el contador.

Estos campos los ponemos al final del documento, después de cerrar la parte de php y antes de cerrar el formulario y html (luego explico la razón).

```
            }
        ?>
        <!-- Se incluyen campos ocultos para almacenar el número aleatorio
        y el contador de intentos. -->
        <input type="hidden" name="numMaquina" value="<?php echo $numMaquina; ?>">
        <input type="hidden" name="contador" value="<?php echo $contador; ?>">
    </form>
</body>
</html>
```

4. Procesamiento de Datos del Formulario

Cuando el usuario envía el formulario haciendo clic en el botón "Adivinar", se verifica si existe la variable con **isset $_REQUEST["adivinar"]** (lo que indica que se ha enviado el formulario).

Si el formulario se ha enviado, se obtiene el número ingresado por el usuario a través de **$_REQUEST["num"]** y se lo asignamos a la variable **$num.**

Teniendo ya estos datos, vamos a la lógica de la aplicación, que es ir comparando el número de usuario $num con el número aleatorio $numMaquina para determinar si es igual, mayor o menor. Se hace una retroalimentación

En los tres casos, el contador se aumenta en 1 para ir sumando intentos y lo imprimimos por pantalla.

Si el número es igual, si acierta, se saca por pantalla dicho número y un mensaje informativo, y como se acaba el juego, inicializamos variables contador, a 0, y numMaquina, generando nuevo número aleatorio, de esta manera puede seguir jugando desde la misma sesión.

Si el número es mayor o menor, se dice por pantalla con mensaje de que lo vuelva intentar.

```php
// Verifica si se ha enviado el formulario con el botón "Adivinar".
if ( isset($_REQUEST["adivinar"])){
    $num = $_REQUEST["num"];

    // Se comprueba si el número ingresado en casilla es igual al número aleatorio g
    if ($num == $numMaquina){
        $contador++; // aumentamos el contador en 1 para sumar un intento.
        echo "<br> Tu número " . $num . " es correcto<br>¡HAS GANADO!<br>";
        echo "<br> Intentos: " . $contador . "<br>";
        echo "<br> Introduce otro número para volver a jugar.";
        // y como se ha acabado el juego reiniciamos contador de intentos y
        // generamos nuevo núm aleatorio
        $contador = 0;
        $numMaquina = rand(1, 100);

    } elseif ($num > $numMaquina){
        $contador++;
        echo "<br> Tu número " . $num . " es mayor.<br>Vuelve a intentarlo.";
        echo "<br> Intentos: " . $contador . "<br>";
    } else {
        $contador++;
        echo "<br> Tu número " . $num . " es menor.<br>Vuelve a intentarlo.";
        echo "<br> Intentos: " . $contador . "<br>";
    }
}
```

Capturas de ejecución:

5. Dificultades.

Al principio puse los campos ocultos del formulario junto con el resto de campos del formulario, pero claro salía error ya que detectaba que los valores que tenía que almacenar no se actualizaban pues como estaba el php después, cada vez que se enviaba el formulario. Por lo que al colocarlos al final, aseguramos que se actualicen con los valores correctos antes de que el formulario se vuelva a mostrar después de darle a enviar.

Conclusiones

Esta práctica ha sido muy útil para refrescar los conocimientos de creación de formularios HTML y comprender mucho mejor su funcionalidad y cómo funcionan “por dentro”, es decir, como se procesan los datos del usuario. También he aprendido bastante PHP, sobre todo la sintaxis que es diferente al resto de lenguajes, y me ha hecho practicar la lógica y la soltura.