**Explicación previa mediante el uso de una tarea guiada**

**Tarea 1.0**

Enunciado de la Actividad: Implementación de un Chatbot en PHP (sin sesiones)

Descripción de la actividad:

En esta actividad, se te solicita implementar un chatbot simulado utilizando PHP sin recurrir al uso de sesiones. El objetivo es que el chatbot sea capaz de mantener una conversación simple con el usuario, mostrando tanto los mensajes enviados por el usuario como las respuestas generadas por el chatbot en la misma página web. Para mantener el historial de la conversación, utilizarás campos ocultos en el formulario HTML en lugar de sesiones. Toda la interacción deberá ser gestionada exclusivamente en el servidor utilizando únicamente PHP, sin usar tecnologías del lado del cliente como JavaScript o AJAX

Requisitos de la aplicación:

1. Interfaz de usuario:

- La interfaz debe contar con un formulario donde el usuario pueda escribir y enviar mensajes al chatbot.

- Los mensajes enviados por el usuario y las respuestas del chatbot deben mostrarse en la pantalla en la misma página.

- La conversación debe mantenerse visible usando \*\*campos ocultos\*\* en el formulario, que contendrán el historial de los mensajes entre el usuario y el chatbot.

2. Uso de campos ocultos :

- Para almacenar el historial de la conversación, usarás un campo `input` de tipo `hidden` en el formulario. Este campo oculto mantendrá concatenados todos los mensajes previos entre el usuario y el chatbot.

- Al enviar un nuevo mensaje, el contenido del campo oculto se actualizará con los mensajes anteriores más el nuevo mensaje, de manera que todo el historial se envíe nuevamente al servidor con cada petición.

3. Procesamiento de mensajes :

- El chatbot debe ser capaz de interpretar ciertos mensajes clave enviados por el usuario y responder con frases predefinidas. Por ejemplo:

- Si el usuario escribe "hola", el chatbot responderá "¡Hola! ¿Cómo puedo ayudarte hoy?".

- Si el usuario escribe "gracias", el chatbot responderá "¡De nada! Estoy aquí para ayudarte.".

- Si el usuario escribe "adiós", el chatbot responderá "¡Adiós! Que tengas un buen día.".

- Si el chatbot no encuentra coincidencias en el mensaje del usuario, responderá con una frase genérica, como "Lo siento, no entiendo tu pregunta. Intenta decir 'hola' o 'adiós'.".

4. Estilo de la aplicación :

- Usa CSS para darle un estilo visualmente agradable a la página:

- El fondo de la página debe ser de color \*\*amarillo Simpson\*\*.

- Los mensajes del usuario deben mostrarse en \*\*azul fuerte\*\* con fuente Verdana 14px.

- Los mensajes del chatbot se mostrarán en color \*\*gris oscuro\*\*.

- La conversación debe mostrarse en un cuadro que sea visualmente atractivo (con bordes redondeados, un fondo claro, y desplazamiento en caso de mensajes largos).

Funcionamiento esperado:

1. El usuario ingresa un mensaje en un área de texto y lo envía a través de un botón.

2. El formulario envía el mensaje del usuario junto con el historial de mensajes almacenado en un campo oculto .

3. El servidor procesa el mensaje, lo añade al historial junto con la respuesta del chatbot, y devuelve todo el historial actualizado a la página.

4. La página se recarga mostrando tanto el nuevo mensaje del usuario como la respuesta generada por el chatbot, manteniendo el historial de la conversación.

Ejemplo de flujo de trabajo:

1. El usuario abre la página y ve un cuadro vacío donde puede escribir un mensaje.

2. Escribe "hola" y envía el mensaje.

3. La página se recarga y muestra:

- Usuario: hola

- Bot: ¡Hola! ¿Cómo puedo ayudarte hoy?

4. El usuario escribe "gracias" y envía el mensaje.

5. La página se recarga y muestra:

- Usuario: hola

- Bot: ¡Hola! ¿Cómo puedo ayudarte hoy?

- Usuario: gracias

- Bot: ¡De nada! Estoy aquí para ayudarte.

Código básico sugerido:

A partir de aquí, vamos a ver cómo

<?php

// Función para generar la respuesta del chatbot

function generarRespuesta($mensajeUsuario) {

$mensajeUsuario = strtolower(trim($mensajeUsuario));

// Respuestas predefinidas del chatbot

$respuestas = [

"hola" => "¡Hola! ¿Cómo puedo ayudarte hoy?",

"¿quién eres?" => "Soy un chatbot simulado para este proyecto de entorno servidor.",

"¿qué puedes hacer?" => "Puedo responder preguntas básicas. ¡Pregúntame algo!",

"gracias" => "¡De nada! Estoy aquí para ayudarte.",

"adiós" => "¡Adiós! Que tengas un buen día."

];

// Buscar coincidencia y devolver la respuesta

foreach ($respuestas as $pregunta => $respuesta) {

if (strpos($mensajeUsuario, $pregunta) !== false) {

return $respuesta;

}

}

// Respuesta por defecto

return "Lo siento, no entiendo tu pregunta. Intenta decir 'hola' o 'adiós'.";

}

// Historial de mensajes

$historial = "";

// Procesamos el formulario si hay un mensaje enviado

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && !empty($\_POST['mensaje'])) {

$mensajeUsuario = htmlspecialchars($\_POST['mensaje']);

// Recuperar el historial anterior si existe

$historial = isset($\_POST['historial']) ? $\_POST['historial'] : "";

// Añadir el mensaje del usuario al historial

$historial .= "<p class='user'>Usuario: " . $mensajeUsuario . "</p>";

// Generar la respuesta del chatbot

$respuestaBot = generarRespuesta($mensajeUsuario);

// Añadir la respuesta del chatbot al historial

$historial .= "<p class='bot'>Bot: " . $respuestaBot . "</p>";

}

?>

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

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

<title>Chatbot Simulado</title>

<style>

body {

font-family: Verdana, sans-serif;

background-color: #FDD835; /\* Amarillo Simpson \*/

margin: 0;

padding: 0;

}

.chat-container {

width: 400px;

margin: 50px auto;

background-color: #fff;

border: 1px solid #ccc;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

padding: 20px;

}

.chat-header {

font-size: 18px;

font-weight: bold;

color: #003366; /\* Azul fuerte \*/

margin-bottom: 10px;

}

.chat-box {

width: 100%;

height: 300px;

border: 1px solid #ccc;

border-radius: 5px;

overflow-y: auto;

padding: 10px;

background-color: #f9f9f9;

}

.chat-box p {

margin: 10px 0;

}

.chat-box p.user {

color: #003366; /\* Azul fuerte \*/

text-align: right;

}

.chat-box p.bot {

color: #333;

text-align: left;

}

.chat-input {

width: 100%;

padding: 10px;

margin-top: 10px;

border: 1px solid #ccc;

border-radius: 5px;

font-size: 14px;

}

button {

background-color: #003366; /\* Azul fuerte \*/

color: white;

padding: 10px;

border: none;

border-radius: 5px;

cursor: pointer;

margin-top: 10px;

width: 100%;

}

button:hover {

background-color: #002244;

}

</style>

</head>

<body>

<div class="chat-container">

<div class="chat-header">Chatbot Simulado</div>

<div class="chat-box">

<!-- Mostramos el historial de mensajes -->

<?php echo $historial; ?>

</div>

<!-- Formulario para enviar mensajes -->

<form method="post" action="chatbox.php">

<textarea class="chat-input" name="mensaje" placeholder="Escribe tu mensaje..."></textarea>

<!-- Campo oculto para el historial -->

<input type="hidden" name="historial" value="<?php echo htmlspecialchars($historial); ?>">

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

</form>

</div>

</body>

</html>

Objetivos de aprendizaje:

- Practicar la manipulación de datos a través de formularios en PHP.

- Implementar la gestión del historial de conversación utilizando campos ocultos.

- Aplicar la lógica condicional en PHP para procesar mensajes y generar respuestas automáticas.

- Diseñar una interfaz web sencilla utilizando HTML y CSS para mejorar la experiencia visual de la aplicación.

Entregables:

- El archivo `chatbot.php` que contiene tanto el código de la interfaz como la lógica de procesamiento del chatbot, utilizando campos ocultos en lugar de sesiones.

- La aplicación

**Enunciado de la Tarea 1.1: Piedra, Papel o Tijera con Rondas e Imágenes**

“Siempre me he jactado de mi habilidad para jugar con las frases. Las **palabras** son, en mi no tan humilde opinión, **nuestra más inagotable fuente de magia**.”

Albus Dumbledore



Dumbledore quiere que creemos un hechizo que nos permita jugar a piedra, papel o tijera para escoger a quién diseñará los hechizos de las puertas de las diferentes casas de Hogwarts.

**Objetivo:**

Desarrollar un juego de \*\*Piedra, Papel o Tijera\*\* utilizando PHP, que permita jugar varias rondas contra la máquina y muestre el historial de jugadas de cada ronda. Se debe implementar sin el uso de sesiones, manteniendo el estado del juego entre las peticiones mediante formularios y el envío de datos ocultos. Además, el juego debe incluir imágenes representativas de cada jugada (piedra, papel o tijera).

**Instrucciones:**

1. Interfaz inicial:

- El juego debe comenzar mostrando un formulario donde el jugador selecciona cuántas rondas quiere jugar (3, 5 o 7).

- Luego de seleccionar el número de rondas, el jugador podrá elegir entre "piedra", "papel" o "tijera" en cada ronda.

2. Desarrollo del juego:

- Cada vez que el jugador elija su jugada, la máquina debe elegir aleatoriamente entre "piedra", "papel" o "tijera".

- El sistema determinará quién ha ganado la ronda: el jugador, la máquina o si ha habido empate.

- La partida continuará hasta que se completen las rondas elegidas al inicio.

3. Imágenes:

- Deben mostrarse imágenes representativas para las jugadas de piedra, papel o tijera. Utiliza archivos de imagen (como `piedra.gif`, `papel.gif`, y `tijera.gif`) que se mostrarán según la jugada del jugador y de la máquina.

- Ejemplo: Si el jugador elige "piedra" y la máquina "tijera", se deben mostrar las imágenes correspondientes a esas jugadas.

4. Contadores de rondas:

- El sistema debe llevar la cuenta de cuántas rondas ha ganado el jugador y cuántas ha ganado la máquina.

- Al finalizar todas las rondas, se debe mostrar quién ha ganado más rondas y, por lo tanto, el juego completo.

5. Historial de jugadas (ampliación):

- Como ampliación, se debe implementar un historial de jugadas, donde se muestre para cada ronda:

- La jugada del jugador (con la imagen correspondiente).

- La jugada de la máquina (con la imagen correspondiente).

- El ganador de esa ronda (jugador, máquina o empate).

- El historial debe actualizarse en cada ronda y mostrarse al jugador.

6. Restricciones:

- No se deben utilizar sesiones para mantener el estado del juego entre rondas. En su lugar, se deben utilizar campos ocultos en el formulario (`<input type="hidden">`) para enviar y recibir el estado del juego (número de rondas restantes, victorias del jugador, victorias de la máquina, etc.).

7. Resultados finales:

- Al final de todas las rondas, se debe mostrar:

- El número total de victorias del jugador y de la máquina.

- El ganador global del juego (quien haya ganado más rondas).

- Un botón para reiniciar el juego y empezar de nuevo.

**Detalles técnicos:**

- Utiliza \*\*PHP\*\* para procesar las elecciones del jugador y la máquina, así como para determinar el ganador de cada ronda.

- Las imágenes deben estar correctamente cargadas mediante etiquetas `<img>` en el HTML.

- El historial de jugadas es opcional, pero será valorado como ampliación de la tarea.

**Ejemplo de flujo de juego:**

1. El jugador selecciona que quiere jugar \*\*5 rondas\*\*.

2. En la primera ronda, el jugador elige \*\*piedra\*\* y la máquina elige \*\*tijera\*\*. El jugador gana.

3. En la segunda ronda, el jugador elige \*\*papel\*\* y la máquina elige \*\*papel\*\*. Hay un empate.

4. El proceso continúa hasta que se completan las 5 rondas.

5. El juego muestra el historial de todas las jugadas y anuncia el ganador.

#### Ampliaciones opcionales:

1. \*\*Historial de jugadas:\*\* Añadir una tabla al final del juego que muestre el detalle de cada ronda: la jugada del jugador, la de la máquina, y quién ganó la ronda.

2. \*\*Mejor de X rondas:\*\* Permitir al jugador elegir si desea jugar al mejor de 3, 5 o 7 rondas, en lugar de un número fijo de rondas.

**Evaluación:**

La tarea se evaluará en función de los siguientes criterios:

- Funcionalidad básica: Correcto funcionamiento del juego con múltiples rondas, manejo de imágenes y contadores de victorias.

- Correcto uso de PHP: Uso adecuado de arrays, funciones, formularios y estructura condicional.

- Interfaz de usuario: Mostrar de forma clara las jugadas y el resultado de cada ronda.

- Ampliaciones: Historial de jugadas y posibilidad de seleccionar el número de rondas será considerado como ampliación de la tarea y aportará puntos adicionales.

**Recursos:**

- Imágenes: Deberás encontrar imágenes adecuadas como `piedra.gif`, `papel.gif` y `tijera.gif` que debes usar en el juego.

- Lenguajes y tecnologías: Solo se permite el uso de PHP y HTML (CSS opcional).

Ejemplo de ejecución del programa:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Otro ejemplo de ejecución:

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente