Actividad de desarrollo en Servidor.

Lectura de archivos JSON , BASES DE DATOS y SESION

Índice de apartados

Descripción de la actividad

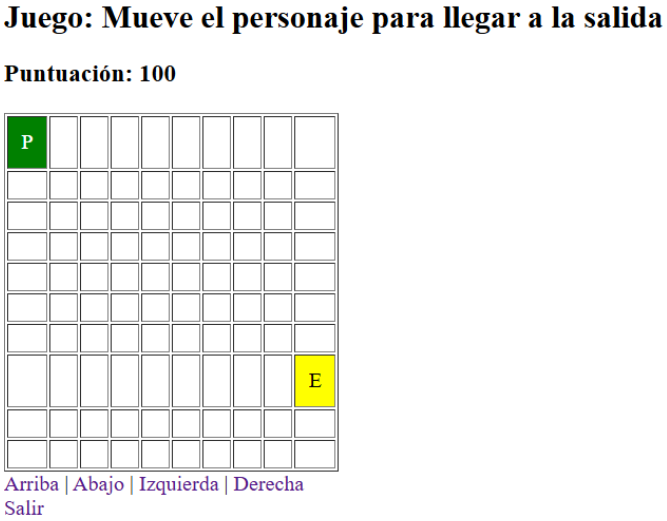
En esta actividad **vamos a desarrollar un pequeño juego siguiendo una arquitectura cliente y servidor**, usando bases de datos relacionales , la sesión para guardar los datos de interacción y la lectura de archivos JSON.

La aplicación se compone de los siguientes componentes:

- **index.php** . Este módulo php inicializa la sesión del juego y visualiza la imagen del tablero
- **controalorJuego.php** . Tiene toda la lógica del juego : avanzar la posición, comprobar si se ha caído en una casilla de trampa, comprobar si se ha llegado a la meta,etc
- **Clase VistaTablero.php y Clase Movimiento.php:** Clases para representar el tablero y un movimiento respectivamente

Interfaz gráfica del juego (index.php)

La interfaz del juego se representa con el código HTML necesario para mostrar una cuadricula, donde la posición del personaje y la salida están dibujadas con distinto color

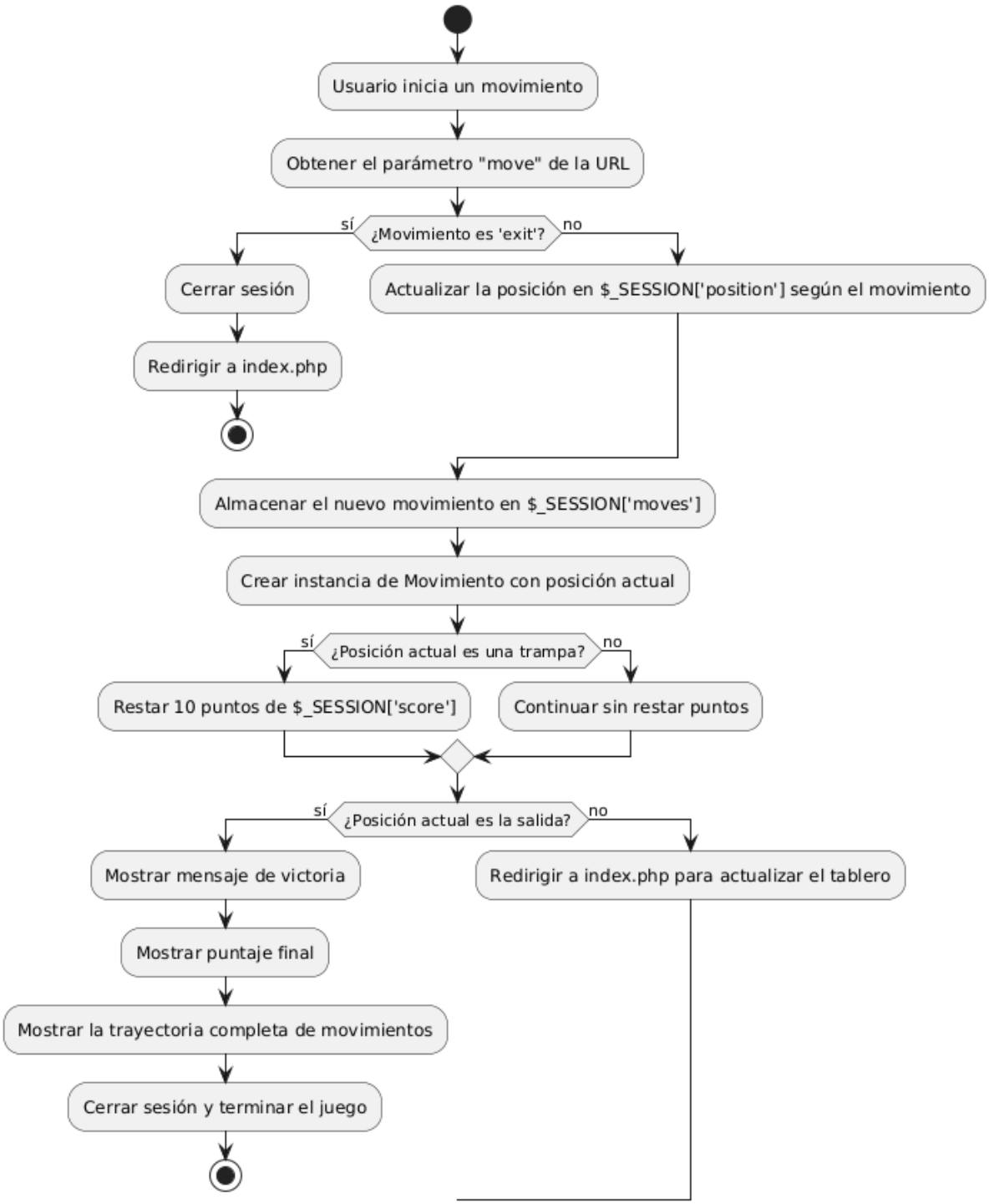


El objetivo es que jugador debe llegar hasta la casilla de salida (E) sin caer en las trampas que están ocultas en el tablero (ver archivo de configuración del juego). Cada vez que cae en una se le restan 10 puntos

Para moverse por el tablero tiene 4 controles (Arriba, Abajo, Izquierda y Derecha).

Lógica de control del juego (controladorJuego.php)

La lógica de control del juego se desarrolla en este módulo que **es llamado en cada movimiento desde la interfaz cliente**. Cada movimiento generar una solitud GET al servidor, el cual realiza la siguiente lógica de control.



IMPORTANTE. La sesión guarda la posición actual del jugador, luego cada movimiento supone acceder a este valor y actualizarle.

Su código es el siguiente

```php
<?php
require_once 'Movimiento.php';


session_start();
// Si se ha mandado un movimiento
if (isset($_GET['move'])) {
    // 1.- Obtener la posición actual y procesar el movimiento para actualizar posición
    $position = &$_SESSION['position']; // OJO  fíjate en el operador &
    $move = $_GET['move'];


    switch ($move) {
        case 'up':
            if ($position['row'] > 0) $position['row']--;
            break;
        case 'down':
            if ($position['row'] < 9) $position['row']++;
            break;
        case 'left':
            if ($position['col'] > 0) $position['col']--;
            break;
        case 'right':
            if ($position['col'] < 9) $position['col']++;
            break;
        case 'exit':
            session_destroy();
            header('Location: index.php');
    }


    // 2.- Guardar la trayectoria del movimiento
    //$_SESSION['moves'][] = ["row" => $position['row'], "col" => $position['col']];
    $_SESSION['moves'][] = new Movimiento($position['row'], $position['col']);
```

```
    // 3.- Verificar si cayó en una trampa
    $isTrap = false;
    foreach ($_SESSION['traps'] as $trap) {
        if ($trap['row'] == $position['row'] && $trap['col'] == 
$position['col']) {
            $_SESSION['score'] -= 10; // Restar puntos
            $isTrap = true;
            break;
        }
    }

    // 4.- Verificar si ha llegado a la salida
    $isExit = ($position['row'] == $_SESSION['exit']['row'] && 
$position['col'] == $_SESSION['exit']['col']);

    // 5.- si el usuario llegó a la salida terminar partida cerrando la 
sesión
    if ($isExit) {
        echo "<h2>Felicidades! Has llegado a la salida.</h2>";
        echo "<h3>Puntuación final: {$_SESSION['score']}</h3>";
        echo "<h4>Trayectoria recorrida:</h4>";
        //echo "<pre>" . json_encode($_SESSION['moves'], JSON_PRETTY_PRINT) . 
"</pre>";
        foreach ($_SESSION['moves'] as $movimiento) {
            echo "<pre>" . $movimiento->getFila() . "," . $movimiento-
>getColumna();
        }

        session_destroy(); // Terminar la sesión y la partida
        exit();
    } else {

        // Si no se ha llegado al final se vuelve a mostrar el tablero a 
través de index
        // Redirigir a index.php para actualizar el tablero
```

```
        header('Location: index.php');
    }
}
```

Archivo config.json

La configuración del juego se guarda en un archivo JSON. Cómo puedes observar, se define la posición inicial del jugador, donde se colocarán las trampas y donde estará la salida.

```
{
    "init":{"row":0,"col":0},
    "traps": [
        {"row": 2, "col": 3},
        {"row": 5, "col": 6},
        {"row": 8, "col": 7},
        {"row": 6, "col": 7}
    ],
    "exit": {"row": 7, "col": 9}
}
```

Módulo index.php

Este módulo se encarga **de inicializar la sesión de juego** , sino lo está **y visualizar el tablero con los datos de configuración y actualizados.** Para ello se apoya en la clase **VistaTablero.php** que permitirá mostrar el tablero en función de las posiciones actualizadas por el juego.

NOTA: Este módulo es reentrante con diferente comportamiento cada vez. La primera vez que entra, guarda e inicializa todos los datos de la sesión. La segunda vez sólo se limita a mostrar la interfaz del tablero.

NOTA: fíjate como el elemento $_SESSION[‘moves’] está vacío. Ya veremos que cada movimiento se guardará como un objeto en este array.

```
<?php
require_once 'VistaTablero.php';
session_start();
$vista = new VistaTablero();
// Inicializar o reiniciar el juego
// Leer configuración de trampas y salida al iniciar y meterlas en la sesión si no están
if (!isset($_SESSION['traps']) || !isset($_SESSION['exit'])) {
    $config = json_decode(file_get_contents("config.json"), true);
    $_SESSION['traps'] = $config['traps'];
    $_SESSION['exit'] = $config['exit'];
    $_SESSION['position']=$config['init'];
    $_SESSION['exit']=$config['exit'];
    $_SESSION['score'] = 100;
    $_SESSION['moves'] = [];
}

….
```

Clase Movimiento.php

NOTA: La clase está incompleta

Representa un movimiento que será guardado en la sesión del jugador para luego imprimirlos todos al finalizar el juego.

```
<?php
class Movimiento{

    private $fila,$columna;

    public function __construct(int $fila,int $columna){
    }
    public  function getFila():int{
    }

    public  function getColumna():int{

    }
}
```

Clase VistaTablero.php

Esta clase representa un modulo funcional para imprimir el tablero en función de los datos actuales de posición

```
<?php
class VistaTablero
{
    public function __construc() {}
    public function mostrarTablero($pos, $meta, $score)
    {
        // Mostrar el tablero de 10x10 y los controles de movimiento
        echo "<h2>Juego: Mueve el personaje para llegar a la salida</h2>";
        echo "<h3>Puntuación: {$score}</h3>";
        echo "<table border='1' cellpadding='10'>";
        for ($i = 0; $i < 10; $i++) {
            echo "<tr>";
            for ($j = 0; $j < 10; $j++) {
                $position = $pos; // Posición del personaje
                $exit = $meta; // Posición de la salida, suponiendo que está definida en la sesión
                if ($i == $position['row'] && $j == $position['col']) {
                    // Mostrar el personaje
                    echo "<td style='background-color: green; color: white;'>P</td>";
                } elseif ($i == $exit['row'] && $j == $exit['col']) {
                    // Mostrar la salida
                    echo "<td style='background-color: yellow; color: black;'>E</td>"; // E para "Exit"
                } else {
                    echo "<td></td>"; // Casilla vacía
                }
            }
```

```
            echo "</tr>";
        }
        echo "</table>";
        // Controles de movimiento
        echo "<div>";
        echo "<a href='controladorJuego.php?move=up'>Arriba</a> | ";
        echo "<a href='controladorJuego.php?move=down'>Abajo</a> | ";
        echo "<a href='controladorJuego.php?move=left'>Izquierda</a> |
";
        echo "<a href='controladorJuego.php?move=right'>Derecha</a>";
        echo "</br>";
        echo "<a href='controladorJuego.php?move=exit'>Salir</a>";
        echo "</div>";
    }
}
```

Cuestiones

1. Prueba la aplicación . Estudia cómo se usa la sesión en este ejercicio

2. Necesitamos que la aplicación guarde cada partida en la base de datos al finalizar con la puntuación obtenenida . para ello tendremos una tabla con un **id_sesion, una puntuación y la fecha en la que se jugó la partida.**

   Sigue los siguientes pasos:

   a) Añade una clase (RepositoryMSQL) que encapsule toda la lógica para conectar con la base de datos y acceder e insertar datos. La base de datos se llama **'game'**

```
class RepositoryMySQL
{
   private $conn; // Conexión a MYSQL
```

```
// Constructor con los parámetros de conexión
    public function __construct($servername, $dbname, $username,
$password) {...]

// Método para guardar los datos de cada partida
    public function saveGame( $points ):int {... }
}

public function findAllScores(): array {...}
```

Para lanzar la creación de la base de datos al contenedor Docker

```
docker exec -it your_mysql_container mysql -u root -p'1234' <
game_table.sql
```

El siguiente script permite crear la base de datos

```
-- Crear la base de datos si no existe
CREATE DATABASE IF NOT EXISTS game_matrix;
USE game_matrix;

-- Crear la tabla `game`
CREATE TABLE IF NOT EXISTS game (
    id_session INT AUTO_INCREMENT PRIMARY KEY,
    score INT NOT NULL,
    date_session DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Insertar algunos datos de ejemplo en la tabla `game`
INSERT INTO game (score, date_session) VALUES
(150, '2024-10-30 10:00:00'),
(200, '2024-10-30 11:30:00'),
(75, '2024-10-30 12:00:00'),
(120, '2024-10-31 09:15:00'),
(250, '2024-10-31 14:45:00');
```

b) Ahora modifica la interfaz gráfica de usuario para que cada vez que se acceda a la opción "**ver datos de partidas guardadas** se muestre el listado de puntuaciones obtenidas.

c) Crea ahora módulo php denominado **verPartidasGuardadas.php** para extraer todas las partidas y mostrarlas