Trabajo Práctico N°3

Nombre: Marcos Rafael Quispe  
2.1

<?php

if (isset($\_GET['numeros'])) {

$numeros\_str = $\_GET['numeros'];

$numeros = explode(',', $numeros\_str);

foreach ($numeros as $numero) {

if (!is\_numeric($numero)) {

echo json\_encode(['error' => 'Uno o más valores no son números']);

exit;

}

}

$media = array\_sum($numeros) / count($numeros);

sort($numeros);

$count = count($numeros);

$mitad = floor($count / 2);

if ($count % 2 == 0) {

$mediana = ($numeros[$mitad - 1] + $numeros[$mitad]) / 2;

} else {

$mediana = $numeros[$mitad];

}

$frecuencias = array\_count\_values($numeros);

$moda = array\_search(max($frecuencias), $frecuencias);

$resultados = [

'media' => $media,

'mediana' => $mediana,

'moda' => $moda

];

echo json\_encode($resultados);

} else {

echo json\_encode(['error' => 'No se proporcionaron números']);

}

2.2

<?php

class Figura {

public function perimetro() {

return 0;

}

public function area() {

return 0;

}

}

<?php

class Circulo extends Figura {

protected $radio;

public const PI = 3.1415;

public function \_\_construct($radio) {

$this->radio = $radio;

}

public function perimetro() {

return 2 \* PI \* $this->radio;

}

public function area() {

return PI \* pow($this->radio, 2);

}

}  
<?php

class Triangulo extends Figura {

protected $lado1;

protected $lado2;

protected $lado3;

public function \_\_construct($lado1, $lado2, $lado3) {

$this->lado1 = $lado1;

$this->lado2 = $lado2;

$this->lado3 = $lado3;

}

public function perimetro() {

return this->lado1 + this->lado2 + this->lado3;

}

public function area() {

$s = $this->perimetro() / 2;

$a = $this->lado1;

$b = $this->lado2;

$c = $this->lado3;

return sqrt($s \* ($s - $a) \* ($s - $b) \* ($s - $c));

}

}

2.3

<?php

class Saludador {

public function saludar($nombre = "") {

if ($nombre === "") {

echo "Hola, mundo.";

} else {

echo "Hola, " . $nombre . ".";

}

}

}

$s = new Saludador();

$s->saludar();

$s->saludar("Fulano");

?>

2.4

//Iniciar seción  
<?php

session\_start();

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

$nombre = $\_POST['nombre'];

$\_SESSION['nombre'] = $nombre;

} elseif (!isset($\_SESSION['nombre'])) {

header('Location: index.php');

exit;

}

$nombre = $\_SESSION['nombre'];

?>

<!DOCTYPE html>

<html>

<head>

<title>Bienvenido</title>

</head>

<body>

<h1>Bienvenido, <?php echo htmlspecialchars($nombre); ?></h1>

<p><a href="cerrar\_sesion.php">Cerrar Sesión</a></p>

</body>

</html>

// Cerrar Seción  
<?php

session\_start();

session\_unset();

session\_destroy();

header('Location: index.php');

exit;

?>

// Index

<!DOCTYPE html>

<html>

<head>

<title>Iniciar Sesión</title>

</head>

<body>

<h1>Iniciar Sesión</h1>

<form method="POST" action="inicio.php">

<label for="nombre">Nombre:</label>

<input type="text" name="nombre" id="nombre">

<input type="submit" value="Iniciar Sesión">

</form>

</body>

</html>

2.5

|  |  |
| --- | --- |
| **Línea 1** | Se declara la función route, de forma pública y no retorna nada |
| **Línea 2** | Se calcula la ruta actual eliminando la parte de la url correspondiente a $this->base\_path |
| **Línea 3** | Crea una variable para saber si hay una coincidencia de ruta |
| **Línea 4** | Itera a través de las rutas definidas en el objeto $this->routes |
| **Línea 5** | Verifica si la ruta actual coincide con la expresión regular definida en $regex |
| **Línea 6** | Itera a través de los métodos http definidos en $route y las funciones asociadas |
| **Línea 7** | Verifica que el método http de la solicitud es igual al método definido en la ruta |
| **Línea 8** | Cambia el valor de $match si hay una coincidencia |
| **Línea 9** | Verifica que el método no sea ni GET ni POST |
| **Línea 10** | Se procesa la solicitud y se almacena en una variable. El nombre está compuesto por un \_ y el nombre del método |
| **Línea 11** | Itera a través de las funciones a ejecutar definidas para la ruta y el método específico |
| **Línea 12** | Ejecuta la función $todo |
| **Línea 13** | Se cierra el if |
| **Línea 14** | Verifica que $match no sea false  Si match es falso, establece el código de respuesta http en 404 |
| **Línea 15** | Se cierra la función route |

2.6

<?php

class GestorBD {

private $conexion;

public function \_\_construct($host, $database, $usuario, $contrasena) {

try {

$dsn = "mysql:host={$host};dbname={$database}";

$this->conexion = new PDO($dsn, $usuario, $contrasena);

$this->conexion->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

throw new Exception("Error al conectar a la base de datos: " . $e->getMessage());

}

}

public function \_\_destruct() {

if ($this->conexion) {

$this->conexion = null;

}

}

}

<?php

require\_once('GestorBD.php');

try {

$gestorBD = new GestorBD("localhost", "nombre\_base\_de\_datos", "usuario\_ejemplo", "contrasena\_ejemplo");

} catch (Exception $e) {

echo "Error: " . $e->getMessage();

}

2.7

public function escribir($tabla, $valores) {

try {

$columnas = implode(', ', array\_keys($valores));

$valores = array\_values($valores);

$placeholders = rtrim(str\_repeat('?, ', count($valores)), ', ');

$sql = "INSERT INTO {$tabla} ({$columnas}) VALUES ({$placeholders})";

$stmt = $this->conexion->prepare($sql);

$stmt->execute($valores);

} catch (PDOException $e) {

throw new Exception("Error al escribir en la base de datos: " . $e->getMessage());

}

}

2.8

public function leer($tabla, $criterio = array()) {

try {

$sql = "SELECT \* FROM {$tabla}";

if (!empty($criterio)) {

$condiciones = array();

foreach ($criterio as $columna => $valor) {

$condiciones[] = "$columna = ?";

}

$sql .= " WHERE " . implode(' AND ', $condiciones);

}

$stmt = $this->conexion->prepare($sql);

$stmt->execute(array\_values($criterio));

return $stmt->fetchAll(PDO::FETCH\_ASSOC);

} catch (PDOException $e) {

throw new Exception("Error al leer desde la base de datos: " . $e->getMessage());

}

}

2.9

public function borrar($tabla, $criterio = array()) {

try {

$sql = "DELETE FROM {$tabla}";

if (!empty($criterio)) {

$condiciones = array();

foreach ($criterio as $columna => $valor) {

$condiciones[] = "$columna = ?";

}

$sql .= " WHERE " . implode(' AND ', $condiciones);

}

$stmt = $this->conexion->prepare($sql);

$stmt->execute(array\_values($criterio));

} catch (PDOException $e) {

throw new Exception("Error al borrar desde la base de datos: " . $e->getMessage());

}

}

2.10

public function editar($tabla, $valores, $criterio = array()) {

try {

$columnas = implode(' = ?, ', array\_keys($valores)) . ' = ?';

$criterios = array();

foreach ($criterio as $columna => $valor) {

$criterios[] = "$columna = ?";

}

$sql = "UPDATE {$tabla} SET {$columnas} WHERE " . implode(' AND ', $criterios);

$stmt = $this->conexion->prepare($sql);

$stmt->execute(array\_merge(array\_values($valores), array\_values($criterio)));

} catch (PDOException $e) {

throw new Exception("Error al editar en la base de datos: " . $e->getMessage());

}

}