Base de datos y servicios PHP

Especialidad PHP

Felipe Izquierdo Romero

Índice

1. Descripción del problema

Crear una aplicación web para la gestión de tareas. Esta aplicación podrá registrar los usuarios que trabajan durante una jornada laboral. La aplicación también podrá crear tareas y poder asignarlas a un usuario durante la creación de esta o posteriormente.

Los usuarios y las tareas asignadas y sin asignar se podrán visualizar en forma de tablas y para asignar tareas deberá de tener un desplegable con las tareas sin asignar y los usuarios a los que se les puede asignar tareas que serán quienes no tengan tareas asignadas.

2. Desarrollo

Para comenzar con el siguiente laboratorio he reutilizado parte del código del paso a paso para la gestión de usuarios. He añadido otro archivo ".php" con contenido "html" para visualizar las tareas asignadas, sin asignar y los formularios para crear una tarea o asignar tareas existentes. Para cambiar entre estos ".php" con una etiqueta "<a>" podrás navegar entre las dos vistas.

```
36      <nav><a href="administrador_tareas.php">Tareas</a></nav>
```

Ilustración 1- Navegación hacia administrador_ tareas.php

```
60      <nav><a href="index.php">Index</a></nav>
```

Ilustración 2- Navegación hacia index.php

El administrador de tareas contiene dos formularios para agregar una tarea o para asignar una tarea existente a un usuario libre.

```
63  <form method="post" action="administrador_tareas.php">
64      <input type="hidden" name="form_type" value="agregar_tarea">
65      <label for="nombre_tarea">Nombre de la tarea:</label>
66      <input type="text" name="nombre_tarea" required><br>
67
68      <label for="descripcion">Descripción:</label>
69      <input type="text" name="descripcion" required><br>
70
71      <label for="asignar_usuario">Asignar usuario:</label>
72      <select id="asignar_usuario" name="asignar_usuario">
73          <option value="null">Sin Asignar</option>
74          <?php foreach ($usuarios as $user) { ?>
75          <option value="<?php echo $user['id']; ?>"><?php echo $user['nombre']; ?></option>
76          <?php } ?>
77      </select>
78
79      <br>
80      <input type="submit" name="agregar_tarea" value="Agregar Tarea">
81  </form>
```

Ilustración 3- Formulario agregar tarea

```
84    <form method="post" action="administrador_tareas.php">
85        <input type="hidden" name="form_type" value="asignar_tarea">
86
87        <label for="id_tarea">Tarea:</label>
88        <select id="id_tarea" name="id_tarea">
89            <option value="null">Sin Asignar</option>
90            <?php foreach ($tareas_no_asignadas as $tarea) { ?>
91            <option value="<?php echo $tarea['id']; ?>"><?php echo $tarea['nombre']; ?></option>
92            <?php } ?>
93        </select>
94
95        <label for="asignar_usuario">Asignar usuario:</label>
96        <select id="asignar_usuario" name="asignar_usuario">
97            <option value="null">Sin Asignar</option>
98            <?php foreach ($usuarios as $user) { ?>
99            <option value="<?php echo $user['id']; ?>"><?php echo $user['nombre']; ?></option>
100           <?php } ?>
101       </select>
102
103       <input type="submit" name="asignar_tarea" value="Asignar Tarea">
104   </form>
```

Ilustración 4- Asignar tarea

Contiene dos objetos que se encargarán de la conexión con la base de datos "ConexionDB" y el objeto que recogerá la conexión y tiene definidas las sentencias y retornará los valores "MySQLRepository".

```
7   $conexionDB = new ConexionDB();
8   $tareaRepository = new MySQLRepository($conexionDB);
```

Ilustración 5- Objetos relaccionados con la BBDD

Para visualizar las tablas y el combo box para la asignación de tareas a usuarios que están disponibles existen las siguientes variables.

```
47  $usuarios = $tareaRepository->obtenerSinTareas();
48  $tareas_asignadas = $tareaRepository->obtenerAsignadas();
49  $tareas_no_asignadas = $tareaRepository->obtenerNoAsignadas();
```

Ilustración 6- Variables para visualizar datos

Estos dos formularios en el mismo documento para diferenciar entre que formulario fue quien desencadeno la llamada tienen el mismo atributo name "form_type" pero el atributo value es diferente de esa manera se distinguen con un bloque "if".

```
12      if (isset($_POST['agregar_tarea']) && $form_type === 'agregar_tarea') {
```

Ilustración 7- Condicional agregar tarea

Posteriormente recogerá los valores del formulario, creará un objeto Tarea y agregará la tarea en la base de datos e imprimirá un mensaje de éxito o de error según el resultado.

```php
if (isset($_POST['agregar_tarea']) && $form_type === 'agregar_tarea') {
    $nombre_tarea = $_POST['nombre_tarea'];
    $descripcion = $_POST['descripcion'];
    $id_usuario = $_POST['asignar_usuario'] !== 'null' ? $_POST['asignar_usuario'] : null;
    //$id_usuario = $_POST['asigar_usuario'];

    $tarea = new Tarea($nombre_tarea, $descripcion);

    // Insertar el objeto de usuario en la base de datos
    if ($tareaRepository->agregarTarea($tarea->getNombre(), $tarea->getDescripcion(), $id_usuario)) {
        echo "Tarea agregada correctamente.";
    } else {
        echo "Error al agregar la tarea.";
    }
}
```

Ilustración 8- Bloque de agregar tarea

Si es asignar tarea además comprueba que no haya un campo Sin Asignar que por defecto devolverá una cadena de caracteres con la palabra null.

```php
if (isset($_POST['asignar_tarea'])
    && $form_type === 'asignar_tarea'
    && isset($_POST['id_tarea'])
    && isset($_POST['asignar_usuario'])
    && $_POST['id_tarea'] !== 'null'
    && $_POST['asignar_usuario'] !== 'null') {
```

Ilustración 9- Condicional asignar tarea

Si los datos están correctamente introducidos recogerá el id de la tarea y del usuario para insertar en el registro de tarea el id del usuario al que corresponde la tarea.

```php
if (isset($_POST['asignar_tarea'])
    && $form_type === 'asignar_tarea'
    && isset($_POST['id_tarea'])
    && isset($_POST['asignar_usuario'])
    && $_POST['id_tarea'] !== 'null'
    && $_POST['asignar_usuario'] !== 'null') {

    $id_tarea = $_POST['id_tarea'];
    $id_usuario = $_POST['asignar_usuario'];

    $tareaRepository->asignarTarea($id_tarea,$id_usuario);
}

}
```

Ilustración 10- Bloque asignar tarea

Como parte final de este modulo de php se cierra la conexión con la base de datos para asegurar su integridad y consumir menos recursos.

```php
$conexionDB->cerrarConexion();
```

Ilustración 11- Cierre de la conexión

El modulo de usuarios por nombrar también su funcionamiento contiene un formulario para agregar un usuario y una tabla para visualizar los usuarios existentes.

```php
<!DOCTYPE html>
<html>
<head>
    <title>Administrador de Usuarios</title>
</head>
<body>
    <nav><a href="administrador_tareas.php">Tareas</a></nav>
    <h1>Administrador de Usuarios</h1>
    <table>
        <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Email</th>
            <!-- <th>Acciones</th> -->
        </tr>
        <?php foreach ($usuarios as $user) { ?>
            <tr>
                <td><?php echo $user['id']; ?></td>
                <td><?php echo $user['nombre']; ?></td>
                <td><?php echo $user['email']; ?></td>
            </tr>
        <?php } ?>
    </table>

    <h2>Agregar Usuario</h2>
    <form method="post" action="index.php">
        <label for="nombre">Nombre:</label>
        <input type="text" name="nombre" required><br>

        <label for="email">Email:</label>
        <input type="email" name="email" required><br>

        <input type="submit" name="agregar" value="Agregar Usuario">
    </form>
</body>
</html>
```

Ilustración 12- Cuerpo html de usuarios

```php
<?php
require_once('./service/conexionDB.php');
require_once('./class/mySqlRepository.php');
require_once('./class/usuario.php');

$conexionDB = new ConexionDB();
$usuarioRepository = new MySQLRepository($conexionDB);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['agregar'])) {
        $nombre = $_POST['nombre'];
        $email = $_POST['email'];

        // Crear un objeto de usuario
        $usuario = new Usuario($nombre, $email, $conexionDB);

        // Insertar el objeto de usuario en la base de datos
        if ($usuarioRepository->agregarUsuario($usuario->getNombre(), $usuario->getEmail())) {
            echo "Usuario agregado correctamente.";
        } else {
            echo "Error al agregar el usuario.";
        }
    }
}

$usuarios = $usuarioRepository->obtenerUsuarios();
$conexionDB->cerrarConexion();
?>
```

Ilustración 13- Cuerpo de php de usuarios

Profundizando en las clases y servicios que conforman la aplicación web: Tareas, Usuarios, mySqlRepository y ConexionDB.

mySqlRepository Recoge como propiedad la conexión a la base de datos en su constructor.

```php
<?php
class MySQLRepository {
    private $db;

    public function __construct($db) {
        $this->db = $db;
    }
```

Ilustración 14- Constructor MySqlRepository

Los métodos que contiene son las sentencias sql para interactuar con la base de datos.

```php
    public function obtenerUsuarios() {
        $sql = "SELECT * FROM usuarios";
        $stmt = $this->db->conexion->query($sql);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    public function obtenerSinTareas() {
        $sql = "SELECT u.*
                FROM usuarios u
                LEFT JOIN tareas t ON u.id = t.id_usuario
                WHERE t.id_usuario IS NULL";
        $stmt = $this->db->conexion->query($sql);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    public function agregarUsuario($nombre, $email) {
        $sql = "INSERT INTO usuarios (nombre, email) VALUES (:nombre, :email)";
        $stmt = $this->db->conexion->prepare($sql);
        $stmt->bindParam(':nombre', $nombre);
        $stmt->bindParam(':email', $email);
        $stmt->execute();
    }

    public function agregarTarea($nombre,$descripcion,$id_usuario){
        $sql = 'INSERT INTO tareas (nombre,descripcion,id_usuario) VALUES (:nombre,:descripcion,:id_usuario)';
        $stmt = $this->db->conexion->prepare($sql);
        $stmt->bindParam(':nombre', $nombre);
        $stmt->bindParam(':descripcion', $descripcion);
        $stmt->bindParam(':id_usuario', $id_usuario);
        $stmt->execute();
    }
```

Ilustración 15- Sentencias sql (1)

```php
    public function obtenerTareas(){
        $sql = 'SELECT * FROM tareas';
        return fetchAll(PDO::FETCH_ASSOC);
    }

    public function obtenerAsignadas(){
        $sql = 'SELECT t.*, u.nombre AS nombre_usuario
                FROM tareas t
                JOIN usuarios u ON t.id_usuario = u.id
                WHERE id_usuario IS NOT NULL';
        $stmt = $this->db->conexion->prepare($sql);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
    public function obtenerNoAsignadas(){
        $sql = 'SELECT * FROM tareas WHERE id_usuario IS NULL';
        $stmt = $this->db->conexion->prepare($sql);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    public function asignarTarea($id_tarea,$id_usuario){
        $sql = 'UPDATE tareas SET id_usuario = :id_usuario WHERE id = :id_tarea';
        $stmt = $this->db->conexion->prepare($sql);
        $stmt->bindParam(':id_usuario', $id_usuario);
        $stmt->bindParam(':id_tarea', $id_tarea);
        $stmt->execute();
    }
}
?>
```

Ilustración 16- Sentencias sql (2)

ConexiónDB recoge en el constructor las variables de entorno necesarias para conectar a la base de datos y contiene el método para cerrar conexión con la base de datos.

```php
<?php
use PDO as PDO;
class ConexionDB {

    private $dbhost;
    private $usuario;
    private $contrasena;
    private $database;
    public $conexion;

    public function __construct() {
        try {
            $this->dbhost = getenv("MYSQL_HOST");
            $this->usuario  = getenv("MYSQL_USER");
            $this->contrasena = getenv("MYSQL_PASSWORD");
            $this->database = getenv("MYSQL_DATABASE");

            $this->conexion = new PDO("mysql:host=$this->dbhost;dbname=$this->database", $this->usuario, $this->contrasena);
            $this->conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            echo "Error de conexión: " . $e->getMessage();
        }
    }

    public function cerrarConexion() {
        $this->conexion = null;
    }
}
?>
```

Ilustración 17- Servicio ConexionDB

La clase usuario implementa la interfaz usuario y esta clase recoge el nombre y el email del usuario y tiene métodos para retornar esos valores.

```php
<?php
require_once('./interfaces/usuario.php');

class Usuario implements UsuarioInterface {

    private $nombre;
    private $email;
    public function __construct($nombre, $email) {
        $this->nombre = $nombre;
        $this->email = $email;
    }

    public function getNombre()
    {
        return $this->nombre;
    }

    public function getEmail()
    {
        return $this->email;
    }

}
?>
```

Ilustración 18- Clase usuario

```php
<?php
interface UsuarioInterface {
    public function getNombre();
    public function getEmail();
}
?>
```

Ilustración 19- Interfaz usuario

La clase tarea se comporta de manera similar, pero con otros valores, nombre y descripción.

```php
<?php

require_once('./interfaces/tarea.php');
class Tarea implements TareaInterface{
    private $nombre;
    private $descripcion;

    public function __construct($nombre,$descripcion) {
        $this->nombre = $nombre;
        $this->descripcion = $descripcion;
    }
    public function getNombre(){
        return $this->nombre;
    }
    public function getDescripcion(){
        return $this->descripcion;
    }
}

?>
```

Ilustración 20- Clase Tarea

```php
<?php
interface TareaInterface {
    public function getNombre();
    public function getDescripcion();
}
?>
```

Ilustración 21- Interfaz Tarea

Para finalizar un pequeño apunte es que el contenedor de Docker iniciaría el servicio de base de datos sin tablas, para que se creen las tablas y se cree un usuario de prueba y no me de error el programa he añadido en el “docker-compose.yml“ en la etiqueta “volume” un “init.sql “ que contiene el siguiente código sql.

```sql
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    email VARCHAR(100)
);

CREATE TABLE tareas (
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre VARCHAR(50),descripcion VARCHAR(280),
    id_usuario INT, FOREIGN KEY (id_usuario)
    REFERENCES usuarios(id)
);

INSERT INTO usuarios (nombre,email) VALUES ('Usuario','De Prueba');
```

Ilustración 22- init.sql