Práctica 2: Adaptación de la web de la práctica Evaluable I

UNIVERSIDAD DE GRANADA

María de las Angustias Izquierdo García
Programación Web

1. Tablas MySQL

Para poder implementar el backend de esta práctica, era necesario crear tres tablas: Usuarios, Obras y Comentarios. Las sentencias que he utilizado para ello son las siguientes:

Usuarios

```
CREATE TABLE Usuarios (
  id int(11) NOT NULL AUTO_INCREMENT,
  nombreusuario varchar(50) NOT NULL,
  contrasenia varchar(255) NOT NULL,
  nombre varchar(50) NOT NULL,
  apellidos varchar(50) NOT NULL,
  correo varchar(100) NOT NULL,
  telefono varchar(15) NOT NULL,
  fecha_nacimiento date NOT NULL,
  comunidad_autonoma varchar(50) NOT NULL,
  provincia varchar(50) NOT NULL,
  visita_prado enum('si','no') NOT NULL,
  aceptar_terminos tinyint(1) NOT NULL,
  es_administrador tinyint(1) NOT NULL DEFAULT 0,
  imagen varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
)
```

La columna es_administrador se añade al final de la tabla, y su valor por defecto es 0, lo que significa que por defecto, los usuarios no son administradores. Si un usuario es administrador, se debe insertar un valor 1 en esta columna.

Obras

```
CREATE TABLE Obras (
  id int(11) NOT NULL AUTO_INCREMENT,
  titulo varchar(255) NOT NULL,
  autor varchar(255) DEFAULT NULL,
  epoca varchar(255) NOT NULL,
  imagen varchar(255) NOT NULL,
  tema varchar(255) NOT NULL,
  descripcion mediumtext NOT NULL DEFAULT '',
  PRIMARY KEY (`id`)
)
```

Comentarios

```
CREATE TABLE Comentarios (
  comentario_id int(11) NOT NULL AUTO_INCREMENT,
```

```
  usuario varchar(255) NOT NULL,
  sugerencia text DEFAULT NULL,
  fecha_comentario datetime DEFAULT NULL,
  PRIMARY KEY (`comentario_id`)
)
```

Para poder almacenar datos con tildes he realizado esto en la base de datos: ALTER TABLE Obras CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; y también lo he hecho con Comentarios y Usuarios

2. Alta usuarios

En el archivo alta usuarios.php, se llama a cinco archivos .php: verificarExistencia.php, verificarusuario.php, conexionbd.php, iniciosesion.php y altaexitosa.php.

conexionbd.php

```php
<?php
    // Datos de conexión a la base de datos
    $dsn = "mysql:host=localhost;dbname=";
    $usuario = "";
    $password = "";

    $conexion = new PDO($dsn, $usuario, $password);
    $conexion->exec("SET NAMES utf8mb4");
    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
?>
```

En $dsn defino el tipo de base de datos, el host y el nombre de la base de datos. En $usuario y $password se define el usuario y la contraseña de la base de datos y en $conexion se crea nueva instancia de PDO, la cual se utiliza para interactuar con la base de datos. Se pasa el DSN, el nombre de usuario y la contraseña como parámetros para establecer la conexión. Luego ejecutamos la instrucción SQL con SET NAMES utf8mb4 que configura el conjunto de caracteres para la conexión a la base de datos y finalmente con setAttribute configuramos un atributo de la conexión PDO para que lance excepciones en caso de errores.

verificarExistencia.php

```php
<?php

    // Verificar que se haya proporcionado un campo y un valor
    if (isset($_GET['campo']) && isset($_GET['valor'])) {
        $campo = $_GET['campo'];
        $valor = $_GET['valor'];

        try {
            require 'conexionbd.php';

            $sql = "SELECT COUNT(*) FROM Usuarios WHERE $campo = :valor";
            $stmt = $conexion->prepare($sql);
            $stmt->bindParam(':valor', $valor);
            $stmt->execute();

            // Obtener el número de registros que coinciden
            $count = $stmt->fetchColumn();

            $response = array('existe' => $count > 0);

            echo json_encode($response);

        } catch (PDOException $e) {
            echo json_encode(array('error' => $e->getMessage()));
        }
    }
    else {
        echo json_encode(array('error' => 'Parámetros inválidos'));
    }
?>
```

Este archivo se llama en la función javascript del archivo altausuarios.php para comprobar que el correo y el teléfono no estén asignados ya a un usuario en la base de datos.

En primer lugar se verifica si los parámetros campo y valor se encuentran en la solicitud GET. Accedemos a la base de datos y preparamos una consulta SQL para contar el número de registros en la tabla Usuarios donde el valor de la columna especificada ($campo) coincide con el valor proporcionado ($valor). He utilizado una declaración preparada (prepare) para evitar inyecciones SQL y luego se ejecuta (execute). Luego, con fetchColumn() se obtiene el número de registros que coinciden con la condición proporcionada. Después se construye una respuesta JSON que indica si el valor ya existe y se envía la respuesta JSON.

verificarusuario.php

```php
<?php
    // Verificar si el nombre de usuario ya está en uso
    if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['nombreusuario'])) {
        require 'conexionbd.php';

        $nombreusuario = $_GET['nombreusuario'];
        $sql = "SELECT COUNT(*) AS count FROM Usuarios WHERE nombreusuario = :nombreusuario";
        $stmt = $conexion->prepare($sql);
        $stmt->bindParam(':nombreusuario', $nombreusuario);
        $stmt->execute();
        $row = $stmt->fetch(PDO::FETCH_ASSOC);

        // Devolver respuesta en formato JSON
        header('Content-Type: application/json');
        echo json_encode(['disponible' => ($row['count'] == 0)]);
    }
?>
```

En primer lugar se verifica si la solicitud HTTP es de tipo GET y si el parámetro nombreusuario está presente en la URL. Obtiene el valor del parámetro nombreusuario de la solicitud GET y lo almacena en la variable $nombreusuario. La consulta SQL cuenta cuántos registros en la tabla Usuarios tienen un nombreusuario igual al valor proporcionado. Luego prepara la consulta SQL para su ejecución y asocia el valor de $nombreusuario al marcador de posición :nombreusuario en la consulta preparada. Se ejecuta la consulta y se obtiene el resultado de la consulta en forma de un array asociativo y se almacena en $row. Finalmente se establece el encabezado de la respuesta HTTP para indicar que el contenido es JSON y tilizo json_encode para crear una respuesta JSON que indica si el nombre de usuario está disponible o no. La disponibilidad se determina verificando si el recuento de registros es igual a 0 ($row['count'] == 0). Si es igual a 0, significa que el nombre de usuario no está en uso y, por lo tanto, está disponible.
Este archivo se llama en la función verificarNombreUsuario para mostrar un mensaje por pantalla si el nombre de usuario ya está en uso.

altausuarios.php

Javascript

```
<script>
    async function validarFormulario(event) {
        event.preventDefault();

        // Validar si el formulario está totalmente vacío
        const camposVacios = ['nombreusuario', 'contrasenia', 'nombre', 'apellidos', 'correo', 'telefono', 'fecha', 'comunidadautonoma', 'provincia', 'imagen'];
        if (camposVacios.some(campo => !document.getElementById(campo).value.trim())) {
            alert('Todos los campos son obligatorios.');
            return;
        }

        // Obtener los valores de los campos
        const nombreusuario = document.getElementById('nombreusuario').value;
        const contrasenia = document.getElementById('contrasenia').value;
        const nombre = document.getElementById('nombre').value;
        const apellidos = document.getElementById('apellidos').value;
        const correo = document.getElementById('correo').value;
        const telefono = document.getElementById('telefono').value;
        const fecha = document.getElementById('fecha').value;
        const comunidadautonoma = document.getElementById('comunidadautonoma').value;
        const provincia = document.getElementById('provincia').value;
        const visita_prado = document.querySelector('input[name="visita_prado"]:checked').value;
        const aceptar_terminos = document.getElementById('aceptar_terminos').checked;

        // Validar que los campos no estén vacíos
        if (!nombreusuario || !contrasenia || !nombre || !apellidos || !correo || !telefono || !fecha || !comunidadautonoma || !provincia || !aceptar_terminos) {
            alert('Todos los campos son obligatorios.');
            return;
        }

        // Validar el formato del correo electrónico
        const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailRegex.test(correo)) {
            alert('Por favor, introduce un correo electrónico válido.');
            return;
        }

        // Validar el formato de la fecha
        const fechaRegex = /^\d{2}-\d{2}-\d{4}$/;
        if (!fechaRegex.test(fecha)) {
            alert('Por favor, introduce una fecha válida en el formato DD-MM-AAAA.');
            return;
        }
```

```
        // Validar el formato del teléfono
        const telefonoRegex = /^\d{9}$/;
        if (!telefonoRegex.test(telefono)) {
            alert('Por favor, introduce un número de teléfono válido de 9 dígitos.');
            return;
        }

        // Validar que el nombre de usuario no esté repetido
        const existeUsuario = await verificarExistencia('nombreusuario', nombreusuario);
        if (existeUsuario) {
            alert('El nombre de usuario ya está registrado.');
            return;
        }

        // Validar que el correo no esté repetido
        const existeCorreo = await verificarExistencia('correo', correo);
        if (existeCorreo) {
            alert('El correo electrónico ya está registrado.');
            return;
        }

        // Validar que el teléfono no esté repetido
        const existeTelefono = await verificarExistencia('telefono', telefono);
        if (existeTelefono) {
            alert('El número de teléfono ya está registrado.');
            return;
        }

        // Validar que el usuario ha seleccionado una opción de visita al Prado
        if (!document.querySelector('input[name="visita_prado"]:checked')) {
            alert('Por favor, selecciona si has visitado el Museo del Prado.');
            return;
        }

        // Si todas las validaciones pasan, envía el formulario
        document.getElementById('formularioregistro').submit();
    }
```

La función validarFormulario se encarga de realizar la validación de los datos ingresados en un formulario de registro antes de que este sea enviado al servidor para su procesamiento. El primer paso dentro de la función es llamar al método event.preventDefault() para evitar que el formulario se envíe automáticamente cuando se hace clic en el botón de enviar. Se verifica si algún campo obligatorio está vacío. Si se encuentra algún campo vacío, se muestra una alerta indicando que todos los campos son obligatorios y se detiene la ejecución de la función. Seguidamente se obtienen los valores ingresados por el usuario en cada campo del formulario y luego se realizan las validaciones correspondientes con expresiones regulares, de tal manera que se compruebe que el correo tenga un formato adecuado y no esté ya asignado a otro usuario (llamando a verificarExistencia), que el teléfono tenga 9 dígitos y tampoco esté repetido, que la fecha tenga formato DD-MM-AAAA y validando también que el

nombre de usuario no esté repetido. En caso de que estas condiciones no se cumplan, se detiene la función con el return y se muestra un mensaje del error. Finalmente, si todo ha funcionado correctamente, se envía el formulario con document.getElementById('formularioregistro').submit();
La función validar formulario se incluye en el formulario de la siguiente forma:

```
Unset
<form id="formularioregistro" method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>"
onsubmit="validarFormulario(event)">
```

<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" indica que al procesar el formulario, se ejecute el codigo php que se encuentra en el propio archivo

```
async function verificarExistencia(campo, valor) {
    const response = await fetch(`verificarExistencia.php?campo=${campo}&valor=${valor}`);
    const data = await response.json();
    return data.existe;
}

async function verificarNombreUsuario() {
    const nombreusuario = document.getElementById('nombreusuario').value;
    const mensajeUsuario = document.getElementById('mensaje-usuario');

    // Verificar el nombre de usuario solo si hay algo escrito
    if (nombreusuario.trim() !== '') {
        // Enviar solicitud al servidor para verificar la disponibilidad del nombre de usuario
        const response = await fetch(`verificarusuario.php?nombreusuario=${nombreusuario}`);
        const data = await response.json();

        // Mostrar mensaje según la disponibilidad del nombre de usuario
        if (data.disponible) {
            mensajeUsuario.textContent = 'Nombre de usuario disponible.';
            mensajeUsuario.style.color = 'green';
        } else {
            mensajeUsuario.textContent = '¡El nombre de usuario ya está en uso!';
            mensajeUsuario.style.color = 'red';
            return;
        }
    } else {
        // Limpiar mensaje si el campo está vacío
        mensajeUsuario.textContent = '';
    }
}
```

La función verificarExistencia realiza una solicitud asíncrona al servidor utilizando el método fetch para verificar si un valor específico ya existe en la base de datos. Toma dos parámetros: el campo que se va a verificar y el valor que se desea comprobar. Luego, espera la respuesta del servidor en formato JSON y devuelve un booleano indicando si el valor existe o no.

Por otro lado, la función verificarNombreUsuario se encarga de verificar la disponibilidad de un nombre de usuario en tiempo real mientras el usuario escribe en el campo correspondiente del formulario. Extrae el valor del campo de nombre de usuario, y si este no está vacío, realiza una solicitud al servidor para verificar su disponibilidad. Dependiendo de la respuesta del servidor, actualiza un mensaje en la interfaz de usuario indicando si el nombre de usuario está disponible o no. Finalmente, si el campo de nombre de usuario está vacío, limpia el mensaje. Esta función se incluye en el formulario de registro en el input de la siguiente manera:

```
Unset
<input type="text" name="nombreusuario" id="nombreusuario"
oninput="verificarNombreUsuario()" >
                    <p id="mensaje-usuario"></p>
```

PHP

El siguiente código php es el que se utiliza para procesar el formulario de registro:

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener los datos del formulario
    $nombreusuario = $_POST['nombreusuario'];
    $contrasenia = $_POST['contrasenia'];
    $hashed_password = password_hash($contrasenia, PASSWORD_DEFAULT); // Crear un hash de la contraseña
    $nombre = $_POST['nombre'];
    $apellidos = $_POST['apellidos'];
    $correo = $_POST['correo'];
    $telefono = $_POST['telefono'];
    $fecha_nacimiento = $_POST['fecha'];
    $comunidad_autonoma = $_POST['comunidadautonoma'];
    $provincia = $_POST['provincia'];
    $visita_prado = $_POST['visita_prado'];
    $aceptar_terminos = isset($_POST['aceptar_terminos']) ? 1 : 0;
    $es_administrador = 0; // Por defecto, el usuario no es administrador
    $imagen = $_POST['imagen'];

    $fecha_nacimiento = date("Y-m-d", strtotime(str_replace('/', '-', $fecha_nacimiento)));

    try {
        // Conexión a la base de datos
        require 'conexionbd.php';

        // Consulta SQL para insertar el nuevo usuario
        $sql = "INSERT INTO Usuarios (nombreusuario, contrasenia, nombre, apellidos, correo, telefono,
                fecha_nacimiento, comunidad_autonoma, provincia, visita_prado, aceptar_terminos, es_administrador, imagen)
                VALUES (:nombreusuario, :contrasenia, :nombre, :apellidos, :correo, :telefono, :fecha_nacimiento, :comunidad_autonoma,
                :provincia, :visita_prado, :aceptar_terminos, :es_administrador, :imagen)";

        // Preparar la consulta
        $stmt = $conexion->prepare($sql);
        // Enlazar los parámetros
        $stmt->bindParam(':nombreusuario', $nombreusuario);
        $stmt->bindParam(':contrasenia', $hashed_password);
        $stmt->bindParam(':nombre', $nombre);
        $stmt->bindParam(':apellidos', $apellidos);
        $stmt->bindParam(':correo', $correo);
        $stmt->bindParam(':telefono', $telefono);
        $stmt->bindParam(':fecha_nacimiento', $fecha_nacimiento);
        $stmt->bindParam(':comunidad_autonoma', $comunidad_autonoma);
        $stmt->bindParam(':provincia', $provincia);
        $stmt->bindParam(':visita_prado', $visita_prado);
        $stmt->bindParam(':aceptar_terminos', $aceptar_terminos);
        $stmt->bindParam(':es_administrador', $es_administrador);
        $stmt->bindParam(':imagen', $imagen);
```

```php
        // Ejecutar la consulta
        $stmt->execute();

        // Iniciar sesión automáticamente después de registrar al usuario
        echo '<form id="loginForm" action="iniciosesion.php" method="post" style="display: none;">
        <input type="hidden" name="usuario" value="' . $nombreusuario . '">
        <input type="hidden" name="contraseña" value="' . $contrasenia . '">
        <input type="hidden" name="registro" value="true">
        </form>
        <script type="text/javascript">
            document.getElementById("loginForm").submit();
        </script>';

        exit();

    } catch (PDOException $e) {
        // Manejo de errores
        echo "Error: " . $e->getMessage();
    }
}
?>
```

Este código PHP se ejecuta cuando se envía el formulario de registro mediante el método POST. En primer lugar se recuperan los datos del formulario, pudiendo destacar que he utilizado la función password_hash para cifrar la contraseña y así proporcionar más seguridad. Posteriormente, se establece una conexión a la base de datos y se prepara la consulta SQL para insertar los datos del nuevo usuario en la tabla Usuarios. Los parámetros se vinculan a los valores obtenidos del formulario utilizando bindParam(). Después de ejecutar la consulta, se genera un formulario oculto de inicio de sesión que se completa automáticamente con el nombre de usuario y la contraseña proporcionados en el formulario de registro. Este formulario se envía automáticamente mediante JavaScript. Luego, el script PHP termina la ejecución utilizando exit(), por lo que cualquier código después de este punto no se ejecutará. En caso de que ocurra alguna excepción durante el proceso, se captura y se muestra un mensaje de error.

3. Inicio de sesión y cierre de sesión

iniciosesión.php

En este archivo aparece el código php encargado de iniciar la sesión del usuario, el cual se llama cuando se procesa el formulario de inicio de sesión.

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    require 'conexionbd.php';

    $nombreusuario = $_POST['usuario'];
    $contrasenia = $_POST['contraseña'];
    $es_registro = isset($_POST['registro']) && $_POST['registro'] == 'true';

    $sql = "SELECT * FROM Usuarios WHERE nombreusuario = :nombreusuario";

    $stmt = $conexion->prepare($sql);
    $stmt->bindParam(':nombreusuario', $nombreusuario);
    $stmt->execute();
```

```php
    if ($stmt->rowCount() > 0) {
        $usuario = $stmt->fetch(PDO::FETCH_ASSOC);

        // Verificar la contraseña
        if (password_verify($contrasenia, $usuario['contrasenia'])) {
            $_SESSION['usuario'] = $nombreusuario;
            $_SESSION['es_administrador'] = $usuario['es_administrador'];
            $_SESSION['mensaje_bienvenida'] = "Bienvenido, $nombreusuario";
            $_SESSION['imagen'] = $usuario['imagen'];

            if ($es_registro) {
                header("Location: altaexitosa.php");
            } else {
                header("Location: index.php");
            }
            unset($_SESSION['error_login']);
            exit();
        } else {
            $_SESSION['error_login'] = "Usuario o contraseña incorrecto.";
            header("Location: index.php");
        }
    } else {
        $_SESSION['error_login'] = "Usuario o contraseña incorrecto.";
        header("Location: index.php");
    }

    $conexion = null;
}

?>
```

En primer lugar verifico si la sesión está iniciada, y en caso contrario la inicio. Luego, compruebo si se ha enviado una solicitud POST, lo que indica un intento de inicio de sesión. Si es así, se conecta a la base de datos y busca al usuario por su nombre de usuario. Si se encuentra un usuario, se procede a verificar si la contraseña proporcionada coincide con la almacenada en la base de datos utilizando

password_verify(). Si las credenciales son correctas, se establecen las variables de sesión, las cuales son usuario, si es administrador (para comprobar así si puede dar de alta obras o no), la imagen del usuario y un mensaje de bienvenida. Posteriormente, se redirige al usuario a la página correspondiente, ya sea la de éxito en el registro o el inicio. Sin embargo, si las credenciales son incorrectas, se establece un mensaje de error de inicio de sesión en la variable de sesión error_login y se redirige de nuevo a la página de inicio. Finalmente, se cierra la conexión a la base de datos.

formularioinicio.php

En primer lugar, el formulario de inicio se sesión lo he incluido en un archivo aparte llamado formularioinicio.php, el cual he incluido en el resto de archivos con require 'formularioinicio.php'

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
?>
<!doctype html>
<html lang="es">
<head>
    <script>
        function validarFormulario(event) {

            const usuario = document.getElementById('usuario').value;
            const contrasena = document.getElementById('contraseña').value;

            if (!usuario || !contrasena) {
                alert('Ambos campos son obligatorios.');
                event.preventDefault();
            }
        }
    </script>
</head>
```

En primer lugar aparece el código javascript para validar el formulario y que los campos hayan sido rellenados.

```php
<body>
    <?php if (!isset($_SESSION['usuario'])): ?>
        <form class="formulariousuario" method="post" action="iniciosesion.php" onsubmit="validarFormulario(event)">
            <label for="usuario">Usuario:</label>
            <input type="text" name="usuario" id="usuario">

            <label for="contraseña">Contraseña:</label>
            <input type="password" name="contraseña" id="contraseña">

            <input type="submit" value="Iniciar sesión" class="iniciosesion">

            <?php
                if (isset($_SESSION['error_login'])) {
                    echo '<p class="error">' . $_SESSION['error_login'] . '</p>';
                    unset($_SESSION['error_login']);
                }
            ?>

            <p>¿No dispones de cuenta?</p>
            <a href="altausuarios.php">Regístrate</a>
        </form>
    <?php else: ?>
        <section id="usuario-inicio">
            <article id="iconousuario">
                <a href="usuario.php" >
                    <img src="imagenes/usuario.jpg" alt="icono usuario" id="icono-usuario">
                </a>
            </article>
            <figure id="usuariofigure">
                <img src="<?php echo $_SESSION['imagen']; ?>" alt="Imagen del usuario" id="imagenusuario">
                <p class="mensaje_bienvenida"><?php echo $_SESSION['mensaje_bienvenida']; ?></p>
            </figure>
            <form method="post" action="logout.php">
                <input type="submit" value="Cerrar sesión" class="cerrarsesion">
            </form>
        </section>
    <?php endif; ?>
</body>
</html>
```

En el código PHP, establezco un if else, y en el caso de que la sesión no esté iniciada, muestro el formulario para que el usuario inicie sesión y en el caso de que esté iniciada, muestro la ficha del usuario. En el caso de que el introducir la contraseña y el usuario, estos no coincidan en la base de datos, se imprime la variable de sesión error_login indicando que el usuario o la contraseña son incorrectos. En la ficha de usuario se encuentra un icono que nos dirije al archivo usuario.php, lo cual he añadido como algo opcional que explicaré más adelante, y también se encuentra el botón cerrar sesión que procesará el archivo logout.php para así cerrar la sesión.

logout.php

```php
<?php
// Iniciar sesión
session_start();

// Destruir todas las variables de sesión
$_SESSION = array();

//Borrar cookies
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}

// Destruir la sesión
session_destroy();

header("Location: index.php");
exit();
?>
```

En primer lugar inicia una sesión y luego destruye todas las variables de sesión al vaciar el array $_SESSION. En caso de que se usen cookies, se elimina la cookie de sesión. Para terminar, se destruye la sesión y se redirige al usuario a la página "index.php".

4. Gestión de las obras

Para la gestión de las obras, puesto que sólo los usuarios administradores pueden gestionarlas, en colección he añadido el botón añadir obra sólo en el caso de que la variable de sesión es_administrador sea 1 y el usuario esté identificado:

```
Unset
<?php
    if (isset($_SESSION['usuario']) && $_SESSION['es_administrador'] == 1) {
            echo '<a href="altaobra.php" id="addobra">Añadir obra</a>';
    }
?>
```

y en el archivo correspondiente a cada obra he añadido el botón modificar obra y eliminar obra utilizando las mismas condiciones que antes:

```
Unset
if (isset($_SESSION['usuario']) && $_SESSION['es_administrador'] == 1) {
    echo '<a href="#" onclick="confirmarEliminar(' . $obra['id'] . ')"
id="boton-eliminar-obra">Eliminar</a>';
    echo '<a href="editarobra.php?id=' . $obra['id'] . '"
id="boton-editar-obra">Editar</a>';
          }
```

altaobra.php

En este archivo, he añadido básicamente un formulario para que el administrador realice el alta de obras. El formulario se procesa en altaobrabd.php, que se explicará posteriormente.

```
<section id="altaobra">
    <h2>Dar de alta una obra</h2>
    <form id="form-alta-obra" method="post" action="altaobrabd.php" onsubmit="validarFormulario(event)">
        <label for="titulo">Título:</label>
        <input type="text" name="titulo" id="titulo">

        <label for="autor">Autor:</label>
        <input type="text" name="autor" id="autor">

        <label for="epoca">Época:</label>
        <input type="text" name="epoca" id="epoca">

        <label for="tema">Tema:</label>
        <input type="text" name="tema" id="tema">

        <label for="imagen">Imagen (URL en el servidor):</label>
        <input type="text" name="imagen" id="imagen">

        <label for="descripcion">Descripción:</label>
        <textarea name="descripcion" id="descripcion"></textarea>

        <input type="submit" value="Dar de alta" id="boton-alta-obra">
    </form>
</section>
```

y he validado el formulario con la siguiente función de javascript:

```javascript
function validarFormulario(event) {
    event.preventDefault(); // Prevenir el envío del formulario

    // Obtener los valores de los campos
    const titulo = document.getElementById('titulo').value.trim();
    const autor = document.getElementById('autor').value.trim();
    const epoca = document.getElementById('epoca').value.trim();
    const tema = document.getElementById('tema').value.trim();
    let imagen = document.getElementById('imagen').value.trim();
    const descripcion = document.getElementById('descripcion').value.trim();

    // Validar que los campos no estén vacíos
    if (!titulo || !autor || !epoca || !tema || !imagen || !descripcion) {
        alert('Todos los campos son obligatorios.');
        return;
    }

    // Validar que la URL de la imagen tenga la forma correcta
    const baseURL = 'http://bahia.ugr.es/~angustias16/pe2/imagenes/';
    if (!imagen.startsWith(baseURL)) {
        imagen = baseURL + imagen;
    }

    // Actualizar el campo de la imagen con la URL completa
    document.getElementById('imagen').value = imagen;

    // Validar si el título ya existe en la base de datos
    const xhr = new XMLHttpRequest();
    xhr.open('POST', 'validartituloObra.php', true);
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.onload = function() {
        if (xhr.status === 200) {
            const response = JSON.parse(xhr.responseText);
            if (response.exists) {
                alert('El título ya existe en la base de datos.');
            } else {
                // Si el título no existe, enviar el formulario
                document.getElementById('form-alta-obra').submit();
            }
        } else {
            alert('Error al verificar el título. Inténtelo de nuevo más tarde.');
        }
    };
    xhr.send('titulo=' + encodeURIComponent(titulo));
}
```

Esta función previene el envío automático del formulario, obtiene y valida los valores de los campos del formulario asegurándose de que no estén vacíos, verifica si la URL de la imagen es correcta y la completa si es necesario. Luego, crea una solicitud XMLHttpRequest para verificar si el título de la obra ya existe en la base de datos enviando una solicitud POST a validartituloObra.php. Si la respuesta indica que el título ya existe, muestra un mensaje de alerta; de lo contrario, si el título es único, envía el formulario.

validartituloObra.php

El código de este archivo recibe un título de obra enviado mediante una solicitud POST y verifica si dicho título ya existe en la base de datos. Utiliza una consulta preparada para contar las filas en la tabla Obras donde el título coincide con el proporcionado. Si encuentra al menos una coincidencia,

actualiza una respuesta JSON indicando que el título ya existe. En caso de error envía una respuesta JSON con el mensaje de error.

```php
<?php
require 'conexionbd.php'; // Archivo de conexión a la base de datos

$titulo = $_POST['titulo'];
$response = ['exists' => false];

try {
    $stmt = $conexion->prepare("SELECT COUNT(*) FROM Obras WHERE titulo = :titulo");
    $stmt->bindParam(':titulo', $titulo, PDO::PARAM_STR);
    $stmt->execute();
    $count = $stmt->fetchColumn();

    if ($count > 0) {
        $response['exists'] = true;
    }

    echo json_encode($response);
} catch (PDOException $e) {
    // Registrar el error en un archivo
    echo json_encode(['error' => $e->getMessage()]);
}
?>
```

altaobrabd.php

Este es el código al que se redirige el formulario de altaobra.php para así registrar la obra en la base de datos.

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener los datos del formulario
    $titulo = $_POST['titulo'];
    $autor = $_POST['autor'];
    $epoca = $_POST['epoca'];
    $imagen = $_POST['imagen']; // URL de la imagen en el servidor
    $tema = $_POST['tema'];
    $descripcion = $_POST['descripcion'];

    // Insertar los datos en la base de datos
    try {
        // Conexión a la base de datos
        require 'conexionbd.php';

        // Consulta SQL para insertar la nueva obra
        $sql = "INSERT INTO Obras (titulo, autor, epoca, imagen, tema, descripcion) VALUES (:titulo, :autor, :epoca, :imagen, :tema, :descripcion)";

        // Preparar la consulta
        $stmt = $conexion->prepare($sql);
        // Enlazar los parámetros
        $stmt->bindParam(':titulo', $titulo);
        $stmt->bindParam(':autor', $autor);
        $stmt->bindParam(':epoca', $epoca);
        $stmt->bindParam(':imagen', $imagen);
        $stmt->bindParam(':tema', $tema);
        $stmt->bindParam(':descripcion', $descripcion);

        // Ejecutar la consulta
        $stmt->execute();

        // Redirigir a una página de éxito
        header("Location: coleccion.php");
        exit();
    } catch (PDOException $e) {
        // Manejo de errores
        echo "Error: " . $e->getMessage();
    }
}
?>
```

Este archivo maneja el envío del formulario para agregar una nueva obra a una base de datos, obteniendo los datos del formulario (título, autor, época, imagen, tema y descripción), estableciendo una conexión a la base de datos a través de un archivo de conexión separado, preparando y ejecutando

la consulta SQL para insertar estos datos en la tabla Obras, y finalmente redirigiendo al usuario a coleccion.php en el caso de que se haya insertado correctamente; si ocurre un error durante el proceso, se muestra el mensaje de error correspondiente.

editarobra.php

Este archivo es prácticamente igual que altaobra.php, lo único que en los campos del formulario, ya aparecen los datos de la obra en cuestión. Al intentar realizar algún cambio, se comprueba de la misma forma que en altaobra.php si el título ya está asignado a otra obra, y en ese caso, no se envía el formulario.

editarobrabd.php

```php
<?php
require 'conexionbd.php';

// Obtener los datos del formulario
$id = isset($_POST['id']) ? $_POST['id'] : null;
$titulo = isset($_POST['titulo']) ? $_POST['titulo'] : null;
$autor = isset($_POST['autor']) ? $_POST['autor'] : null;
$tema = isset($_POST['tema']) ? $_POST['tema'] : null;
$epoca = isset($_POST['epoca']) ? $_POST['epoca'] : null;
$descripcion = isset($_POST['descripcion']) ? $_POST['descripcion'] : null;
$imagen = isset($_POST['imagen']) ? $_POST['imagen'] : null;

if ($id && $titulo && $autor && $tema && $epoca && $descripcion && $imagen) {
    try {
        $sql = "UPDATE Obras SET titulo = :titulo, autor = :autor, tema = :tema, epoca = :epoca, descripcion = :descripcion, imagen = :imagen WHERE id = :id";
        $stmt = $conexion->prepare($sql);
        $stmt->bindValue(':id', $id, PDO::PARAM_INT);
        $stmt->bindValue(':titulo', $titulo, PDO::PARAM_STR);
        $stmt->bindValue(':autor', $autor, PDO::PARAM_STR);
        $stmt->bindValue(':tema', $tema, PDO::PARAM_STR);
        $stmt->bindValue(':epoca', $epoca, PDO::PARAM_STR);
        $stmt->bindValue(':descripcion', $descripcion, PDO::PARAM_STR);
        $stmt->bindValue(':imagen', $imagen, PDO::PARAM_STR);
        $stmt->execute();

        // Redirigir a la página de la obra después de actualizarla
        header('Location: obra.php?id=' . $id);
        exit;
    } catch (PDOException $e) {
        echo "Error al actualizar la obra: " . $e->getMessage();
    }
} else {
    echo "No se proporcionaron todos los datos necesarios para actualizar la obra.";
}

$conexion = null;
?>
```

El código en primer lugar obtiene los campos del formulario, comprueba que todos estén presentes, y actualiza los campos con la sentencia sql correspondiente. Si los campos se han actualizado correctamente, nos redirige a la página de la obra correspondiente.

eliminarobra.php

```php
<?php
require 'conexionbd.php';

// Obtener el ID de la obra de la URL
$id = isset($_GET['id']) ? $_GET['id'] : null;

if ($id) {
    try {
        $sql = "DELETE FROM Obras WHERE id = :id";
        $stmt = $conexion->prepare($sql);
        $stmt->bindValue(':id', $id, PDO::PARAM_INT);
        $stmt->execute();

        // Redirigir a la página de la colección después de eliminar la obra
        header('Location: coleccion.php');
        exit;
    } catch (PDOException $e) {
        echo "Error al eliminar la obra: " . $e->getMessage();
    }
} else {
    echo "No se proporcionó el ID de la obra.";
}

$conexion = null;
?>
```

En primer lugar obtiene el id de la obra en la URL mediante el método GET, y en caso de que se haya proporcionado, se elimina esa obra de la base de datos y se redirige al usuario a coleccion.php.

5. Colecciones

colecciones.php

En primer lugar, el submenu que aparece a la izquierda se construye de forma dinámica tal y como se pide en la práctica. Para cada categoría (autores, temas y épocas), se ejecuta una consulta SQL que selecciona los valores distintos en la tabla Obras, y los resultados se procesan dentro de bloques PHP que crean elementos de lista HTML con enlaces a una página de colección filtrada por el respectivo autor, tema o época, utilizando urlencode para asegurar que los valores en los enlaces sean válidos, y htmlspecialchars para evitar inyecciones de HTML, manejando posibles excepciones de la base de datos mediante bloques try-catch que capturan y muestran errores.

```php
<nav class="submenu">
    <ul>
        <li>
            <a href="#" class="autores">Autores</a>
            <ul>
                <?php
                    require 'conexionbd.php';

                    try {
                        $sql_autores = "SELECT DISTINCT autor FROM Obras";
                        $stmt_autores = $conexion->prepare($sql_autores);
                        $stmt_autores->execute();
                        $autores = $stmt_autores->fetchAll(PDO::FETCH_ASSOC);

                        foreach ($autores as $autor) {
                            echo '<li><a href="coleccion.php?autor=' . urlencode($autor['autor']) . '">' . htmlspecialchars($autor['autor']) . '</a></li>';
                        }
                    } catch (PDOException $e) {
                        echo "Error: " . $e->getMessage();
                    }
                ?>
            </ul>
        </li>
        <li>
            <a href="#" class="tema">Tema</a>
            <ul>
                <?php
                    try {
                        $sql_temas = "SELECT DISTINCT tema FROM Obras";
                        $stmt_temas = $conexion->prepare($sql_temas);
                        $stmt_temas->execute();
                        $temas = $stmt_temas->fetchAll(PDO::FETCH_ASSOC);

                        foreach ($temas as $tema) {
                            echo '<li><a href="coleccion.php?tema=' . urlencode($tema['tema']) . '">' . htmlspecialchars($tema['tema']) . '</a></li>';
                        }
                    } catch (PDOException $e) {
                        echo "Error: " . $e->getMessage();
                    }
                ?>
            </ul>
        </li>
        <li>
            <a href="#" class="epoca">Época</a>
            <ul>
                <?php
                    try {
                        $sql_epocas = "SELECT DISTINCT epoca FROM Obras";
                        $stmt_epocas = $conexion->prepare($sql_epocas);
                        $stmt_epocas->execute();
                        $epocas = $stmt_epocas->fetchAll(PDO::FETCH_ASSOC);


                        foreach ($epocas as $epoca) {
                            echo '<li><a href="coleccion.php?epoca=' . urlencode($epoca['epoca']) . '">' . htmlspecialchars($epoca['epoca']) . '</a></li>';
                        }

                    } catch (PDOException $e) {
                        echo "Error: " . $e->getMessage();
                    }
                ?>
            </ul>
        </li>
    </ul>
</nav>
<?php
```

```php
<section class="colecciones">
    <?php
        require 'conexionbd.php';

        // Obtener los parámetros de la URL
        $epoca = isset($_GET['epoca']) ? $_GET['epoca'] : null;
        $autor = isset($_GET['autor']) ? $_GET['autor'] : null;
        $tema = isset($_GET['tema']) ? $_GET['tema'] : null;

        // Definir el número de obras por página
        $obras_por_pagina = 9;

        // Obtener la página actual de la URL (si no está establecida, la página es 1)
        $pagina_actual = isset($_GET['pagina']) ? (int)$_GET['pagina'] : 1;

        // Calcular el índice del primer obra en esta página
        $primer_obra = ($pagina_actual - 1) * $obras_por_pagina;

        try {
            // Construir la consulta SQL según los parámetros
            $sql = "SELECT * FROM Obras WHERE 1";
            if ($epoca) {
                $sql .= " AND epoca = :epoca";
            }
            if ($autor) {
                $sql .= " AND autor = :autor";
            }
            if ($tema) {
                $sql .= " AND tema = :tema";
            }

            // Modificar la consulta SQL para limitar el número de resultados
            $sql .= " LIMIT :primer_obra, :obras_por_pagina";

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

```php
            // Asignar los valores a los placeholders (si los hay)
            if ($epoca) {
                $stmt->bindValue(':epoca', $epoca, PDO::PARAM_STR);
            }
            if ($autor) {
                $stmt->bindValue(':autor', $autor, PDO::PARAM_STR);
            }
            if ($tema) {
                $stmt->bindValue(':tema', $tema, PDO::PARAM_STR);
            }

            $stmt->bindValue(':primer_obra', $primer_obra, PDO::PARAM_INT);
            $stmt->bindValue(':obras_por_pagina', $obras_por_pagina, PDO::PARAM_INT);

            $stmt->execute();
            $obras = $stmt->fetchAll(PDO::FETCH_ASSOC);

            foreach ($obras as $obra) {
                echo '<a href="obra.php?id=' . $obra['id'] . '"class="obrax">';
                echo '<figure>';
                echo '<img src="' . $obra['imagen'] . '" alt="Descripción de la imagen">';
                echo '<figcaption>';
                echo '<h4>' . htmlspecialchars($obra['titulo']) . '</h4>';
                echo '<p>' . htmlspecialchars($obra['autor']) . '</p>';
                echo '<p>' . htmlspecialchars($obra['epoca']) . '</p>';
                echo '</figcaption>';
                echo '</figure>';
                echo '</a>';
            }

        } catch (PDOException $e) {
            echo "Error: " . $e->getMessage();
        }
    ?>
```

Las dos imágenes anteriores muestran el código de gestión de las colecciones según los parámetros de la URL. En primer lugar se obtienen los parámetros de la URL, posteriormente se define el número de obras por página y también se obtiene la página actual de la URL (que en caso de que no haya es 1) y finalmente se calcula el índice de la primera obra en esa página. Luego, se prepara la consulta y se añade lo necesario en caso de que se haya proporcionado en la URL autor, tema o época y se limita el

número de resultados por página. Seguidamente se añaden los valores a los placeholders y se ejecuta la consulta y se obtienen las filas en obras usando fetchAll(). Finalmente se itera sobre cada obra y se imprimen por pantalla en sus correspondientes figure. En el enlace de obra se añade el id de la obra para así poder acceder a ella en el archivo obra.php.

```php
<section id="paginacion">
    <?php
        try {
            // Calcular el número total de páginas considerando los filtros aplicados
            $sql_total = "SELECT COUNT(*) as total FROM Obras WHERE 1";
            if ($epoca) {
                $sql_total .= " AND epoca = :epoca";
            }
            if ($autor) {
                $sql_total .= " AND autor = :autor";
            }
            if ($tema) {
                $sql_total .= " AND tema = :tema";
            }

            $stmt_total = $conexion->prepare($sql_total);

            if ($epoca) {
                $stmt_total->bindValue(':epoca', $epoca, PDO::PARAM_STR);
            }
            if ($autor) {
                $stmt_total->bindValue(':autor', $autor, PDO::PARAM_STR);
            }
            if ($tema) {
                $stmt_total->bindValue(':tema', $tema, PDO::PARAM_STR);
            }

            $stmt_total->execute();
            $total_obras = $stmt_total->fetch(PDO::FETCH_ASSOC)['total'];
            $total_paginas = ceil($total_obras / $obras_por_pagina);

            // Mostrar los botones de paginación
            if ($pagina_actual > 1) {
                echo '<a href="coleccion.php?pagina=' . ($pagina_actual - 1);
                if ($epoca) echo '&epoca=' . urlencode($epoca);
                if ($autor) echo '&autor=' . urlencode($autor);
                if ($tema) echo '&tema=' . urlencode($tema);
                echo '" id="boton-anterior">Anterior</a>';
            }
            if ($pagina_actual < $total_paginas) {
                echo '<a href="coleccion.php?pagina=' . ($pagina_actual + 1);
                if ($epoca) echo '&epoca=' . urlencode($epoca);
                if ($autor) echo '&autor=' . urlencode($autor);
                if ($tema) echo '&tema=' . urlencode($tema);
                echo '" id="boton-siguiente">Siguiente</a>';
            }

        } catch (PDOException $e) {
            echo "Error: " . $e->getMessage();
        }
        $conexion = null;
    ?>
</section>
```

En la imagen anterior se realiza la gestión de la paginación de 9 obras por página. En primer lugar, e construye una consulta SQL para contar el total de obras que cumplen con los filtros (epoca, autor, tema) aplicados. Después, se prepara la consulta y se enlazan los valores de los parámetros si los filtros están definidos. Luego, se ejecuta la consulta para contar el total de obras y se calcula el número total de páginas, dividiendo el total de obras por el número de obras por página y redondeando hacia arriba. Para terminar, se generan los enlaces HTML para los botones "Anterior" y "Siguiente", incluyendo los parámetros de la URL (pagina, epoca, autor, tema) para mantener los filtros aplicados. Estos enlaces sólo se generan si hay más páginas para mostrar.

Finalmente en el archivo coleccion.php está la función de javascript encargada de que al pasar el ratón por la imagen asociada a cada obra, aparezca una ventana emergente con el título de la obra y la categoría a la que pertenece.

```
<script>
    var figures = document.querySelectorAll('figure');
    var tooltip = document.querySelector('.tooltip');

    for (var i = 0; i < figures.length; i++) {
        figures[i].addEventListener('mouseover', function(event) {
            var titulo = this.querySelector('h4').textContent;
            var categoria = this.querySelector('p').textContent;

            tooltip.innerHTML = '<h4>' + titulo + '</h4><p>Categoría: ' + categoria + '</p>';
            tooltip.style.display = 'block';

            // Posiciona la tooltip a la derecha del cursor
            tooltip.style.left = event.pageX + 10 + 'px';
            tooltip.style.top = event.pageY + 10 + 'px';
        });

        figures[i].addEventListener('mouseout', function() {
            tooltip.style.display = 'none';
        });
    }
</script>
```

Primero se seleccionan todos los elementos <figure> del documento y luego se selecciona el primer elemento con la clase tooltip del documento. Se inicia un bucle que recorre cada elemento <figure> seleccionado anteriormente y para cada figura, se añade un event listener que escucha el evento mouseover. Cuando se activa el evento mouseover, se extrae el texto del título (dentro de una etiqueta <h4>) y la categoría (dentro de una etiqueta <p>) de la figura sobre la que se pasa el ratón. Luego se actualiza el contenido del tooltip con el título y la categoría extraídos. Se hace visible el tooltip configurando su estilo display a 'block' y e posiciona el tooltip cerca del cursor del ratón, ajustando las propiedades left y top en base a las coordenadas del evento pageX y pageY. Finalmente, Para cada figura, se añade un event listener que escucha el evento mouseout y cuando se activa el evento mouseout, se oculta el tooltip configurando su estilo display a 'none'.

obra.php

```
<main>
  <?php
    require 'conexionbd.php';

    // Obtener el ID de la obra de la URL
    $id = isset($_GET['id']) ? $_GET['id'] : null;

    if ($id) {
      try {
        $sql = "SELECT * FROM Obras WHERE id = :id";
        $stmt = $conexion->prepare($sql);
        $stmt->bindValue(':id', $id, PDO::PARAM_INT);
        $stmt->execute();
        $obra = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($obra) {
          // Mostrar la información de la obra
          echo '<figure>';
          echo '<img src="' . $obra['imagen'] . '" class="imagenobra" alt="Imagen de la obra">';
          echo '</figure>';
          echo '<section class="fichayobra">';
          echo '<article class="ficha-tecnica">';
          echo '<h3>' . htmlspecialchars($obra['titulo']) . '</h3>';
          echo '<p>' . htmlspecialchars($obra['autor']) . '</p>';
          echo '<p>' . htmlspecialchars($obra['tema']) . '</p>';
          echo '<p>' . htmlspecialchars($obra['epoca']) . '</p>';
          echo '<p>' . htmlspecialchars($obra['descripcion']) . '</p>';
          echo '</article>';
          echo '<aside class="obrasrelacionadas">';
          echo '<nav>';
          echo '<h3>Obras Relacionadas</h3>';
          echo '<ul>';

          // Obtener las obras relacionadas con el mismo tema
          $tema = $obra['tema'];
          $sql_relacionadas = "SELECT * FROM Obras WHERE tema = :tema AND id != :id";
          $stmt_relacionadas = $conexion->prepare($sql_relacionadas);
          $stmt_relacionadas->bindValue(':tema', $tema, PDO::PARAM_STR);
          $stmt_relacionadas->bindValue(':id', $id, PDO::PARAM_INT);
          $stmt_relacionadas->execute();
          $obras_relacionadas = $stmt_relacionadas->fetchAll(PDO::FETCH_ASSOC);
```

```php
            foreach ($obras_relacionadas as $obra_relacionada) {
            echo '<li><a href="obra.php?id=' . $obra_relacionada['id'] . '">' . htmlspecialchars($obra_relacionada['titulo']) . '</a></li>';
            }

            echo '</ul>';
            echo '</nav>';
            echo '</aside>';
            echo '</section>';
        }
        else {
            echo '<p>No se encontró la obra.</p>';
        }
    }
    catch (PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
    $conexion = null;
}
else {
    echo '<p>No se proporcionó el ID de la obra.</p>';
}
// Mostrar el botón de eliminar dentro del figure si el usuario es administrador
if (isset($_SESSION['usuario']) && $_SESSION['es_administrador'] == 1) {
    echo '<a href="#" onclick="confirmarEliminar(' . $obra['id'] . ')" id="boton-eliminar-obra">Eliminar</a>';
    echo '<a href="editarobra.php?id=' . $obra['id'] . '" id="boton-editar-obra">Editar</a>';
}
?>
```

Este archivo sirve para mostrar cada obra independientemente. En primer lugar se obtiene el id de la URL, y si este no es nulo, se ejecuta el try. Luego se prepara la consulta SQL para seleccionar la obra con el ID especificado y se ejecuta la consulta y se almacena el resultado en $obra. Después se genera el HTML para mostrar la imagen y los detalles de la obra (título, autor, tema, época y descripción). Después aparece el código para mostrar las obras relacionadas, donde en primer lugar se obtiene el tema de la obra actual y se realiza una consulta SQL para seleccionar las obras con el mismo tema, excluyendo la obra actual. Se itera sobre las obras relacionadas y se van mostrando en el aside.

Finalmente, al final del documento se muestra el botón eliminar obra, que tiene una verificación en una función de javascript muy simple, y el botón editar obra, todo esto en caso de que el usuario haya iniciado sesión y sea administrador.

6. Experiencias

experiencias.php

```php
<main id="opinionesysugerencias">
    <section id="opiniones">
        <h2>Opiniones y sugerencias</h2>
        <?php
        require 'conexionbd.php';
        $sql = "SELECT Comentarios.usuario, Comentarios.sugerencia, Usuarios.imagen
                FROM Comentarios
                INNER JOIN Usuarios ON Comentarios.usuario = Usuarios.nombreusuario
                ORDER BY Comentarios.fecha_comentario DESC";

        $stmt = $conexion->prepare($sql);
        $stmt->execute();
        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            echo '<article>';
            echo '<figure class="fotoynombre">';
            echo '<img src="' . htmlspecialchars($row['imagen']) . '" alt="Foto de usuario">';
            echo '<figcaption>' . htmlspecialchars($row['usuario']) . '</figcaption>';
            echo '</figure>';
            echo '<p>' . htmlspecialchars($row['sugerencia']) . '</p>';
            echo '</article>';
        }
        ?>
    </section>
    <section id="sugerencias">
        <h2>Buzón</h2>
        <p>¿Te gustaría enviar una sugerencia u opinión?</p>
        <?php
            if (isset($_SESSION['usuario'])) { // Verificar si el usuario está logeado
                ?>
                <form id="form-sugerencia" action="registrosugerencia.php" method="post">
                    <label for="sugerencia">Sugerencia/opinión:</label>
                    <textarea id="sugerencia" name="sugerencia" ></textarea>
                    <input type="submit" value="Enviar sugerencia">
                </form>
                <?php
            } else {
                echo "<p>Para enviar una sugerencia, debes iniciar sesión.</p>";
            }
        ?>
    </section>
</main>
```

En el <section> opiniones, en primer lugar se realiza una consulta a la base de datos (tablas Comentarios y Usuario) para obtener los datos del comentario (usuario, sugerencia, imagen y fecha_comentario). Para cada fila obtenida en $row, se va mostrando cada sugerencia en un article. Por otra parte, en el segundo <section>, sólo se muestra el text area de sugerencia en caso de que el usuario haya iniciado sesión. En caso de que el usuario haya inciado sesión, se procesará el formulario form-sugerencia ejecutando el código en registrosugerencia.php. El anterior formulario es validado por la siguiente función JavaScript:

```javascript
<script>
    document.getElementById('form-sugerencia').addEventListener('submit', function(event) {
        var sugerencia = document.getElementById('sugerencia').value.trim();
        if (sugerencia.length < 50) {
            event.preventDefault(); // Evita el envío del formulario
            alert('Por favor, ingresa una sugerencia de al menos 50 caracteres antes de enviar.');
        }
    });
</script>
```

la cual previene el envío del formulario en caso de que el número de caracteres sea menor que 50.

registrosugerencia.php

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    require 'conexionbd.php';

    $sugerencia = $_POST['sugerencia'];
    $fecha_actual = date('Y-m-d H:i:s'); // Obtiene la fecha y hora actual


    $sql = "INSERT INTO Comentarios (usuario, sugerencia, fecha_comentario) VALUES (:usuario, :sugerencia, :fecha_comentario)";

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

    $stmt->bindValue(':usuario', $_SESSION['usuario']);
    $stmt->bindValue(':sugerencia', $sugerencia);
    $stmt->bindValue(':fecha_comentario', $fecha_actual); // Vincula la fecha actual


    if ($stmt->execute()) {
        header("Location: experiencias.php"); // Redirige de nuevo a la página de experiencias
    } else {
        echo "<p>Error al enviar la sugerencia.</p>";
    }

    $stmt = null;
    $conexion = null;
}
?>
```

Este archivo en primer lugar verifica si la solicitud es de tipo POST, indicando que el formulario fue enviado. Si es así, obtiene la sugerencia del formulario y la fecha y hora actual. Prepara una consulta SQL para insertar la sugerencia en la tabla Comentarios, vinculando los valores del usuario actual de la sesión, la sugerencia y la fecha. Ejecuta la consulta y, si tiene éxito, redirige al usuario de nuevo a la página de experiencias. Si la ejecución falla, muestra un mensaje de error. Finalmente, cierra la declaración y la conexión a la base de datos.

7. Gestión de usuarios

Como innovación, he añadido la gestión de usuarios, donde cada usuario podrá modificar sus datos, eliminar su cuenta y también eliminar las sugerencias que ha realizado.

usuario.php

En ese documento, he añadido un formulario donde en cada campo aparecen los datos del usuario, y el usuario registrado podrá modificarlos. La siguiente sección de código obtiene los datos del usuario de la base de datos y los guarda en variables para posteriormente mostrarlos en el formulario. Los comentarios los he ordenado para que primero salgan los más recientes.

```php
<?php
    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }

    // Redirigir a la página de inicio de sesión si no hay un usuario en la sesión
    if (!isset($_SESSION['usuario'])) {
        header("Location: index.php");
        exit();
    }

    // Conexión a la base de datos
    require 'conexionbd.php';

    try {

        // Obtener los datos del usuario
        $nombreusuario = $_SESSION['usuario'];
        $sql_usuario = "SELECT * FROM Usuarios WHERE nombreusuario = :nombreusuario";
        $stmt_usuario = $conexion->prepare($sql_usuario);
        $stmt_usuario->bindParam(':nombreusuario', $nombreusuario);
        $stmt_usuario->execute();
        $usuario = $stmt_usuario->fetch(PDO::FETCH_ASSOC);

        // Obtener los comentarios del usuario
        $sql_comentarios = "SELECT * FROM Comentarios WHERE usuario = :nombreusuario ORDER BY fecha_comentario DESC";
        $stmt_comentarios = $conexion->prepare($sql_comentarios);
        $stmt_comentarios->bindParam(':nombreusuario', $nombreusuario);
        $stmt_comentarios->execute();
        $comentarios = $stmt_comentarios->fetchAll(PDO::FETCH_ASSOC);

    } catch (PDOException $e) {
        echo "Error: " . $e->getMessage();
        die();
    }
?>
```

El siguiente código contiene el formulario con los datos obtenidos anteriormente, y al final aparece un botón de modificar contraseña, el cual nos llevará a una página específicamente para modificar la contraseña, y un botón para eliminar la cuenta, que nos muestra un mensaje de confirmación con onclick. Este formulario se valida también con una función de Javascript tal y como se ha explicado anteriormente.

```php
<section id="modificar-usuario">
    <h2>Editar Datos del Usuario</h2>
    <section id="datos-usuario">
        <form id="formulario-modificar-usuario" method="post" action="actualizarusuario.php">
            <label for="nombre">Nombre:</label>
            <input type="text" id="nombre" name="nombre" value="<?php echo htmlspecialchars($usuario['nombre']); ?>">

            <label for="apellidos">Apellidos:</label>
            <input type="text" id="apellidos" name="apellidos" value="<?php echo htmlspecialchars($usuario['apellidos']); ?>">

            <label for="correo">Correo:</label>
            <input type="email" id="correo" name="correo" value="<?php echo htmlspecialchars($usuario['correo']); ?>">

            <label for="telefono">Teléfono:</label>
            <input type="text" id="telefono" name="telefono" value="<?php echo htmlspecialchars($usuario['telefono']); ?>">

            <label for="fecha_nacimiento">Fecha de Nacimiento:</label>
            <input type="date" id="fecha_nacimiento" name="fecha_nacimiento" value="<?php echo htmlspecialchars($usuario['fecha_nacimiento']); ?>">

            <label for="comunidad_autonoma">Comunidad Autónoma:</label>
            <input type="text" id="comunidad_autonoma" name="comunidad_autonoma" value="<?php echo htmlspecialchars($usuario['comunidad_autonoma']); ?>">

            <label for="provincia">Provincia:</label>
            <input type="text" id="provincia" name="provincia" value="<?php echo htmlspecialchars($usuario['provincia']); ?>">

            <label for="imagen">Imagen (URL):</label>
            <input type="text" id="imagen" name="imagen" value="<?php echo htmlspecialchars($usuario['imagen']); ?>">


            <input type="submit" value="Actualizar Datos" id="actualizar-datos">
        </form>
        <section id="cambios">
            <article id="cambiar-contraseña">
                <p>¿Quieres modificar la contraseña?</p>
                <a href="cambiar_contrasenia.php" id="boton-modificar-contraseña" >Modificar contraseña</a>
            </article>
            <article id="eliminar-cuenta">
                <p>¿Deseas eliminar tu cuenta?</p>
                <a href="eliminar_cuenta.php" id="boton-eliminar-cuenta" onclick="return confirm('¿Estás seguro de que deseas eliminar tu cuenta? Esta acción no se puede deshacer.');">Eliminar Cuenta</a>
            </article>
        </section>
    </section>
</section>
```

Finalmente, en este archivo incluyo los comentarios correspondientes a este usuario, y un botón en cada comentario por si el usuario quiere eliminar el mismo

```php
<section id="comentarios-usuario">
    <h2>Comentarios Realizados</h2>
    <?php if (count($comentarios) > 0): ?>
        <ul>
            <?php foreach ($comentarios as $comentario): ?>
                <article >
                    <p><?php echo htmlspecialchars($comentario['sugerencia']); ?></p>
                    <form method="post" action="eliminar_comentario.php" id="eliminarcomentario" >
                        <input type="hidden" name="comentario_id" value="<?php echo $comentario['comentario_id']; ?>">
                        <input type="submit" value="Eliminar comentario" onclick="return confirm('¿Estás seguro de que deseas eliminar el comentario?');">
                    </form>
                </article>
            <?php endforeach; ?>
        </ul>
    <?php else: ?>
        <p>No has realizado ningún comentario.</p>
    <?php endif; ?>
</section>
```

actualizarusuario.php

Este es el archivo que ejecuta cuando un usuario actualiza los datos:

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    session_start();
    require 'conexionbd.php';

    // Recibir datos del formulario
    $nombre = $_POST['nombre'];
    $apellidos = $_POST['apellidos'];
    $correo = $_POST['correo'];
    $telefono = $_POST['telefono'];
    $fecha_nacimiento = $_POST['fecha_nacimiento'];
    $comunidad_autonoma = $_POST['comunidad_autonoma'];
    $provincia = $_POST['provincia'];
    $nombreusuario = $_SESSION['usuario'];
    $imagen = $_POST['imagen'];

    // Actualizar datos
    $sql = "UPDATE Usuarios SET nombre=:nombre, apellidos=:apellidos, correo=:correo,
    telefono=:telefono, fecha_nacimiento=:fecha_nacimiento, comunidad_autonoma=:comunidad_autonoma,
    provincia=:provincia, imagen=:imagen WHERE nombreusuario=:nombreusuario";
```

```php
    $stmt = $conexion->prepare($sql);
    $stmt->bindParam(':nombre', $nombre);
    $stmt->bindParam(':apellidos', $apellidos);
    $stmt->bindParam(':correo', $correo);
    $stmt->bindParam(':telefono', $telefono);
    $stmt->bindParam(':fecha_nacimiento', $fecha_nacimiento);
    $stmt->bindParam(':comunidad_autonoma', $comunidad_autonoma);
    $stmt->bindParam(':provincia', $provincia);
    $stmt->bindParam(':nombreusuario', $nombreusuario);
    $stmt->bindParam(':imagen', $imagen);

    if ($stmt->execute()) {
        echo "Datos actualizados correctamente.";
    } else {
        echo "Error al actualizar los datos.";
    }

    $_SESSION['imagen'] = $imagen;

    header("Location: usuario.php");
    $conexion = null;
}
?>
```

en el cual se actualizan los parámetros de igual manera a la que se ha explicado anteriormente en editarobrabd.php.

cambiar_contrasenia.php

En este archivo se muestra un formulario para introducir la contraseña actual, y dos veces la nueva contraseña. Si se valida correctamente, se ejecutaría cambiar_contraseniabd.php.

```php
<main>
    <h2>Cambiar Contraseña</h2>
    <?php
        if (isset($_SESSION['mensaje_error'])) {
            echo "<p style='color:red'>" . $_SESSION['mensaje_error'] . "</p>";
            unset($_SESSION['mensaje_error']);
        }
    ?>
    <form id="form-cambiar-contrasenia" action="cambiar_contraseniabd.php" method="post" onsubmit="validarFormulario(event)">
        <label for="contrasena_actual">Contraseña Actual:</label>
        <input type="password" id="contrasena_actual" name="contrasena_actual">

        <label for="nueva_contrasena">Nueva Contraseña:</label>
        <input type="password" id="nueva_contrasena" name="nueva_contrasena">

        <label for="confirmar_contrasena">Confirmar Nueva Contraseña:</label>
        <input type="password" id="confirmar_contrasena" name="confirmar_contrasena">

        <input type="submit" value="Actualizar Contraseña">
    </form>
</main>
```

La función JavaScript que valida el formulario es la siguiente:

```
<script>
    function validarFormulario(event) {
        event.preventDefault(); // Prevenir el envío del formulario

        // Obtener los valores de los campos
        const contrasenaActual = document.getElementById('contrasena_actual').value;
        const nuevaContrasena = document.getElementById('nueva_contrasena').value;
        const confirmarContrasena = document.getElementById('confirmar_contrasena').value;

        // Validar que los campos no estén vacíos
        if (!contrasenaActual || !nuevaContrasena || !confirmarContrasena) {
            alert('Todos los campos son obligatorios.');
            return;
        }

        // Validar que las nuevas contraseñas coincidan
        if (nuevaContrasena !== confirmarContrasena) {
            alert('Las nuevas contraseñas no coinciden.');
            return;
        }

        // Realizar la validación de la contraseña actual con JavaScript (opcional)
        fetch('verificar_contraseña_actual.php', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/x-www-form-urlencoded',
            },
            body: 'contrasena_actual=' + encodeURIComponent(contrasenaActual)
        })
        .then(response => response.json())
        .then(data => {
            if (!data.valida) {
                alert('La contraseña actual no es correcta.');
            } else {
                // Si la contraseña actual es correcta, enviar el formulario
                document.getElementById('form-cambiar-contrasenia').submit();
            }
        })
        .catch(error => {
            console.error('Error al validar la contraseña:', error);
        });
    }
</script>
```

La función en primer lugar verifica que los campos no estén vacíos y luego verifica que las nuevas contraseñas coincidan. Finalmente se utiliza fetch para enviar una solicitud POST al archivo verificar_contraseña_actual.php, enviando la contraseña actual para validación y una vez que se recibe la respuesta, se convierte en JSON y se comprueba si la contraseña actual es válida (data.valida).

verificar_contraseña_actual.php

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
require 'conexionbd.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $contrasena_actual = $_POST['contrasena_actual'];
    $nombreusuario = $_SESSION['usuario'];

    $stmt = $conexion->prepare("SELECT contrasenia FROM Usuarios WHERE nombreusuario = :nombreusuario");
    $stmt->bindParam(':nombreusuario', $nombreusuario);
    $stmt->execute();
    $resultado = $stmt->fetch();

    if ($resultado && password_verify($contrasena_actual, $resultado['contrasenia'])) {
        echo json_encode(['valida' => true]);
    } else {
        echo json_encode(['valida' => false]);
    }
}
?>
```

Obtiene la contraseña actual y la verifica realizando una consulta a la base de datos. La respuesta la devuelve en JSON.

cambiar_contraseniabd.php

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
require 'conexionbd.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $contrasena_actual = $_POST['contrasena_actual'];
    $nueva_contrasena = $_POST['nueva_contrasena'];
    $confirmar_contrasena = $_POST['confirmar_contrasena'];

    // Verificar si las nuevas contraseñas coinciden
    if ($nueva_contrasena != $confirmar_contrasena) {
        $_SESSION['mensaje_error'] = 'Las nuevas contraseñas no coinciden.';
        header("Location: cambiar_contrasenia.php");
        exit;
    }

    // Obtener la contraseña actual del usuario de la base de datos
    $nombreusuario = $_SESSION['usuario'];
    $stmt = $conexion->prepare("SELECT contrasenia FROM Usuarios WHERE nombreusuario = :nombreusuario");
    $stmt->bindParam(':nombreusuario', $nombreusuario);
    $stmt->execute();
    $resultado = $stmt->fetch();

    // Verificar la contraseña actual
    if (!$resultado || !password_verify($contrasena_actual, $resultado['contrasenia'])) {
        $_SESSION['mensaje_error'] = 'La contraseña actual no es correcta.';
        header("Location: cambiar_contrasenia.php");
        exit;
    }

    // Actualizar la contraseña
    $nueva_contrasena_hash = password_hash($nueva_contrasena, PASSWORD_DEFAULT);
    $stmt = $conexion->prepare("UPDATE Usuarios SET contrasenia = :nueva_contrasena WHERE nombreusuario = :nombreusuario");
    $stmt->bindParam(':nueva_contrasena', $nueva_contrasena_hash);
    $stmt->bindParam(':nombreusuario', $nombreusuario);
    if ($stmt->execute()) {
        header("Location: usuario.php");
    } else {
        $_SESSION['mensaje_error'] = 'Error al actualizar la contraseña.';
        header("Location: cambiar_contrasenia.php");
    }
}
?>
```

El código anterior verifica si la solicitud proviene de un formulario enviado mediante el método POST. Extrae las contraseñas actual y nuevas del formulario y verifica si las nuevas coinciden. Si no coinciden, establece un mensaje de error en la sesión y redirige a la página de cambio de contraseña. Luego, obtiene la contraseña actual del usuario de la base de datos y verifica si la ingresada coincide usando password_verify. Si no coincide, establece otro mensaje de error y redirige. Si la contraseña es correcta, se genera un hash de la nueva contraseña usando password_hash y se actualiza en la base de datos. Finalmente, redirige al usuario a su perfil si la actualización es exitosa o establece un mensaje de error y redirige nuevamente a la página de cambio de contraseña en caso contrario.

eliminar_cuenta.php

Este es el código que se ejecuta al pulsar el botón de eliminar cuenta en usuario.php.

```php
<?php
session_start();

if (!isset($_SESSION['usuario'])) {
    header("Location: index.php");
    exit();
}

// Conexión a la base de datos
require 'conexionbd.php';

try {
    $nombreusuario = $_SESSION['usuario'];

    // Eliminar los comentarios del usuario
    $sql_comentarios = "DELETE FROM Comentarios WHERE usuario = :nombreusuario";
    $stmt_comentarios = $conexion->prepare($sql_comentarios);
    $stmt_comentarios->bindParam(':nombreusuario', $nombreusuario);
    $stmt_comentarios->execute();

    // Eliminar el usuario
    $sql_usuario = "DELETE FROM Usuarios WHERE nombreusuario = :nombreusuario";
    $stmt_usuario = $conexion->prepare($sql_usuario);
    $stmt_usuario->bindParam(':nombreusuario', $nombreusuario);
    $stmt_usuario->execute();

    // Cerrar sesión y redirigir a la página principal
    session_destroy();
    header("Location: index.php");
    exit();
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
    die();
}
?>
```

En primer lugar obtiene el usuario de la variable de sesión usuario y seguidamente elimina todos los comentarios realizados por el usuario y después elimina el usuario de la tabla Usuarios. Finalmente destruye la sesión y nos redirige al index.php.

eliminar_comentario.php

```php
<?php
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}

// Verificar si el usuario está logeado
if (!isset($_SESSION['usuario'])) {
    header("Location: index.php");
    exit();
}

// Conexión a la base de datos
require 'conexionbd.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['comentario_id'])) {
    $comentario_id = $_POST['comentario_id'];
    $usuario = $_SESSION['usuario'];

    try {
        // Verificar que el comentario pertenece al usuario logeado
        $sql = "DELETE FROM Comentarios WHERE comentario_id = :comentario_id AND usuario = :usuario";
        $stmt = $conexion->prepare($sql);
        $stmt->bindParam(':comentario_id', $comentario_id);
        $stmt->bindParam(':usuario', $usuario);
        $stmt->execute();

        if ($stmt->rowCount() > 0) {
            $_SESSION['mensaje'] = "Comentario eliminado correctamente.";
        } else {
            $_SESSION['mensaje'] = "Error al eliminar el comentario.";
        }
    } catch (PDOException $e) {
        $_SESSION['mensaje'] = "Error: " . $e->getMessage();
    }

    header("Location: usuario.php");
    exit();
}
?>
```

Este código en primer lugar verifica si la solicitud es POST y contiene el ID del comentario, y extrae dicho ID y el nombre de usuario de la sesión. Intenta eliminar el comentario de la base de datos, asegurándose de que pertenece al usuario logueado, mediante una consulta SQL con parámetros. Si la eliminación es exitosa (es decir, si se ha afectado alguna fila), establece un mensaje de éxito en la sesión; de lo contrario, establece un mensaje de error. En caso de una excepción PDO, también establece un mensaje de error.