Alumno: Raúl Rodríguez Rodríguez

Práctica 2
Adaptación de una página web para un museo

1. Introducción

Para realizar esta práctica se han debido cambiar prácticamente todos los archivos a .php, para poder hacer uso de las herramientas que ofrece el lenguaje como el uso de funciones para "pre-programar" y reusar código sin escribir el mismo cada vez que se use. Obviamente también se ha cambiado las direcciones a todos los "nuevos" ficheros para que no haya direcciones incorrectas.

Se han creado varios archivos php para poder implementar el uso de la base de datos proporcionada por la escuela para poder hacer consultas y peticiones SQL para poder leer datos de la BD y escribir en ella. También se han creado varios documentos JavaScript o .js para darle dinamismo a varias comprobaciones y más funcionalidad a las diferentes páginas que tiene la web. En cada punto se verán los cambios y las adiciones a cada parte de la web.

Antes de empezar a detallar cada punto hago alusión a dos archivos que uso en todos los archivos para la cabecera y el footer, que son **header.php** y **footer.php**, los cuales tienen dos funciones con el código html necesario para generar la estructura html de la cabecera (logo, menú), pero para el form de identificación y el mensaje de usuario ya identificado he necesitado repetirlo en cada archivo para tener la sesión en cada uno y sepa qué datos tiene para funcionar correctamente. Al mensaje de usuario se le ha dado un estilo en el archivo css index.css acorde al estilo de la página, con un mensaje de bienvenida con el nombre de usuario y el rol/tipo de usuario y un botón de cierre de sesión (cerrar_sesion.php) con un simple código de creación de sesión, destrucción de sesión, redirección a index y la finalización del script.

Para que cambien la aparición entre el **form** de identificación y el **mensaje**, se ha añadido un archivo js, **script.js**, que se encarga de hacer **desaparecer** el **mensaje**, al principio invisible, hasta que se haya identificado el usuario (tenga información user-info) y también pone en tipo password al textarea de la contraseña cambiando los caracteres por puntos. El

footer tiene la misma estructura que en la primera práctica. Para hacer uso de ellos se debe indicar el **include** de ambos **archivos** al **principio** del documento.

El form de identificación tiene enlazado otro documento php al hacer submit (botón de envío), **identificar.php**, que como tendrán todos los documentos php, empezarán iniciando la sesión, luego dando las credenciales a la BD crea el objeto PDO para manejar consultas y errores de la BD y con el email y la contraseña identifican al usuario con la consulta de selección de usuarios con estas credenciales, donde una vez comprobada la contraseña obtiene el tipo de usuario, pudiendo ser usuario, lo cual accede a index con el mensaje de bienvenida, o admin (administrador), el cual enviará directamente al panel de administrador (**CRUD_admin.php**).

2. Index

No hay muchos cambios más que los indicados antes, con la inclusión del script.js, los includes de (header y footer).php y el código necesario para la identificación y mensaje de usuario.

3. CRUD_admin

Tendrá un estilo minimalista, con el mensaje de bienvenida al administrador, un botón para cerrar sesión y un menú en el main con:

- Una lista de dos elementos (botones) con las opciones:
  - Crear Obra (redirige a crear_obra.php)
  - Lista obras (redirige a lista_obras.php)
- Dos forms:
  - Editar obra (redirige a editar_obra.php) por id
  - Eliminar obra (redirige a eliminar_obra.php) por id con un mensaje de confirmación

3.1. Crear_obra

Siguiendo una estructura similar a la explicada en la introducción, comprueba de nuevo que el usuario sea administrador, y obtiene las imágenes disponibles en la BD para asignarlas a la nueva obra, a que deben estar ya en la BD. Una vez hecho esto, teniendo un header similar a CRUD_admin.php con un botón para volver al panel, en el main hay un form con las características de la obra. Cuando se envía/pulsa el botón de envío se envía a **guardar_obra.php**.

3.2. Guardar_obra

De forma idéntica a la ya dicha, tras tener el PDO se obtienen los datos del form de crear_obra.php y se prepara la consulta o petición SQL de inserción de la nueva obra con las diferentes características: `$stmt = $pdo->prepare("INSERT INTO obras (titulo, autor, fecha, imagen, epoca, tema) VALUES (:titulo, :autor, :fecha, :imagen, :epoca, :tema)");`

Una vez hecho esto se "bindean" los valores a las variables y se ejecuta la consulta quedando guardada la nueva obra en la BD. Una vez hecho esto te redirige al panel del administrador.

3.3. Lista_obras

Tras los pasos iniciales, lee de la BD todas las obras para su presentación en la página. En el main de la página hay un **section** con un **article** por **obra** dentro de un **bucle**, mostrando la imagen, título, autor y fecha, junto a dos botones de **editar** y **eliminar** que apuntan a los mismos documentos php que estas opciones en el panel de administrador.

3.4. Editar_obra

Obtiene el id de la obra a editar y al igual que crear_obra, obtiene las imágenes disponibles en la BD. Se le ha añadido un botón para volver a la lista de obras en el header. En el main hay un **form** similar al de **crear_obra** solo que "**pre-rellena**" los **campos** de los **atributos** de la obra a editar con los que tiene en la BD. Una vez satisfecho, hay un botón que lleva a **actualizar_obra.php**.

3.5. Actualizar_obra

Crea una sentencia SQL con UPDATE con los atributos modificados a la BD: `$stmt = $pdo->prepare("UPDATE obras SET titulo = :titulo, autor = :autor, fecha = :fecha, imagen = :imagen, epoca = :epoca, tema = :tema WHERE id = :id");`

Una vez hecho esto, se redirige a la lista de obras.

3.6. Eliminar_obra

Con el id indicado de la obra a eliminar crea una sentencia SQL con DELETE para borrar la obra por el id: `$stmt = $pdo->prepare("DELETE FROM obras WHERE id = :id");`

Una vez hecho esto, se redirige a la lista de obras.

4. Colección

Tras el PDO, se obtienen todas las opciones de filtro disponibles según los diferentes campos en la BD (epoca, autor y tema),y si no hubiera filtro se obtienen todos. Después se prepara la página con el número total de obras por página (9) y por el filtro seleccionado.

Después en el main habrá varios apartados:

- **Menú de filtros de búsqueda de obras**: en un aside hay una lista donde cada elemento tiene otra lista con todas las posibilidades de filtrado de obras según el tipo de filtro. Cuando se aplica un filtro se obtiene la colección de obras según el filtrado.
- **Section con las tablas de las obras** donde se controla el número de filas y hacer el salto de fila cada 3 elementos para tener una tabla de 3x3 elementos. Dentro de cada elemento/casilla de la tabla (td) hay un article con toda la información de las obras. También se tendrá guardada la información en variables que se utilizan más adelante. Si no se llegara a llenar una fila se crean articles vacíos para llenar la fila. Debajo de la tabla hay un nav con el cambio de páginas disponible según el filtro y para cambiar a la página anterior y siguiente.

Tras el main, se crea otro article como tooltip para dar la información cuando se tiene el ratón encima de una imagen, dando la información de la obra en un pequeño recuadro encima de la imagen. Para que funcione, tanto el **menú desplegable** para que calcule correctamente cuánto debe desplegarse, como para el funcionamiento del evento de poner el **ratón encima** de la **imagen** de las **obras** muestra el **tooltip**, se hace uso de **menu.js**.

5. Experiencias

Prepara una sentencia SQL de inserción para añadir un comentario a la BD: `$stmt = $pdo->prepare("INSERT INTO comentarios (usuario_id, comentario) VALUES (:usuario_id, :comentario)");`

Tras esto se encarga de leer los comentarios ya en la BD: `$stmt = $pdo->prepare("SELECT c.*, u.nombre AS nombre_usuario FROM comentarios c INNER JOIN usuarios u ON c.usuario_id = u.id ORDER BY c.fecha_com DESC");`

En el main hay un section con un article para cada comentario en la BD y debajo del mismo un form con un textarea para escribir un nuevo mensaje con un botón de envío y otro de borrado del borrador del mensaje. Para que aparezca este form y poder escribir un mensaje, antes debes estar identificado en la web. Las comprobaciones del textarea están en **val_exp.js**. En este js se previene envío por defecto, longitudes mínimas y máximas de texto y un ejemplo de filtro de palabras/lista negra de palabras.

6. Alta_usuarios

En este php se han cambiado las comprobaciones que se hacían en el propio código html al documentos js, **val_reg.js**. Para introducir al nuevo usuario después de pasar todas las comprobaciones oportunas, se enlaza al documento **procesar.php** para comunicarse con la BD.

6.1 Procesar

Después de preparar el PDO, se obtienen los datos del form, correspondientes a los campos en la tabla de la BD. Después se prepara la sentencia SQL para insertar el nuevo usuario en la BD: `$stmt = $pdo->prepare("INSERT INTO usuarios (nombre, apellidos, email, password, fecha_nac, tipo_doc, num_doc, cod_postal) VALUES (:nombre, :apellidos, :mail, :password, :fecha_nac, :tipo_doc, :num_doc, :cod_postal)");`

Cuando realiza con éxito la operación SQL, envía al usuario al archivo **exito_registro.php**, que simplemente notifica que se ha registrado con éxito y permite volver al index.

6.2 Val_reg

Empieza poniendo el campo de contraseña a tipo contraseña para esconder la contraseña con puntos. Continúa comprobando todos los valores de los campos del form preparando todas las variables para almacenar los valores de los campos para ser comprobados posteriormente. También se crean diferentes reglas o expresiones para validar los valores en las variables.

En las validaciones se usa la función test() junto a cada campo y la regla correspondiente. En cada una de estas comprobaciones hay un mensaje de error personalizado que queda guardado en el array de errores.

7. Visita, Exposiciones e Información

No hay ningún cambio mayor, solo la adición de las funciones php de código, el inicio de sesión, mensaje de bienvenida, junto con los scripts js.

8. Innovaciones

Se puede considerar la actualización a php y js del menú desplegable que crece según la cantidad y tamaño de los elementos de la lista.