Práctica 2:
Adaptación de la web del gestor de bibliotecas digitales

Programación Web

Tercer Curso – Grado en Ingeniería Informática
CURSO 2019-20
ESPECIALIDAD EN SISTEMAS DE INFORMACIÓN

ALONSO BUENO HERRERO
alonsobueno13@correo.ugr.es

Escuela Técnica Superior de
Ingenierías Informática y de Telecomunicación

UNIVERSIDAD
DE GRANADA

Índice

Índice de figuras

Resumen

En este documento se presenta una breve documentación sobre la práctica evaluable 2 de la asignatura Programación Web del Grado en Ingeniería Informática de la UGR.

El contenido está orientado a justificar a grandes rasgos cómo se ha conseguido «dinamizar» y «dar cohesión y coherencia» a todo el sitio web utilizando esencialmente el lenguaje PHP, el acceso a bases de datos (MariaDB en este caso) y complementando el desarrollo con acciones en el lado del cliente de la mano del lenguaje JavaScript.

En Granada, a 2 de junio de 2020.

Antes de empezar con la descripción, aclaro algunas ideas sobre lo explicado en esta memoria:

1. He denominado *ficheros intermedios* a los ficheros que **no son visibles** por parte del usuario, y que están dedicados en general a **procesar los formularios**: hacer las conexiones a la base de datos necesarias para completar la tarea de éstos.
2. Los ficheros con extensión *.html* de la práctica anterior se han cambiado a la extensión PHP donde ha sido necesario para permitir su «dinamización», de cara a cumplir los objetivos de la práctica. He evitado cambiarles los nombres para generar la menor confusión posible. El resto de ficheros PHP se han creado para la presente práctica.

   La lista completa de ficheros usados en la práctica se puede ver en el Apéndice A.
3. Todas las validaciones de formularios mediante Javascript se hacen asociando una función `validar()` concreta a cada formulario mediante el atributo `onsubmit=return validar()"` de `<form>`.

1. Cambios sobre la página *index.php*

La cabecera ha cambiado, albergando dos pequeños formularios:

- Un *primer formulario* para **iniciar sesión**, que tiene tres elementos HTML *input*, para introducir el nombre de usuario, la contraseña y un elemento *submit* para enviar los datos e iniciar sesión. El botón lleva a un fichero «intermedio» que accede a la base de datos y comprueba si el par usuario/contraseña introducido es correcto. En este caso, el usuario **será redigido directamente a la página heredada de la práctica 1 *gestorbd.php***, donde llegará con la ***sesión recién iniciada***.

  Al formulario se le ha indicado qué hacer cuando se pulsa el submit (dónde y cómo se envían los datos), con qué método (GET/POST), etc. Muestro ahora cómo se ha redefinido este formulario para esta práctica:

```
<form action="login.php" method="POST" onsubmit="return validar()" action="login.php">
   <!-- usuario, clave y submit -->
</form>
```

  Se ha usado una breve función **Javascript** para comprobar que el usuario no envía los campos usuario/contraseña vacíos para iniciar sesión.

- Un *segundo formulario* consistente en un único botón para darse de alta, el cual, cuando se pulsa, lleva a un formulario para darse de alta. El nombre del formulario es *darse_de_alta.php*.

En cuanto al cuerpo principal, he conseguido dinamizar las bibliotecas mostradas, mediante un fragmento PHP que accede a la tabla `bd` de la base de datos, la cual contiene las bibliotecas digitales, e imprime el código HTML necesario para mostrar correctamente cada biblioteca.

El fragmento PHP se muestra a continuación. Es similar que el que se usará (cambiando el código necesario) para mostrar los recursos, las secciones, etc...

```
$resultado = conexionBD('select * from bd') ;
foreach ( $resultado as $fila ) {
   $nombre_bd = $fila['nombre'];
   echo '
      <article class="unabiblio">
         <img class="im_biblio" src="' .$fila["imagen"] . '"/>
         <p class="tit_biblio">
            <a href="previo_bd1.php?bd=' . $nombre_bd . '" > '. $nombre_bd . '</a>
         </p>
      </article>
      <br>
   ';
}
```

El resto del cuerpo principal, así como el *footer* se mantienen igual que en la práctica 1, en estilos y contenido.

1.1. El fichero intermedio *login.php*

La tarea fundamental de este fichero es la de «validar» el par usuario/contraseña que ha introducido el usuario en el formulario de *index.php*.

¿Cómo se realiza la validación? La idea es sencilla: una vez que se han recuperado los datos usuario/contraseña (par) en dos variables PHP, se hace una consulta a la tabla de `usuarios` pidiéndole la fila que tiene como valor de

la columna `nick` (nombre de usuario) el nombre que el usuario ha introducido. Es decir, la consulta SQL es, tal y como se ha redactado desde PHP:

```
$consulta = "select * from usuario where nick='" . $_POST["nombre_"] . "' "
```

A continuación, se crean dos variables a las que se les asigna la cadena vacía, se extrae el *nombre de usuario* y la *contraseña* de la tabla y se comparan ambos pares (el del formulario y el de la base de datos). Si coinciden, **la identificación ha resultado satisfactoria y se inicia sesión**. En otro caso, se desprecian los datos recibidos y se lleva al usuario de nuevo a *index.php*.

Para iniciar la sesión se fija una variable en el array `$_SESSION` llamada `tu_nick` que se mantendrá con el nombre de usuario hasta que se cierra la sesión (desde un botón en *gestorbd.php*).

1.2. Darse de alta con *darse_de_alta.php*

Contiene un formulario HTML para que cualquiera se pueda dar de alta.

Una vez que Javascript comprueba los campos, se enviará el formulario a un fichero intermedio que insertará una fila en la tabla `usuario`.

2. Cambios y funcionalidades en *gestorbd.php*

Esta página ha sufrido algunos cambios:

1. En la ***cabecera***, se muestra el usuario actual (su nombre de usuario registrado) y dos botones: uno para «Cerrar sesión» (*log_out.php*) y otro para que el usuario modifique sus datos (*editar_mis_datos.php*).
2. El ***cuerpo*** ahora también se ha dinamizado, mostrando cada biblioteca digital mediante código PHP tal y como se hacía en *index.php*. A los enlaces para editar/borrar una biblioteca se les asignan formularios «útiles» con enlaces a *editarbd.php* y *borrarbd.php*, respectivamente.
3. Dentro del cuerpo había un enlace a un formulario para **crear una biblioteca digital**, *crearbd.php*, que también ha sido desarrollado.

2.1. Modificar datos del usuario actual o *editar_mis_datos.php*

El diseño y contenido es prácticamente el mismo que en *darse_de_alta.php*, sólo que ahora:

- En lugar de mostrar las cajas de texto del formulario vacías se muestran los datos asociados del usuario de la sesión. Esto se consigue con una conexión a la base de datos previa para extraer de la tabla `usuarios` la fila correspondiente. Después, se ha usado el atributo `value` de los *input* correspondientes para poder mostrar esos valores actuales. Un ejemplo:

```
<input type="text" value='<?php echo $nombre ?>' ... />
```

- Al enviar los datos, se lleva a un nuevo fichero intermedio llamado *editar_mis_datos2.php* que extrae los datos del formulario (todos), y hace un `UPDATE` sobre la tabla `usuarios` para actualizar la fila correspondiente.

  Como cabe esperar, la validación en Javascript es exactamente igual que en el formulario de *darse_de_alta.php*, pues los campos y restricciones son los mismos.

2.2. Cerrar sesión o *log_out.php*

Cuando se pulsa sobre este botón, actuará un fichero intermedio PHP que destruirá las variables de sesión y la propia sesión con:

```
session_unset();
session_destroy();
```

y justo después se redirige al usuario a *index.php*, ya que se ha cerrado su sesión.

2.3. Crear una nueva biblioteca con *crearbd.php*

Se trata, de nuevo, de un formulario HTML que solicita al usuario varios campos, todos ellos obligatorios: una imagen (*input* de tipo *file*), el título, su autor y el campo de conocimiento de la biblioteca (con un menú desplegable).

Al pulsar en el botón de *submit* («Enviar datos») actuará otro fichero PHP (intermedio) que se encarga de ejecutar la sentencia `INSERT INTO` sobre una tabla llamada `bd` que guarda las bibliotecas digitales, como se hacía con los usuarios.

2.4. Botones para cada biblioteca: *editarbd.php* y *borrarbd.php*

El contenido y funcionalidad de estos fichero es muy sencilla:

- En el caso de ***editarbd.php***, los campos aparecen rellenos con la información de la biblioteca que se quiere modificar. Al pulsar en *submit* actuará de nuevo un fichero intermedio que capturará los datos del formulario y hará un `UPDATE` sobre la tabla `bd`. También se hace la validación oportuna con Javascript.
- El formulario ***borrarbd.php*** es muy sencillo, pues únicamente pregunta al usuario si realmente quiere borrar la biblioteca asociada al botón que ha pulsado (estaba dentro de su cuadro en *gestorbd.php*).

Ambos dos, *editarbd.php* y *borrarbd.php*, necesitaban saber el nombre de la biblioteca sobre la que iban a trabajar por razones obvias. Lo conseguían porque *gestorbd.php* se la pasaba en el propio enlace (`href`). Lo vemos a continuación:

```
1 echo "
2 <a href='editarbd.php?esta_bd=" .$nombre_bd. "'><img ... title='Editar biblioteca digital'></a>
3 <a href='borrarbd.php?esta_bd=" .$nombre_bd. "'><img ... title='Borrar biblioteca digital'></a>
4 ";
```

3. Cambios en la página de una biblioteca digital: *bd1.php*

Ahora esta página será compartida por todas las bibliotecas digitales, ya que todo el contenido que se muestra será dependiente de la biblioteca seleccionada.

He usado dos ***cookies*** llamadas `bd_actual` e `imagen_bd` que se rellenan en un fichero intermedio llamado *previo_bd.php*, al cual se llega seleccionando el título o la imagen de una biblioteca en *gestorbd.php* (como se accedía hasta ahora a *bd1.php*).

Este fichero PHP se encarga de establecer esas *cookies*, y a continuación redirige al usuario a la página que espera (*bd1.php*).

La cookie `bd_actual` contiene el nombre de la biblioteca accedida e `imagen_bd`, como cabría esperar, la imagen de dicha biblioteca[1].

En cuanto al **diseño y contenido** de *bd1.php*, los cambios han sido los siguientes, todos ellos a hacer todo su contenido «dinámico»:

1. El título de la biblioteca se obtiene con un fragmento PHP que imprime la biblioteca seleccionada desde *gestorbd.php* (realmente desde *previo_bd.php*, que la ha capturado y vuelto a pasar a *bd1.php*).
2. El menú también se obtiene mediante un fragmento PHP incrustado que consulta la tabla `seccion` y saca las secciones (la columna *titulo*) asociadas a la biblioteca actual. Cada título en el menú sigue «apuntando» a la página de la sección.
3. En cuanto al ***lateral izquierdo de la página***, se muestran cinco de los recursos de esta biblioteca que aparecen en la tabla `recurso`.
4. En el ***lateral derecho de la página***, se ha dinamizado el contador de recursos de la biblioteca (con una consulta a la base de datos en PHP9. También se indica ahora el campo de conocimiento de la biblioteca siguiendo la misma técnica.

   El enlace para regresar a la página *gestorbd.php* ahora sólo se muestra si el usuario ha iniciado sesión (creo que así tiene más coherencia el enlace), y si no había iniciado sesión, vuelve a *index.php*.

   También se ha dado funcionalidad a todos los formularios de gestión de recursos y secciones. Para no saturar en exceso esta memoria, explicaré el patrón general que siguen cada tipo de formulario en los dos casos, ya que es la misma, sólo cambian levemente los campos:

[1] Esta última *cookie* se usa para mantener actualizado en todo momento el logotipo de la cabecera

- un formulario de **alta y de borrado** pide rellenar un determinado formulario (con la validación procedente con Javascript) y procesa dicho formulario en un segundo fichero (intermedio);
- un formulario de **edición**, que se divide en *dos pasos*: (1) seleccionar el recurso/sección a editar de un menú desplegable, seleccionar «*Siguiente paso*» y (2) entonces se editará el recurso o sección seleccionado/a en el paso previo. Entonces, se editará lo que se quiera (de nuevo validado por Javascript) y al hacer el *submit*, se lleva al fichero intermedio para procesar el formulario completo (ejecutar el `UPDATE`, esencialmente).

Cuando se selecciona cualquier sección del menú horizontal, se accederá a la página *recursosseccion1.php*.

4. Cambios y dinamización de *recursosseccion1.php*

En cuanto a las secciones, he creado una nueva *cookie* llamada `$_COOKIE['seccion_actual']` que se establece en el *fichero intermedio previo_seccion.php*, el cual actúa cada vez que se accede a una sección desde el menú. Tras fijar esa *cookie*, se redirige al usuario a la página *recursosseccion1.php actualizada* para la sección escogida.

Los cambios que ha sufrido la página se resumen a continuación:

1. El **menú**, como en el resto de páginas, también está dinamizado y cuando se pulsa se accederá a la misma página actual pero con el contenido de la sección seleccionada.
2. El recuadro centrado que aparece en esta página justo debajo del menú indicando la **sección mostrada** también se actualizará en base a la sección actual.
3. La «***cuadrícula***» que representaba recursos en filas de tres ya no es un elemento `table`, sino que ahora es un conjunto de `article` que se van colocando en línea y apilados hasta ocupar tres por fila.

5. Cambios sobre *recurso1.php*

En este caso los cambios son (en general) evidentes, y algunos ya heredados de las páginas anteriormente explicadas: el título central del encabezado, el menú de secciones y el pie.

En cuanto al ***cuerpo***, se han sustituido los valores de ejemplo que se mostraban en la práctica 1 por valores reales que se han extraído de la tabla de `recurso` de la base de datos mediante PHP **a partir del título del recurso seleccionado en *recursosseccion1.php*** (se le pasa a *recurso1.php* explícitamente en el enlace).

5.1. Sobre los botones «Anterior» y «Siguiente»

Para el caso del botón «**Anterior**», se llama a una función en PHP que buscará en la tabla de recursos el recurso que es inmediatamente anterior (si el actual es el primero, no aparecerá el botón, por coherencia). La función se llama `funcion_anterior` y está implementada en el fichero *funcionalidades.inc*.

En cuanto al botón «**Siguiente**», la idea es la misma, pero rescatando el recurso inmediatamente posterior en la tabla al recurso actual. La función PHP que lo implementa se llama `funcion_siguiente`, y también está en *funcionalidades.inc*.

6. Innovaciones

Realmente, son complementos o funcionalidades básicas adicionales que he añadido al sitio web:

- Las páginas *recursosseccion1.php* y *recurso1.php* contienen un enlace de vuelta a la página *bd1.php*, para facilitar la navegación del usuario. Este enlace es el propio **logotipo de la biblioteca**, en la esquina superior derecha de las páginas.
- Las páginas *gestorbd.php* y *bd1.php* incluyen, en las imágenes asociadas a cada biblioteca mostrada, un enlace directo a la misma (obviamente, también se mantiene dicho enlace a través del título de la biblioteca ya presente).

APÉNDICES

A. Lista de ficheros creados en esta práctica

En la tabla 1 se intenta dar una visión general de los ficheros PHP/CSS (básicamente) que se usan en esta práctica, y que por tanto están contenidos en el directorio `/bdII` de mi carpeta en el servidor `betatun.ugr.es`:

Funcionalidad del fichero	**Nombre del fichero**
Página principal	*index.php*
Iniciar una sesión (procesar login)	*login.php*
Gestor de bibliotecas (tras iniciar sesión)	*gestorbd.php*
«Darse de alta»	*darse_de_alta.php*
«Darse de alta» (procesar formulario)	*darse_de_alta2.php*
«Editar mis datos»	*editar_mis_datos.php*
«Editar mis datos» (procesar formulario)	*editar_mis_datos2.php*
Cerrar sesión	*log_out.php*
Crear biblioteca	*crearbd.php*
Crear biblioteca (procesar formulario)	*crearbd2.php*
Editar una biblioteca	*editarbd.php*
Editar una biblioteca (procesar formulario)	*editarbd2.php*
Borrar la biblioteca	*borrarbd.php*
Borrar la biblioteca (procesar formulario)	*borrarbd2.php*
Fijar cookie de la biblioteca seleccionada	*previo_bd1.php*
Página de una biblioteca	*bd1.php*
Fijar la cookie de la sección seleccionada	*previo_seccion.php*
Página de una sección	*recursosseccion1.php*
Página de un recurso	*recurso1.php*
Dar de alta un recurso	*altarecurso.php*
Dar de alta un recurso (procesar formulario)	*altarecurso2.php*
Dar de alta una sección	*altaseccion.php*
Dar de alta una sección (procesar formulario)	*altaseccion2.php*
Editar un recurso (paso 1)	*editarrecurso.php*
Editar un recurso (paso 2)	*editarrecurso_paso2.php*
Editar un recurso (procesar formulario)	*editarrecurso2.php*
Editar una sección (paso 1)	*editarseccion.php*
Editar una sección (paso 2)	*editarseccion_paso2.php*
Editar una sección (procesar formulario)	*editarseccion2.php*
Borrar un recurso	*borrarrecurso.php*
Borrar un recurso (procesar formulario)	*borrarrecurso2.php*
Borrar una sección	*borrarseccion.php*
Borrar una sección (procesar formulario)	*borrarseccion2.php*
Ficheros de funciones PHP	*funcionalidades.inc*
Hoja de estilos	*estilos.css*

Cuadro 1: Distribución de los ficheros de la práctica.

B. Tablas de la base de datos.

El siguiente código muestra la estructura de las tablas creadas para gestionar los diferentes elementos de la práctica:

```
MariaDB [db76067525_pw1920]> describe usuario;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| nombre     | varchar(60) | YES  |     | NULL    |       |
| email      | varchar(40) | YES  |     | NULL    |       |
```

```
| nick       | varchar(30) | YES  |     | NULL    |       |
| contrasena | varchar(30) | YES  |     | NULL    |       |
| pais       | varchar(30) | YES  |     | NULL    |       |
| imagen     | varchar(40) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

```
MariaDB [db76067525_pw1920]> describe bd;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| nombre             | varchar(100) | NO   | PRI | NULL    |       |
| autor              | varchar(30)  | YES  |     | NULL    |       |
| imagen             | varchar(200) | YES  |     | NULL    |       |
| campo_conocimiento | varchar(50)  | YES  |     | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
```

```
MariaDB [db76067525_pw1920]> describe seccion;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| titulo | varchar(100) | NO   | PRI | NULL    |       |
| autor  | varchar(80)  | YES  |     | NULL    |       |
| pagina | varchar(50)  | YES  |     | NULL    |       |
| bd     | varchar(100) | NO   | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```

```
MariaDB [db76067525_pw1920]> describe recurso;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| nombre      | varchar(100) | NO   | PRI | NULL    |       |
| autor       | varchar(40)  | YES  |     | NULL    |       |
| tipo        | varchar(30)  | NO   | MUL | NULL    |       |
| imagen      | varchar(60)  | YES  |     | NULL    |       |
| descripcion | varchar(300) | YES  |     | NULL    |       |
| seccion     | varchar(100) | NO   | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

```
MariaDB [db76067525_pw1920]> describe tipos_recursos;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| nombre | varchar(30) | NO   | PRI | NULL    |       |
+--------+-------------+------+-----+---------+-------+
```