Práctica 2

Adaptación de la web de la gestión de bibliotecas digitales

Francisco Javier Merchan

Índice

Introducción

Esta página web se hace como resultado a un problema explicado en clase, para su posterior evaluación de la parte práctica de la teoría. El problema es cuestión en el que nos enfrentamos en esta parte coger como base la practica anterior y añadirle las funcionalidades requeridas por el guion de esta segunda parte.

Como experiencia ha sido enriquecedora, pero la curva de dificultad con respecto al anterior aumento bastante ya que la parte de debuging y de aprendizaje dependía de partes que no controlábamos, como es el servidor, esa a sido mi mayor dificultad. Por suerte una vez fui descubriendo entre tutoriales en internet y apuntes facilitados por el profesorado el funcionamiento de los idiomas de programación y que podía hacer con cada lenguaje, me resulto mucho mas llevadera la práctica y conseguir superar esa curva de aprendizaje inicial.

Los principales errores y dificultades que he tendido han sido a la hora de ejecutar la sentencias en la base de datos asi como establecer la conexión con esta, las expresiones regulares de los formularios de JS y como pasar variables y datos entre guiones. Esto último está hecho de varias maneras, como poder ser una variable de sesión o pasado como parámetro en la URL.

Ahora procederemos a explicar como hemos hecho los distintos puntos que se requieren en la práctica, asi como la inclusión de cosas que no se requerían formando parte de la innovación del proyecto.

Alta de un usuario

Para este punto partiendo de la página de altagestor que usamos en la práctica anterior le añadimos un par de mejoras con php y JS.

Con php, lo primero que hacemos al principio es un require para la conexión, en el cual configura una conexión e inicia una sesión (sin usuario). Este require esta en todas las páginas del sitio web. Hay varias filosofías para implantar php en nuestro código, a lo largo de la practica hemos usados varias, aquí nos encontramos el código php incrustado en las partes necesarias de HTML. Al no realizar muchas funciones salvo mostrar información o algún condicional, se hace mucho mas sencillo y mas visual ponerlo de esta manera que en función de echo.

En el barner de arriba hemos visualizado dos posibilidades. La primera que se este dando de alta habiendo iniciado sesión con otro usuario y la segunda haciéndolo sin haber iniciado sesión. Para poder acceder a esta función (la de dar de alta a un usuario) tanto en esta como en index.php se ha incluido un botón extra llamado "*Registrarse*" el cual redirige a la página altagestor para poder dar de alta al usuario. Para hacer esta distinción se usa la función *isset* de php que nos dice si existe o no la variable pasada como parámetro, en este caso es la variable de sesión que establece el nombre de usuario de la persona que ha iniciado sesión. Si se ha iniciado sesión muestra la configuración que muestra en las paginas posteriores como gestordb, bd1, etc, en cambio si no, muestra el mismo formulario que en index.

El formulario valida con JS los datos tal y como se pide en los puntos, al ser todos muy parecidos los explicaremos más adelante todos juntos.

El formulario validado con JS es enviado al servidor y que llama al archivo *procesarUsuario.php* para procesar los datos del formulario.

Formularios de Alta

procesarUsuario junto con los de *procesarSeccion, procesarRecurso, procesarBiblioteca*, son los que suben a la base de datos los datos introducidos con sus respectivos formularios alas respectivas tablas. Pese a que son formularios para distintos objetos, realizan la misma función y por lo tanto sus estructuras son iguales, excepto cosas como datos que introducen en la tabla o la tabla en la que lo introducen.

Otra particularidad son que para biblioteca y Recursos se incluye de forma comentada el subir un archivo (por que da error al subir el archivo, he intentado posibles soluciones, pero es un error generalizado en la clase hasta donde sé). Debido a un error ajeno, he optado por poner la ruta a una foto por defecto, pero dejando el código necesario para la subida comentado y listo para funcionar.

El guion de subida esta estructurado con condiciones if/else. En altagestor como puede hacerse habiendo o no iniciado sesión no se comprueba, pero primero se incluye un condicionante con isset si existe la variable de sesión "*useranme*" que indica si se ha iniciado sesión, en caso contrario en el else notificamos del error y redirigimos al index. Después nos encontramos el if en el que comprobamos que existen los datos del formulario que se han introducido, en caso contrario tenemos un else que muestra un mensaje de error y redijimos otra vez al formulario de alta correspondiente.

Dentro del este if está la parte anteriormente mencionada para subir el archivo al servido que esta comentada. La secuencia esta cogida de la pagina de referencia W3Schools, con algunas modificaciones como el nombre de las variables o la ruta elegida para subir el archivo. Aun dentro del if (el que comprueba si se han introducido todos los datos), encontramos la funcionalidad propiamente dicha, primero asignamos variables a cada dato introducido por el formulario, seguidamente encontramos la variable con sentencia que guarda los datos en la tabla correspondiente. Aun dentro del if y debajo de la sentencia el fi que nos asegura que todo ha ido correctamente, ya que la condición de este es la sentencia *mysqli_query*, la cual devuelve falso si no se ejecutado correctamente, si es el caso se redirige devuelta al formulario de alta correspondiente, en caso contrario se redirige a la página de inicio, que es distinta para cada elemento(Recursos y secciones es bd1, Biblioteca es gestodb y para Usuario index.php.

Cuando nos referimos a redirigir es usar la sentencia php *header*.

Inicio y cierre de sesión

Como hemos dicho anteriormente el formulario que ya existía le hemos dado funcionalidad pudiendo iniciar sesión en la pagina principal, asi como un botón adicional en todas las demás paginas para desconectar o cerrar la sesión del usuario actual.

concetarseUsuario

Su estructura es parecida a las altas, primero hay un condicional if-else que comprueba que los datos de inicio de sesión introducidos en el formulario existen, si no redirigimos al index.

Dentro del if, nos encontramos con el bloque que inicia la sesión del usuario. Primero creamos variables con los datos del formulario para que trabajemos más cómodamente y más fácil con los datos. Después construimos la sentencia en la que comprobamos que el par usuario-contraseña existe y lo guardamos en una variable. Seguidamente conectamos con la base de datos, a través de la sentencia *msqli_query* y guardamos el resultado en una variable. Después de esto nos encontramos con dos casuísticas, que exista en la base de datos o no, para ello

colocamos en el if la sentencia php *mysqli_num_rows* y comprobamos que sea mayor que 0, esta sentencia da el numero de filas, es decir de coincidencias, del resultado de la consulta pasada como parámetro. Si no pasa la condición significa que no es correcto el par por lo que se redirige a la página index. En caso contrario y pasa la condición, iniciamos sesión con la sentencia php de *sesión_start* y utilizamos el resultado de la consulta, guardándolo en variables de sesión (*$_SESSION ['nombre de la variable']*) para su posterior uso algunos datos, guardamos el nombre de usuario y el email, seguidamente cerramos conexión con la base de datos con la sentencia *mysql_close* y redijimos a la página gestordb. Gracias a las variables que acabos de crear pondremos fácilmente el nombre del usuario en el lugar designado en las anteriores prácticas.

desconectarUsuario

En este guion básicamente destruimos la sesión con la sentencia php *sesión_destroy* seguidamente destruimos las variables que hemos usado durante la sesión como con la función unset. Por último, redirigimos con al index.

Formularios de Modificación

La primera idea que surgió para este apartado fue hacer una magina extra intermedia para elegir el objeto que se quería seleccionar de una lista. Para crear esta pagina intermedia use la misma la de borrar, pero cambiando borrar por modificar y asignándole distintos guiones a los de borrado en el servidor. Ahora cuando pulsamos en editar un elemento, nos llevara a una página que muestra una lista de la cual tendremos que elegir el que queramos modificar. Una vez esto se reducirá a la página de edición.

En términos generales la de selección no tiene mucho mas contenido distinto a la de borrar que se explicara más adelante. Cuando seleccionamos una de la lista que nos muestra pulsamos el botón de Modificar, el servidor ejecutara el guion *TratarSeleccionarBiblioteca, TratarSeleccionarRecurso* o *TratarSeleccionarSeccion*, según el objeto que queramos modificar.

TratarSeleccionar

En este guion tenemos lo mismo que todo los demás una encadenación de if-else. El primero como siempre comprobamos que haya una sesión de usuario activa y si no redireccionamos a index. Si tenemos una sesión activa pasamos al segundo if-else encadenado. En este caso se comprueba que se haya pasado un elemento del formulario de elección anterior, si no se redirigiera de nuevo al dicha pagina para que se seleccione una opción. Si se ha seleccionado uno, creamos una variable y le damos valor que ha seleccionado el usuario y redireccionamos a la pagina editar correspondiente al objeto (*edtarbd, edicionRecurso, edicionSeccion).*

editarbd, *edicionRecurso, edicionSeccion*

Como dijimos en la practica anterior las paginas son iguales a sus correspondientes altas, aunque esta vez añadiendo algo más, ya que hay que seleccionar todos los componentes almacenados en la BD y plasmarlos en el formulario para que el usuario los vea y los cambie si lo cree oportuno.

Antes de la pagina en si tenemos unas cuantas sentencias php para poder sacar la información asi como comprobaciones para que todo sea correcto.

La primera es una condición if que comprueba si existe la variable de inicio de sesión "*username*", si no existe redirige al usuario a la página index. Después comprobamos que hay un elemento seleccionado con la variable creada en el guion anterior, si no pues redirigimos al usuario otra vez a la página de selección que corresponda a cada elemento. Si existe continuamos dentro del if.

Dentro de él asociamos a la variable de sesión del elemento seleccionado a una variable local y la utilizamos para hacer una consulta a la base de datos y guardamos el resultado en una variable. A través de una condición if y la función *mysqli_num_rows* miramos si el resultado tiene menos o es igual a 0 filas, en cuyo caso significa que no hay ningún objeto con el identificador seleccionado por lo que se redirige al usuario a la pantalla de la base de datos, bd1. Si existe pues cogemos la fila que ha salido como resultado con *mysqli_fech_assoc* y la guardamos en una variable. Después pasamos a variables locales dicho resultado facilitando asi después utilizarlas.

En este punto hay un poco de división entre los guiones ya que por ejemplo en editarbd no nos encontramos ante el menú de navegación por lo que tenemos que hacer una consulta para ella, en cambio en edicionSeccion y ediciónRecursos sí. Esto se resuelve fácilmente con dos consultas extras (la otra es para su uso en los datos) en edicionRecursos y una para edicionSeccion. Por último, en edicionRecusos también comprobamos que el resultado de la búsqueda de las secciones que hay en la biblioteca sea igual a 0, en ese caso redirigiremos al usuario a la pagina bd1 ya que no se puede crear un Recurso sin asociarlo a una sección.

Ahora pasamos a la página, en ésta el bloque de usuario donde está el botón de desconexión y el nombre del usuario es igual para todas las paginas, al igual que el menú de navegación que se encuentra en todas excepto en la de gestorbd, y lo que tiene que ver con las bd asi como el alta de los usuarios. El primer bloque ya lo explicamos antes y simplemente es un echo de las variables de sesión correspondientes. Para el menú de navegación hemos usado una sentencia while que recorre una de las consultas que hemos hecho anteriormente para ello a la tabla de secciones. En esa consulta hemos cogido el titulo de todas las secciones de la biblioteca que hay en la variable de sesión. Con esto pasamos como variable al enlace el titulo de la sección para que pagina *recursosseccion1* pueda saber que sección a pulsado el usuario.

A continuación, tenemos el formulario, el cual rellenamos con los datos obtenidos anteriormente, solo hay un campo que no se puede cambiar y es el nombre del objeto, ya que esta a su vez es la clave primaria. Para todos los campos hemos usado el atributo value junto con la variable correspondiente para añadirle el valor, con la lista seleccionable hemos añadido un campo extra con el valor que trae el objeto, esto puede causar confusión al usuario, pero no hay problema por que se seleccione cualquiera de las opciones, es la forma menos complicada que se me ha ocurrido para solucionar esta casuística. Estos formularios también se verifican con JS usando el mismo que para las altas. Cuando el usuario lo envía al servidor este lo trata con el formulario escogido dependiendo del objeto que estemos tratando.

TratarEditarSeccion, TratarEditarBiblioteca, TratarEditarRecrusos

Estas páginas son iguales que las de tratar el formulario de alta salvo que algunas de las variables que no se usan y de que la sentencia usada en la base de datos es distinta, pasando de un INSERT a un UPDATE. Además de que en los errores redirigen paginas distintas por coherencia como cuando no hay dato en vez redirigir a la de alta del elemento redirige a la de seleccionar, por lo demás tienen las mismas sentencias y estructura.

Borrar

Para esta pagina tenemos la misma estructura y diseño a nivel de la página web que en el proyecto anterior. Para innovar y hacerlo por un método distinto al de alta y editar, el documento entero está escrito en PHP, excepto las partes de gestión de usuario y el índice de secciones, este último en borrarRecursos y borrarSecciones, que esta escrito como en los guiones anteriores (mezcla de PHP y HTML). Todo el texto HTML esta escrito con echo. Al principio tenemos la comprobación habitual de que la sesión esta iniciada como en los anteriores guiones. Después nos encontramos con diferentes sentencias dependiendo de los objetos que tengamos entre manos, para borarbd solo encontramos una búsqueda en la base de datos sin embargo para borrarRecurso y borrarSeleccion encontramos 2, una de ellas es para el menú de navegación, la otra es para obtener la lista de objetos para mostrar en la lista.

Al seleccionar una opción de la lista ofrecida y pulsar el botón borrar, el servidor ejecuta el guion correspondiente dependiendo del objeto que queramos borrar.

TratarBorrarbiblioteca, TratarBorrarSeccion, TratarBorrarRecurso

Su estructura el igual que los guiones que trataban los formularios de alta, es decir misma estructura, salvo que cambia la sentencia que compongo para la base de datos (ahora es DELETE), la asignación de variables (en este caso solo 1 ya que solo usamos 1 estrada, esto incluye la condición del if) y cambios en los header (la cambiamos por sus homónimos en borrar y dependiente de casa objeto).

Alta Recurso, Sección y Biblioteca

Para las altas hemos usado la plantilla y la estructura que la practica anterior. En este caso no hay mucha diferencia, es decir se añaden pocas cosas. Para los tres recursos son muy parecidos y hemos añadidos las mimas cosas, aunque con sus personalizaciones pertinentes (no todas tienen los mismos atributos), por ejemplo el creabd solo incluye la parte de comprobación de que se ha iniciado sesión, en las otras dos aparte tenemos las comprobaciones de que se haya seleccionado una biblioteca asi como consultas a la bases de datos para la barra de navegación. Las 3 sin embargo tienen el mismo bloque de gestión de usuarios que hemos añadido en las otras, asi como su correspondiente script para verificar el formulario de forma local. Específicamente altarecruso tiene una consulta más, con la misma sentencia SQL que para la barra de navegación, pero esta vez se usa para rellenar una lista para seleccionar a la sección que pertenece el recurso.

Cuando el usuario rellena el formulario y pulsa enviar, después de pasar el filtro de JS este formulario llega al servidor, que lo tratara con su guion correspondiente dependiendo del objeto como hemos explicado en uno de los puntos de arriba (*Formularios de Alta*).

Visualización de recurso

Para visualizar recurso hemos cogido la misma plantilla que para edicionRecusos, hemos eliminado todas las opciones en las listas de opciones y en los demás elementos hemos usado el atributo *readonly* para que no sea posible cambiarlo. Por lo demás es igual que la referencia que hemos usado, tanto para consultas (aunque esta vez hay una consulta menos debido a que ya no se necesita buscar las secciones de nuevo para rellenar el formulario) en la base de datos como para rellenar los campos del formulario.

Otra cosa que no es igual es en vez de el botón de enviar, hay dos botones, uno para el recurso anterior y otro para el botón siguiente. Estos no son necesarios para la visualización completa por lo que lo he supuesto como una innovación. A través de un campo extra numérico en el que numeramos los recurso con numero individuales, cogemos el numero del recurso actual y buscamos el siguiente y comprobamos que sea de la misma biblioteca en las que estamos. Si existe pues colocamos el botón, si no pues no lo ponemos.

recrusosseccion1

En esta pagina aparece cuando seleccionamos una de las secciones en la barra de navegación que esta dentro del sitio web.

Primero comprobamos el que una sesión activa como en todos guiones, después guardamos la variable de sesión del nombre de la biblioteca en una variable local para asi facilitarnos su uso. Seguidamente hacemos la consulta para la barra de navegación (la misma que en todas donde aparece esta).

Después nos encontramos el bloque donde miramos si hay una variable pasada como parámetro a través de la URL (esta variable es el titulo de la sección que queremos visualizar), si existe la guardamos una variable de sesión, y si no existe comprobamos que no existe su homónimo como variable de sesión, si sigue siendo negativo pues redirigimos a la página principal de la biblioteca(bd1). Si a pasado las condiciones anteriores, hacemos una consulta a la base de datos en la que consultamos los recursos disponibles para esa sección. Por último, buscamos que la sección elegida en la variable existe en la base de datos y si existe la guardamos una variable local para su posterior uso en los títulos, si en cambio no da resultado redirigimos a la pagina principal de la biblioteca(bd1).

En la pagina web encontramos los mismos elementos característicos como son el barner de arriba y el menú de navegación que se generan igual que las veces anteriores.

Para generar la parte central de la pagina hemos hecho una consulta a la base de datos extra y muy específica con los datos que hemos dicho anteriormente, es decir, buscamos los recursos cuya sección y biblioteca sea igual a la definidas y comprobadas anteriormente. Con el resultado, primeramente, comprobamos que hay recursos disponibles si no escribimos que no hay recursos. Si hay recursos, tenemos dos bucles for anidados. El primero de ellos hace siempre 3 vueltas, aunque lo que escribe no influye en el aspecto final visualmente, por lo que no he considerado añadirle mas clausulas para su detención. En cambio, en el bucle de dentro sí que tiene dos condiciones para seguir, la primera es para controlar que solo se ejecuta 3 veces, y la segunda condición es para coger una fila del resultado que hemos obtenido de la consulta anterior y que si no hay ninguna pues que no de mas vueltas. Con estas dos condiciones, aunque el bucle externo de las 3 vueltas completas el bucle del interior solo dará las necesarias. Asi obtenemos la matriz de recursos requerida.

Validación de Formularios con JS

Para la validación de formularios hemos usa el lenguaje JavaScript, el cual a sido incluido en todos los guiones necesarios con una función que comprueba cada dato individualmente. Me plantee hacerlo más eficiente y en vez de hacerlo como una función gigante con muchas condiciones if hacerlo con llamadas a mini funciones, y ponerlo todo en un solo archivo JS, pero debido a la falta de tiempo no me ha sido posible. Aun asi la manera que esta puesta también tiene sus beneficios, como que cada formulario tiene unas comprobaciones personalizadas he

independientes por lo que ayuda a su seguridad ya que no permite conocer los campos de los otros formularios.

En todos los campos comprobamos que su valor no sea nulo, que su longitud sea 0, una expresión lógica para comprobar que no sea rellenado solo con espacios y que no tenga una longitud que supere un máximo a convenir para cada objeto. Después tenemos otras comprobaciones como son una regla para comprobar un email que sea bien formado o que los nombres solo tengan caracteres permitidos, en el caso de los títulos y descripciones se añaden también a los caracteres permitidos los numero. Si incumple alguna de estas reglas la función devuelve false y muestra un mensaje de error al usuario para que corrija el respectivo error.

Base de datos

Para la base de datos me he declarado por 4 tablas de las cuales 3 están conectadas entre si ya que los objetos están relacionados. Las sentencias con las que creo estas son:

```
CREATE TABLE TABLA_RECURSOS(
    IDRecurso int NOT NULL UNIQUE AUTO_INCREMENT,
    titulo varchar(30) UNIQUE NOT NULL,
    genero ENUM('Action', 'Comedia', 'Suspense', 'Animacion') NOT NULL,
    subGenero ENUM('Action', 'Comedia', 'Suspense', 'Animacion'),
    anio varchar(10)NOT NULL,
    duracion int NOT NULL,
    nombreDirector varchar(40) NOT NULL,
    pais enum('ESP', 'USA', 'FRA', 'UK', 'MX') NOT NULL,
    protagonista varchar(40),
    tipo enum('Audio', 'Video', 'Texto', 'Imagen') NOT NULL,
    descripcion varchar(200),
    archivo varchar(200) NOT NULL,
    SeccionTitulo varchar(30),

CONSTRAINT PK_recursos PRIMARY KEY (IDRecurso),
CONSTRAINT FK_TituloSeccion FOREIGN KEY (SeccionTitulo) REFERENCES
TABLA_SECCIONES(titulo),
CONSTRAINT UNQ_tituloRecurso UNIQUE(titulo)
);

CREATE TABLE TABLA_SECCIONES(
    IDSeccion int NOT NULL AUTO_INCREMENT,
    titulo VarChar(30) UNIQUE NOT NULL,
    descripcion varchar(200),
    numeroRecursos int,
    bibliotecaTitulo varchar(30),

CONSTRAINT PK_seccion PRIMARY KEY (IDSeccion),
CONSTRAINT FK_TituloBiblioteca FOREIGN KEY (bibliotecaTitulo) REFERENCES
TABLA_BIBLIOTECAS(titulo),
CONSTRAINT UNQ_tituloSeccion UNIQUE(titulo)
);
```

```
CREATE TABLE TABLA_BIBLIOTECAS(
        IDBiblioteca int NOT NULL AUTO_INCREMENT,
        titulo VarChar(30) UNIQUE NOT NULL,
        descripcion varchar(200),
        numeroSecciones int,
        numeroRecursos int,
        archivo varchar(200) NOT NULL,

CONSTRAINT PK_biblioteca PRIMARY KEY (IDBiblioteca),
CONSTRAINT UNQ_tituloBiblioteca UNIQUE(titulo)
);

CREATE TABLE TABLA_USUARIOS(
        nombre varchar(50) NOT NULL,
        apellidos varchar(70) NOT NULL,
        usuario varchar(10) NOT NULL,
        email varchar(70) NOT NULL,
        pasword varchar(12) NOT NULL,
CONSTRAINT PK_usuario PRIMARY KEY (usuario),
CONSTRAINT UNQ_email UNIQUE (email)
);
```

El contener las claves primarias de por ejemplo la sección a la que pertenece el recurso, nos ayuda a crear sentencias de búsqueda más rápidas a la vez que nos ayuda a crear las restricciones necesarias como que no puede existir un recurso sin sección asignada.

Acceso a la base de datos

Para ello hemos creado un guion que se referencia al principio de todos los guiones con un require. En ese guion definimos unas variables para poder utilizarlas con las tablas de nuestra base de datos asi como de los datos necesarios para la conexión. El método de conexión que hemos utilizado ha sido el procedural que es distinto al enseñado en clase (me pareció el más correcto debido a su sencillez), el cual asignamos a una variable una conexión creada con la sentencia php *mysqli_connect* junto con varios parámetros que hemos definido arriba. Después comprobamos que esa conexión es correcta, si no pues informamos de ello y dirigimos a la página principal.

Notas adicionales

Como anotaciones adicionales también se ha hecho dinámico las bases de datos mostradas en la página de inicio, index, del mismo modo que se ha usado para generarlas en gestorbd. Otro apunte también es que en la mayoría de las listas si no hay elementos muestra una frase como que no hay elementos disponibles. Asi como las múltiples modificaciones extras recalcadas en los puntos anteriores (dinamismo de la barra de control, etc).

También está pensada para poder hacerla más modularizada, por ejemplo crear el pie o el barner superior en otro guion y con los requires incluirlo en la pagina que queremos dando asi mayor sencillez y rapidez a la hora de modificar algo de estos elementos, pero debido a la falta de tiempo por examenes y trabajos de otras asignaturas no me ha sido posible implementarlo. Otra función que se ha quedado a medias debido a la falta de tiempo, son los datos adicionales para estadistas, por ejemplo el numero de recursos de sección, que están

incluidos en las base de datos pero que no se actualizan. también me hubiera gustado poder haber usado algún framework o api para la realización de la práctica, pero a la hora de cuadrar el tiempo que tenía disponible para ello no lo vi viable, aun asi no descarto explorarlo en un futuro por mi cuenta.