Servidores Web de Altas Prestaciones

3º Grado en Ingeniería Informática (Tecnologías de la información)
Pedro Antonio Mayorgas Parejo (50643882K) y Adrián Acosa Sánchez (52418989B)

Proyecto Web

ÍNDICE

Fichero de restauración de la BBDD

La base de datos la restauramos al inicio del uso de la aplicación con el método DB_init(), el cual accede a los ficheros **reset.sql** que contiene las sentencias de creación de las tablas, **triggers.sql** que inicializa los triggers usados en la aplicación, y por último llama a **basicdata.sql** que contiene los datos iniciales y propuestos en el guión del proyecto. Estos tres ficheros se encuentran en la carpeta **bd** en el directorio del proyecto.

La aplicación, como se pide en el guión, permite que los usuarios administradores hagan operaciones en la BBDD desde la misma web permitiendo ejecutar operaciones de borrado, restauración completa por defecto, restauración a partir de fichero y generar un archivo para copia de seguridad para poder restaurar la BBDD desde el mismo.

Un detalle a tener en cuenta es que los triggers se borran al momento de borrarse las tablas a las que accedían, por lo que hemos optado por añadir a la operación de regeneración un bloque de código adicional encargado de regenerar dichos triggers desde el fichero comentado anteriormente (**triggers.sql**).

Comentar que en **basicdata.sql** hemos optado por codificar las fotos en formato 'base64' en lugar de binario puro debido a que al obtener la copia de seguridad nos estaba dando problemas ya que se insertaba de manera literal y no podíamos acceder a las fotos.

El directorio de assets contiene las fotos utilizadas en la práctica. Dentro de ésta carpeta tenemos la carpeta *usuarios* en la que almacenamos las fotos de los distintos usuarios del sistema, la carpeta *listas* donde almacenamos las fotos de las listas, y dos carpetas temporales en la que almacenamos las nuevas fotos que los usuarios introduzcan para sus perfiles o sus listas.

Diseño previo de la aplicación

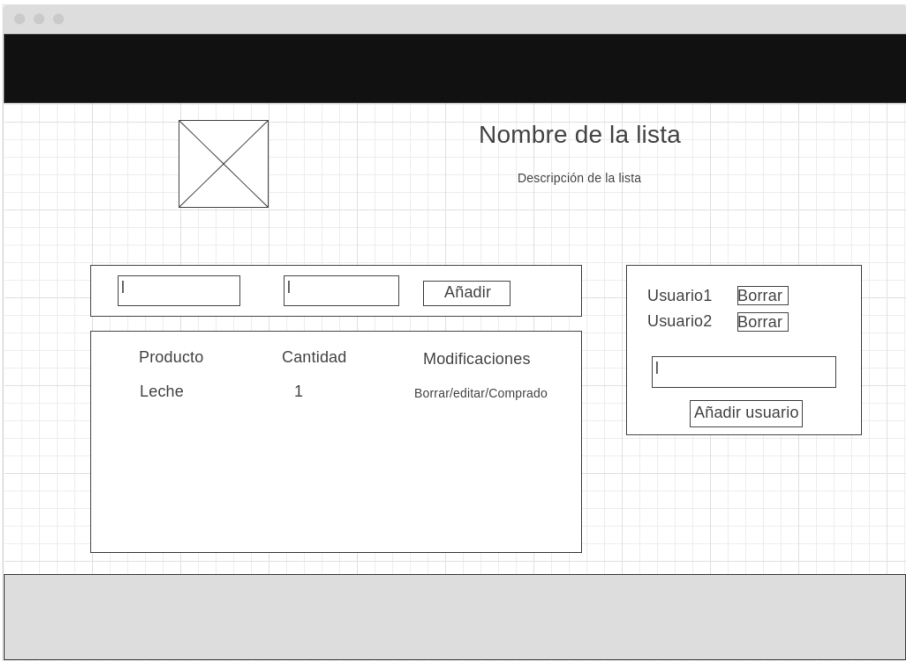
Para el aspecto del diseño, habíamos pensado usar "Bootstrap" en un principio. La idea era que todo el contenido principal fuese en medio de la pantalla, dejando arriba la barra de navegación y abajo el footer con la información relacionada.

El contenido principal en cada una de las páginas iba a ser distinto en función de las necesidades. Para ello, en primer lugar, decidimos que la única página que iba a tener un aspecto un poco más complejo iba a ser la pantalla de la lista.

En esa página nos basamos en el diseño que se muestra en la guía para el proyecto y le dimos nuestro estilo, siendo éste el uso de las "cards" de Bootstrap, el cual nos parece un ítem

bastante útil y que centra la atención del usuario en lo que está haciendo. Usando el posicionamiento basado en “grid” hemos dividido en 2 filas el contenido principal, siendo la primera fila la que contendrá el contenido de la información de la lista y una segunda fila dividida en dos columnas que contendrá el contenido de dicha lista y un input para introducir elementos, además de una caja con el nombre de los usuarios que participan en la lista colaborativa.

Por tanto el diseño de la página más compleja tendría el siguiente aspecto:



Para la página que muestra las listas que pertenecen al usuario, hemos usado un diseño basado también en cartas que nos parece bastante llamativo y visual. En primer lugar tenemos una cabecera que mostrará las opciones de “Crear lista” y las de filtrado de sus listas. Cuando le damos a crear lista, nos lleva a un formulario sencillo parecido al de los usuarios pero sólamente con los campos de foto, nombre y descripción (información básica de la lista). Los filtros se mostrarán justo debajo de éste botón, los cuales son exactamente iguales a los que salen en el guión.

Para mostrar las distintas listas que puede visualizar el usuario hemos implementado una solución que consiste en mostrar 3 cartas en las que aparecerá en la parte superior la foto de la lista, en el medio su nombre y debajo las opciones que tiene el usuario en función de sus privilegios sobre la lista. Si el usuario es propietario, podrá modificar la información de la lista (foto, nombre y descripción) en un formulario aparte y podrá también borrar la lista y visualizarla. Si los privilegios son de editor o de lector, el usuario sólamente podrá entrar a visualizar su contenido y ya dentro de la misma tendrá distintas funcionalidades en función de sus privilegios.

El resto de las páginas contendrían tanto el footer como el header, pero el contenido central se mostrará directamente en el main.

En el caso de los formularios, hemos seguido el mismo maquetado que se propone en el guión pero conteniendo éste en una carta de Bootstrap como hemos comentado anteriormente. Lo mismo pasa con la visualización del log del sistema, en la que en la propia carta aparecerá una tabla en la que cada entrada consiste en un mensaje de la tabla LOG de la base de datos.

Una vez decidido el diseño de la página, procedemos a explicar la organización de la base de datos.

Diseño de la BBDD

Diagrama de entidad relación

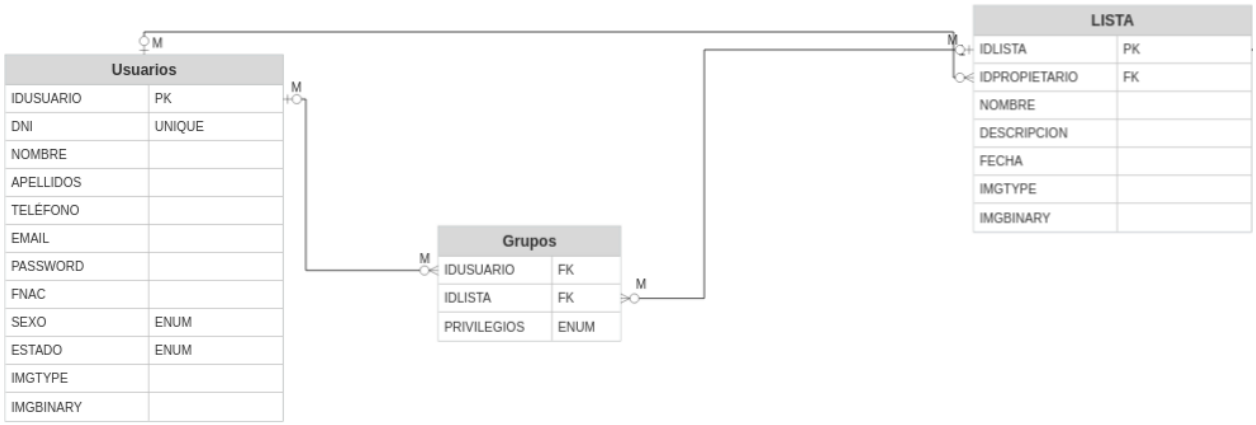
Hemos realizado algunos cambios en el modelo E/R para poder adaptarlos a nuestra aplicación.

En la relación de Lista y Productos concluimos que tiene una cardinalidad de:

- Da una cardinalidad de N:M que hace necesario una tabla relacional.
    - Porque puede haber de 0 a N productos que existan en una Lista.
    - Puede haber de 0 a N listas que tengan un producto..

En la relación de Listas y Usuario concluimos que tiene una cardinalidad de:

- Da una cardinalidad de N:M que hace necesario una tabla relacional.
    - Porque puede haber de 0 a N usuarios que tengan acceso a una o ninguna Lista.
    - Puede haber de 0 a N listas que tengan un usuario..
    - En el gráfico sale mal pero hay una relación adicional de 1:N de usuario a Lista directamente conocida como Relación Usuario Propietario de Lista.
        - Donde se determina que 0 o 1 usuario es propietario de N listas.



Explicaciones técnicas

Hemos usado el motor de plantillas de PHP llamado "*Twig*", que facilita la modularización de las vistas y poder separar código HTML y PHP con mayor facilidad. Para poder mostrar y ocultar los ítems según el rol del usuario hemos usado un control de estados basado en variables pasadas a la vista desde el controlador correspondiente para facilitar la validación de formularios. Con respecto a esto, según el estado que envíe el controlador, se muestran unos controles u otros.

Para poder mostrar la foto en la validación, usamos una COOKIE que almacena la información de ésta y la usamos para mostrarla al validar todos los datos introducidos. Una vez se ha completado el proceso correspondiente al formulario, la COOKIE se elimina para no dejar rastro.

Decir que hemos repetido algunas vistas de los formularios pero ha sido en favor de hacer un código más legible, ya que si hacemos todas las comprobaciones en un mismo formulario se ensucia mucho el código y también ha sido en favor de no perjudicar las comprobaciones de los distintos formularios.

Para el estilo de la página hemos usado el framework de CSS llamado *Bootstrap*, que nos ha facilitado el diseño de la página muchísimo debido a la facilidad de aprendizaje que tiene. Para poder hacer correctamente el diseño, hemos ido consultando la documentación oficial según las necesidades de diseño que hemos tenido. Un ejemplo de ésto es al mostrar los errores en los formularios, en los que hay dos clases que sólo funcionan si una es hija de la otra. Las dos clases son 'is-invalid' (que es la clase padre) e 'invalid-feedback' (la clase hija). Si se encuentra la padre, se muestra el mensaje de la clase hija debajo del padre.

En cuanto a detalles de implementación de PHP, comentar que hemos agrupado todas las operaciones que tienen que ver con la base de datos (consultas, inserciones, borrados y

actualizaciones) en un solo fichero llamado **bd.php** contenido en la carpeta **model**. Éste archivo es incluído en todos los ficheros de los distintos controladores que hemos creado.

Decir también que los controladores deniegan el acceso a los visitantes a las páginas no accesibles si no se ha iniciado sesión o a los usuarios si no tienen permisos de administrador. Ésto lo hemos realizado guardando en una variable de sesión el id del usuario logueado (en caso de ser visitante no se asigna nada y sólo tendría acceso a la página principal) y comprobando en cada una de las vistas qué cosas puede ver y qué cosas no en función de su rol.

También para mantener la información de las listas al momento de hacer operaciones en ésta, mantenemos una sesión con el id de la lista en el momento que entramos a añadir o eliminar productos de dicha lista. Así conseguimos que al realizar un POST en cualquiera de los formularios contenidos en la página no perdamos el contexto de la lista en la que nos encontramos.