Memoria Prctica 2

Jose Manuel de Frutos Porras
josem.frutos@estudiante.uam.es

Contents

1 Recomendaciones para funcionamiento **2**

2 Estructura de ficheros y funcionalidad **2**
2.1 Cabeceras 2
2.2 Registro 2
2.3 Login 3
2.4 Indice 3
2.5 Búsquedas, peticiones y resultados 3
2.6 Carrito 4
2.7 Historial 4

3 Detalles de implementación **4**

1 Recomendaciones para funcionamiento

Para el correcto funcionamiento de la práctica, recomendamos darle los máximos permisos al directorio raíz en el que se encuentre el proyecto. Sino la creación de fichero y carpetas no se realizará correctamente y fallarán entre otras cosas la funcionalidad de registrarse, de apostar ...

2 Estructura de ficheros y funcionalidad

En esta práctica podemos encontrar ficheros semejantes a la anterior pero con la extensión cambiada. La mayoría de ficheros de esta práctica son de tipo php debido al carácter dinámico entre el cliente y servidor. Además, ha resultado útil a nivel de programación. He podido factorizar código de ficheros relacionados con la interfaz (ej: ./miscelaneos/cabeceras.php o ./printTabla), evitando repeticiones innecesarias de código html. El php, por tanto, ha resuelto tanto el procesamiento de peticiones con el servidor como la factorización del código. Para el dinamismo en la parte del cliente tendremos porciones de javascript dentro de los php, y algún archivo .js independiente.

A continuación procedemos a describir los distintos ficheros, y comentaremos tanto su funcionalidad como los posibles problemas que se derivan.

2.1 Cabeceras

En el fichero cabeceras.php presente en ./miscelaneos/ se encuentra código que tienen en común muchos ficheros. Encontramos una función común que añade el pie de página y otras para las cabeceras, que aunque parecidas, cambian por ejemplo en los botones presentes de una página a otra. También encontramos una función para pintar el menú lateral con las distintas categorías.

El propósito de este fichero es evitar repeticiones de código y hacer también más legible la página.

2.2 Registro

Existen varios archivos relacionados con el registro de un nuevo usuario. Esta acción es una de las que tienen más controles de errores de toda la práctica. El fichero que muestra el formulario, registro.php, utiliza las comprobaciones de campos disponibles en HTML. En otro fichero, validarRegistro.js hacemos comprobaciones más exhaustivas del lado del cliente, hasta tal punto que podemos detectar todos los errores antes de mandarlo a verificar al servidor (para el nombre del usuario ya en uso, utilizamos Ajax, comprobando del lado cliente lo que nos devuelve el servidor al buscar ese nombre respuestaNombreResgistro.php). En otro fichero, validarRegistro.php, hacemos las comprobaciones del lado del servidor, con php, que incluye las reglas del javascript anteriores. Si todo ha ido correctamente el usuario debería ser redirigido a la página de inicio y deberia poder ver en esta un mensaje de bienvenida. Cabe destacar que he creado un

css que muestren los distintos mensajes de error en forma de burbuja de texto, y que cambia el borde del input en función de la verificación. También hemos usado funciones que verifican si una tarjeta de crédito es válida, por lo que para probarlo recomendamos probar con un generador de tarjetas de crédito online. Hemos implementado el sistema de fortaleza de contraseñas.Para ello hemos utilizado una biblioteca ya existente. Esta esta en el archivo css2. Se trata de un template del framework Boostrap desarrollado para twitter. Hemos considerado que las funcionalidades aportadas por este plugin que utiliza jQuery eran más que suficientes para lo que se pedía.

2.3 Login

El login es parecido al registro debido a que es un formulario. Solo tiene dos ficheros propios, el que muestra la página para iniciar sesión y el que valida la pareja nombre de usuario y contraseña. Quizá también podríamos contar en este apartado logout.php que simplemente cierra la sesión. En este caso, no usamos la validación del lado del cliente, pues no le vemos la utilidad. Además, he implementado el sistema de cookies que permiten recordar la información del último usuario conectado. Cabe destacar que tras un login con éxito, se cambiará dinámicamente otras páginas como el index. Se han utilizado variables de sesión para recordar la información del nuevo usuario logeado.

2.4 Indice

En el índice encontramos la presentación principal de nuestra aplicación. Podemos en él realizar busquedas de apuestas. También se hay enlaces hacia el login, el registro y hacia el carrito si no se está dado de alta. En caso de ya estar logeado se puede realizar logout y ver además un enlace hacia el carrito y el historial. Se modifica por tanto dinámicamente en función de si el usuario está logueado. Todas las funcionalidades del índice las comentaremos en los respectivos ficheros.

2.5 Búsquedas, peticiones y resultados

En esta sección incluyo todo lo relacionado a la interacción cliente-servidor-base de datos (lo que en un futuro será la base de datos, que de momento está en xml). Para realizar las busquedas he utilizado Ajax. El planteamiento no es quizás el mejor (ya el cliente debe descargarse el fichero de apuestas entero y realizar la búsqueda en él). Es muy costosa, aunque en esta práctica todavía no supone ningún problema importante ya que la base de datos es pequeña. Lo ideal sería una mezcla php Ajax y php, siendo el servidor el que realiza la búsqueda y el cliente se encarga de recoger el resultado mediante Ajax. No he implementado esto ya que simplemente no he tenido tiempo. Considero que es bastante fácil el cambio, sin embargo al realizar esta práctica solo no me ha dado tiempo. Aún así una de las ventajas de este diseño es la intección en tiempo real con los queries del usuario. Por ejemplo podemos ir poniendo

los resultados obtenidos mientras el usuario esta introduciendo el nombre del deportista que quiere, apareciendo al principio resultado que coinciden con las primeras letras del nombre. Los ficheros de busqueda y filtrado se encuentran en busquedasYfiltros.js son llamados desde el índice con los argumentos de la búsqueda.

He decidido crear un fichero printTabla.php, que se encarga de pintar las distintas tablas de resultados (historial, detalles de apuesta, búsquedas) a partir de ciertos parámetros.

Los resultados de una búsqueda son eventos, que pinchando en ellos nos llevan a las distintas apuestas de dicho evento.

2.6 Carrito

El carrito consta de dos ficheros, carrito.php, que se encargará de mostrar las distintas apuestas del carrito y añadir la funcionalidad de quitar (y apostar si está logueado) en el cliente, que hará peticiones AJAX al ficherocarritoFuncionalidades.php. Este se encargará de modificar el carrito , eliminarApuestaDelCarrito y apostarCarrito carrito entero (escribiendo las apuestas si el usuario está logeado en historial.xml. Si el usuario no está logeado se volverá al carrito un mensaje indicando que se ha de logear para poder apostar el carrito.)

2.7 Historial

Consta de los ficheros historico.php para mostrarlo e historicoFuncionalidad.php El historial muestra una tabla en la zona de contenidos, también detalles del usuario (Nombre, n de cuenta, saldo,mail). Además existe la posibilidad de añadir desde esta página dinero a la cuenta. La función que carga el historico se encuentra en historicoFuncionalidad.php. También en este fichero encontramos una función que se encarga del añadido de dinero a la cuenta del usuario.

3 Detalles de implementación

En este apartado comento detalles de la práctica más que su funcionamiento general. Hemos introducido algunos cambios en nuestra página respecto al guión proporcionado en prácticas que pasamos a explicar.

- Como explicado antes la busqueda y filtrado no lo he realizado con php y usando arays asociativos sino que al contrario hemos usado Ajax, dejando toda la aplicación de filtrado y busqueda por parte del usuario. Esto tiene inconvenientes. El envío del xml si es grande (no es el caso) puede llegar a tardar mucho al principio (Tras la primera descarga probablemente este se encuentre en caché web o algo similar y la segunda petición será mucho más liviana en tiempo). Luego el terminal cliente no tiene la potencia del servidor y búsquedas en bases de datos grandes pueden llegar a ser muy costosas para un ordenador normal. Aún así hay que recordar que cada día los clientes son más potentes y por esta razón muchas aplicaciones

trasladan parte de la lógica de la aplicación al cliente (cliente pesado). El uso ideal sería una mezcla Ajax y php, como hemos explicado antes, haciendo los tiempos de espera aceptables y proporcionando la alta interactividad propia de Ajax. No he llegado a realizar esto por falta de tiempo, pues he realizado la práctica solo. Pero creo que es relativamente sencillo la adaptación. Basicamente consiste en reescribir el codigo javascript en php y con Ajax mostrar los resultados de la busqueda y filtrado en php.

- Otro punto técnico es que he utilizado para la barra de fortaleza de contraseñas he utilizado una biblioteca ya existente. Esta está en el archivo css2. Se trata de un template del framework Boostrap desarrollado para twitter. He considerado que las funcionalidades aportadas por este plugin que utiliza jQuery eran más que suficientes para lo que se pedía.

- En la página de estilo he añadido unos media queries. No están todos completos (La aplicación no está todavía depurada para ser vista en monitores pequeños) pero una simples pruebas con monitores de distintos tamaños ajustando ciertos parámetros bastaría para que la aplicación fuera adaptable a casi todos los monitores estandar. No he llegado a hacerlo por falta de tiempo y de medios!

- Una vez más el uso de boostrap me ha ayudado enormemente el diseño y a la hora de hacer "resposive" mi aplicación.

- Por lo demás creo que todo está hecho tal y como se pedía!