Proyecto Final Desarrollo web en entorno servidor

Ibai Helguera de la hera

Kaietz Olabarri Candelas

Desarrollo de Aplicaciones Web 2º Curso

Planteamiento

En este proyecto conjunto centrado en el desarrollo de la metodología modelo-vista-controlador, nos hemos centrado en canalizar a través de una única página, el index, todo el código necesario para el tratamiento de una base de datos con el objetivo de realizar un menú que, en vez de permitirnos navegar por dististas páginas de la aplicación, sirva de filtro para la información que queremos plasmar en el index.

Para ello, hemos dividido el código de la aplicación en 3 carpetas:

* Modelo, con aquellos archivos que contienen las clases, en este caso cines, horarios y películas, cada de una de ellas con sus métodos correspondientes para el tratamiento de la base de datos. Debido al uso de la metodología MVC, todos los métodos se encuentran o bien dentro de las tres clases enunciadas anteriormente o dentro de una cuarta clase llamada "conexión" cuyo único propósito es el de permitirnos crear el propio objeto mysqli dentro de los diferentes controladores.
* Vista, con el código HTML y el código PHP indispensable (en su basta mayoría, echos de variables) para visualizar el contenido de la base de datos.
* Controlador, con los archivos PHP que contienen las instrucciones para cargar las variables que son utilizadas en las vistas o cargar una u otra vista dependiendo de las elecciones del usuario a través de la interfaz provista por la misma. Aunque muchos de los controladores dentro de la aplicación son muy parecidos, hemos creído conveniente su estricta compartimentalización para facilitar la legibilidad del código.

Por último, en cuanto a la temática de la aplicación, nos hemos decantado por realizar tablas que recuerdan vagamente a la cartelera de un cine o, en este caso, de varios, ya que nos permitía una excusa para realizar filtros en la base de datos de la aplicación web.

División de tareas

Para facilitar el desarrollo de la aplicación, el código ha sido dividido de tal manera que, aunque el index y el css ha sido un trabajo conjunto, Kaietz se ha centrado en la creación de la base de datos y las vistas e Ibai se ha ocupado de los controladores y modelos.

Desarrollo del código

Como se ha expuesto más arriba, el código está expresado de tal manera que siempre parte del index. Es decir, pase lo que pase en la aplicación, el flujo que sigue es:

"index->controlador\_único\_general->controlador\_específico->vista"

Todo comienza en un index compuesto únicamente por un menú lateral que permite al usuario interacturar con 3 tablas (cines, peliculas y horarios) de 4 formas distintas (añadir información, eliminarla, modificarla y mostrarla).

Una vez el usuario ha introducido en el formulario del menú qué desea hacer, el index llama al controlador principal o, mejor dicho, siempre lo llama, y sólo cuando detecta, comprobando si existen distintas variables del array $\_POST, que el usuario ha introducido sus preferencias es cuando deriva, haciendo uso de dos switch anidados, a los controladores particulares encargados de cada tarea.

Es necesario apuntar que, debido a que algunas vistas, como las encargadas de modificar datos en las tablas, o aquellas con algún filtro, tienen formularios en su interior que recargan el index, era necesario poder guardar las elecciones del usuario en el menú hasta que escogiese otras distintas (ya que, al recargar la página, el contenido del array $\_POST que nos permitía cargar dicha vista se perdía), por lo que se ha recurrido al uso del array $\_SESSION para poder mantener en memoria datos relevantes (como las elecciones del usuario, si los formularios de añadir, eliminar o modificar han sido rellenados o dos datos genéricos utilizados para el paso de información a los métodos que modifican información en la base de datos).

A partir de aquí, y dado que las 4 formas distintas de interactuar con las tablas son, en esencia, muy parecidas, no describiremos cada controlador de forma individual, si no que se explicará por bloque y se matizará si alguno de ellos contiene alguna particularidad.

Primero, en cuanto a los controladores encargados de añadir información a la base de datos, tienen la función principal de detectar si el formulario de su correspondiente vista ha sido o no rellenado. Si no lo ha sido, el controlador se limitará a cargar la vista con el formulario y, si por el contrario, ha sido rellenado, ejecutará la función correspondiente del modelo que se encarga de interactuar con la tabla de la base de datos. Dichas funciones, que de manera interna hacen un INSERT a la base de datos, devuelven un mensaje de éxito o error que se recoge en una variable y se muestra en la vista.

En segundo lugar, los controladores a cargo de eliminar contenido de la base de datos realizan el mismo proceso pero llamando a la instrucción encargada de eliminar datos.

En tercer lugar, es importante señalar que los controladores que manejan la modificación de información en la base de datos comparten variables del array $\_SESSION para no tener que crear una variable por cada campo de la base de datos, si no tener dos $\_SESSION que reutilizo para cada tabla. Esto no supondría ningún problema, de hecho, no lo ha supuesto en los casos expuestos en los párrafos anteriores, si no fuera porque la vista encargada de mostrarme el formulario con los datos a modificar tiene como value de los inputs un echo de dichas $\_SESSION, por lo que podría ocurrir que, ya con dicha vista cargada, el usuario decidiera moverse a otra parte de la aplicación y modificar los datos de otra tabla, lo que supondría que, como la variable de $\_SESSION que el controlador utiliza para detectar si se ha elegido que fila modificar es la lisma para todos los controladores, se cargue directamente la vista con el formulario para cambiar una fila utilizando los datos seleccionados en la primera tabla.

(Es decir, si yo elijo modificar la fila 4 de la tabla peliculas, y luego decido que no voy a cambiarlo y me voy directamente a modificar la tabla de cines, arrastro a ese controlador los valores de la fila 4 de la tabla películas, causando una serie de errores no previstos en la ejecución del formulario).

Para evitar esto, hemos creado una variable de $\_SESSION adicional que solo se rellena tras detectar que se ha enviado un formulario de modificación permitiendo que la $\_SESSION que utilizamos para comprobar el envío del formulario se libere dentro de ella y evite que se pueda saltar la vista de selección en los demás formularios de modificación.

Por último, los controladores a cargo de mostrar información por pantalla, simplemente llaman al método que recoge la información de las tablas y carga la vista correspondiente. En el caso de las películas y horarios, como hemos querido añadir un pequeño filtro para poder utilizar otras sentencias SQL, también detectan si se ha enviado el formulario de búsqueda. Especial atención al método que recoge información en la clase Horarios\_Modelo, que no recoge estrictamente información de la tabla horarios, si no que utiliza información de todas las tablas con JOIN y ORDER para facilitar la visualización del contenido.