Cómo se hizo: Página web del Centro Deportivo, segunda práctica

Juan José Sierra González

En este documento se incluye una pequeña guía para entender cómo se ha realizado la página web dedicada a la segunda práctica de la asignatura Programación Web de la Universidad de Granada (UGR).

Alta de usuarios en la base de datos de MySQL

Para almacenar los usuarios he creado una tabla en la base de datos llamada **Usuarios**. Esta tabla contiene un campo para almacenar cada uno de los datos básicos de un usuario como su nombre, apellidos, nombre de usuario, contraseña, correo electrónico o fecha de nacimiento, además de haber incluido campos adicionales para los otros datos que se pueden rellenar en mi formulario. Para cada uno de los checkbox seleccionables he creado un campo que almacena una variable booleana, al igual que para la variable del *"radio button"* que simboliza si el usuario ha pertenecido antes a un centro deportivo.

Estos datos se recogen desde el formulario HTML y se procesan en el archivo `procesar_formulario_alta.php`. En este archivo se hace uso de la clase Usuario creada en PHP mediante el paradigma dataObject que se explicó en el guión de prácticas.

La clase Usuario contiene un array de datos con los mismos campos que se pueden almacenar en la tabla de la base de datos. Además de las funcionalidades heredadas de dataObject, para este caso he implementado un método *crearUsuario* que realiza un INSERT en la tabla correspondiente. Primero preparo la consulta y después voy asignando a cada campo el valor correspondiente, de los recibidos por el array `$_POST` tras enviar el formulario. Cabe destacar que la contraseña del usuario se almacena en la base de datos encriptada, en este caso por el algoritmo SHA512, lo que impide que alguien que acceda a la base de datos pueda leerla.

Una vez que se han asignado todos los campos de la consulta, esta se ejecuta, y se añade el usuario a la base de datos, siempre y cuando el nombre de usuario (la clave primaria de la tabla) no sea repetido. Si todo ha salido con éxito se redirige al usuario a una página nueva, `registrado.php`, que le permite regresar a la página del índice mediante un enlace.

Tras la creación del usuario, antes de mandarle a la página de registrado, se le añade a la sesión para que pueda comenzar a navegar por la página.

Manejo de sesiones con PHP

Para el manejo de sesiones he hecho utilidad de la función `session_start` de PHP, y que permite el uso de un array, `$_SESSION`, que puede almacenar la información necesaria en cada sesión, hasta que un usuario la cierre. Gracias a ello he guardado el nickname del usuario que ha iniciado la

sesión, y ello me ha permitido decidir qué secciones de la web deben mostrarse cuando hay un usuario registrado navegando o cuando en su lugar es un visitante anónimo.

Utilizando código PHP y reproduciendo código HTML usando la funcionalidad `echo` he conseguido este propósito para, por ejemplo, decidir si mostrar el formulario de login o por el contrario el nombre de usuario y un enlace para cerrar la sesión. Puede comprobarse que en cada página se hace esta comprobación.

El formulario de login, que recibe un nombre de usuario y una contraseña, ejecuta el código de un archivo llamado `procesar_login.php`, que comprueba la veracidad de las credenciales introducidas y o bien da acceso al usuario con la sesión iniciada o bien lo devuelve a la pantalla de índice sin cambio alguno.

La comprobación de la contraseña se hace con un método de la clase Usuario llamado *validarLogin*, cuya tarea es hacer un select a la tabla de Usuarios de aquellos con un nickname y una contraseña iguales a las del formulario. Por supuesto, es necesario cifrar la contraseña con el mismo algoritmo con el que se almacenó en la tabla antes de buscar en ella. Si el número de usuarios encontrado es distinto de 1, quiere decir que ha habido algún problema y el login no es válido. En caso contrario, se puede proceder a abrir la sesión para el usuario.

Para procesar la desconexión del usuario simplemente hay que hacer nulo el valor que se está comprobando, que es el del nickname del usuario que está navegando por la página.

Modificar datos del perfil

Del mismo modo que en la esquina superior derecha de la página aparece un formulario de login o un botón de desconexión dependiendo de si hay un usuario registrado o no, la pestaña de Alta de Usuario cambia para dejar sitio a la de **Perfil**. Desde esta pestaña un usuario podrá modificar algunos de los datos de su perfil, y lógicamente sólo será accesible para un usuario registrado.

El diseño de esta página es similar al de la página de alta de usuario, simplemente cambian los campos. El usuario solamente podrá editar su nombre, sus apellidos, su correo electrónico, fecha de nacimiento y su actividad preferida. Para modificar estos campos, el usuario deberá introducir, al final del formulario, su contraseña actual. Si esta no es correcta, no se validará el cambio. Los campos del usuario se recogen desde su entrada en la base de datos. Haciendo uso de la funcionalidad de dataObject, podemos pedir a la base de datos todos los datos del usuario con el nickname que ha iniciado sesión e incluirlos como valor por defecto en los distintos campos.

Además, de forma distinta, se permite que el usuario cambie su actual contraseña, incluyendo un nuevo campo que debe rellenar con la nueva contraseña elegida. Este campo nunca se rellenará por defecto, y el usuario puede elegir no rellenarlo si no quiere cambiar la contraseña.

Una vez que el usuario ha modificado los datos deseados y pulsa el botón de modificar perfil, se llama al archivo `modificar_perfil.php`, que es quien comprueba que la contraseña sea correcta o no. En caso de que sea correcta, realiza una llamada al método *actualizarDatos*, de la clase Usuario,

que se encarga de modificar los datos de la tabla. Mediante una sentencia del tipo UPDATE, y que contiene aquellos datos que pueden ser modificados en el formulario, se puede guardar la nueva información en la base de datos.

En caso de que no se haya aportado una nueva contraseña, se vuelve a guardar la contraseña que ya existía. Estas comprobaciones se hacen con PHP directamente dentro del método. De nuevo, la contraseña se guarda cifrada.

Validación de formularios con JavaScript

Para la validación de formularios se han utilizado scripts de JavaScript, fragmentos de código que aseguren que los formularios se utilizan correctamente, por ejemplo para evitar campos vacíos o inyecciones de código. Estos scripts se generan en el head de la página y se llaman en el `onsubmit` del propio formulario. Esto permite que si el valor del script es false, no se continúe con la ejecución del procesamiento del formulario en PHP.

Para los formularios de perfil y alta de usuario se han tenido en cuenta en primer lugar que aquellos campos que deben ser obligatorios no estén vacíos. He conseguido esto de forma genérica utilizando un array con los campos requeridos y un bucle for-of de JavaScript. Además para algunos de estos campos se ha dado un requerimiento especial.

En el caso del alta de usuario, la longitud del nickname debe ser al menos de 4 caracteres, y no más larga de 15. La contraseña tiene un requerimiento similar, pero sin estar limitada por larga. En cuanto al correo electrónico, se exige que incluya al menos un @, y que no esté ubicado al inicio de la cadena, para preservar el formato habitual de las direcciones de correo. Y además, más importante, se obliga a que no contenga espacios en blanco, debe ser una única cadena.

Para el formulario de modificación del perfil se han añadido requerimientos similares, pero con una especificación más clara y particular en el mensaje de alerta cuando no se indica la contraseña al modificar los datos. Además, se tiene en cuenta para la nueva contraseña que, aunque no pueda ser menor de 4 caracteres, puede estar vacía.

Foro

La implementación del foro está pendiente de hacer, por limitaciones de tiempo y problemas surgidos durante la práctica que se tratarán en la defensa de la misma.