

APLICACIONES WEB

MEMORIA DEL PROYECTO

Práctica 2 - Arquitectura y prototipo del proyecto del grupo UniTools

Miembros del grupo :

Luis Cepeda, Carlos
Bilbao, Hugo Ribeiro,
Daniel Canseco, Fernando
Ruiz, Bruno Mayo



Índice general

1.	Descripción general de la práctica	2
2.	Scripts para las vistas	3
3.	Scripts adicionales	7
4.	Estructura de la Base de Datos	10

1. Descripción general de la práctica

En esta memoria, describiremos cómo hemos decidido estructurar nuestra página web llegado este punto de su desarrollo. Se trata de una práctica importante para nosotros, ya que su desarrollo y las decisiones que tomáramos iban a condicionar el resto de las prácticas. Es decir, la estructura general, así como la Base de Datos y las abstracciones que empleamos van a servirnos como base en sucesivas versiones.

Durante esta práctica hemos centrado nuestros esfuerzos en dos aspectos:

1. Organizar nuestra página web haciendo uso de **un esquema que divide en cuatro partes (cabecera, navegación ó menú, contenido y pie)**, siendo el contenido el único actualizado dinámicamente, en función de las peticiones recibidas.

A continuación, hemos dividido entre scripts de vista, de apoyo y de lógica. Esta distribución ha resultado ser muy provechosa y nos ha permitido avanzar más agilmente. Además, hemos ahorrado muchas líneas de código gracias a ella.

2. Desarrollar tres funcionalidades principales. Para comenzar a darle forma a nuestra web, hemos elegido por un lado funcionalidades esenciales: Es decir, sin ellas lo demás no sería posible, este es el caso de **Login y Registro**.

Al mismo tiempo, hemos desarrollado otras dos funcionalidades que, sin depender otras de ellas, no planteaban un gran reto técnico. Estas han sido **Herramientas y Foro**.

Además, y para facilitar nuestro trabajo en sucesivas versiones del proyecto, hemos añadido utilidades que nos servirán para administrar la página web. Para ello, hemos creado un usuario extra para la Base de Datos (aparte del propio root, con total acceso) con diferentes permisos y privilegios, así como una tabla extra en la Base de Datos **como log de los accesos fallidos** en el Login, lo que pensamos que puede ser un feature interesante para mejorar la seguridad de la web resultante. Discutiremos esto en más detalle en la sección 4 "*Estructura de la Base de Datos*".

A la hora de dividir el trabajo, decidimos dejar a cada miembro del equipo encargado de dirigir cada una de las partes. Dicho lo cual, hemos hecho un esfuerzo por realizar videollamadas semanalmente para coordinar el trabajo y asegurarnos de que todos estábamos al corriente del funcionamiento de lo demás. Hemos utilizado GitHub para gestionar las versiones y commits del proyecto. **El código mostrado en esta memoria, podría no estar completamente actualizado debido a pequeños cambios para pulir detalles, no obstante la idea general explicada será la misma en cualquier caso.**

2. Scripts para las vistas

Las vistas posibles de esta versión son entre las que permite navegar el Menú de la página web, aunque algunas de ellas no mostrarán ningún contenido si el usuario no ha sido previamente registrado.

- (scripts/perfil.php) Perfil incluirá en sucesivas versiones toda la información del usuario, incluido si se trata de un usuario Premium.
- (scripts/proyectos.php) Proyectos permitirá a los estudiantes subir código de sus proyectos.
- (scripts/foro.php) **Foro permite a los usuarios postear mensajes.**
- (scripts/mensajes.php) Mensajes requerirá estar loggeado.
- (scripts/herramientas.php) **Herramientas incluye algunas utilidades para los estudiantes** a modo de funcionalidades. En esta versión hemos añadido varias, que serán explicadas luego.
- (scripts/login.php) Login permite iniciar sesión al usuario, tal y como se explicará más tarde.
- (scripts/registro.php) Registrar permite crear un nuevo usuario, mediante una conexión a la Base de Datos, como se explicará a continuación.

En realidad, en nuestro portal sólo se accede a una página, que es index.php, cuyo código se muestra a continuación. Se ha incluido un caso condicional para hacer *session_start()*, para que en el caso que el usuario se acabe de conectar ó su sesión haya caducado, se puedan restaurar las variables globales.

Después, utilizando *require()* se incluyen las partes que toda vista debe tener: cabecera (estructura/cabecera.php), navegación (estructura/menu.php), un contenido (del que ahora se hablará) y un pie de página (estructura/pie.php).

```
1 <?php
2     if(!isset($_SESSION))
3     {
4         session_start();
5     }
6 ?>
7 <!DOCTYPE html>
8 <html>
9 <head>
10     <link rel="stylesheet" type="text/css" href="../css/hoja.css">
11     <title>INDEX</title>
12     <meta charset="UTF-8">
```

```
13 </head>
14 <body>
15 <div id="contenedor">
16     <?php
17         require("cabecera.php") ;
18         require("navegacion.php") ;
19     ?>
20     <div id="contenido">
21         <?php require("contenido.php");?>
22     </div>
23     <?php
24         require("pie.php") ;
25     ?>
26 </div> <!-- Fin del contenedor -->
27 </body>
```

Por tanto, lo único que cambiará según cual sea la vista es el contenido, administrado en estructura/contenido.php, el cual no solo se encarga de cambiar el contenido en función de la petición que le llegue, sino que además realiza el control de seguridad para comprobar que el usuario ha sido correctamente loggeado, y en caso contrario no mostrarle dicho contenido.

Para ilustrar esto, asumamos que alguien que no está ni siquiera loggeado intenta acceder a un contenido sólo disponible para administradores, manipulando la petición en la url para acceder a admin.php (Es decir, escribe *index.php?page=admin*). En ese caso, tal y como se muestra en el siguiente código, contenido.php no le dejará acceder tras comprobar con los parámetros de la variable global SESSION que, efectivamente, ni era administrador ni había iniciado sesión.

```
1 <?php
2
3 if(isset($_GET["page"])) {
4     (...)
5
6     else if($_GET["page"] == "admin") {
7
8         if ((!isset($_SESSION["login"]))&&(!isset($_SESSION["esAdmin"]))) {
9             echo "<p>No puedes ver este contenido, tienes que estar
10                loggeado y ser Administrador para visualizarlo.</p>";
11         }
```

Otra funcionalidad que hemos implementado es **Herramientas**, en *script/herramientas.php*. Se trata de una serie de utilidades sencillas que quisimos añadir para practicar nuestra programación en JavaScript y al mismo tiempo proveer de alguna herramienta que puedan encontrar nuestros compañeros interesante.

Por ahora, se han implementado cuatro herramientas:

1. Un formateador de palabras.
2. Un contador de palabras.
3. Un conversor de decimal a hexadecimal.
4. Un conversor de decimal a binario.
5. Un conversor de binario a decimal.

Todos salvo el primero son sencillas funciones de JavaScript cuyo desarrollo no merece la pena explicar. Lo único reseñable es que las tres últimas se han implementado, en realidad, con la misma función pero distintos parámetros.

La herramienta más interesante es la primera, ya que para ella se ha hecho uso de **Canvas**, que permite dibujar gráficas y similares en JavaScript como se muestra.

(1) Introduzca su palabra para darle formato:

Ejemplo

Convertir texto

EJEMPLO

(1) Introduzca su palabra para darle formato:

Hola :)

Convertir texto

HOLA :)

```

1 var c = document.getElementById("myCanvas");
2 var ctx = c.getContext("2d");
3 ctx.font = "italic small-caps 30px Verdana";
4 ctx.strokeText("Ejemplo",30,70);
5 function texto(){
6     $str = document.getElementById('valor_d4').value;
7     ctx.clearRect(0,0,c.width,c.height).fillText($str, 30, 70);
8     alert("Palabra cambiada!");
9 }
    
```

La otra funcionalidad que hemos implementado durante esta práctica, al menos en su versión más básica, es el Foro. Una vez un usuario se ha registrado e iniciado sesión, puede **tanto revisar la lista de Posts publicados como escribir un nuevo Posts**, con scripts/post.php.

Para mostrar los posts del foro, basta con realizar una query a la Base de Datos que ordene las entradas por antigüedad, cosa que se puede hacer sin problema gracias a los id, que ascienden automáticamente. Una vez se tienen las entradas, estas pueden asociarse a su usuario, para luego crear una tabla con su información (id y Nombre de Usuario).

```
1 require_once 'connectdb.php';
2
3 $query = mysqli_query($conn, "SELECT * FROM forumposts ORDER BY
4     id_post DESC");
5 while($res = mysqli_fetch_array($query)){
6
7     $userQuery=$res['user'];
8     $userById = mysqli_query($conn, "SELECT * FROM user WHERE
9     id_user=$userQuery");
10    $resultUser = mysqli_fetch_array($userById);
11    echo "<table class=\"posts\">";
12    echo "<tbody>";
13    echo "<tr>";
14    echo "<td>ID del post: " . $res['id_post'] . "</td>";
15    (...)
```

Publicar entradas en el foro es un poco más intrincado, ya que requiere coger el texto escrito por el usuario y realizar una inserción en la Base de Datos.

```
1 $sql = "INSERT INTO forumposts(usuario, content) VALUES ('$username'
2     ', '$contenido')";
```

La conexión a la Base de Datos tendrá que hacerse con los permisos necesarios para insertar, por lo que hemos decidido dar al usuario no root permisos de inserción. De todo ello se hablará más detenidamente en el Capítulo dedicado a la BBDD.

3. Scripts adicionales

Dos scripts que es importante mencionar son el de Login y el de Registro, ambos estrechamente ligados con la Base de Datos que será después descrita.

El contenido de scripts/registrar.php es simplemente un form sencillo, cuya action es procesarRegistro.php, en la misma carpeta. A continuación se describirán algunas de las partes más importantes de este script.

Primero, se crean variables que serán utilizadas para la conexión con la Base de Datos. Estas son los campos de la tabla usuario, que fueron previamente rellenos en registro.php. Además, se utilizan las funciones *trim* y *strip_tags* para hacer "sanitize" del input. Esto incluye eliminar espacios, e **impedir que el usuario introduzca código malicioso en las entradas**.

```
1 $servername = "localhost";
2 $username = htmlspecialchars(trim(strip_tags($_REQUEST["username"])));
3 $email = htmlspecialchars(trim(strip_tags($_REQUEST["email"])));
4 $password = htmlspecialchars(trim(strip_tags($_REQUEST["password"])));
5 $nick = htmlspecialchars(trim(strip_tags($_REQUEST["nick"])));
6 $rol = ($_REQUEST["rol"]);
7 $Premium = ($_REQUEST["premium"]);
8 require_once 'connectdb.php';
```

Acabado esto, toca conectarse a la Base de Datos, utilizando la última línea del código anterior, que llama a connectdb.php. En este script se intentará conectar a la Base de Datos, especificando el nombre del servidor, el usuario y la Base de Datos. Para evitar que un estudiante cualesquiera pueda modificar las tablas de la Base de Datos, **este acceso se lleva a cabo con el nombre de un usuario con pocos privilegios**.

```
1 <?php
2 $conn = mysqli_connect("localhost", "usuario1", "", "unitoolsdb");
3 if( mysqli_connect_error ()){
4     die ("Acceso a BBDD fallido : " . mysqli_connect_error());
5 }else{ echo "Connected successfully"; }
6 ?>
```


Una vez se haya establecido la conexión, el script procesarRegistro.php podrá realizar una query e incluir al usuario en las tablas de la Base de Datos. No obstante, esto lo hará tras primero comprobar que se ha podido conectar a la Base de Datos satisfactoriamente. Como se puede comprobar a continuación, la petición SQL se trata de una sencilla inserción en la tabla user, utilizando todas las variables que fueron previamente revisadas.

```
1 $sql = "INSERT INTO
2   user(id_User, email, password, Nick, Rol, Premium)
3   VALUES('$username','$email','$password','$nick','$rol','$Premium')
4   ";
5 if ($conn->query($sql) === TRUE) {
6     echo "New record created successfully";
7 } else { echo "Error: " . $sql . "<br>" . $conn->error; }
```

El script de procesarLogin.php es muy similar. Se reciben los valores que han sido introducidos en el form de login.php y estos se utilizan para realizar la query. Eso sí, como era de esperar en esta ocasión la petición SQL no es una inserción, sino un SELECT que comprueba que los valores son idénticos.

```
1 $query = mysqli_query($conn,
2   "SELECT * FROM user
3   WHERE Nick='$username'
4   AND password='$password'");
5
6 if(!$query){
7     echo mysqli_error($conn);
8     // Si falla la consulta, mejor no seguir ejecutando.
9     exit;
10 }
11
12 // Validamos los datos introducidos en el login
13 if($user = mysqli_fetch_assoc($query)) {
14     $conn->close();
15     $_SESSION['access_success'] = $username;
16     header("Location:index.php");
17 } else {
18     $_SESSION['access_error'] = '1';
19     header("location: login.php");
20 }
```

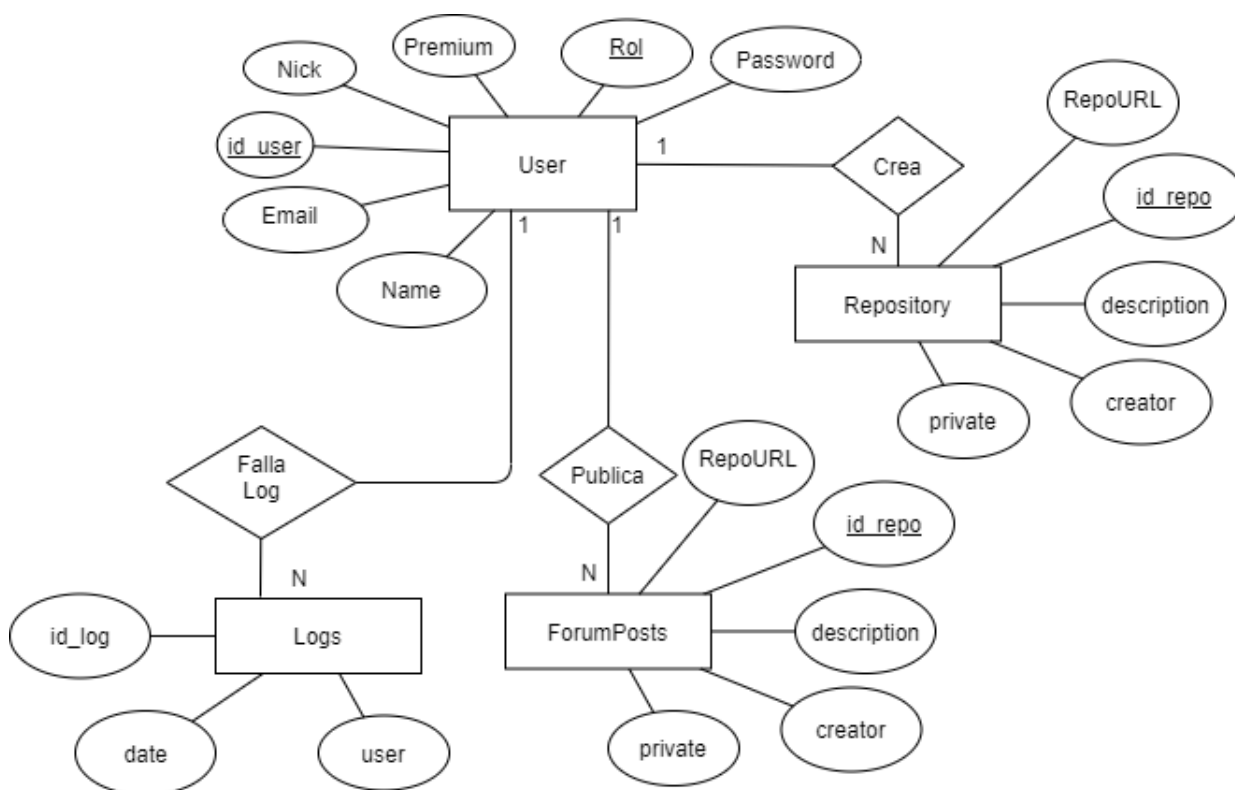
Como se puede apreciar en la línea 19, se emplea una variable global para marcar que el acceso ha fracasado en caso de que los datos sean erróneos, y otra para la

situación opuesta. Estas variables son luego revisadas por login.php, que llamó a procesarLogin.php, para saber si ha devuelto un valor la petición.

```
1 <?php
2 if(isset($_SESSION['access_error'])){
3     $try = $_SESSION['access_error'];
4     if($try == '1'){
5         echo "\n";
6         echo " <h3> <font color = 'red'> Datos incorrectos.</font>
7     </h3>";
8         $_SESSION['access_error'] = '0';
9     }
10 }
```

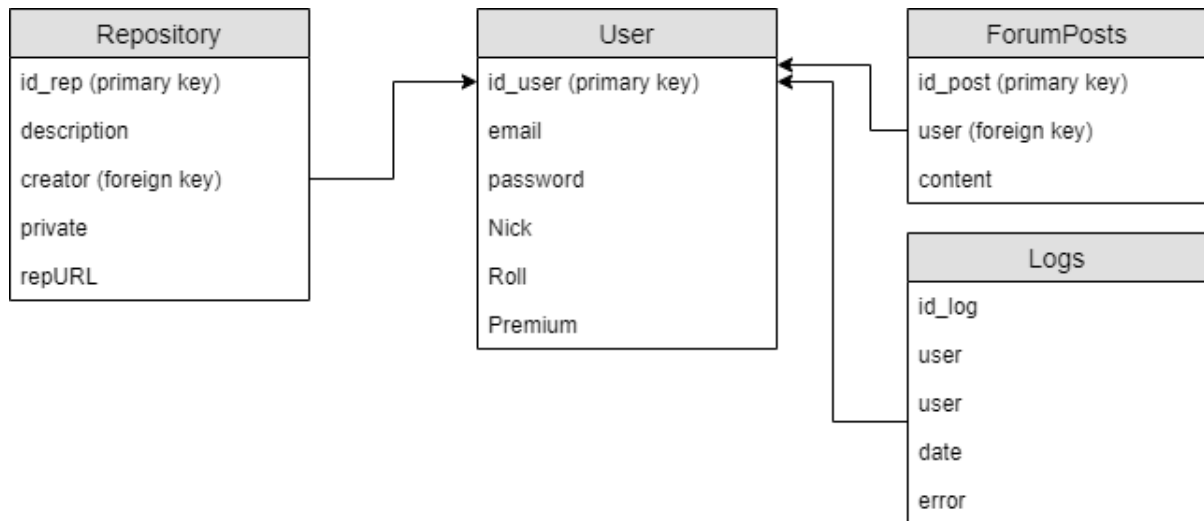
4. Estructura de la Base de Datos

A continuación repasaremos la estructura que hemos seguido para nuestra Base de Datos inicial, apoyándonos en PhpMyAdmin. Para ello, hemos decidido diseñar e incluir dos modelos muy utilizados para la definición de las BBDD, el Modelo Relacional y el Modelo Entidad Relación, de más alto nivel. También hablaremos de cómo hemos gestionado los permisos.



Como se puede ver, la Base de Datos consta de cuatro tablas: User, Repository, Logs y ForumPosts, cada una de ellas con su clave primaria y relacionada con otra (u otras, en el caso de User). La tabla User es la única con más de una primary key, ya que dos usuarios podrán tener el mismo id pero diferentes roles.

Además, se pueden apreciar las restricciones de cantidad. Un usuario puede crear N repositorios, así como generar N logs de fallos y publicar N posts en un foro. Todas las relaciones son para con el Usuario y de tipo (1,N). Desgraciadamente, **esta información se pierde al traducir el Modelo ER a un modelo relacional**, por culpa de las limitaciones inherentes al segundo. Este segundo esquema quedaría entonces como se muestra a continuación.



Estas cuatro tablas luego son pobladas en nuestra Base de Datos, los ids son incrementados por orden de apariencia y estos no pueden ser nulos, tal y como se puede apreciar en el siguiente código.

Además, como puede apreciarse en el código mostrado abajo, se han añadido restricciones para las foreign key y políticas CASCADE para las actualizaciones.

```

1  --
2  -- AUTO_INCREMENT de la tabla 'repository'
3  --
4  ALTER TABLE 'repository'
5      MODIFY 'id_rep' int(11) NOT NULL AUTO_INCREMENT;
6  (...)
7  --
8  -- Filtros para la tabla 'repository'
9  --
10 ALTER TABLE 'repository'
11     ADD CONSTRAINT 'repository_ibfk_1' FOREIGN KEY ('creator')
12     REFERENCES 'user' ('id_User') ON UPDATE CASCADE;
13 COMMIT;

```

Además, hemos tenido que tener en mente los permisos para hacer una Base de Datos segura y restrictiva. Para eso, hemos decidido usar **dos cuentas de usuarios**.

Los permisos del usuario root son los que eran de esperar, totales. La siguiente imagen muestra una captura de pantalla de los parámetros utilizados en PhpMyAdmin para añadir al usuario root con total privilegio y permisos de acceso.

Como se puede ver, el usuario root tiene todos los permisos tanto desde un punto de vista de Administración , como de Estructura (crear,alterar entradas...), y Datos para realizar queries a la Base de Datos (Seleccionar, Insertar, Actualizar, Eliminar, etc).

Por otro lado, los privilegios del usuario normal (**usuario1**) son mucho más limitados, como se puede ver debajo. En este caso, la contraseña que usamos para el usuario normal es unitoolsdb.

La BBDD (disponible en ***DataBase/unitoolsdb.sql***) se complicará y crecerá en sucesivas prácticas. Dicho esto, consideramos que la estructura inicial, así como el sistema de privilegios por el que hemos optado, son un buen punto de partida y no esperamos grandes cambios ni re-estructuraciones completas.