

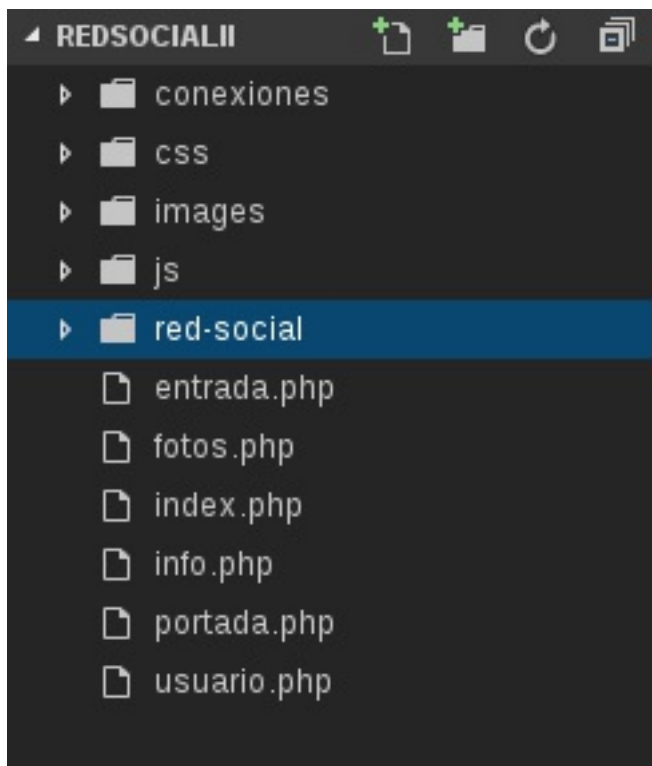
DOCUMENTACIÓN PRÁCTICA EVALUABLE II

RED SOCIAL

Por: Rubén Calvo Villazán

Distribución de ficheros

La distribución de ficheros para esta práctica es la siguiente:



En **conexiones** están todos los ficheros php que realizan una petición a la base de datos y que son llamados desde los ficheros php de la red social. Estos son:

- amistad.php
- borrar_usuario.php
- cambiar_foto.php
- cambiar_info.php
- cargar_entradas.php
- dbconexion.php
- registro.php
- salir.php
- sesion.php
- subir_comentario.php
- subir_entrada.php
- usuarios_activos.php

En **css** encontramos todos los ficheros que dan el estilo visual a la red social.

En **images** se encuentran todas las imagenes que usa la red social, así como las que usan los usuarios para su perfil. Cuando un usuario suba una foto a la red social para cambiar la que tiene de perfil, será almacenada en este directorio.

En **js** tenemos un fichero *include.js*. Este fichero contiene varias funciones en javascript para validar formularios y mostrar mensajes en el navegador. Estas funciones son usadas por los demás ficheros de la red social.

En **red-social** se encuentra el fichero *contacto.php*, que muestra en la red social el nombre del autor, y esta documentación.

Finalmente en el directorio general tenemos una serie de ficheros php, estos son los distintos directorios donde el usuario podrá navegar por la red social. Los ficheros son:

- entrada.php
- fotos.php
- index.php

- info.php
- portada.php
- usuario.php

Son la parte visible para el usuario de la red social.

Base de datos

Antes de entrar en detalle en los demás ficheros, si vemos la base de datos tenemos la siguiente estructura:



Como podemos ver, encontramos las siguientes tablas:

- amigos
- comentarios
- entradas
- usuarios

Si vemos detalladamente cada tabla:


amigos


#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id_amistad	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
2	actual	varchar(20)	utf8_spanish_ci		Yes	NULL	
3	amigo	varchar(20)	utf8_spanish_ci		Yes	NULL	

;

Siendo *actual* el usuario que realiza la petición de amistad, y *amigo* el que la recibe.


comentarios

	#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	1	id_com 	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
<input type="checkbox"/>	2	id_ent	int(10)		UNSIGNED	No	None	
<input type="checkbox"/>	3	user_id	int(10)		UNSIGNED	No	None	
<input type="checkbox"/>	4	username	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	5	texto_com	varchar(150)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	6	fecha_com	date			No	None	

▲           

;


entradas

	#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	1	id_ent 	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
<input type="checkbox"/>	2	user_id	int(10)			No	None	
<input type="checkbox"/>	3	username	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	4	titulo	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	5	texto	varchar(250)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	6	media_ent	varchar(150)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	7	fecha_public	date			No	None	

;

En el campo *media_ent* guardamos la url del contenido multimedia de la entrada en caso de que lo tenga.

usuarios

#	Name	Type	Collation	Attributes	Null	Default	Extra
<input checked="" type="checkbox"/>	1 user_id 	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
<input type="checkbox"/>	2 nombre	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	3 apellidos	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	4 username	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	5 email	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	6 password	varchar(20)	utf8_spanish_ci		No	None	
<input type="checkbox"/>	7 last_login	date			Yes	NULL	
<input type="checkbox"/>	8 fecha_registro	date			No	None	
<input type="checkbox"/>	9 foto_usuario	varchar(50)	utf8_spanish_ci		No	images/usuarios/default.png	
<input type="checkbox"/>	10 activo	int(1)		UNSIGNED	No	None	

;

En el campo *foto_usuario* almacenamos el directorio donde el usuario tendrá guardada su foto. El campo *activo* guarda dos posibles valores, 0 o 1, siendo 0 si el usuario no está activo o 1 si lo está.

Red social

Para explicar el funcionamiento de la red social, vamos a suponer el recorrido que haría un usuario normal.

dbconexion.php

El primer paso es conectarnos a la base de datos. La forma en que vamos a realizar todas las consultas y la conexión es por **PDO**

En este fichero definimos el host, el usuario, la contraseña y el nombre de la base de datos.

Eliminamos los errores para que no aparezcan por pantalla (por seguridad):

```
error_reporting( ~E_DEPRECATED & ~E_NOTICE );
```

Y realizamos la conexión:

```
$conn = new PDO("mysql:host=$DBHOST;dbname=$DBNAME",  
$DBUSER, $DBPASS);  
$conn->setAttribute(PDO::ATTR_ERRMODE,  
PDO::ERRMODE_EXCEPTION);
```

index.php

Este es el lugar de partida, lo primero que encontramos son dos formularios. El formulario de registro que rellena los campos para el registro y los envía mediante un **post** al fichero *registro.php*.

En este fichero (y en todos los que procesan una entrada del usuario) lo que hacemos es asegurarnos de que no se trata de un sql injection. Esto lo hacemos limpiando la entrada con:

```
// Limpiar la entrada para evitar sql injection  
  
$user_firstname = trim($_POST['user_firstname']);  
$user_firstname = strip_tags($user_firstname);  
$user_firstname = htmlspecialchars($user_firstname);  
  
$user_lastname = trim($_POST['user_lastname']);  
$user_lastname = strip_tags($user_lastname);  
$user_lastname = htmlspecialchars($user_lastname);  
  
$user_name = trim($_POST['user_name']);  
$user_name = strip_tags($user_name);  
$user_name = htmlspecialchars($user_name);  
  
$user_email = trim($_POST['user_email']);  
$user_email = strip_tags($user_email);  
$user_email = htmlspecialchars($user_email);  
  
$user_password = trim($_POST['user_password']);
```

```
$user_password = strip_tags($user_password);  
$user_password = htmlspecialchars($user_password);
```

Para guardar la contraseña en la base de datos, se le hace un **encriptado** sha256, de forma que cuando el usuario la introduzca para iniciar sesión, se compararían las contraseñas encriptadas.

```
$user_password = hash('sha256', $user_password);
```

Después guardamos la fecha de registro y realizamos una consulta previa con el *email* y el *username*, como ambos son **unique** hacemos un **select** de los dos y comprobamos el número de filas resultado. Si nos devuelve filas distintas de 0, el usuario ya existe, sino realizamos la inserción.

Con esto, ya estarían todos los datos del usuario guardados en la base de datos y por lo tanto quedaría registrado. Finalmente hacemos los unset correspondientes.

El formulario de iniciar sesión realiza un **post** al fichero *sesion.php* ubicado en *conexiones*. En este fichero leemos todos los campos del formulario, (Comprobando siempre sql injection como hemos visto antes). Además hacemos un **select** de la base de datos donde el usuario y la contraseña deben de ser iguales a los que ha introducido el usuario, de forma que comprobando las filas resultado podemos saber si el usuario existe o no.

Si el usuario existe, realizamos el inicio de sesión. Guardamos la fecha del login, la actualizamos y ponemos su campo de activo a 1.

Posteriormente inicializamos el array **\$_SESSION** con todos los campos del usuario correspondiente en la tabla de usuarios:

```

$_SESSION['user_id'] = $array[0]['user_id'];
$_SESSION['nombre'] = $array[0]['nombre'];
$_SESSION['apellidos'] = $array[0]['apellidos'];
$_SESSION['username'] = $array[0]['username'];
$_SESSION['email'] = $array[0]['email'];
$_SESSION['password'] = $array[0]['password'];
$_SESSION['last_login'] = $array[0]['last_login'];
$_SESSION['fecha_registro'] = $array[0]
['fecha_registro'];
$_SESSION['foto_usuario'] = $array[0]
['foto_usuario'];
$_SESSION['activo'] = $array[0]['activo'];

```

Finalmente hacemos un redireccionado a *portada.php?bienvenida=si*.

En el formulario de inicio de sesión, tenemos un botón de *recordar*. Si el usuario inicia sesión con este botón seleccionado, una función javascript guarda dos **cookies** en el navegador. La cookie *name* y la cookie *pass* con el nombre y la contraseña. La próxima vez que el usuario vaya a *index.php* y estén las cookies en el navegador, el formulario de sesión se rellenará automáticamente.

```

var user_name = getCookie("name");
var user_pass = getCookie("pass");
.....
document.getElementsByClassName("header-form")[0]
[0].value = user_name;
document.getElementsByClassName("header-form")[0]
[1].value = user_pass;

```

Estas cookies se establecen en la misma función que valida el formulario, poniendoles una fecha de expiración de 10 días.

En cuanto a las funciones para validar formularios, se encuentran en

include.js.

Las funciones comprueban que:

- todos los campos son rellenados
- las contraseñas si se introducen dos veces, deben coincidir
- el email tiene formato '@gmail.com'
- la contraseña debe tener un tamaño mayor a 6 caracteres
- etc

Además colorean el campo donde el usuario ha introducido mal la información.

Cuando el usuario se registra en la red social, aparece una nueva ventana dandole la bienvenida. Esto se hace ejecutando un **javascript desde php** al ver si se ha realizado el post del formulario de registro.

```
if(isset($_POST['SubmitButton'])){\n    include_once 'conexiones/registro.php';\n    echo "<script> var myWindow =\nwindow.open(\"\\\", \"Welcome\\\",\n\"toolbar=yes,scrollbars=yes,resizable=yes,top=300,left=\n500,width=300,height=100\\");\n        myWindow.document.write(\"<h4>¡Gracias\npor registrarte en Social Network UGR! Ya puedes iniciar\nsesión escribiendo tus datos en el formulario de la\ncabecera</h4>\\");\n        </script>";\n}
```

portada.php

Una vez el usuario inicie sesión, pasará a esta página. Lo primero que realizamos es comprobar si existe *bienvenida*. El fichero index como

hemos visto antes nos redirecciona hasta esta página **con un parámetro**, *bienvenida=si*. Esto es para que aparezca una ventana que se desplaza hacia la derecha dandonos la bienvenida. Para ello volvemos a ejecutar **javascript desde php**, haciendo que la función que abre la ventana solo aparezca si está el `$_GET['bienvenida']` En esta función se da anchura de 100% a una ventana (ya existente), y en la zona de texto se le añade el mensaje de bienvenida.

```
document.getElementById("myNav").style.width = "100%";
```

En la función de cerrar la ventana de bienvenida, se le vuelve a dar width de 0%.

Si seguimos analizando el fichero, vemos que en la cabecera de la red social generamos el contenido según el usuario actual, esto es que usamos los valores del array `$_SESSION` para crear el enlace y foto al perfil del usuario actual.

La zona en la que aparecen foto y nombre de todos los usuarios, es un muro de la fama. Esto significa que apareceran según el número de entradas que hayan publicado. Esto se hace con la siguiente consulta:

```
$sql = "SELECT usuarios.username, usuarios.foto_usuario,
entradas.username, entradas.fecha_public FROM entradas,
usuarios WHERE usuarios.username = entradas.username
GROUP BY entradas.username HAVING entradas.fecha_public >=
CURDATE()-10 ORDER by count(*) DESC;";
```

Después generamos dinámicamente todos los usuarios con el resultado de la consulta, así como la función javascript que nos mostrará el mensaje con los títulos de sus entradas cuando pasemos el ratón por encima.

Todas las consultas se realizan de la forma:

```
$sql = "" $result = $conn->query($sql); $array = $result->fetchAll(PDO::FETCH_ASSOC);
```

Teniendo en *\$array* los valores resultado de la consulta.

Para ver los usuarios activos, se realiza una consulta buscando los que el campo *activo* lo tengan a 1. Este campo se actualiza mediante **ajax**, es decir, no hace falta refrescar la página, basta con pulsar el botón.

Esto se hace mediante **jquery**, cuando se pulsa el botón de usuarios activos, se realiza una petición a *conexiones/usuarios_activos.php*, que realiza la consulta y devuelve el resultado en forma de **json**

```
echo json_encode($array);
```

Recibimos la información y la leemos como si se tratase de un array. Generamos el código html para darle formato a la información y finalmente hacemos un **append** con javascript para fijar el contenido que hemos leído. Todo esto sin recargar la página. Antes de hacer el **append**, realizamos:

```
$("#recent-users-profiles").html(
    "<section></section>"
);
```

Fijando un *section* vacío, esto es para que no aparezcan los usuarios repetidos en cadena y limpiar la zona con los usuarios previos, de forma que no se repita información.

Las entradas de los usuarios se cargan de una forma similar. Tenemos dos botones que nos llevan a paginas con las entradas antiguas o las más nuevas. Esto se hace dinámicamente estableciendo un enlace según el parámetro que reciba *portada.php*. Las entradas van de 6 en 6,

así que al botón le ponemos que nos lleve a origen+6 entradas, de forma que en la consulta hacemos:

```
$sql = "SELECT DISTINCT entradas.id_ent, usuarios.foto_usuario,
entradas.username, usuarios.username, entradas.titulo,
entradas.texto, entradas.fecha_public FROM entradas, usuarios
WHERE entradas.username = usuarios.username ORDER BY
entradas.id_ent DESC LIMIT 6 OFFSET $siguiente;";
```

Seleccionando con el offset, las 6 siguientes entradas de donde partimos. De forma que cada vez que pasamos de página con el botón, se incrementa en 6 y se hace por lo tanto una consulta con las 6 entradas siguientes.

En las entradas encontramos también un botón de actualizar las entradas. Esto se hace también con **ajax**. Hacemos una petición al fichero *conexiones/cargar_entradas.php*, que como hemos visto antes nos devuelve la información (las primeras 6 entradas) en formato **json**. Después procesamos el **json** como si fuese un array y generando el html para darle formato. Finalmente en la zona de entradas hacemos un **append** del contenido.

En las entradas si el texto supera los 150 caracteres, lo cortamos y le añadimos '...'.

Para ver los títulos de las entradas de cada usuario al pasar el ratón por su foto, volvemos a hacer una petición con **ajax**. Sin embargo esta es distinta ya que no se hace en el propio ajax. Lo que hacemos es a la hora de cargar cada foto de usuario en la página, le ponemos como título a la etiqueta el título de las entradas. Después con **jquery** llamamos a la función **qtip()** que lo que hace es crear una ventana emergente mostrando el contenido que el receptor tenga en el tag *title*, por lo tanto solo nos queda comprobar cuando el usuario pase el ratón sobre la foto. Esto lo hacemos con la función *hover* de **jquery**.

Por lo tanto nos queda lo siguiente:

```
$(document).ready(function () {  
    $("#get_content").hover(function () {  
        $('#usuario').qtip();  
    });  
});
```

Nota: Haremos uso de ajax y jquery en todas las páginas de la red social, aunque solo lo explicaré aquí

usuario.php

En todas las páginas de la red social comprobamos primero si existe una **sesión activa**, si no es así, redirigimos a *index.php*

```
if (!isset($_SESSION["username"])) {  
    header("location: index.php");  
}
```

En esta página lo que hacemos para cargar el contenido es comprobar con qué parámetro hemos accedido. Esto es con un *if else* del valor de usuario. Si venimos con:

```
$_SESSION['username'] == $_GET['usuario']
```

Cargamos la página del usuario actual, sino la de otro usuario.

Si es la página de otro usuario, hacemos una consulta como la de *portada.php* para cargar todas las entradas donde el autor sea *\$_GET['usuario']*, el usuario correspondiente. Guardamos la información en el array resultado y dinámicamente generamos el html con la información para escribirlo en la página.

Si por el contrario nos encontramos en la página del usuario que tiene abierta la sesión, generamos también sus entradas pero además generamos un formulario para que pueda escribir una entrada nueva. Este formulario lo procesa el fichero *conexiones/subir_entrada.php* Que comprueba sql injection, realiza la inserción en la base de datos pero además comprueba si se ha subido **contenido multimedia**. Esto es buscando en el **texto** de la entrada si existe un link (http). Si existe, se corta el texto desde ahí hasta el final y nos quedamos con la primera palabra separada por espacios, es decir, con el propio link.

```
if($tiene_media){
    $index = strrpos($texto, 'http');

    $length = $index;
    $aux_text = "";
    while (isset($texto[$length])) {
        $aux_text .= $texto[$length];
        $length++;
    }
    $result = explode(" ", $aux_text, 2);
    $media = $result[0];

    $texto = str_replace($media, "", $texto);
    $media = str_replace('watch?v=', 'embed/',
    $media);
```

De forma que **según tenga foto, video o nada, la entrada se generará de forma distinta**.

Hacemos el **insert** correspondiente según tenga media o no, y volvemos a la página *usuario.php*.

info.php

Esta página carga la información del usuario correspondiente, tiene una

estructura similar a *usuario.php*, según el valor del parámetro usuario, cargaremos una página o otra. Si es el usuario que tiene iniciada la sesión, cargaremos su foto, un boton para poder subir una imagen nueva (que se guardará en el directorio *images/usuarios/*) Además cargaremos cada campo de su base de datos para que consulte su información, así como campos de entrada para que pueda modificar la informacion que desee. Esta petición la recibirá el fichero *conexiones/cambiar_info.php*. Para poder cambiar los campos por separado, lo que hacemos en este fichero es para cada valor del `$_POST` ver si está asignado, y si lo está, realizamos el correspondiente **update**. De forma que el usuario pueda actualizar su información de forma individual.

Si el campo a cambiar es el nombre de usuario, se actualiza también el valor en las tabla de entradas, ya que para mostrarlas en la página se usa el nombre de usuario como campo de búsqueda. Cambiando siempre el valor del campo actualizado en la variable `$_SESSION`.

Si el usuario quiere cambiar su foto de perfil, nos iremos al script *conexiones/cambiar_foto.php*, que recibirá la imagen, procesará que tenga el tamaño correspondiente y la guardará en el directorio correspondiente, en este caso en *images/usuarios/*. Además actualizaremos el valor de `$_SESSION['foto_usuario']`.

En la página encontraremos también un botón para borrar el usuario, que hará un **post** a *conexiones/borrar_usuario.php* haciendo un **delete** de las tablas de *usuarios* y *entradas*, de forma que se borre toda la información del usuario y sus entradas asociadas. Si el usuario pulsa el botón para borrarse su cuenta, nos saldrá un *alert* en javascript para preguntarnos si estamos seguros de querer borrarla. Esta función se encuentra en *js/include.js*.

Si estamos en *info.php* de otro usuario, cargaremos su información correspondiente además de un botón que nos permitirá agregarlo como amigo. Este botón llama al script *amistad.php*, que hará un **insert**

en la tabla de *amigos*, con el nombre de usuario del usuario de la sesión, y el receptor de la petición.

fotos.php

En esta página mostramos el contenido multimedia de todas las entradas que lo tengan, para eso hacemos una consulta con un **select** y procesamos el array resultado con todo el contenido multimedia seleccionado, le damos el formato correspondiente (si es una imagen o si hay que cargar un *iframe* para reproducir el video) y generamos el html.

entrada.php

En este fichero se muestra la correspondiente entrada, lo primero que hacemos es comprobar que haya una sesión activa (como en todos los ficheros), después definimos dos funciones, *parsearMedia* y *setVideoOFoto* en php que nos servirán para ver si la entrada tiene contenido multimedia, y en caso de que lo tenga dar formato a la entrada dinámicamente.

Después generamos la entrada correspondiente que recibimos por parámetro. Desde *portada.php* o cualquier otra página con entradas, enlazamos cada entrada con *entrada.php?entrada=* y el id de la entrada, de forma que para obtenerla debemos hacer un **select** de la tabla *entradas* con el id correspondiente.

Una vez tenemos toda la información de la tabla, parseamos el texto para ver si tiene contenido multimedia. Si no tiene, cargamos el texto normal, si tiene foto la mostramos y si tiene vídeo mostramos el reproductor con un *iframe*. Para ver si tiene media o no, basta con hacer algo parecido a lo que hicimos en la página *usuario.php* para ver si al subir la entrada tenía media o no, es decir, comprobamos si contiene *http* y si contiene *youtube* es vídeo, y sino es imagen.

Una vez cargada la entrada, debajo cargamos la zona donde el usuario puede dejar su comentario. Antes de mostrar el formulario,

comprobamos que el usuario y el autor de la entrada son amigos. Esto se hace con un **select**. Si el usuario es el autor de la entrada, o el autor de la entrada es amigo del usuario, éste puede dejar un comentario, sino nos sale un mensaje de aviso.

Para subir un comentario, se hace un **post** a *conexiones/subir_comentario.php* En este fichero se leen las variables del `$_POST`, (autor, id de la entrada, ...) y se realiza el **insert** correspondiente.

Posteriormente en *entrada.php* se realiza un **select** de la tabla *comentarios* con el id de la entrada actual, y si encuentra los muestra todos en el html.

salir.php

Finalmente, en *conexiones/salir.php* tenemos el fichero que se llama cuando el usuario cierra su sesión (pincha en *bye, bye...* en la cabecera), en este fichero lo que hacemos es un **update** del campo *activo* del usuario para ponerlo a 0, así como los *unset* correspondientes al array `$_SESSION`. Luego hacemos una redirección a *index.php*