ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

APLICACIÓN WEB PARA EL MANTENIMIENTO DEL LABORATORIO DE FOTÓNICA

Maider Huarte Razquin

Alayn Loayssa Lara

Pamplona, 26 de noviembre de 2010

ÍNDICE

Resumen del PFC

El objeto del proyecto ha sido la creación de una aplicación web para el control de los componentes y equipos disponibles en el laboratorio de Fotónica, situado en el edificio de los Pinos del campus de Arrosadía.

Este laboratorio disponía de una base de datos basada en *FileMaker Pro*. Para acceder a ella era necesaria la instalación de este software en cualquier ordenador con acceso a la misma y la versión de FileMaker Pro era muy antigua. Además los manuales y hojas de características de los componentes y equipos estaban en archivadores, con la posibilidad de pérdida que esto conlleva. Finalmente, un usuario al acceder a esta aplicación tenía el control absoluto sobre la base de datos, teniendo que tener cuidado con los usuarios que tenían acceso, ya que podían realizar cambios en ésta, e incluso eliminarla. Por ello se requirió la creación de una aplicación nueva para tener acceso a la base de datos. Se creyó que lo más útil sería una aplicación accesible desde cualquier ordenador de la universidad, sin necesidad de instalar ningún software en el ordenador del cliente. Debido a esto se pensó en una aplicación web, o sea, una página web dinámica. Para llevarla a cabo se tuvieron en cuenta diferentes lenguajes de programación, posibles bases de datos y servidores sobre los que poner en funcionamiento la aplicación. Debido a que PHP es un lenguaje de programación que permite crear páginas web dinámicas, tiene un gran parecido con los lenguajes más comunes de programación estructurada, puede ser desplegado en la mayoría de servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno y permite la conexión con diferentes tipos de servidores de bases de datos, fue el lenguaje elegido para la programación de esta aplicación. Además se eligió un servidor de bases de datos MySQL, ya que es un sistema muy utilizado en aplicaciones web y se puede acceder a él a partir de aplicaciones escritas en diferentes lenguajes de programación, entre ellos PHP. El servidor web seleccionado fue Apache porque es el servidor web HTTP más utilizado y es muy robusto. Para que la aplicación sea lo más segura posible, la conexión cliente-servidor se ha hecho mediante un protocolo seguro (HTTPS).

El acceso a la aplicación está restringido, y los usuarios con acceso a ella tienen diferentes grados de privilegios. Hay tres tipos de usuarios, dependiendo de los privilegios que tengan; los de tipo *usuario*, *intermedio*, y *administrador*. Los de tipo *usuario* pueden acceder a la base de datos de una forma muy restringida, los de tipo *intermedio* tienen acceso a una mayor parte de la aplicación y los de tipo *administrador* son los que tienen el control absoluto de la aplicación, lo que conlleva el control absoluto de la base de datos. Desde esta aplicación se va a tener acceso a la base de datos que almacena todos los componentes y equipos del laboratorio y además se van a poder realizar copias de seguridad de ésta, posibilidad que no existía en la base de datos que había anteriormente en el laboratorio. En definitiva, dependiendo del tipo de usuario que tenga acceso a la aplicación, se va a poder crear, y eliminar usuarios, además de editar los datos relativos a éstos. También se pueden insertar nuevos registros en la base de datos, editarlos y eliminarlos y realizar copias de seguridad de la base de datos. Además los manuales y hojas de características de los componentes y equipos del laboratorio serán accesibles desde la propia aplicación, pudiendo tener acceso a ellos diferentes personas simultáneamente sin necesidad de ir al laboratorio a buscarlos.

Para acceder a la aplicación hay que hacerlo escribiendo en la barra de direcciones del navegador lo siguiente:

https://172.18.67.26/FOTONICA ó https://172.18.67.26/FOTONICA/index.php

1. Descripción del proyecto

En este apartado se va a describir el contexto en el que se llevó a cabo el proyecto, así como las fases de su elaboración, la estructura del mismo y sus especificaciones.

1.1. Requerimientos

Este proyecto se lleva a cabo por la necesidad de actualizar la forma de acceso a la base de datos del laboratorio de Fotónica. Este laboratorio disponía de una base de datos, basada en FileMaker Pro, en el que se almacenaban las características de todos los componentes y equipos del laboratorio. Sin embargo, todas las hojas de características y los manuales de los mismos estaban en archivadores, con el riesgo que esto implica de poder ser extraviados. Además la versión de FileMaker Pro utilizada era muy antigua.

Con este proyecto se quiso conseguir una aplicación que diera acceso a esta base de datos. Lo primero que se tuvo en cuenta para la realización de la misma fue que se tuviera acceso a ella en cualquier ordenador, sin necesidad de instalar ningún software en el equipo cliente. Por esta razón se pensó en una aplicación web, es decir, en una página web dinámica. A consecuencia de esto se pensó en qué lenguajes de programación podían utilizarse, que servidor de bases de datos utilizar y sobre que servidor web ponerla en funcionamiento. Entre los lenguajes de programación existentes se optó por utilizar PHP, ya que originalmente fue diseñado para la creación de páginas web dinámicas, puede ser desplegado en la mayoría de servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Además su gran parecido con los lenguajes más comunes de programación estructurada, permiten crear aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones. PHP permite la conexión con diferentes tipos de servidores de bases de datos tales como MySQL, Oracle y Microsoft SQL Server, entre otras. Además este lenguaje de programación soporta la programación orientada objetos, la cual es una manera de programar que trata de modelar los procesos de programación de una manera cercana a la realidad: tratando a cada componente de un programa como un objeto con sus características y funcionalidades. El servidor de bases de datos elegido para esta aplicación fue MySQL debido a que es muy utilizado en aplicaciones web y está muy ligado a PHP, aunque se puede acceder a él a partir de aplicaciones escritas en diversos lenguajes de programación, incluyendo C, C++, Pascal, Java y PHP, entre otros. También puede ser utilizado bajo diferentes sistemas operativos. En él las tablas están compuestas por registros (cada fila de la tabla es un registro) y columnas (también llamadas campos). MySQL permite a la aplicación realizar varias tareas a la vez y permite proveer servicio y procesamiento a múltiples usuarios simultáneamente. Además es un software libre. Se optó por elegir como servidor web Apache, ya que funciona muy bien en combinación con PHP y MySQL y es el servidor web más extendido. Para que la aplicación sea lo más segura posible el protocolo de transferencia entre cliente-servidor utilizado es HTTPS, para evitar ataques en los que se puedan conseguir contraseñas de acceso y la base de datos pueda ser modificada o incluso eliminada. Este protocolo de transferencia de hipertexto es un protocolo de red basado en HTTP, destinado a la transferencia segura de datos, es decir, es la versión segura de HTTP. La idea principal de HTTPS es la de crear un canal seguro sobre una red insegura. Esto proporciona una protección razonable contra ataques, siempre que se

empleen métodos de cifrado adecuados y que el certificado del servidor sea verificado y resulte de confianza. El sistema HTTPS utiliza un cifrado basado en SSL/TLS para crear un canal cifrado más apropiado para el tráfico de información sensible que el protocolo HTTP. De este modo consigue que esta información sensible (usuarios y contraseñas) no pueda ser usada por un atacante que haya conseguido interceptar la transferencia de datos de la conexión, ya que lo único que obtendrá será un flujo de datos cifrados que le resultará imposible de descifrar. Tanto cliente como servidor se ponen de acuerdo de antemano sobre la clave a usar para cifrar y descifrar la información que es enviada por la red. Entonces el remitente cifra la información usando esa clave, la envía al destinatario y éste la descifra con la misma.

En definitiva, se optó por utilizar un servidor web Apache y programar un conjunto de scripts en PHP y SQL que se comunicaran con la base de datos MySQL.

1.2. Fases del proyecto

Para la elaboración del proyecto se llevaron a cabo las siguientes fases:

- Primera fase. Estudio de los lenguajes de programación.
- Segunda fase. Instalación del software necesario para la realización de la aplicación.
- Tercera fase. Creación de la aplicación.
- Cuarta fase. Migración de datos de una base de datos a otra.
- Quinta fase. Configuración de las copias de seguridad.
- Sexta fase. Modificación de la apariencia de la aplicación.
- Séptima fase. Migración de la aplicación al servidor.

A continuación se van a explicar cada una de estas fases.

Primera fase. Estudio de los lenguajes de programación

Durante esta primera fase se estudiaron los lenguajes de programación que se han utilizado para la realización de esta aplicación, PHP y SQL. Para aprender a programar utilizando estos lenguajes se estudiaron diversos tutoriales. El lenguaje de programación PHP es el utilizado para programar todos los scripts que forman la aplicación y gracias al lenguaje de consulta estructurado SQL se tiene acceso a la base de datos. Este lenguaje de programación permite efectuar consultas con el fin de recuperar información de la base de datos, así como hacer cambios en ella.

Segunda fase. Instalación del software necesario para la realización de la aplicación

Durante la segunda fase se estudió qué software era necesario instalar para poder programar la aplicación y que ésta funcionara correctamente. Se llegó a la conclusión de que había que instalar un servidor web Apache, un módulo para programación PHP, el servidor de bases de datos MySQL y un editor PHP. Como el servidor se iba a instalar en un equipo cuyo sistema operativo fuera Windows se encontró un paquete llamado EasyPHP que instala el servidor Apache, junto con el módulo para programación PHP, el servidor de bases de datos MySQL y PHPMyAdmin. Esta última aplicación es muy útil ya que es una aplicación web para administrar las bases de datos de MySQL.

Además se instaló el PHP Editor, que fue muy útil a la hora de programar los scripts en PHP. También se tuvo que instalar el programa FileMaker Pro, para tener acceso a la base de datos antigua.

Tercera fase. Creación de la aplicación

Durante esta fase se crearon los 65 scripts que forman la aplicación, con un total de 5800 líneas de código. Además se creó una base de datos llamada *fotonica* compuesta por 11 tablas. La creación de la base de datos se hizo con la herramienta PHPMyAdmin, así como las tablas que la componen y los campos que componen cada tabla.

Cuarta fase. Migración de datos de una base de datos a otra

Cuando ya toda la aplicación estuvo programada se migraron todos los datos que había en la base de datos antigua a la realizada para esta aplicación.

Quinta fase. Configuración de las copias de seguridad

En esta fase se estudió cuál sería la mejor forma de realizar las copias de seguridad de la base de datos. Se llegó a la conclusión de que sería útil que el usuario *administrador* pudiera realizar y restaurar copias de seguridad siempre que él quisiera. Además debería poder elegir el directorio, dentro del servidor, donde se almacenan esas copias. También se opto por programar la realización de una copia de seguridad semanalmente de manera automática.

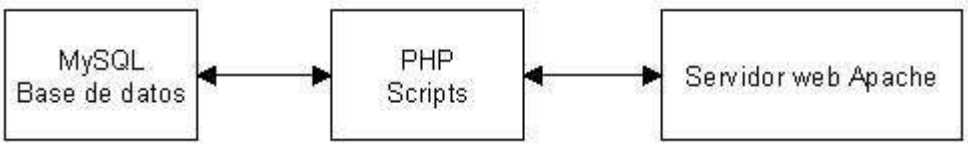
Sexta fase. Modificación de la apariencia de la aplicación

Durante esta fase se crearon las hojas de estilo que darían a la aplicación su apariencia final.

Séptima fase. Migración de la aplicación al servidor

Por último se migró toda la aplicación al servidor. Para ello se instaló en él el paquete EasyPHP y se configuró de la manera más adecuada. Además se copiaron en él todos los scripts, así como los archivos relacionados con la base de datos. También se hizo alguna modificación en un script para que la aplicación funcionara en el servidor.

1.3. Estructura de la aplicación



Esta aplicación, como se puede observar en el diagrama anterior, está compuesta por una base de datos MySQL a la que se tiene acceso a través de unos scripts programados en PHP, los que se comunican con el servidor web Apache.

La base de datos creada se llama *fotonica* y está compuesta por 11 tablas:

- config_backup
- config_mail
- elementos
- categoria
- localizacion
- estado
- usuario
- uso
- usuarios
- privilegios_usuarios
- categorias_usuarios

A continuación se describe qué información almacena cada tabla:

Tabla *config_backup*

Esta tabla se compone de un campo llamado Directorio, y siempre va a almacenar un único registro. Este va a ser el directorio en el que se van a almacenar las copias de seguridad en el servidor, tanto la/s que hace/n el/los administrador/es, como las que se hacen automáticamente semanalmente. Además va a ser el directorio desde el que se van a seleccionar los archivos necesarios a la hora de restaurar la base de datos a partir de una copia de seguridad. Por ejemplo, C:\Archivos de programa\EasyPHP-5.3.2i\FOTONICA\backup.

Tabla *config_mail*

Esta tabla está compuesta por 6 campos; Host, Port, Username, Password, From_dir y From_name. Al igual que la tabla anterior siempre va a almacenar un único registro. Ésta tiene la configuración necesaria para el envío de correos electrónicos desde la aplicación a partir de una dirección de destino determinada. Esta información es el servidor que se va a utilizar para el envío de correos, el puerto que utiliza, el nombre de usuario y la contraseña de esa cuenta, la dirección de correo desde la que se envían los emails y el nombre que se quiere que el destinatario vea al recibir el correo electrónico. Por ejemplo,

- Host: smtp.gmail.com
- Port: 587
- Username: upna.fotonica
- Password: labfotonica
- From_dir: upna.fotonica@gmail.com
- From_name: Administrador

Tabla *elementos*

Esta tabla está compuesta por 14 campos; Numero, Fecha_entrada, Nombre, Numero_serie, Categoria, Descripcion, Responsable, Localizacion_hoja_caracteristicas, Estado, Comentarios, Usuario, Uso, Fecha_inicio, Fecha_estimada_fin. Esta tabla es la

que almacena todos los componentes y equipos que hay en el laboratorio de Fotónica, con toda su información asociada. El campo Numero almacena el número del elemento, número correlativo que sirve para ordenar los elementos. Otro campo es la fecha de entrada, para saber cuándo se ha introducido cada elemento en la base de datos. Además hay que introducir el nombre y el número de serie de cada uno de ellos. En el campo Categoria se introduce uno de los registros de la tabla *categoria*. Hay otro campo en el que se hace una breve descripción. En el campo Responsable, se introduce el nombre de la persona responsable de ese elemento y en la localización de la hoja de características se almacenan valores obtenidos de la tabla *localizacion*. El campo Estado puede tener unos valores determinados, que están definidos en la tabla *estado* y definen el estado del elemento, si está bien, averiado… Se pueden escribir comentarios sobre cada elemento en el campo Comentarios y además, si un elemento está siendo usado por alguien tiene que quedar constancia de quién lo está usando (Usuario), qué uso le está dando (Uso) y durante qué periodo de tiempo lo va a utilizar (Fecha_inicio y Fecha_estimada_fin). Los campos Usuario y Uso están relacionados con las tablas *usuario* y *uso*. Un ejemplo de esta tabla es el siguiente registro:

- Numero: 100456
- Fecha_entrada: 10-06-2010
- Nombre: CIRCULADORES
- Numero_serie: 100303/910087/9
- Categoria: COMPONENTES FIBRA ÓPTICA
- Descripción: CIRCULADORES DE 3 Y 4 PUERTOS (2 unidades de cada uno), 1550nm
- Responsable: Montse Fernández Vallejo
- Localizacion_hoja_caracteristicas: Archivador
- Estado: O.K.
- Comentarios: Comprado a OEMARKET
- Usuario: Manuel Lopez-Amo
- Uso: INVESTIGACION
- Fecha_inicio: 10-06-2010
- Fecha_estimada_fin:

Tabla *categoia*

Esta tabla está compuesta de un único campo llamado Nombre, en el que se almacenan las posibles categorías que pueden asociarse a los elementos guardados en esta base de datos. Por ejemplo, *DIODO LASER*, *FILTROS OPTICOS*, *FUENTES DE LUZ*...

Tabla *localización*

Esta tabla está constituida por un campo llamado Localizacion. En él se almacenan las posibles localizaciones de las hojas de características y manuales de cada equipo o componente. Los registros almacenados en la actualidad son: Archivador, No disponible, Otros, Versión electrónica y Archivador/Versión electrónica.

Tabla *estado*

La tabla *estado* consta de un único campo (Estado) y almacena los posibles estados de los elementos almacenados en la base de datos. Por ejemplo, Averiado, O.K., K.O. u Otros.

Tabla *usuario*

Esta tabla también tiene un sólo campo y éste tiene el nombre de Nombre. En ella se almacenan los nombres de las personas que pueden estar utilizando alguno de los elementos del laboratorio. Por ejemplo, Manuel Lopez-Amo, Alayn o LIBRE.

Tabla *uso*

Esta tabla almacena, en un campo llamado Uso, los posibles usos que se le pueden dar a los elementos del laboratorio. Estos pueden ser P.F.C., INVESTIGACION, PRACTICAS o LIBRE.

Tabla *usuarios*

En esta tabla se almacena toda la información relacionada con los usuarios que tienen acceso a la aplicación. Se compone de 9 campos; Usuario, Password, Nombre, Apellidos, DNI, email, Categoria, Otros y Privilegios. El campo Usuario almacena el nombre del usuario, en Password la contraseña de ese usuario, en Nombre y Apellidos el nombre y los apellidos del usuario, en DNI su DNI, en email una dirección de correo electrónico para poder recibir correos de esta aplicación. En Categoria se muestra la categoría del usuario, uno de los registros almacenados en la tabla *categorias_usuarios*. En el campo Otros se puede añadir información por si la categoría seleccionada es Otros. Finalmente en Privilegios se almacena el nivel de privilegios que tiene cada usuario (Administrador, Intermedio o Usuario). Un ejemplo de registro almacenado en esta tabla es:

- Usuario: prueba1
- Password: prueba1
- Nombre: Prueba
- Apellidos: 1
- DNI: 12345678A
- Email: huarte.56181@e.unavarra.es
- Categoria: Otros
- Otros: prueba de otros
- Privilegios: Administrador

Tabla *privilegios_usuarios*

Esta tabla, con un único campo llamado Privilegios, almacena los tres posibles tipos de privilegios que puede tener un usuario. Estos son Administrador, Intermedio y Usuario.

Tabla *categorias_usuarios*

Esta tabla también está compuesta de un único campo llamado Categorias. En ella se almacenan las posibles categorías a las que puede pertenecer un usuario, como por ejemplo *Estudiante*, *PFC*, *Técnico laboratorio*, *Profesor* u *Otros*.

Explicado ya el formato y uso de cada tabla, se va a hacer una breve explicación del formato de la página web.

La aplicación tiene el acceso restringido y los usuarios que tienen acceso a ella lo van a tener más o menos restringido, dependiendo del tipo de privilegios de los que disponga cada uno. Hay usuarios con un nivel de privilegios muy bajo, los del tipo *usuario*, que tienen acceso a una parte muy pequeña de la aplicación. Hay otro tipo de usuarios que ya tienen acceso a una mayor parte de la aplicación, estos son los de tipo *intermedio*. Por último están los usuarios que tienen control absoluto sobre la aplicación, son los *administradores*.

Al acceder a la aplicación la primera página que se muestra es una en la que se pide al usuario que se identifique. Una vez se ha introducido un nombre de usuario y una contraseña, se comprueba si esta información es correcta, comprobando si existe ese usuario con esa contraseña en la base de datos. Dependiendo del tipo de privilegios que disponga el usuario se abre una página u otra. Independientemente del tipo de usuario la estructura de la página es siempre la misma, con la diferencia de que unos van a tener más enlaces activos que otros, ya que éstos les van a dirigir hacía las tareas que puedan llevar a cabo. Los usuarios con el nivel de privilegios menor van a poder siempre volver a la página de inicio (*menu_usuario.php*). Los intermedios, sin embargo van a tener siempre acceso a su página de inicio (*menu_intermedio.php*) y a la de configuración de su cuenta (*config_cuenta.php*). Por otro lado, los usuarios administradores siempre van a tener acceso a su página de inicio (*menu_administrador.php*), a la de configuración de su cuenta (*config_cuenta.php*) y a la de tareas del administrador (*tareas_administrativas.php*). De esta forma, cada usuario, independientemente de la página en la que se encuentre siempre, va a tener acceso a sus menús correspondiente.

A continuación se muestra una descripción de aquellas tareas que puede llevar a cabo cada usuario, dependiendo del nivel de privilegios que posea.

Usuario:

- Ver todos los elementos de la base de datos.
- Buscar elementos dentro de la base de datos.

Intermedio:

- Ver todos los elementos de la base de datos.
- Buscar elementos dentro de la base de datos.
- Editar los campos que definen cada elemento.
- Eliminar elementos de la base de datos.
- Almacenar nuevos elementos en la base de datos.

- Ver su perfil, donde se almacena la información relativa a su usuario: nombre, apellidos, DNI, dirección de correo electrónico asociada ese usuario, categoría (estudiante, PFC, técnico de laboratorio, profesor u otros) y nivel de privilegios (usuario, intermedio o administrador). En el caso de que la categoría sea *otros* se muestra, si existe, una descripción más detallada.
- Cambiar su contraseña de acceso.

Administrador:

- Ver todos los elementos de la base de datos.
- Buscar elementos dentro de la base de datos.
- Editar los campos que definen cada elemento.
- Eliminar elementos de la base de datos.
- Almacenar nuevos elementos en la base de datos.
- Ver su perfil, donde se almacena la información relativa a su usuario: nombre, apellidos, DNI, dirección de correo electrónico asociada ese usuario, categoría (estudiante, PFC, técnico de laboratorio, profesor u otros) y nivel de privilegios (usuario, intermedio o administrador). En el caso de que la categoría sea *otros* se muestra, si existe, una descripción más detallada.
- Cambiar su contraseña de acceso.
- Configurar la dirección de correo electrónico que se utiliza para enviar emails a los usuarios cuando son creados y eliminados.
- Cambiar el directorio en el que se hacen las copias de seguridad.
- Crear nuevos usuarios.
- Editar la información asociada a cada usuario.
- Eliminar usuarios.
- Crear copias de seguridad.
- Restaurar copias de seguridad.

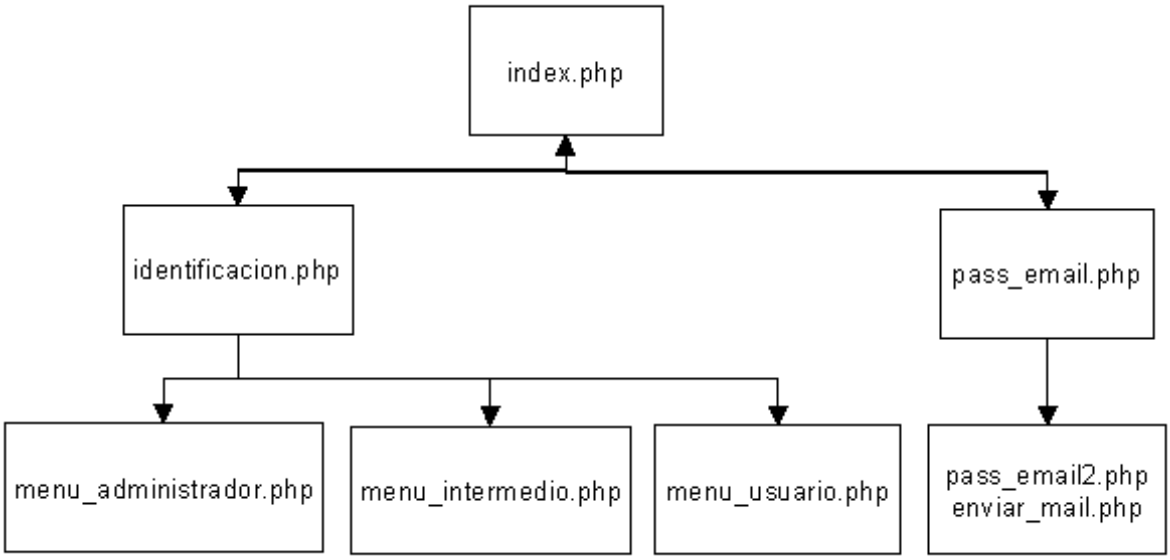
Además de todas estas tareas, si un usuario se olvida de su nombre de usuario o contraseña puede pedir a la aplicación que le envíe esta información por correo electrónico.

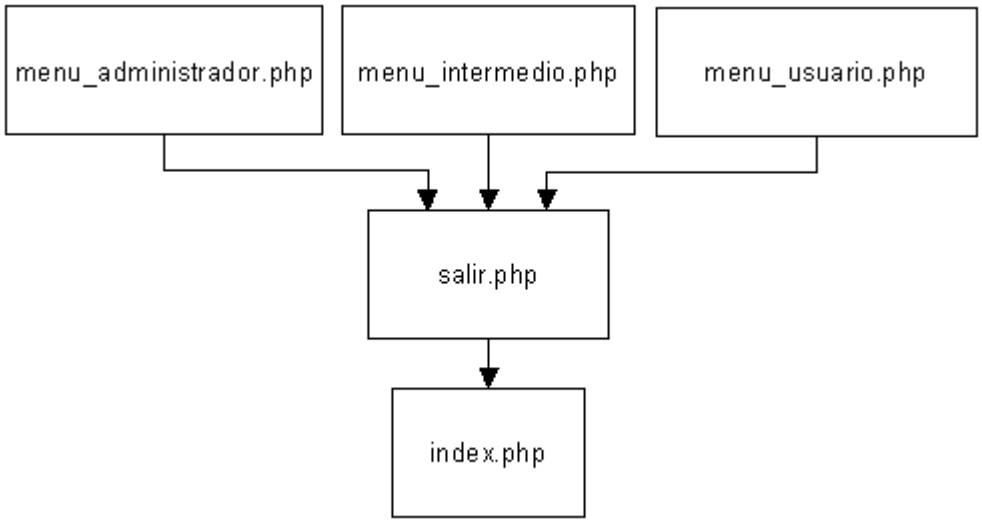
Todas las páginas están asociadas a una hoja de estilo (*estilo1.css*), excepto *index.php*, que está asociada a *index.css*. Estas hojas de estilo hacen que sea muy sencillo modificar la apariencia de la aplicación.

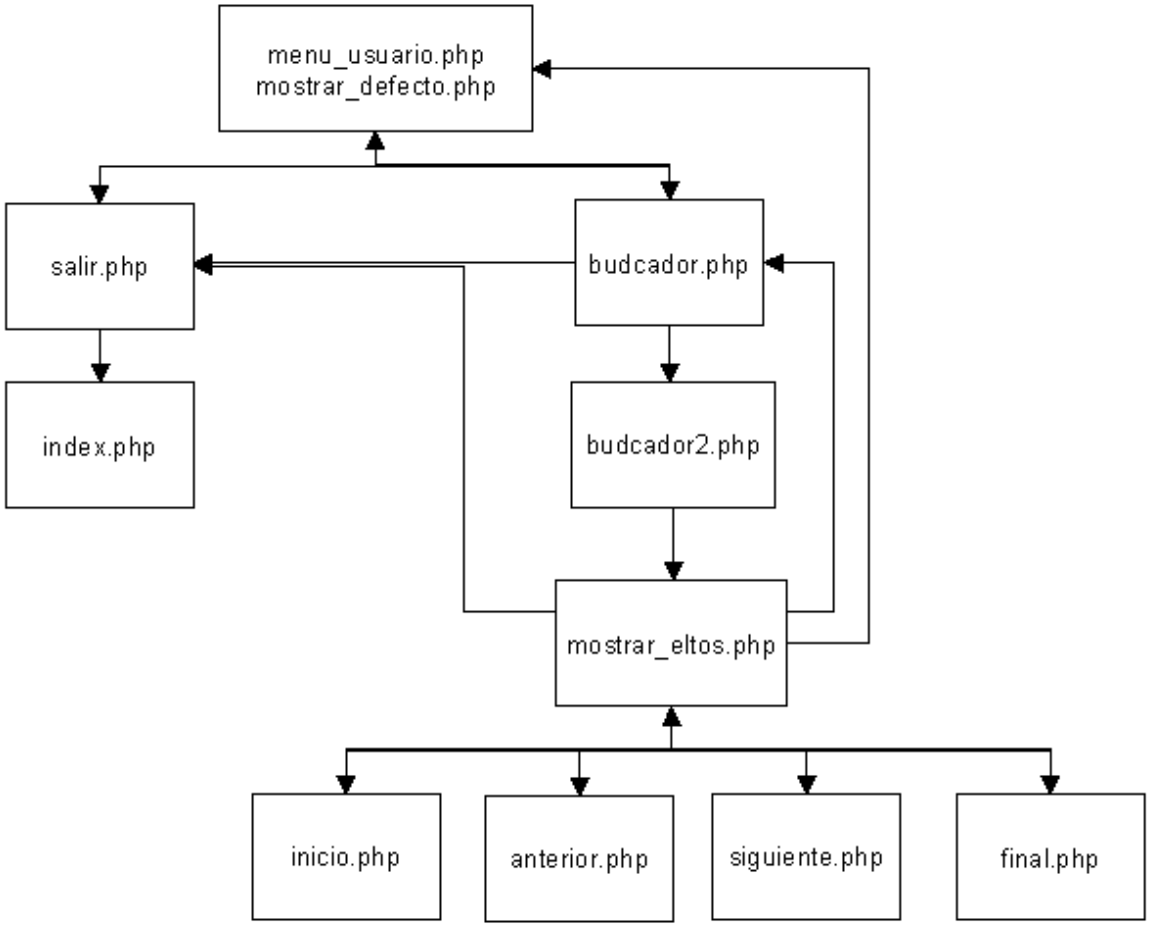
Todas estas tareas se han programado en un total de 65 scripts que forman la aplicación, programados en PHP, con un total de 5800 líneas de código.

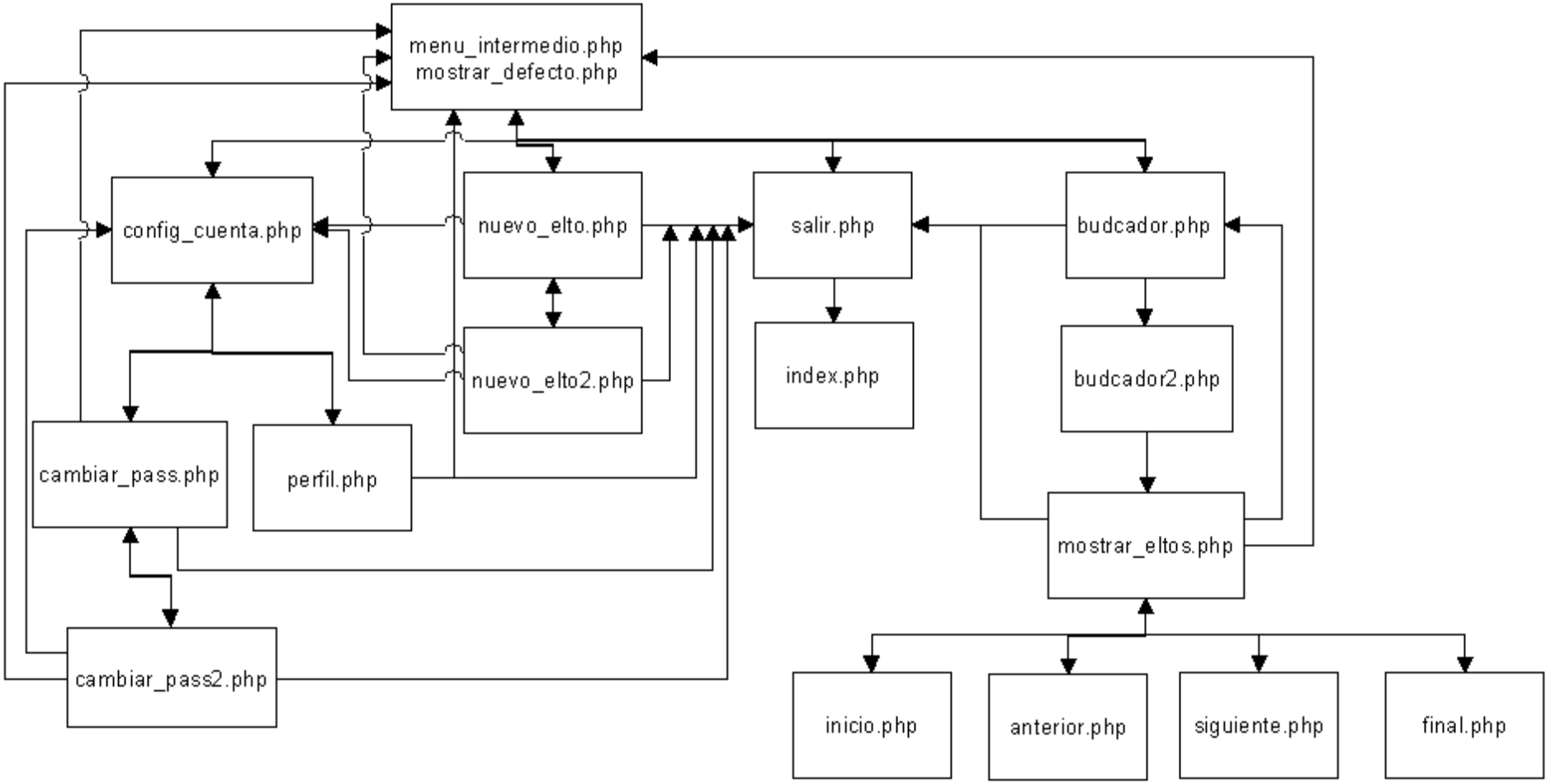
1.4. Diagramas

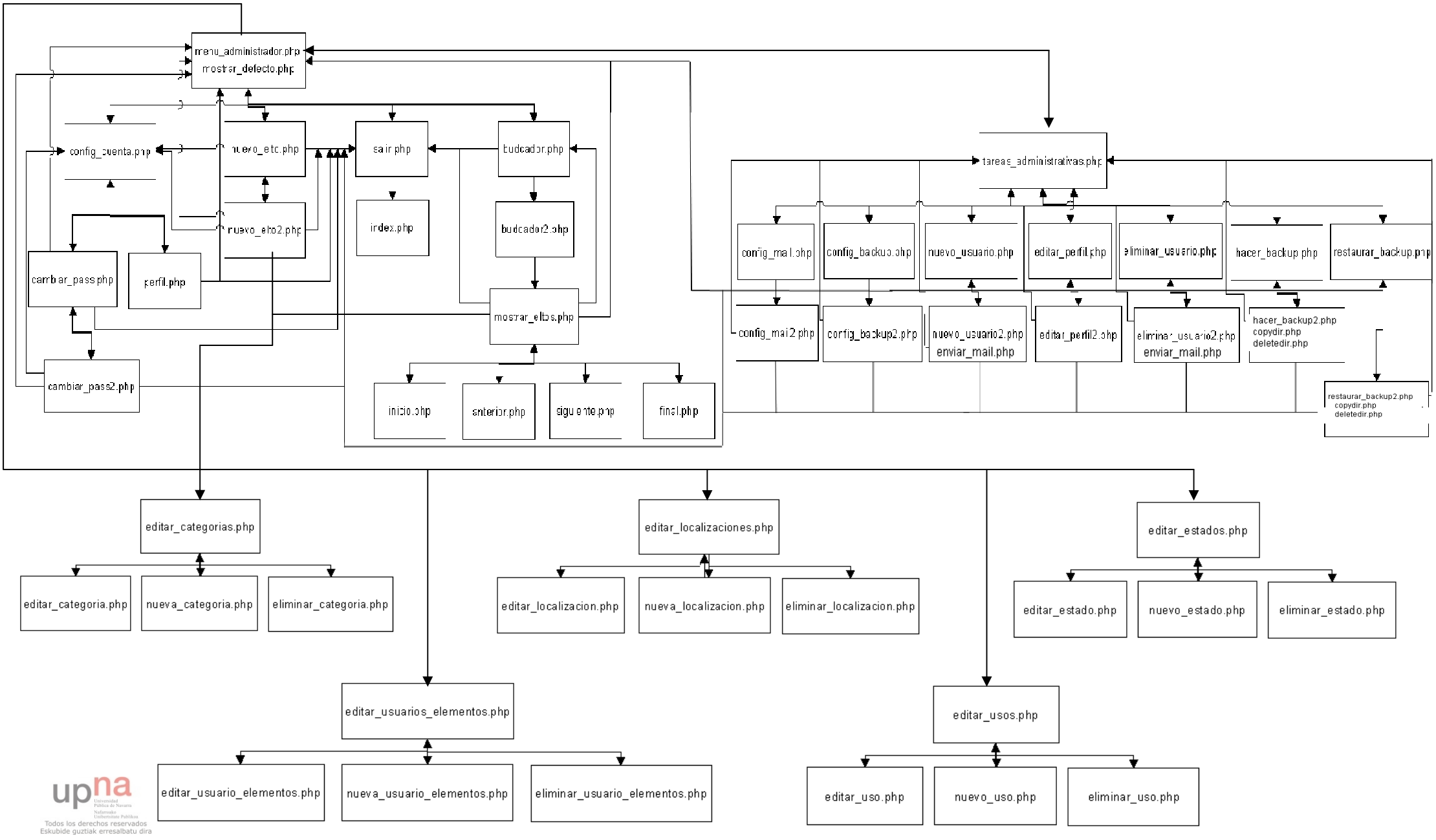
A continuación se van a mostrar unos diagramas que indican la relación entre los scripts.











Cabe destacar que desde cualquier página el usuario puede volver a la página de inicio así como cerrar sesión y salir de la aplicación. Los usuarios *intermedios*, además podrán en todo momento ir a *config_cuenta.php*, donde van a tener acceso a diferentes opciones para configurar su cuenta. Por otro lado, los usuarios *administradores*, además de poder acceder en todo momento a *config_cuenta.php*, podrán también acceder a *tareas_administrativas.php*, donde tendrán acceso a todas las tareas permitidas por la aplicación.

Todos los scripts utilizados en esta aplicación tienen una estructura similar. Al principio, se inicia una sesión y se incluyen, si es necesario, otros script cuyas variables se van a utilizar en el script. El script que se incluye en casi todos es *config.*php, ya que es el que dispone de la información necesaria para conectarse a la base de datos y para redirigir la página automáticamente a otras páginas. Después, se comprueba que se ha accedido a la página de forma correcta, es decir, un usuario se ha identificado y tiene los privilegios necesarios para acceder. Si no es así la página se redirige a la página principal (*index.php*), para que el usuario se identifique correctamente. Si el usuario ha accedido correctamente, la siguiente comprobación es calcular el tiempo que el usuario lleva inactivo. Si el usuario lleva inactivo 10 minutos o más se destruye la sesión y se redirige a la página principal para que se vuelva a identificar. De este modo, si un usuario no cierra sesión, ni cierra el navegador la sesión caducará en 10 minutos. Si todavía no han pasado 10 minutos se modifica el valor de la fecha en el que ha tenido lugar el último acceso. Inmediatamente cada script tiene las sentencias necesarias para la realización de su tarea principal.

A continuación se expone la descripción de cada script, explicando qué se hace en cada uno de ellos.

config.php

Este script tiene la información necesaria para que cuando algún script se tenga que conectar con la base de datos MySQL esta conexión se pueda llevar a cabo. Esta información es el servidor (localhost), el usuario (root) y la contraseña (upna) de la base de datos. Además tiene otra variable que almacena la dirección del servidor y el directorio en el que están almacenados todos los scripts (172.18.67.26/FOTONICA).

index.php

Esta es la página a la que se accede al entrar en la aplicación. Lo primero que se hace en este script es poner a cero la variable de sesión que almacena el nivel de privilegios del usuario identificado, ya que en este momento todavía ningún usuario se ha identificado. En ella hay un formulario en el que se solicita un usuario y una contraseña. Cuando estos datos se introducen esta información se envía a *identificacion.php*. Además, en esta página hay un enlace al que se accede si el usuario ha olvidado su usuario y/o contraseña. Al pinchar sobre él la página va a *pass_email.php*.

pass_email.php

En esta página hay otro formulario en el que se solicita una dirección de correo electrónico. Una vez introducida esta dirección la información se envía al archivo *pass_email2.php*.

pass_email2.php

En este script se utiliza una función llamada *enviar_mail* y por eso se incluye el archivo que la almacena (*enviar_mail.php*) al inicio del mismo. Lo primero que se hace en este script es comprobar si algún usuario tiene esa dirección de correo electrónico. Si existe, en dos variables se almacenan el nombre de ese usuario y su contraseña y se utilizan para formar el cuerpo de un correo electrónico. Además se especifica el titulo del correo. Seguidamente se llama a la función *enviar_mail*, pasándole como parámetros la dirección de correo electrónico del destinatario, el título del mensaje y el cuerpo del mismo. Si ningún usuario tiene asociada esa dirección de correo electrónico sale un mensaje en la página indicando que la dirección introducida es errónea y un enlace para volver a *pass_email.php* y volver a introducir la dirección. Si la dirección existe se envía un mensaje con el nombre de usuario y su contraseña.

enviar_mail.php

En este script se almacena la función *enviar_mail*. Ésta va a ser la encargada de enviar emails a las direcciones de correo, con los títulos y cuerpos de los mensajes que se le pasen como parámetros de entrada. Para ello se hace uso de la clase *PHPMailer*, así que al principio de la función se incluye el archivo *clas.phpmailer.php*. Lo primero que se hace es obtener el nombre y los apellidos del usuario al que se le va a enviar el correo electrónico. Después, se obtienen los datos de configuración de la cuenta de correo desde la que se va a hacer el envío, datos que están almacenados en la tabla *config_mail* de la base de datos. Estos datos son el servidor, puerto, nombre de usuario, contraseña, dirección y nombre de la persona titular de la cuenta. Con toda esta información se crea un objeto de la clase *PHPMailer* y se envía un correo electrónico con la información que se le ha especificado en los parámetros de entrada de la función.

identificación.php

Lo primero que se hace en este script es iniciar una sesión y dar valores a las variables de sesión que van a almacenar el nivel de privilegios del usuario identificado y la fecha en la que ha tenido lugar el acceso. La variable de sesión que almacena el nivel de privilegios se inicializa a 0. Una vez hecho esto comprueba si el usuario existe y comprueba el nivel de privilegios que posee. Con esta información da valor a la variable de sesión que va a almacenar el nombre de usuario que está identificado y dependiendo del nivel de privilegios que éste posea asignará un valor u otro a la variable de sesión que va a almacenar esta información. Si es *administrador*, a la variable se le asigna valor 1, si es *Intermedio*, se le asigna 2 y si es *Usuario* se le asigna 3. Además dependiendo de esto se redirige la página a una u otra, ya que dependiendo de los privilegios que tenga cada usuario tendrá acceso a unas acciones o a otras.Las páginas a las que se puede redirigir son *menú_administrador.php* si se ha identificado como un *Administrador*, *menú_intermedio.php* si se ha identificado como *Intermedio* y

menú_usuario.php si es un U*suario*. Si por el contrario, no existe ningún usuario con ese nombre y contraseña se redirige a *index.php*.

menu_usuario.php

Este script es sólo accesible por aquellos usuarios de tipo *Usuario*. En él puede acceder al buscador programado para esta aplicación (*buscador.php*) o puede ver directamente uno a uno todos los elementos almacenados en la base de datos. Esto es posible debido a que se ha incluido un script (*mostrar_defecto.php*) que muestra uno a uno todos estos elementos.

buscador.php

Este script muestra un formulario, en el que están todos los campos que definen cada elemento de la base de datos. De esta manera el usuario puede buscar un elemento a través de cualquiera de sus campos, puede buscar desde por su nombre, como por la categoría a la que pertenece, hasta por la fecha en la que se añadió a la base de datos. Además si el usuario que ha accedido a la página es un *Administrador*, puede editar los posibles valores de los campos *Categoria* (*editar_categorias.php*), *Localizacion_hoja_caracteristicas* (*editar_localizacione.php*), *Estado* (*editar_estados.php*), *Usuario* (*editar_usuarios_elementos.php*) y *Uso* (*editar_usos.php*), modificando los registros de las tablas correspondientes desde las que se obtienen los valores. Una vez introducidos en el formulario los datos en los campos por los que se quiere buscar esta información es enviada a *buscador2.php*.

buscador2.php

En él se almacena toda la información introducida en el formulario en diferentes variables y se busca por separado cada elemento que tenga alguna coincidencia con alguno de los campos buscados. Después todos los elementos que han tenido alguna coincidencia se introducen en un vector y si existen elementos los cuales todos los campos buscados tienen el valor buscado se almacenan en otro vector, en orden creciente de sus números. Entonces, en este último vector quedan almacenados los números de aquellos elementos que cumplen las condiciones de la búsqueda. Este vector se almacena como variable de sesión, se inicializa el puntero que va a pasar de un elemento a otro dentro de este vector y se redirige la página a *mostrar_eltos.php*.

mostrar_eltos.php

Este archivo va a mostrar de uno en uno todos los elementos que se han encontrado en la búsqueda, con todos los campos que los caracterizan. Lo primero es añadir el archivo *nombre_archivo.php*. Éste almacena la función *nombre_archivo*, que va a ser la encargada de devolver, si existe, el nombre del material electrónico asociado del elemento que se esté mostrando. A una variable se le asigna el número total de elementos encontrados. Si no se han encontrado elementos se muestra un mensaje diciendo que no se encontraron coincidencias. Si se ha encontrado algún elemento hay un menú de navegación para que según lo que seleccione el usuario, se muestre el elemento siguiente (*siguiente.php*), el anterior (*anterior.php*), el primero (*inicio.php*) o el último (*final.php*) de los elementos encontrados, siempre ordenados según su número.

A continuación se muestra toda la información (los 14 campos) del elemento seleccionado. Además si el usuario que ha accedido a esta página se ha identificado como *Administrador* puede editar los posibles valores de los campos *Categoria* (*editar_categorias.php*), *Localizacion_hoja_caracteristicas* (*editar_localizacione.php*), *Estado* (*editar_estados.php*), *Usuario* (*editar_usuarios_elementos.php*) y *Uso* (*editar_usos.php*), modificando los registros de las tablas correspondientes desde las que se obtienen los valores.

Además se comprueba si existe material electrónico asociado a ese elemento con la función *nombre_archivo*, pasándole como parámetro el nombre de este archivo (elto_xxx, donde xxx es el número del elemento), sin extensión, ya que ésta no se sabe de antemano. Si existe se muestra un enlace para que el usuario pueda verlo. Si el usuario que ha accedido se ha identificado como *Administrador* o *Intermedio* además puede editar los valores de cada elemento, así como subir material electrónico asociado nuevo (*editar_elto.php*) y eliminar el elemento (*eliminar_elto.php*).

mostrar_defecto.php

Este archivo va a mostrar de uno en uno todos los elementos almacenados en la base de datos, con todos los campos que los caracterizan. Lo primero es añadir el archivo *nombre_archivo.php*. Éste almacena la función *nombre_archivo*, que va a ser la encargada de devolver, si existe, el nombre del material electrónico asociado del elemento que se esté mostrando. Seguidamente se introduce en un vector todos los números de todos los elementos que hay en la base de datos, a continuación se ordena este vector de menor a mayor. A una variable de sesión se le asigna el número total de elementos. A continuación hay un menú de navegación para que según lo que seleccione el usuario se muestre el elemento siguiente (*siguiente.php*), el anterior (*anterior.php*), el primero (*inicio.php*) o el último (*final.php*) de la base de datos, siempre ordenados según su número. Seguidamente se muestra toda la información (los 14 campos) del elemento seleccionado. Además si el usuario que ha accedido a esta página se ha identificado como *Administrador* puede editar los posibles valores de los campos *Categoria* (*editar_categorias.php*), *Localizacion_hoja_caracteristicas* (*editar_localizacione.php*), *Estado* (*editar_estados.php*), *Usuario* (*editar_usuarios_elementos.php*) y *Uso* (*editar_usos.php*), modificando los registros de las tablas correspondientes desde las que se obtienen los valores. Además se comprueba si existe material electrónico asociado a ese elemento con la función *nombre_archivo*, pasándole como parámetro el nombre de este archivo (elto_xxx, donde xxx es el número del elemento). Si existe se muestra un enlace para que el usuario pueda verlo. Si el usuario que ha accedido se ha identificado como *Administrador* o *Intermedio* además puede editar los valores de cada elemento, así como subir material electrónico asociado nuevo (*editar_elto.php*) y eliminar el elemento (eliminar_elto.php).

nombre_archivo.php

En este archivo se almacena la función *nombre_archivo*. Ésta es la encargada de devolver el nombre del material electrónico asociado al elemento actual que se está mostrando, almacenado en el directorio ./manuales/. Para ello guarda en un vector todos los nombres de los archivos que hay en ese directorio. Busca la máxima similitud entre

el nombre que debería tener el archivo asociado y cada uno de los existentes en el directorio, con el fin de encontrar la extensión del archivo. Almacena la extensión del archivo con mayor similitud en una variable. A continuación compara el nombre que debe tener el archivo, ya que es sabido debido a que todos los nombres de los archivos tienen la forma elto_xxx, donde xxx es el número del elemento, con el nombre que ha obtenido la máxima similitud con él (sin la extensión). Si éstos son iguales es que ese elemento tiene un archivo asociado, y devuelve el nombre de éste (incluida la extensión). Si no son iguales es que ese elemento no tiene ningún archivo asociado y devuelve cadena vacía.

siguiente.php

Éste lo que hace es incrementar el puntero en una unidad, siempre y cuando éste no indique ya la última posición del vector donde están almacenados los elementos que se van a mostrar, en este caso el puntero no cambia de valor. Cuando el puntero ya tiene el valor que le corresponde la página se redirige a la página de la que viene.

anterior.php

Éste lo que hace es decrementar el puntero en una unidad, siempre y cuando éste no indique ya la primera posición del vector donde están almacenados los elementos que se van a mostrar, en este caso el puntero no cambia de valor. Cuando el puntero ya tiene el valor que le corresponde la página se redirige a la página de la que viene.

inicio.php

Éste lo que hace es asignarle al puntero el valor 0, para que indique la primera posición del vector donde están almacenados los elementos que se van a mostrar. Después la página se redirige a la página de la que viene.

final.php

Éste lo que hace es asignarle al puntero el valor de la última posición del vector donde están almacenados los elementos que se van a mostrar. Después la página se redirige a la página de la que viene.

editar_categorias.php

Éste se abre en una ventana nueva, ya que el usuario quiere editar la lista de categorías, pero puede que no quiera salir de la página donde se encuentra. En él se muestran una a una todas las categorías almacenadas en la tabla *categoria* y a través de dos formularios se da la opción de poder cambiar el nombre de estas categorías (*editar_categoria.php*) o eliminar categorías de la base de datos (*eliminar_categoria.php*). Además existe un nuevo formulario para añadir nuevas categorías a la tabla (*nueva_categoria.php*).

editar_categoria.php

Lo que se hace en este script es recoger el nombre de la categoría a la que se le quiere cambiar el nombre y el nombre nuevo que ha introducido el usuario. De esta forma, se actualiza el nombre de esa categoría en la tabla *categoria* de la base de datos. Después redirige al usuario a la página *editar_categorias.php*.

eliminar_categoria.php

Es el encargado de recoger el nombre de la categoría que se quiere eliminar y la elimina de la tabla. Después redirige al usuario a la página *editar_categorias.php*.

nueva_categoria.php

En este script se recoge el nombre de la categoría que se quiere añadir y se comprueba si ya existe en la tabla *categoria* una con ese nombre, si no es así la añade como un nuevo registro y redirige a la página a *editar_categorias.php*. Si la categoría ya existe muestra un aviso y un enlace para volver a *editar_categorias.php*.

editar_localizaciones.php

Éste se abre en una ventana nueva, ya que el usuario quiere editar la lista de localizaciones donde se pueden encontrar los manuales y hojas de características de los componentes y equipos del laboratorio, pero puede que no quiera salir de la página donde se encuentra. En él se muestran una a una todas las localizaciones almacenadas en la tabla *localizacion* y a través de dos formularios se da la opción de poder cambiar el nombre de estas localizaciones (*editar_localizacion.php*) o eliminar localizaciones de la base de datos (*eliminar_localizacion.php*). Además existe un nuevo formulario para añadir nuevas localizaciones a la tabla (*nueva_localizacion.php*).

editar_localizacion.php

Lo que se hace en este script es recoger el nombre de la localización a la que se le quiere cambiar el nombre y el nombre nuevo que ha introducido el usuario. De esta forma, se actualiza el nombre de esa localización en la tabla *localizacion* de la base de datos. Después redirige al usuario a la página *editar_localizaciones.php*.

eliminar_localizacion.php

Es el encargado de recoger el nombre de la localización que se quiere eliminar y la elimina de la tabla. Después redirige al usuario a la página *editar_localizaciones.php*.

nueva_localizacion.php

En este script se recoge el nombre de la localización que se quiere añadir y se comprueba si ya existe en la tabla *localizacion* una con ese nombre, si no es así la añade como un nuevo registro y redirige a la página a *editar_localizaciones.php*. Si la localización ya existe muestra un aviso y un enlace para volver a *editar_localizaciones.php*.

editar_estados.php

Éste se abre en una ventana nueva, ya que el usuario quiere editar la lista de posibles estados de los elementos de la base de datos, pero puede que no quiera salir de la página donde se encuentra. En él se muestran uno a uno todos los estados posibles de cada elemento almacenados en la tabla *estado* y a través de dos formularios se da la opción de poder cambiar el nombre de estos estados (*editar_estado.php*) o eliminar estados de la base de datos (*eliminar_estado.php*). Además existe un nuevo formulario para añadir nuevos estados a la tabla (*nuevo_estado.php*).

editar_estado.php

Lo que se hace en este script es recoger el nombre del estado al que se le quiere cambiar el nombre y el nombre nuevo que ha introducido el usuario. De esta forma, se actualiza el nombre de ese estado en la tabla *estado* de la base de datos. Después redirige al usuario a la página *editar_estados.php*.

eliminar_estado.php

Es el encargado de recoger el nombre del estado que se quiere eliminar y lo elimina de la tabla. Después redirige al usuario a la página *editar_estados.php*.

nuevo_estado.php

En este script se recoge el nombre del estado que se quiere añadir y se comprueba si ya existe en la tabla *estado* uno con ese nombre, si no es así lo añade como un nuevo registro y redirige a la página a *editar_estados.php*. Si el estado ya existe muestra un aviso y un enlace para volver a *editar_estados.php*.

editar_usuarios_elementos.php

Éste se abre en una ventana nueva, ya que el usuario quiere editar la lista de usuarios que tienen acceso a los elementos del laboratorio, pero puede que no quiera salir de la página donde se encuentra. En él se muestran uno a uno todos los usuarios almacenados en la tabla *usuario* y a través de dos formularios se da la opción de poder cambiar el nombre de estos usuarios (*editar_usuario_elementos.php*) o eliminar usuarios de la base de datos (*eliminar_ usuario_elementos.php*). Además existe un nuevo formulario para añadir nuevos usuarios a la tabla (*nuevo_usuario_elementos.php*).

editar_usuario_elementos.php

Lo que se hace en este script es recoger el nombre del usuario al que se le quiere cambiar el nombre y el nombre nuevo que se ha introducido. De esta forma, se actualiza el nombre de ese usuario en la tabla *usuario* de la base de datos. Después redirige al usuario a la página *editar_usuarios_elementos.php*.

eliminar_usuario_elementos.php

Es el encargado de recoger el nombre del usuario que se quiere eliminar y lo elimina de la tabla. Después redirige al usuario a la página *editar_usuarios_elementos.php*.

nuevo_usuario_elementos.php

En este script se recoge el nombre del usuario que se quiere añadir y se comprueba si ya existe en la tabla *usuario* uno con ese nombre, si no es así lo añade como un nuevo registro y redirige a la página a *editar_usuarios_elementos.php*. Si el usuario ya existe muestra un aviso y un enlace para volver a *editar_usuarios_elementos.php*.

editar_usos.php

Éste se abre en una ventana nueva, ya que el usuario quiere editar la lista de usos que se le pueden dar a los elementos, pero puede que no quiera salir de la página donde se encuentra. En él se muestran uno a uno todos los usos posibles que se le pueden dar a cada elemento y a través de dos formularios se da la opción de poder cambiar el nombre de estos usos (*editar_uso.php*) o eliminar usos de la base de datos (*eliminar_uso.php*). Además existe un nuevo formulario para añadir nuevos usos a la tabla *uso* (*nuevo_uso.php*).

editar_uso.php

Lo que se hace en este script es recoger el nombre del uso al que se le quiere cambiar el nombre y el nombre nuevo que ha introducido el usuario. De esta forma, se actualiza el nombre de ese uso en la tabla *uso* de la base de datos. Después redirige al usuario a la página *editar_usos.php*.

eliminar_uso.php

Es el encargado de recoger el nombre del uso que se quiere eliminar y lo elimina de la tabla. Después redirige al usuario a la página *editar_usos.php*.

nuevo_uso.php

En este script se recoge el nombre del uso que se quiere añadir y se comprueba si ya existe en la tabla *uso* uno con ese nombre, si no es así lo añade como un nuevo registro y redirige a la página a *editar_usos.php*. Si el uso ya existe muestra un aviso y un enlace para volver a *editar_usos.php*.

editar_elto.php

Este script, en el inicio incluye el archivo *nombre_archivo.php*. En él se recogen todos los datos asociados al elemento que se está mostrando, así como el nombre del material electrónico asociado a él, si existe. Primero se comprueba si se ha subido algún archivo nuevo. Si es así le cambia el nombre a elto_xxx, donde xxx es el número del elemento. A continuación con la función *nombre_archivo* comprueba si ese elemento ya

tenía un archivo asociado y si es así lo elimina. Seguidamente guarda el archivo nuevo. A continuación actualiza todos los campos del elemento que el usuario quiere modificar.

eliminar_elto.php

Este script, en el inicio incluye el archivo *nombre_archivo.php*. En éste se recoge el número del elemento que se quiere eliminar. Con la función *nombre_archivo* busca si existe algún archivo asociado a ese elemento y si es así lo elimina. Seguidamente elimina el elemento de la base de datos.

menu_intermedio.php

Este script se diferencia de *menu_usuario.php* en que además de mostrar lo que se muestra en este último se pueden añadir nuevos componentes y equipos a la base de datos (*nuevo_elto.php*). Además estos usuarios pueden ver la información relacionada con su perfil (*perfil.php*) y cambiar su contraseña de acceso (cambiar_pass.php).

nuevo_elto.php

Comienza haciendo una consulta a la tabla *elementos* de la base de datos para saber cuál es el número mayor que se ha utilizado en un elemento. Después le suma uno a este número, obteniendo el número que se le va a asignar al nuevo elemento, numero que el usuario no va a poder modificar. Seguidamente se calcula la fecha actual, para asignársela al campo *Fecha_entrada* de la tabla *elementos*, campo que tampoco va a poder ser modificado por el usuario. A continuación se muestra el formulario, con el resto de campos vacíos, para que el usuario rellene la información asociada al elemento que quiere añadir a la base de datos. Además, los campos *Categoría*, *Localizacion_hoja_caracteristicas*, *Estado*, *Usuario* y *Uso* son listas desplegables con los posibles valores que pueden tomar, obtenidos de sus tablas correspondientes. Además si el usuario se ha identificado como *Administrador* podrá modificar los registros de estas tablas en *editar_categorias.php*, *editar_localizaciones.php*, *editar_estados.php*, *editar_usuarios_elementos.php* y *editar_usos.php*. Finalmente también hay un campo del formulario desde el que se pueden subir archivos para asociarlos al elemento nuevo. Toda la información introducida en este formulario se envía a *nuevo_elto2.php*.

nuevo_elto2.php

En este script se recogen todos los datos introducidos en el formulario anterior y se cambia el nombre del archivo subido a la forma elto_xxx, donde xxx es el número del elemento. Seguidamente se comprueba si el elemento que se quiere introducir en la base de datos ya existe, comprobando la existencia de algún elemento con ese nombre o ese número de serie. Si el elemento ya existe se muestra un aviso y un enlace para poder volver a introducir un nuevo elemento. Si el elemento no existe se crea un nuevo registro en la base de datos y se guarda el archivo asociado en la carpeta que le corresponde (./manuales/).

perfil.php

Este script lo que hace es mostrar información del usuario actual. Muestra su nombre, apellidos, DNI, dirección de correo electrónico, categoría y su nivel privilegios. Si la categoría a la que pertenece es *Otros*, muestra una descripción de ésta.

cambiar_pass.php

Este script muestra al usuario un formulario con dos campos, y en ambos se tiene que introducir la contraseña nueva con la que el usuario quiere tener acceso a la aplicación. La información es enviada a *cambiar_pass2.php*.

cambiar_pass2.php

Aquí se recoge la información introducida en el formulario y se comprueba que en ambos campos se ha introducido la misma contraseña y que ésta no es una cadena vacía. Si esto es así se actualiza el valor del campo *Password* de la tabla *Usuarios*. Si no se cumple aparece un mensaje avisando al usuario y un enlace para acceder a la página anterior, donde pueda volver a introducir una contraseña correcta.

config_cuenta.php

Tanto el acceso al enlace para ver el perfil de usuario como para cambiar la contraseña de acceso se muestran en este script. Script que tiene como única finalidad mostrar estos enlaces (*perfil.php* y *cambiar_pass.php*).

menu_administrador.php

Este script se diferencia de *menu_intermedio.php* en que además de mostrar lo que se muestra en este último se puede configurar el directorio donde almacenar y desde el que restaurar copias de seguridad (*config_backup.php*), configurar la cuenta desde la que la aplicación va a enviar emails (*config_mail.php*). Además estos usuarios van a poder crear nuevos usuarios desde *nuevo_usuario.php*, editar sus perfiles (*editar_perfil.php*) y eliminar usuarios (*eliminar_usuario.php*). También van a poder hacer y restaurar copias de seguridad (*hacer_backup.php* y *restaurar_backup.php*). A todas estas nuevas tareas se va a tener acceso desde *tareas_administrativas.php*.

tareas_administrativas.php

Este script tiene como única finalidad mostrar enlaces para acceder a diferentes tareas a las que únicamente los usuarios administradores tienen acceso:

- Configurar la cuenta de correo desde la que la aplicación envía emails (*config_mail.php*).
- Configurar el directorio en el que se hacen y desde el que se restauran las copias de seguridad (*config_backup.php*).
- Crear nuevos usuarios (*nuevo_usuario.php*).
- Editar los perfiles de los usuarios (*editar_perfil.php*).

- Eliminar usuarios para que dejen de tener acceso a la base de datos (*eliminar_usuario.php*).
- Crear copias de seguridad (*hacer_backup.php*).
- Restaurar copias de seguridad (*restaurar_backup.php*).

config_mail.php

Este script muestra en pantalla un formulario con las opciones necesarias para configurar el envío de un email automáticamente con php con la clase PHPMailer. El formulario se muestra con la información que está almacenada en la base de datos en ese momento, pero se pueden modificar todos y cada uno de los campos. Estos campos son:

- Host
- Puerto
- Username
- Password
- Dirección de correo electrónico
- Nombre

Cuando la información ha sido modificada ésta es enviada al script *config_mail2.php*.

config_mail2.php

Este script es el encargado de modificar los valores de la tabla *config_mail* con los datos introducidos en el formulario anterior.

config_backup.php

Este script es muy parecido a *config_mail.php*. También está formado por un formulario, pero en éste se muestra el directorio donde se guardan las copias de seguridad. Se puede modificar este campo, que por defecto muestra el directorio actual donde se almacenan las copias. La información es enviada a *config_backup2.php*.

config_backup2.php

Éste recoge el dato introducido en el formulario y actualiza la tabla *config_backup* con este valor.

nuevo_usuario.php

Este script muestra un formulario en el que el administrador va a introducir los datos necesarios para crear un nuevo usuario. Los campos que puede rellenar son:

- Usuario
- Contraseña (se escribe dos veces)
- Nombre
- Apellidos

- DNI
- Dirección de correo electrónico
- Categoría (lista desplegable a partir de los registros almacenados en la tabla *categorias_usuarios*)
- Otros (campos que se utiliza para especificar la categoría si en este campo se ha seleccionado la opción *Otros*)
- Privilegios (lista desplegable a partir de los registros almacenados en la tabla *privilegios_usuarios*)

Esta información es enviada a *nuevo_usuario2.php*.

nuevo_usuario2.php

Este script incluye al inicio el archivo *enviar_mail.php*. Lo primero que se hace es comprobar que el usuario introducido no es una cadena vacía. Si lo es se avisa de que no se puede crear un usuario cuyo nombre de usuario es una cadena vacía y se habilita un enlace para volver al formulario para crear un nuevo usuario. Si el usuario no es una cadena vacía se comprueba que la contraseña introducida ha sido la misma las dos veces que se ha escrito, para evitar equivocaciones por parte del usuario. Si se cumple esta condición se comprueba que no exista ningún usuario con ese nombre, en el caso de que ya existiera aparece un aviso. También se comprueba si ya existe algún usuario con esa dirección de correo electrónico, en caso afirmativo se muestra otro aviso. Si por el contrario el usuario y la dirección de correo electrónico no existen se crea el nuevo usuario. Si se ha seleccionado *Otros* en el campo *Categoria* se almacena el valor del campo *Otros*, si no, aunque el administrador haya escrito algo en este campo esta información no será introducida en la base de datos. Finalmente, utilizando la función *enviar_mail* se envía un correo electrónico a la dirección asociada al nuevo usuario, indicándole a éste que su cuenta ha sido creada.

editar_perfil.php

Este script está formado por dos formularios, el primero es una lista desplegable en la que se muestran todos los usuarios que tienen acceso a la aplicación, excepto el usuario actual, ya que no se puede modificar su perfil por ser el usuario activo. Al elegir un usuario de esta lista se carga la información del segundo formulario, vacío hasta el momento. Ésta es el nombre, apellidos, DNI, dirección de correo electrónico, categoría y privilegios que tiene el usuario seleccionado. El administrador puede cambiar esta información, enviando los nuevos datos a *editar_perfil2.php*.

editar_perfil2.php

En éste lo primero que se hace es comprobar que se ha seleccionado algún usuario, si no es así aparece un mensaje diciendo que se debe seleccionar un usuario y muestra un enlace para volver a la página anterior. Si se ha seleccionado un usuario se actualiza la información relativa a ese usuario. El campo *Otros* será modificado únicamente si la nueva categoría elegida ha sido *Otros* o si la categoría antigua era *Otros* y se ha cambiado, en este caso el campo *Otros* se pone en blanco.

eliminar_usuario.php

Este script muestra una tabla con información relativa a todos los usuarios que tienen acceso a la aplicación. Después muestra un formulario en el que se solicita el usuario que se desea eliminar. El nombre de este usuario es enviado a *eliminar_usuario2.php*.

eliminar_usuario2.php

En este script se recoge el usuario que se desea eliminar y se comprueba que no es el usuario activo. Si lo es muestra un aviso de que no se puede eliminar un usuario activo. Si no lo es comprueba que el usuario existe; si no existe muestra un aviso indicando que el usuario introducido no existe, si existe lo elimina de la base de datos y con la función *enviar_mail* le envía un correo electrónico indicándole que su usuario ha sido eliminado de la base de datos y ya no tiene acceso a la aplicación.

hacer_backup.php

Este script está formado por un formulario en el que se solicita que se introduzca el nombre del archivo que va a almacenar la copia de seguridad y el nombre de la carpeta que va a almacenar el material electrónico asociado a los componentes y equipos de la base de datos. Esta información es enviada a *hacer_backup2.php*.

hacer_backup2.php

Este script incluye el archivo *copydir.php*. Lo primero que se hace en este script es añadir al nombre introducido por el usuario para el archivo que va a almacenar la copia de seguridad la extensión *sql*. Después comprueba que tanto el nombre introducido para el archivo como el de la carpeta no sean cadenas vacías, si lo son redirige a la página anterior para que el administrador introduzca un nombre para el archivo y otro para la carpeta. Seguidamente comprueba que el directorio donde se va a almacenar la copia de seguridad existe y si no es así lo crea. Si se ha dado nombre al archivo y a la carpeta hace una copia de seguridad de toda la base de datos y la almacena en el archivo **.sql*, en el directorio almacenado en la tabla *config_backup*. Además llama a la función *copydir* para que haga una copia de los archivos asociados a los elementos en la carpeta cuyo nombre se ha introducido en el formulario y que está en el directorio en el que se ha hecho la copia de seguridad de la base de datos.

copydir.php

Este script incluye otro archivo llamado *deletedir.php*. En *copydir.php* se encuentra la función *copydir* que va a ser la encargada de copiar los archivos y carpetas que haya en un directorio a otro. Tanto el directorio de origen como el de destino son parámetros de entrada de la función. Además esta función lo primero que hace es comprobar que ambos directorios existen, si el de destino existe llama a la función *deletedir* para que elimine los archivos que hay en este directorio destino. A continuación crea el directorio y copia uno a uno todos los archivos y carpetas, con sus correspondientes archivos y subcarpetas si los hubiera, al directorio destino.

deletedir.php

Este script almacena la función *deletedir*, que es la encargada de borrar todos los archivos y subcarpetas, si las hubiera, de un directorio que se le pasa como parámetro de entrada de la función.

restaurar_backup.php

Este script muestra todos los archivos y las carpetas que hay en el directorio en el que se almacenan las copias de seguridad, según la tabla *config_backup*, siempre y cuando este directorio exista y no esté vacío. Si el directorio no existe o está vacío se muestra un aviso al usuario. Si la carpeta existe y no está vacía el usuario tiene que seleccionar un archivo *.sql* y una carpeta con los que restaurar la base de datos. Tanto el nombre del archivo como de la carpeta son enviados a *restaurar_backup2.php*.

restaurar_backup2.php

Este script, al igual que al hacer copias de seguridad va a utilizar las funciones *copydir* y *deletedir*, por eso se incluyen en él los archivos *copydir.php* y *deletedir.php*. Lo primero que se hace es crear una base de datos llamada *fotonica*. Ésta sólo se crea si no existe alguna base de datos en el servidor con ese nombre. Después comprueba si se han introducido los nombres del archivo *.sql* y de la carpeta, si no es así redirige a la página anterior para que se seleccionen nombres para ambos. Si ha seleccionado algún archivo y alguna carpeta de las que hay en el directorio realiza la restauración de la base de datos a partir del archivo seleccionado (**.sql*) y con la ayuda de las funciones *copydir* y *deletedir* copia las copias de los archivos asociados a los elementos en la carpeta *manuales*, incluida en el mismo directorio en los que se encuentran estos scripts.

backup_automatizado.php

Este script es igual que *hacer_backup2.php*, excepto porque los nombres del archivo *.sql* y de la carpeta en la que se van a almacenar los archivos asociados a los elementos se definen en el mismo script, y son *backup_automatico.sql* y *manuales_backup_automatico*. Por lo demás crea una copia de seguridad de la base de datos y una copia de la carpeta en la que se encuentran los manuales y hojas de características de los componentes y equipos de la base de datos de la misma manera que se hacía en *hacer_backup2.php*.

salir.php

Este script lo que hace es destruir la sesión, borrando todas las variables de sesión y redirige al usuario a la página principal *index.php*.

1.5. Seguridad

En esta aplicación el protocolo de transferencia utilizado entre cliente-servidor es HTTPS, lo que hace que esta aplicación tenga un método de comunicación entre el cliente y el servidor seguro, ya que toda la información transmitida entre ambos viaja a través de Internet encriptada. Para configurar este tipo de conexión ha habido que seguir unos pasos, que se numeran a continuación.

Lo primero que hay que hacer es copiar unos archivos de un directorio a otro. El archivo *openssl.cnf*, situado en "C:\Archivos de programa\EasyPHP-5.3.2i\apache\conf" hay que copiarlo a "C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin". Éste es el archivo de configuración de OpenSSL, siendo OpenSSL una aplicación para crear certificados, que autentiquen que el sitio al que se desea acceder es seguro.

Los siguientes archivos que hay que copiar son *libeay32.dll* y *ssleay32.dll* de "C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin" a "C:\WINDOWS\system32". *Libeay32.dll* es un archivo que contiene las funciones del cifrado que permiten las comunicaciones cifradas sobre redes. Por otro lado, *ssleay32.dll* es un módulo asociado a *The OpenSSL Toolkit* de *The OpenSSL Project*. En definitiva, son dos librerías que OpenSSL necesita para crear los certificados.

A continuación, desde símbolo de sistema hay que instalar Apache como servicio y que éste se inicie al iniciar Windows:

```
C:\EasyPHP-5.3.2i\apache\bin>apache -k install -n Apache
```

Desde símbolo de sistema, ejecutar la aplicación OpenSSL:

```
C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin>OpenSSL
```

A continuación se genera la petición de certificado para el servidor:

```
OpenSSL> req -config openssl.cnf -new -out apache.csr
```

Para generar esta petición hay que introducir algunos datos:

- Enter PEM pass phrase: apachessl
- "Country Name": ES
- "State or Province Name": Navarra
- "Locality Name" : Pamplona
- "Organization Name": UPNA
- "Organizational Unit Name": <ENTER>
- "Common Name": 172.18.67.26, siendo ésta la dirección IP del servidor
- "Email Address": administrador@localhost
- "A challange password": <ENTER>
- "An optional company name": <ENTER>

Después se genera una pareja de claves para el servidor:

rsa -in privkey.pem -out apache.key

Vuelve a pedir la contraseña introducida anteriormente:

Enter pass phrase for privkey.pem: apachessl

Seguidamente, se genera el certificado de la clave pública del servidor firmado por la entidad certificadora:

x509 -in apache.csr -out apache.crt -req -signkey apache.key -days 3650

Donde 3650 indica que el certificado caduca en 10 años, tiempo en el cual habrá que volver a generar otro certificado.

Ya se ha creado todo lo necesario, con lo cual se sale de la aplicación y del símbolo del sistema.

A continuación hay que modificar el archivo *httpd.conf* de Apache para que permita que el acceso a la aplicación sea mediante HTTPS y no HTTP. A este archivo se puede acceder pinchando con el botón derecho del ratón sobre el icono que aparece en la barra de tareas de EasyPHP, “Configuration/Apache”. Véase el Anexo 6 para saber cómo debe quedar el archivo *httpd.conf*.

Tras esto, se crea una carpeta llamada “modssl” en "C:\Archivos de programa\EasyPHP-5.3.2i\apache\conf" y copiamos los ficheros *apache.crt* y *apache.key* que se encuentran en "C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin" a "C:\Archivos de programa\EasyPHP-5.3.2i\apache\conf\modssl".

Por último se reinicia el servicio Apache, pulsando con el botón derecho del ratón sobre dicho servicio desde la ventana "Servicios" de las "Herramientas Administrativas" del "Panel de Control", y se selecciona la opción "Reiniciar".

1.6. Configuración de EasyPHP

Una vez instalado EasyPHP en el servidor hay que configurar alguno de sus parámetros. Para ello iniciar EasyPHP y pinchar con el botón derecho del ratón sobre el icono que aparece en la barra de tareas, “Configuration/EasyPHP”. Cuando se abre la ventana de configuración seleccionar “Start on Windows startup”, para que si por alguna razón el servidor se reinicia al iniciar Windows se inicie también EasyPHP. Seleccionar también “Start Apache and MySQL as service (administrator only)”. Además hay que activar la extensión *php_openssl*, para ello pinchar con el botón derecho del ratón sobre el icono que aparece en la barra de tareas, “Configuration/PHP Extension”.

1.7. Migración de la base de datos existente

Como ya se ha explicado, antes de crear esta base de datos ya existía una sobre FileMakerPro, en la que estaba almacenada toda la información relacionada con los equipos y componentes del laboratorio. Entonces, la mejor solución para insertar los datos en la tabla *elementos* de la base de datos *fotonica* es importarlos desde un archivo que haya sido exportado de la base de datos FileMaker Pro.

Para hacer esto lo primero que hay que hacer es cambiar los nombres de los campos de la base de datos *labfot.fp3*, para que sean iguales que los de los campos de la base de datos creada para esta aplicación. Para ello se abre la base de datos y en el menú "Archivo/Definir campos…" se cambian los nombres de los campos, quedando éstos de la siguiente manera:

- Categoria
- Descripcion
- Responsable
- Localizacion_hoja_caracteristicas
- Estado
- Comentarios
- Usuario
- Uso
- Fecha_inicio
- Fecha_estimada_fin
- ESTADO ACTUAL2
- FECHA INICIO2
- USUARIO2

Además se ha cambiado el formato de la fecha, ya que la base de datos de FileMaker Pro utiliza el formato dd-mm-aaaa y MySQL utiliza el formato aaaa-mm-dd. Para cambiar esto se accede al menú "Formato/Fecha…" y se selecciona el formato aaaa-mm-dd. De esta forma todos campos tienen el mismo nombre en ambas bases de datos y el mismo formato. Así se podrá exportar e importar la tabla directamente sobre MySQL. Una vez hecho esto, se exportan todos los campos, excepto *ESTADO ACTUAL2*, *FECHA INICIO2* y *USUARIO2*, ya que en la base de datos *fotonica* no se utilizan. Para ello se selecciona la opción "Archivo/Importar/Exportar/"Exportar registros" en FileMaker Pro y se exporta un archivo "*Texto separado por comas (.csv)*". A continuación se especifica el orden de campos para la exportación que ha de ser el mismo que el orden de los campos en la tabla MySQL. Por último se selecciona la opción *Formatear mediante presentación actual*. Así el formato de las fechas exportadas será el adecuado (aaaa-mm-dd).

Una vez exportado el archivo sólo hay que acceder a PHPMyAdmin y dentro de la base de datos *fotonica*, en la tabla *elementos* seleccionar la pestaña Importar. Seguidamente seleccionar el archivo que se quiere importar (*.*csv*) y seleccionar la opción *CSV usando LOAD DATA.* Seleccionar la opción *Reemplazar los datos de la tabla con los del archivo.*

A continuación configurar de la siguiente manera:

- *Campos terminados en* , (coma).
- *Campos encerrados por* ” (comillas).
- *Carácter de escape* \ (barra invertida).
- *Líneas terminadas en auto.*
- *Usa la palabra clave LOCAL*.

Cuando pulse sobre Continuar los datos se habrán importado correctamente a la tabla *elementos*.

1.8. Programación relacionada

Además de toda la programación en PHP y SQL se ha programado una tarea en Windows XP de manera que todos los viernes a las 9 de la mañana se haga una copia de seguridad de la base de datos, así como de todo el material electrónico asociado a los componentes y equipos de ésta. Para programar esta tarea hay que asociarle un archivo, en este caso es un archivo *.bat* en el que se llama al archivo *backup_automatizado.php*, que es realmente el que realiza la copia de seguridad. El archivo *.bat* sólo tiene una línea de código:

Start iexplore.exe –e https://172/18/67/26/FOTONICA/backup_automatizado.php

2. Equipos necesarios y software

Para llevar a cabo este proyecto se ha utilizado un ordenador en el que hacer toda la programación y las pruebas necesarias para comprobar el correcto funcionamiento de la aplicación. Este ordenador ha utilizado el sistema operativo XP y en él se ha instalado el paquete EasyPHP, que como se ha comentado anteriormente, instala el servidor Apache, junto con el módulo para programación PHP, la base de datos MySQL y PHPMyAdmin. Además se ha instalado un editor de PHP, PHP Editor. También se ha instalado FileMaker Pro, para exportar la base de datos existente a un archivo, para posteriormente ser importado, a través de PHPMyAdmin, a la base de datos MySQL.

Cuando se ha comprobado que toda la aplicación funciona correctamente en modo local, se ha vuelto a instalar EasyPHP en un equipo, que va a ser utilizado como servidor. Una vez copiados en él todos los archivos relacionados con la aplicación (scripts y archivos de la base de datos) se ha hecho una pequeña modificación en el archivo de configuración (*config.php*), donde se ha modificado el directorio de almacenamiento de los scripts en el servidor, en lugar de "localhost/FOTONICA", hay que poner "172.18.67.26/FOTONICA", siendo 172.18.67.26 la dirección IP del servidor.

Como todos los archivos ya están en el servidor, para acceder a la aplicación ya no hay que escribir en la barra de direcciones

https://localhost/FOTONICA o https:// localhost/FOTONICA/index.php

sino que hay que escribir

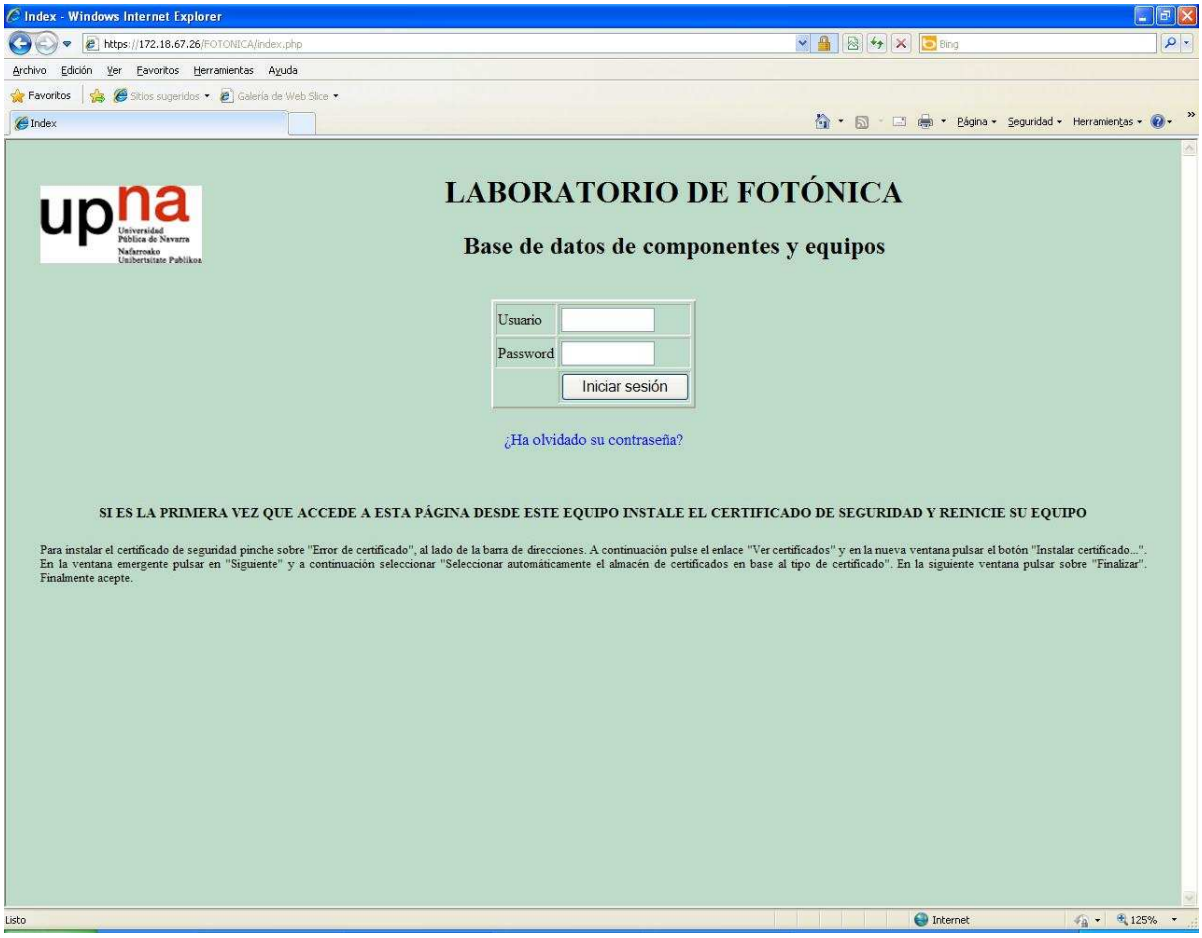
https://172.18.67.26/FOTONICA o https://172.18.67.26/FOTONICA/index.php

La versión de PHP Editor para la realización de este proyecto ha sido: PHP Editor 2.22 – Beta 2. La versión utilizada para abrir la base de datos de FileMaker Pro ha sido FileMaker Pro 3.0E v5. La del paquete EasyPHP utilizada ha sido EasyPHP 5.3.1.0, que incluye las siguientes versiones:

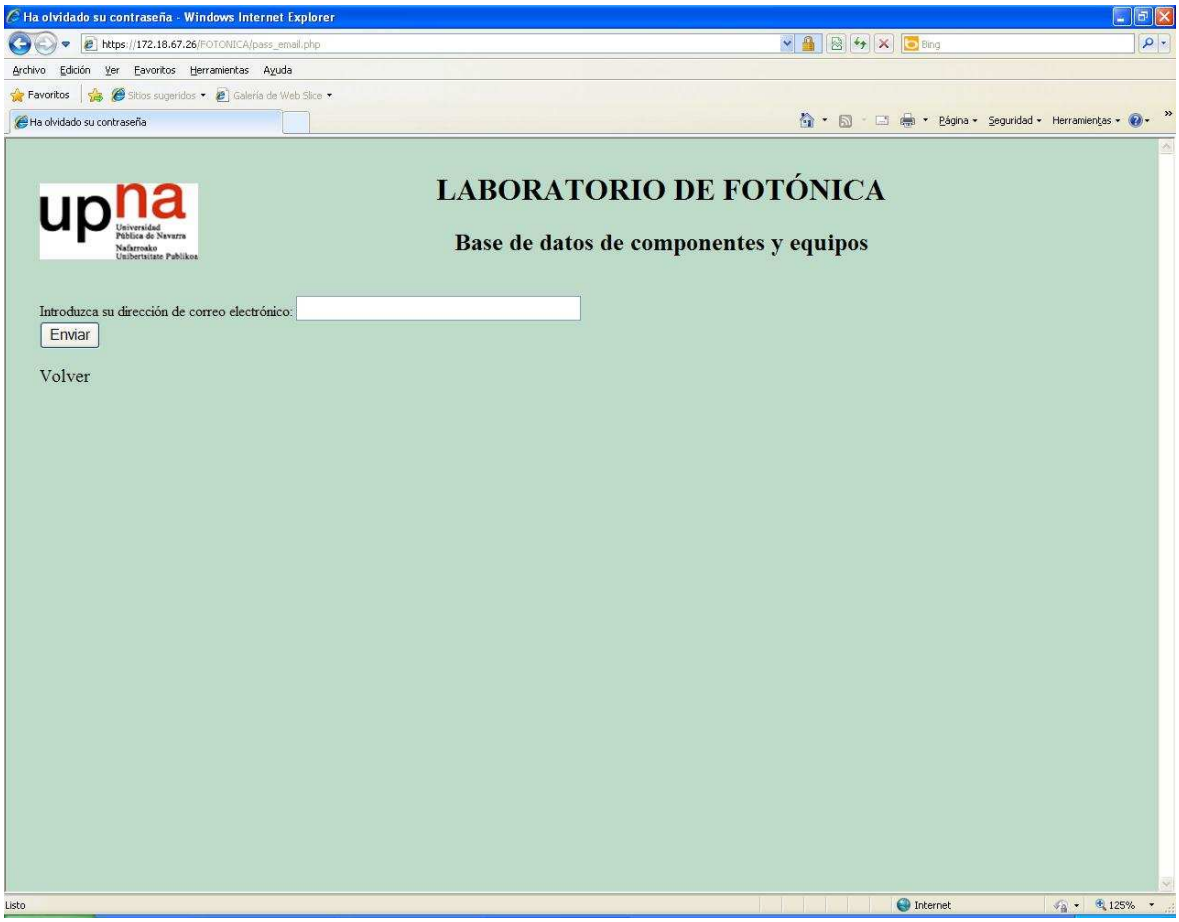
- Apache 2.2.17
- PHP 5.3.3
- MySQL 5.1.52

3. Manual de usuario

Al entrar en la aplicación, a través de https://172.18.67.26/FOTONICA o https://172.18.67.26/FOTONICA/index.php se accede a la siguiente página:



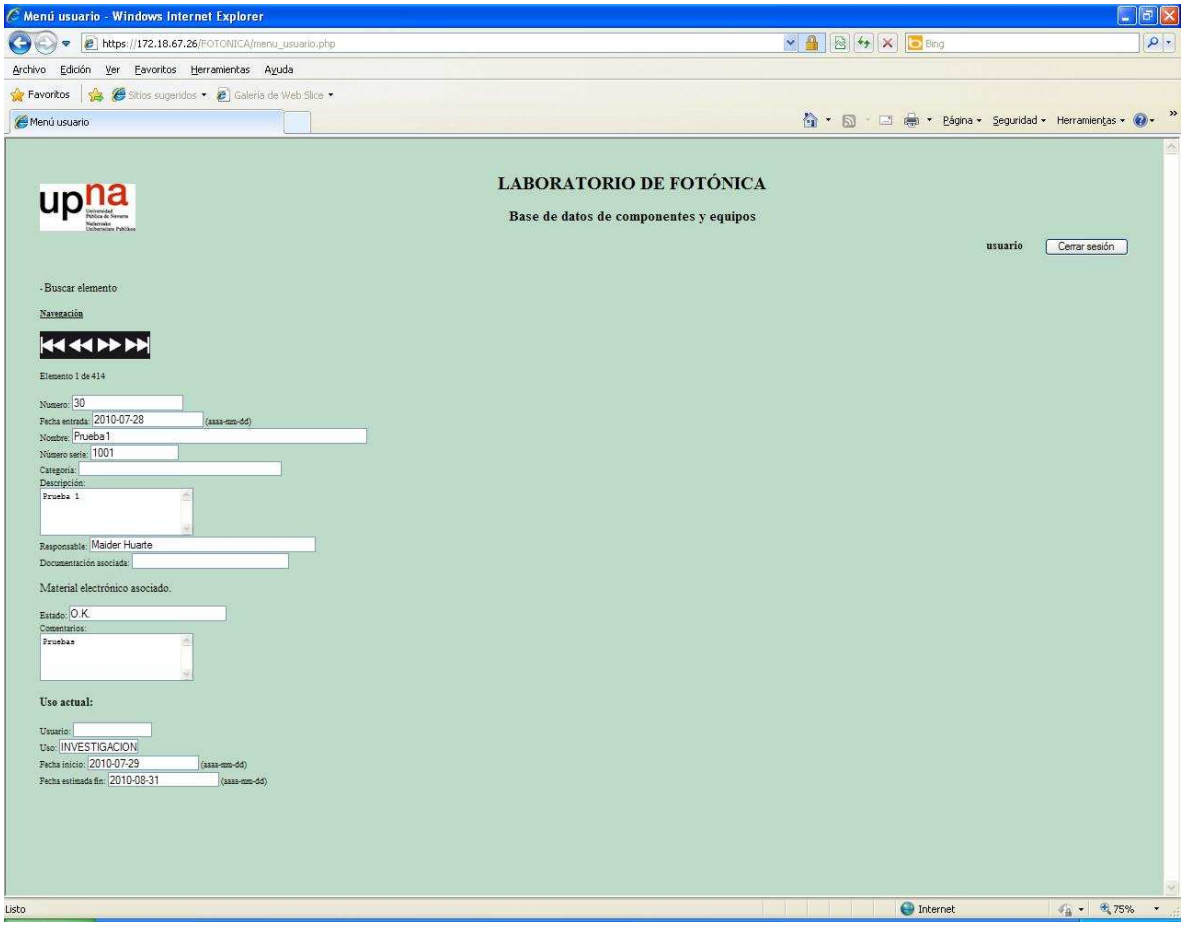
Para acceder hay que introducir un nombre de usuario y una contraseña. Si el usuario ha olvidado su nombre de usuario o su contraseña puede pinchar sobre el enlace que dice *¿Ha olvidado su contraseña?* En ese caso se abre la siguiente ventana:



En esta página se solicita al usuario que introduzca su dirección de correo electrónico. Si algún usuario con acceso a la aplicación tiene asociada esa dirección ese nombre de usuario y su contraseña son enviadas a esa dirección de correo electrónico. Si por el contrario la dirección no existe se muestra un aviso.

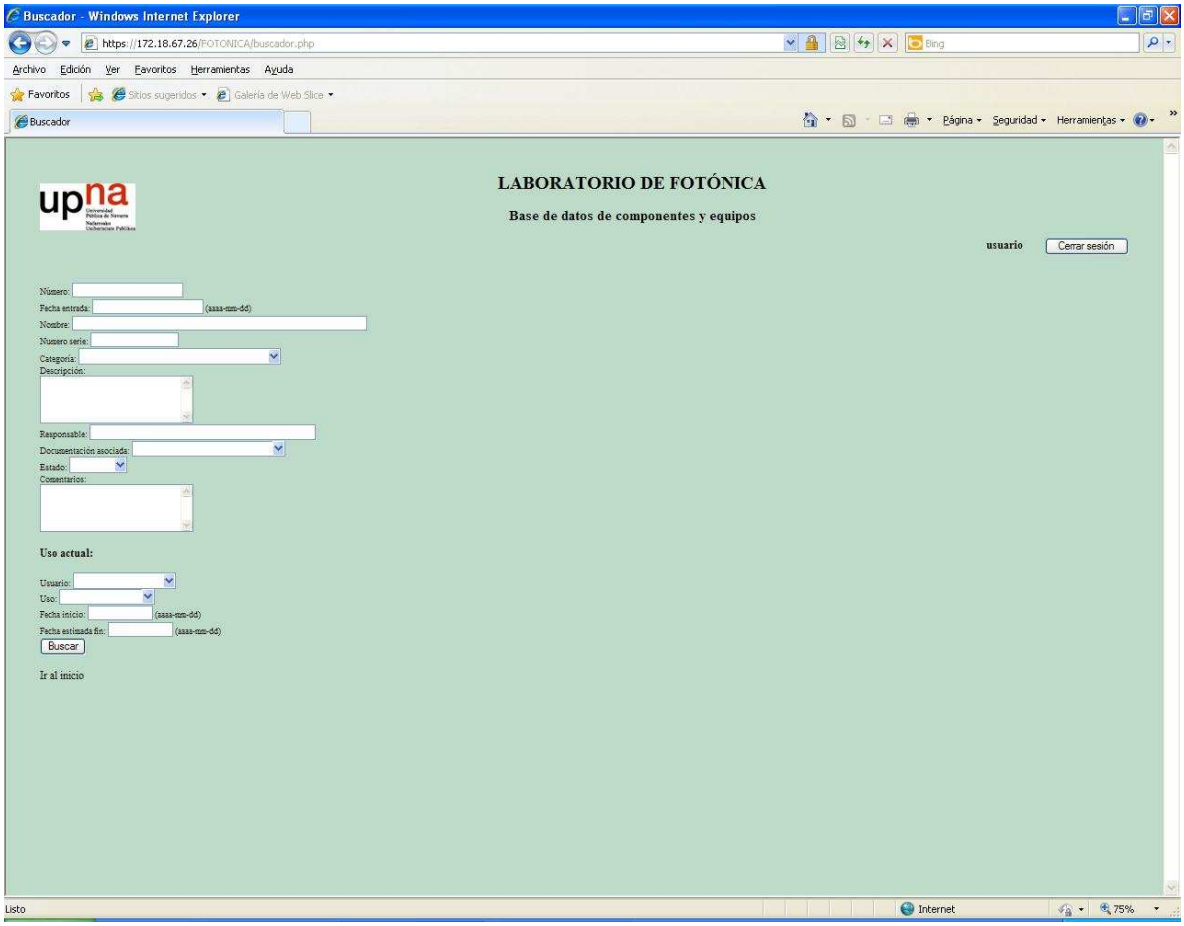
Una vez introducidos el nombre de usuario y la contraseña, dependiendo del nivel de privilegios que tenga ese usuario, se accede a una página o a otra. El usuario de tipo *usuario* es el que menos privilegios tiene en esta aplicación, el *intermedio* tiene alguno más que el anterior y el *administrador* el que tiene el control absoluto de la aplicación.

Si se ha identificado como usuario se accede a la siguiente página:

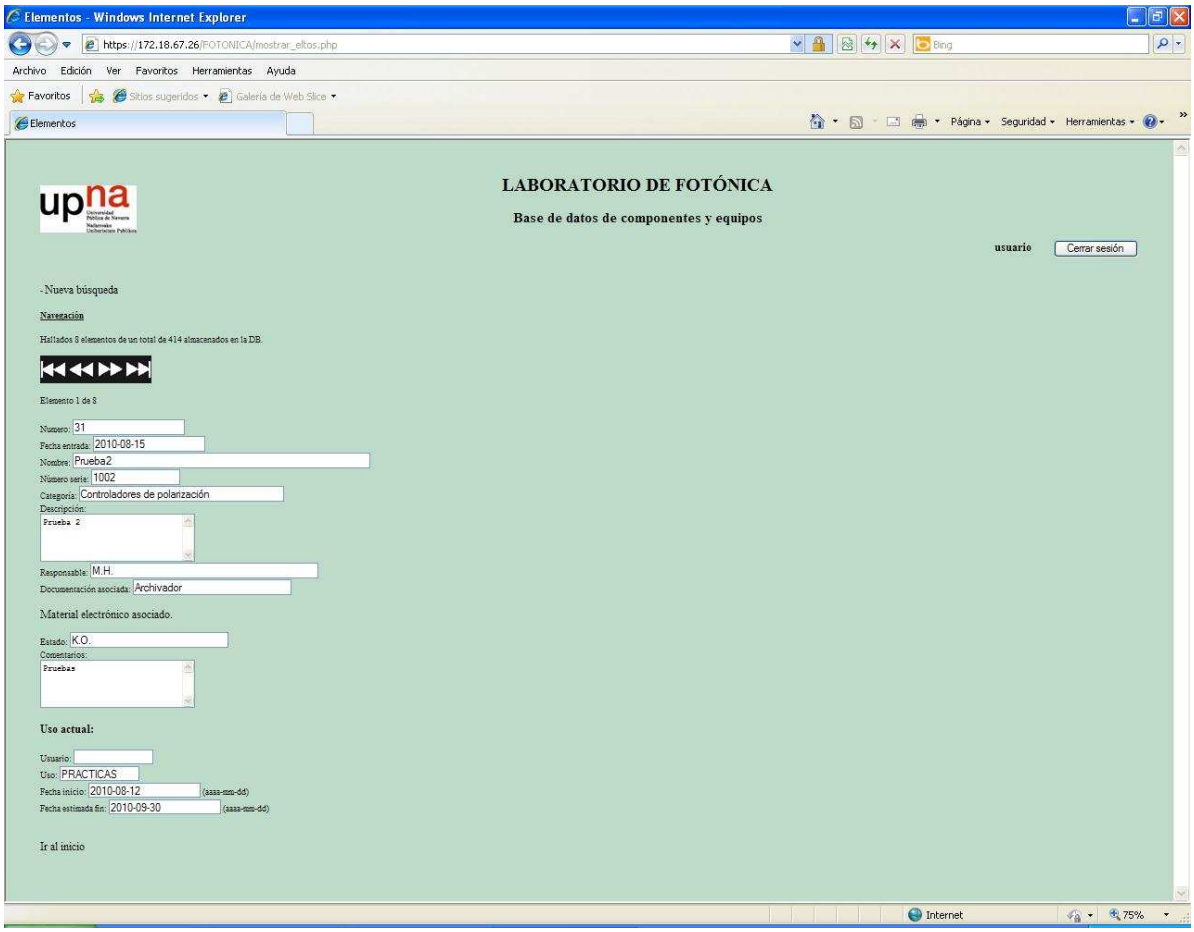


En ella se muestra arriba a la derecha, en negrita, el nombre del usuario que ha accedido a la aplicación y un botón con el que se puede cerrar la sesión. A la izquierda hay un enlace para acceder al buscador de la aplicación. Abajo, toda la información relativa a un elemento almacenada en la base de datos y justo encima de esta información un navegador con el que se puede cambiar el elemento mostrando, pudiendo mostrar el siguiente (siempre ordenados según el número asignado a cada uno), el anterior, el primero o el último.

Si se accede al buscador la página que aparece es la que se muestra a continuación.

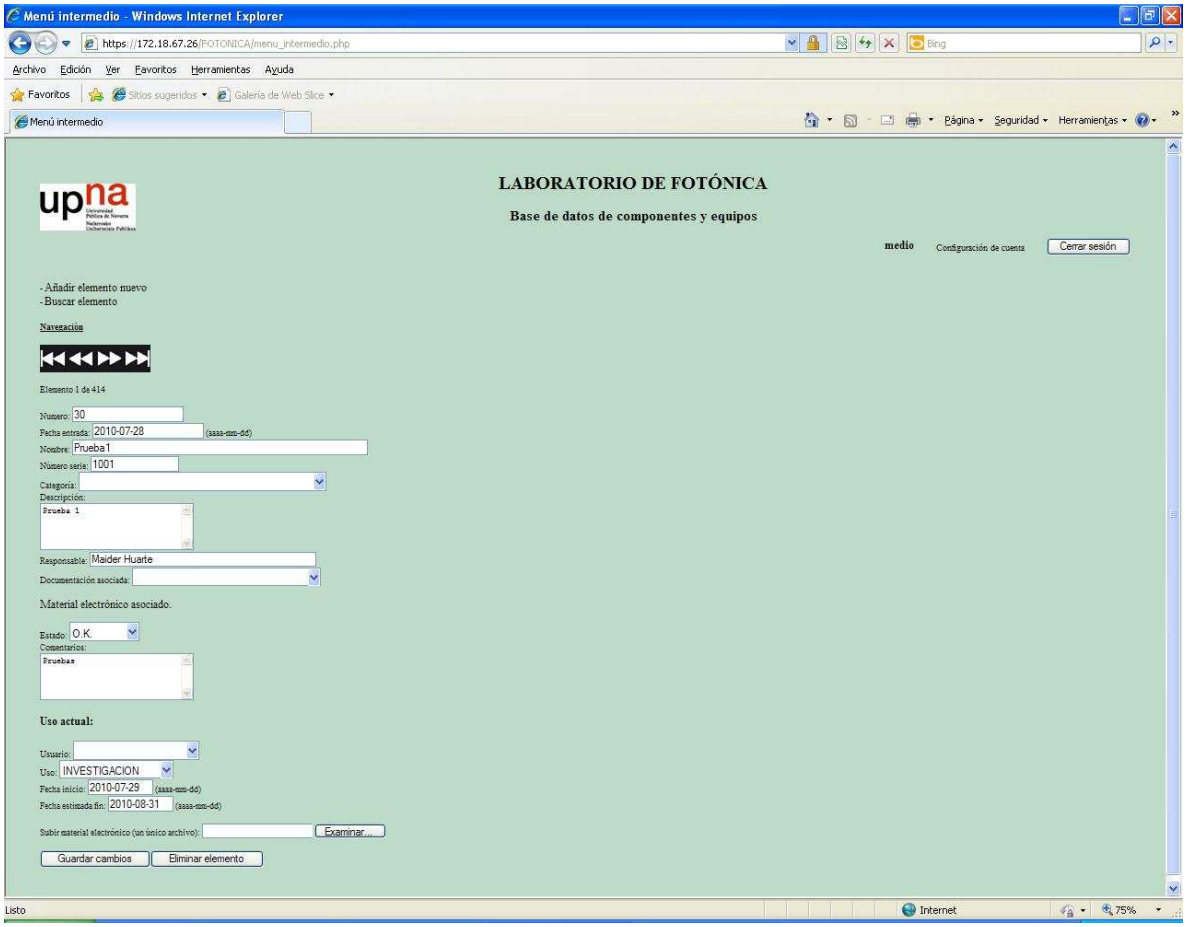


A partir de este buscador se puede buscar cualquier elemento almacenado en la base de datos a partir de cualquiera de sus campos. Si se introduce información en uno de sus campos se buscan todos los elementos cuya información en ese campo se corresponda con la buscada. Si se quiere buscar a partir de más campos se buscan todos los elementos de la base de datos cuyos campos coincidan con los buscados. Una vez realizada la búsqueda aparece una nueva ventana. Si no se han encontrado coincidencias avisa de esto al usuario pero si las ha encontrado se muestra una ventana como la del inicio, pero en este caso los botones del navegador no dan acceso a todos los elementos de la base de datos, sino solamente a los encontrados en la búsqueda. Un ejemplo de esta ventana sería el siguiente:

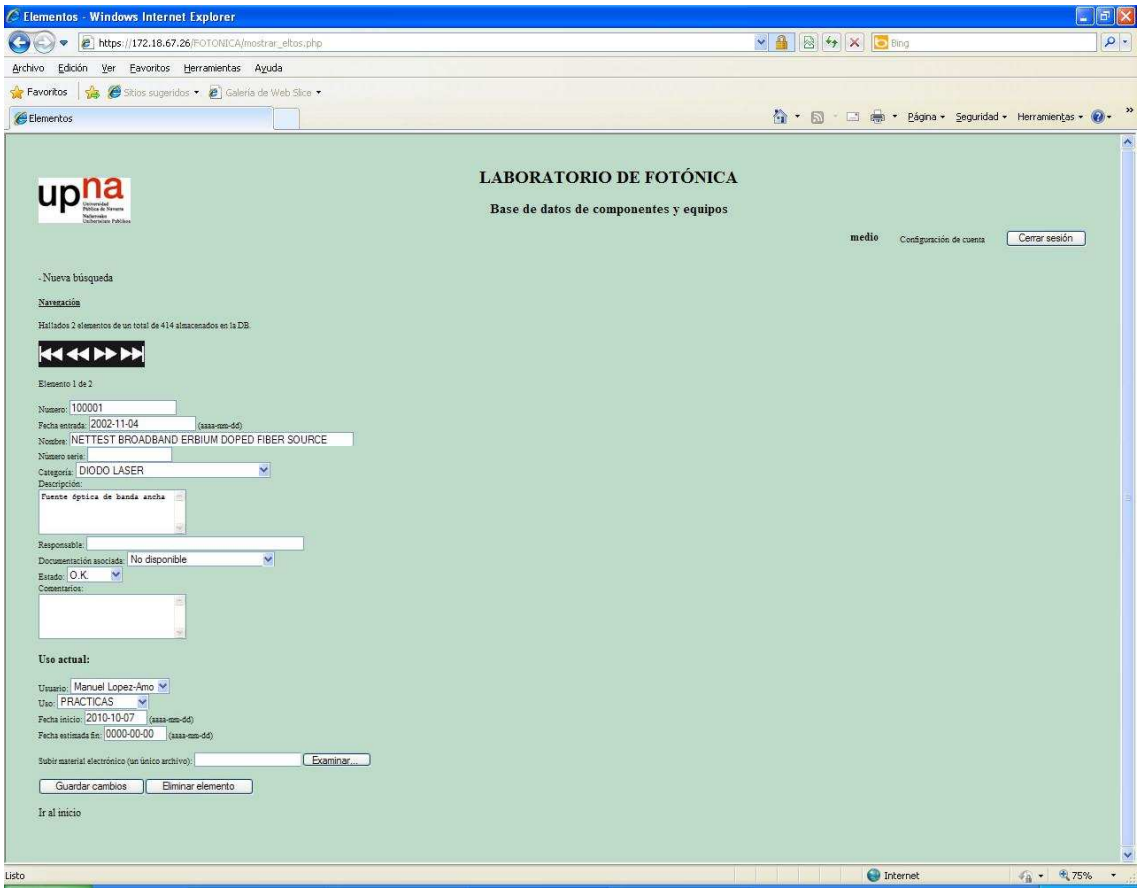


Como se ha podido observar el usuario siempre tiene acceso a su página de inicio (*Ir al inicio*), y a realizar una nueva búsqueda (*Nueva búsqueda*). Además siempre puede cerrar su sesión con el botón *Cerrar sesión*. Los usuarios que pertenezcan a este tipo no tienen acceso a nada más de esta aplicación.

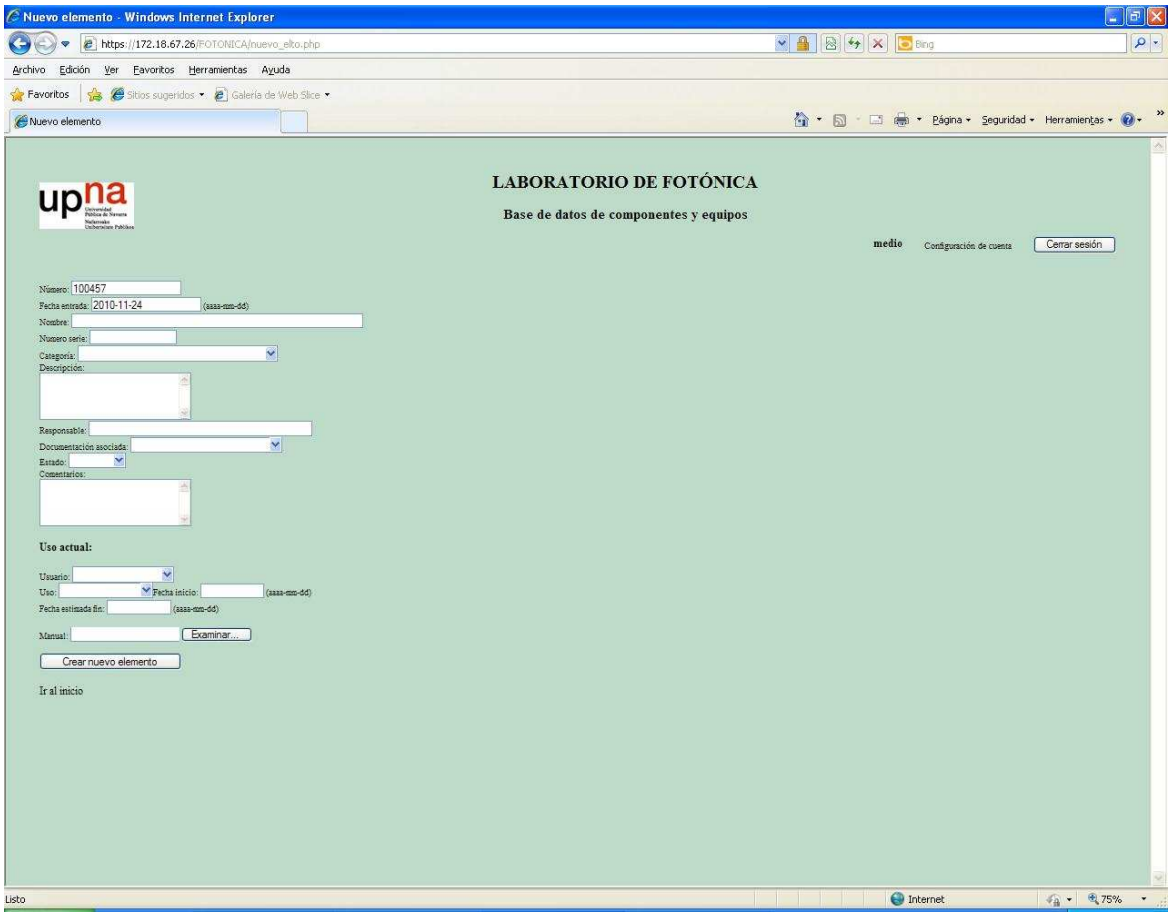
Si el usuario identificado en la página de inicio pertenece a la categoría *intermedio* su página de inicio va a ser la siguiente:



Esta página es igual que a la que acceden los de tipo *usuario* por la salvedad de que tiene la posibilidad de añadir elementos nuevos a la base de datos, editar y eliminar los ya existentes y puede acceder a un menú llamado *Configuración de cuenta*, a través del enlace situado al lado de su nombre de usuario. La realización de búsquedas se hace igual que si se tratara de un *usuario*, pero a la hora de mostrar los elementos encontrados además se tiene la posibilidad de modificar los campos de éstos, subir algún archivo asociado al elemento a la base de datos o eliminarlo de la base de datos.

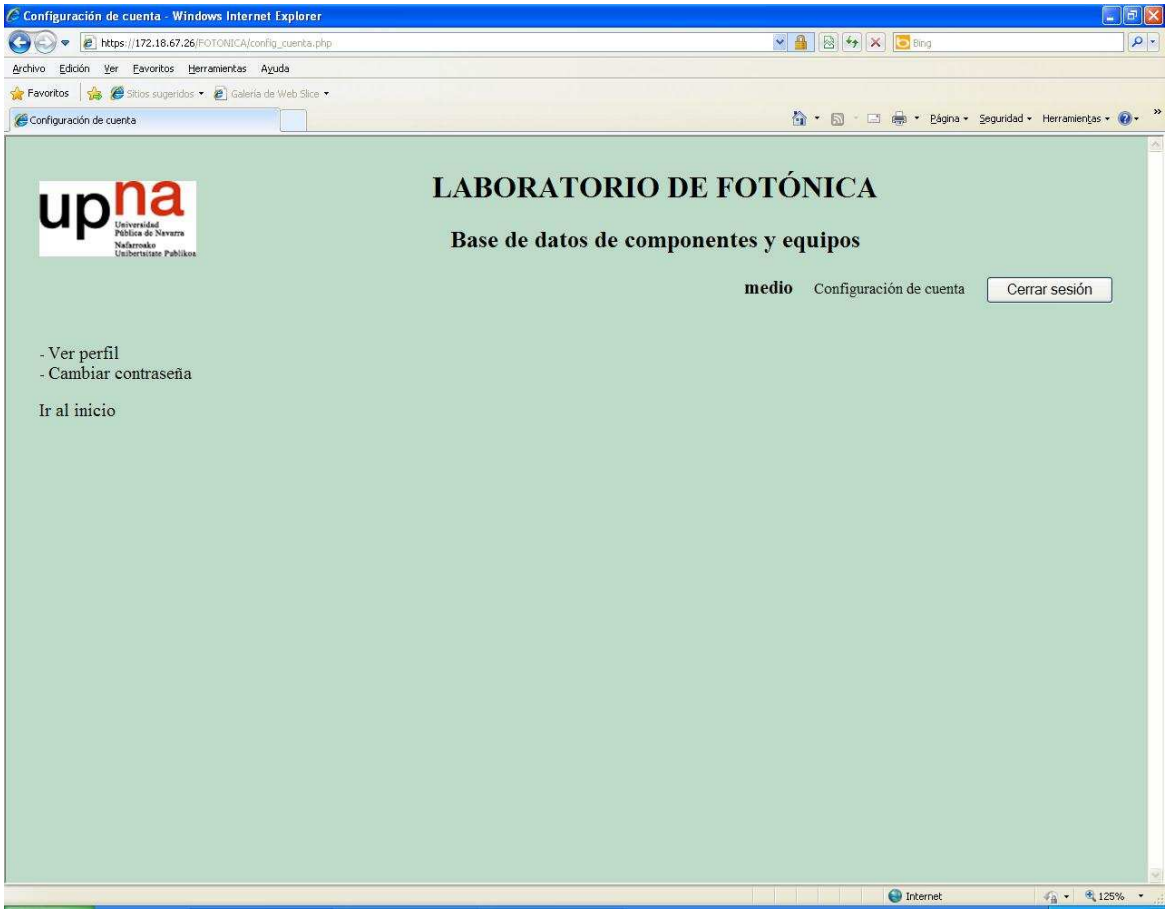


Además, como se puede observar se sigue teniendo acceso a realizar una nueva búsqueda, a volver a su página principal o a acceder a la configuración de su cuenta. Si se vuelve a la página de inicio se puede acceder a un formulario a través del enlace *Añadir elemento nuevo* y añadir un nuevo componente o equipo a la base de datos. El formulario es el siguiente:

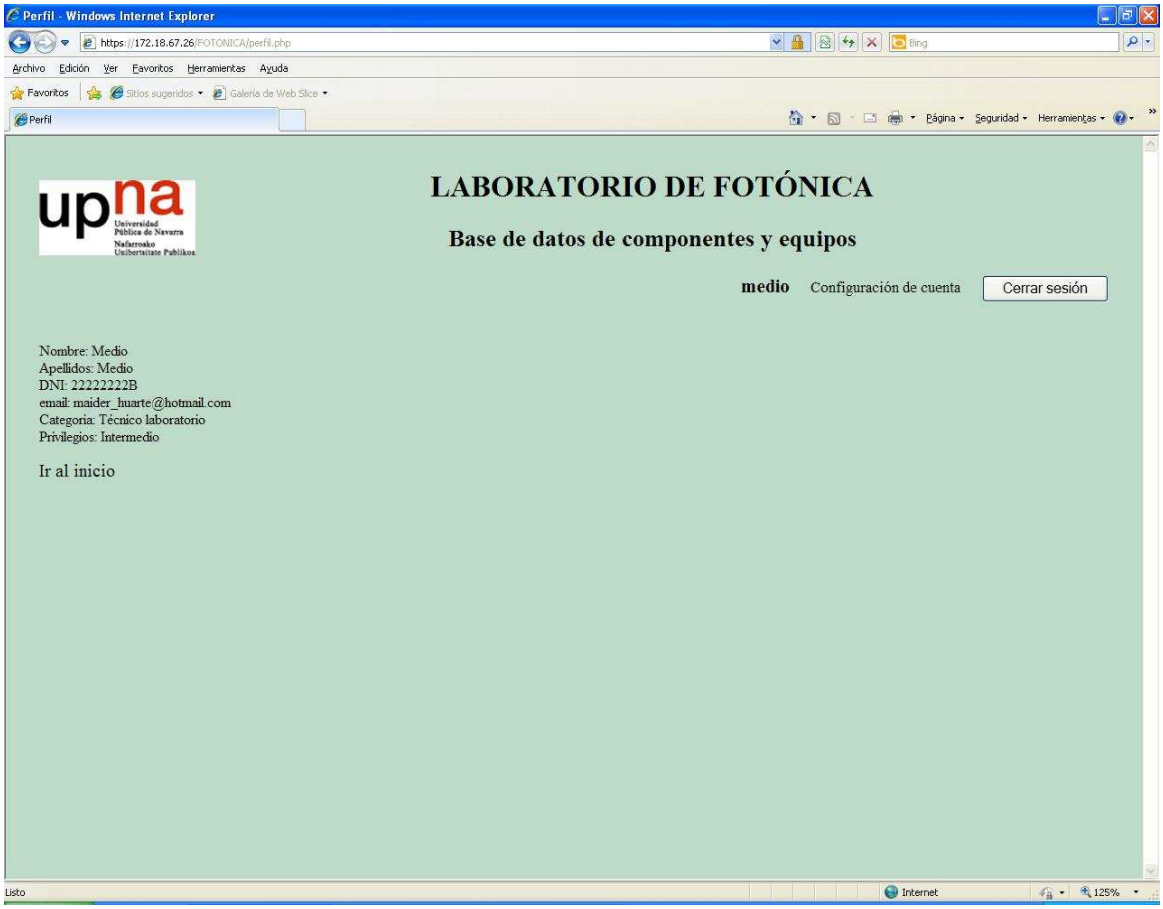


Se pueden introducir datos en cualquiera de los campos, excepto en los dos primeros, que son el número que se le asigna al nuevo elemento y la fecha de entrada en la base de datos. Como se puede observar, se puede subir un archivo para asociarlo al nuevo elemento. Estos archivos suelen ser manuales u hojas de características. Los campos *Categoria*, *Documentación asociada*, *Estado*, *Usuario* y *Uso* son listas desplegables, es decir, las opciones de estos campos están restringidas. Si el nombre o el número de serie del elemento que se quiere añadir a la base de datos ya existe se muestra un aviso y si no lo crea.

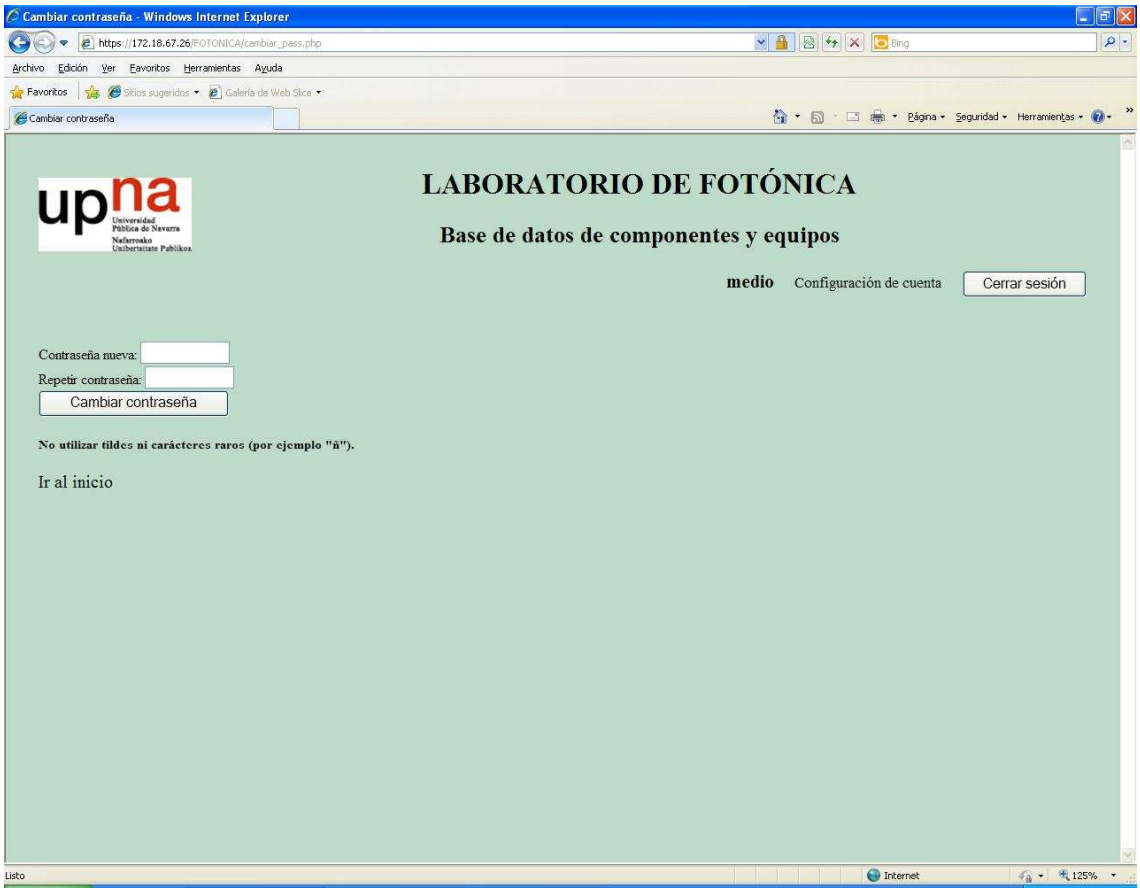
Al acceder al menú *Configuración de cuenta* el usuario puede ver la información relacionada con su perfil o cambiar su contraseña de acceso, tal y como se puede observar en la imagen siguiente:



Al acceder a *Ver perfil* se abre una nueva página en la que se muestra la información personal del usuario.



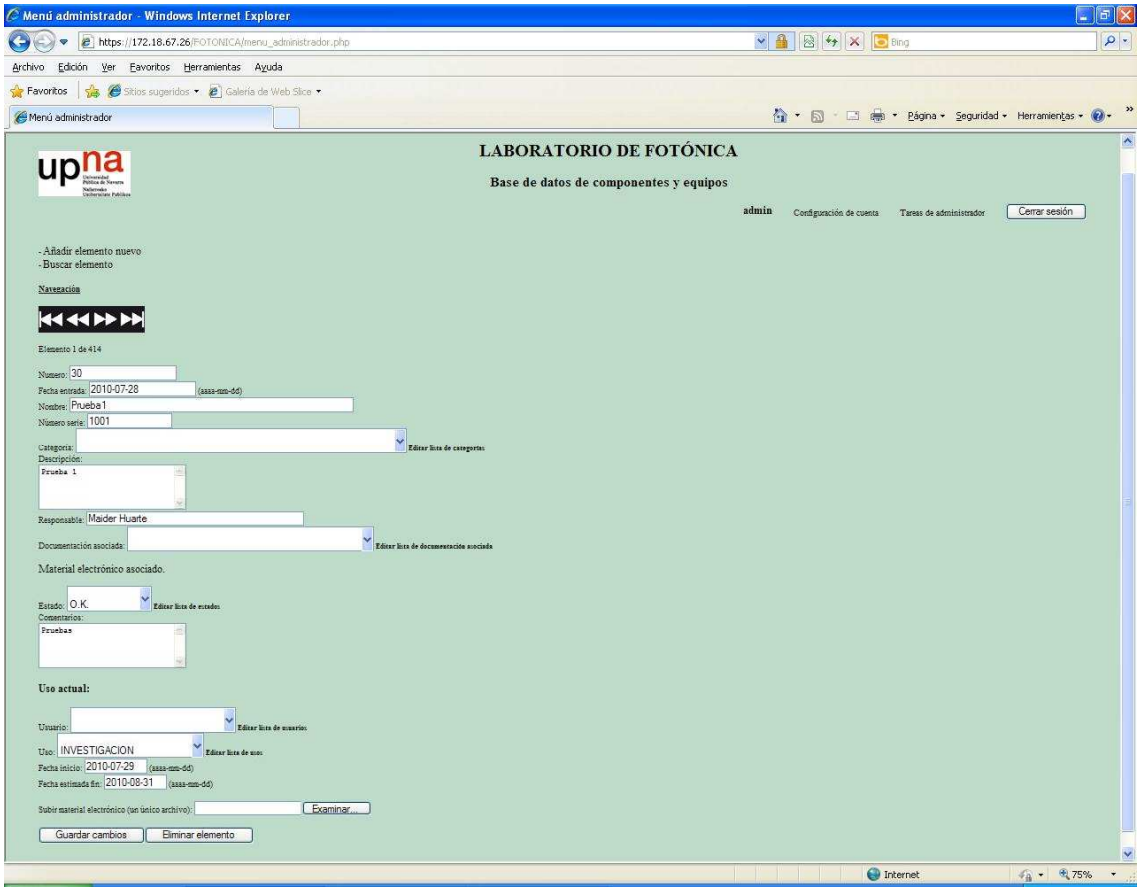
Desde esta página se puede volver al menú *Configuración de cuenta* o a su página de inicio. Si se vuelve a la página anterior y se accede a *Cambiar contraseña* se abre la siguiente ventana:



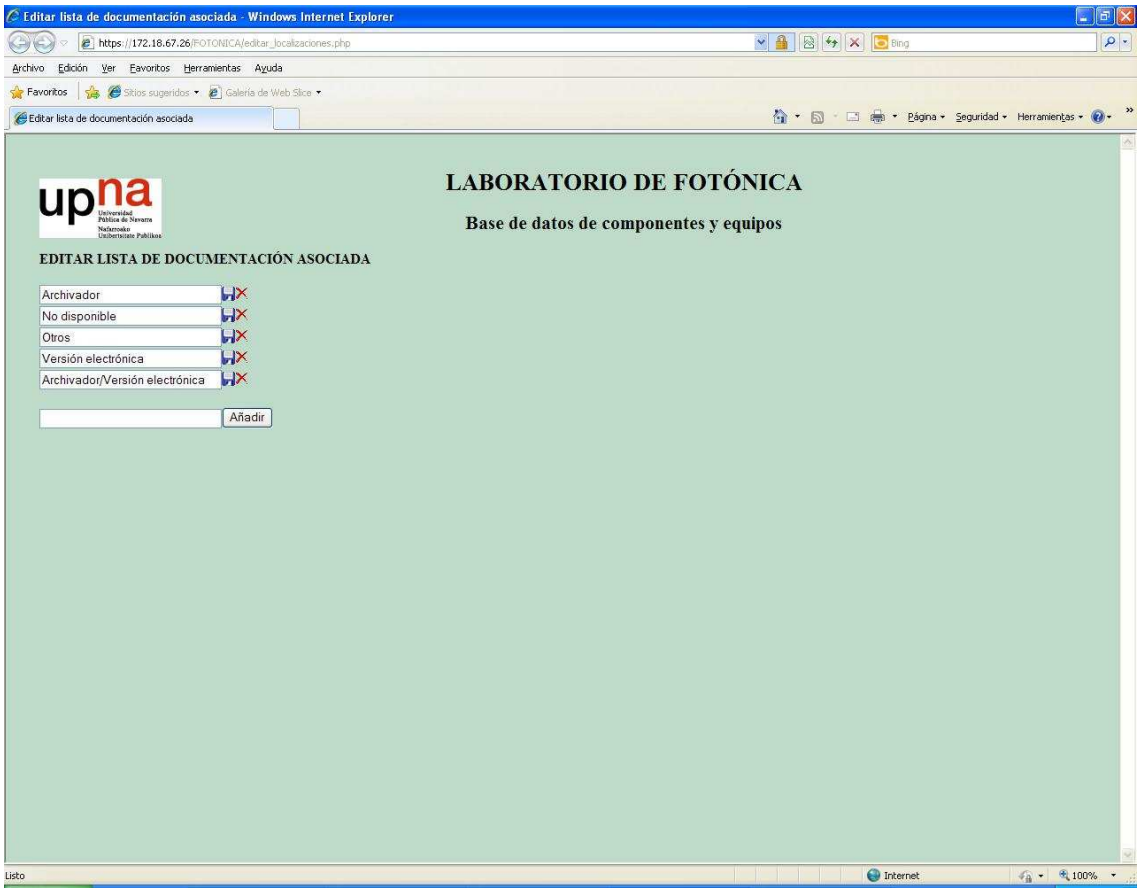
Es en esta donde el usuario puede cambiar su contraseña de acceso, para ello tiene que introducir dos veces la que quiere que sea su nueva contraseña. Se solicita dos veces para evitar posibles errores al escribirla.

Como se ha podido observar el usuario siempre tiene acceso a su página de inicio (*Ir al inicio*), y a *Configuración de* cuenta. Además siempre puede cerrar su sesión con el botón *Cerrar sesión*. Estas son todas las tareas que puede llevar a cabo un usuario de este tipo con esta aplicación.

Finalmente, si el usuario se ha identificado como un *administrador* tiene acceso total a la aplicación y su página de inicio es la siguiente:

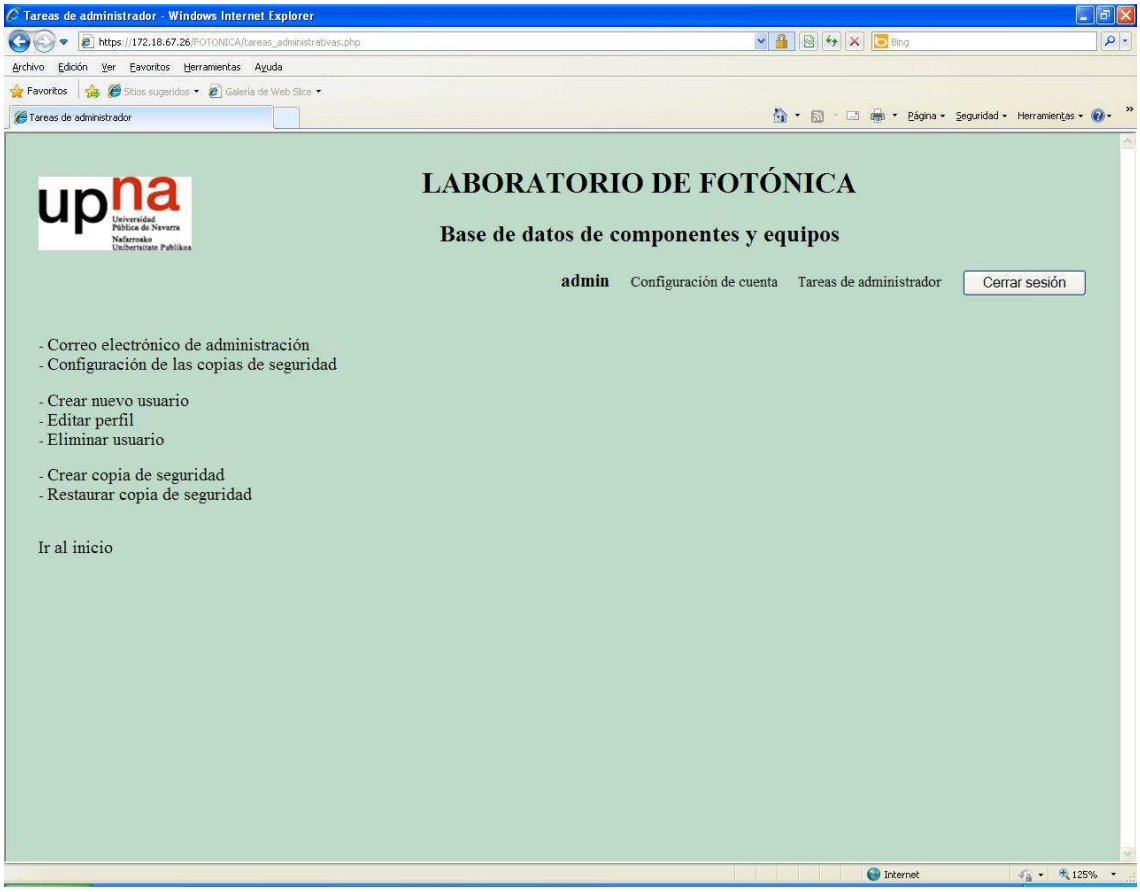


Esta página es igual que la principal de un usuario *intermedio*, exceptuando que los *administradores* pueden modificar la lista desplegable de los campos *Categoria*, *Documentación asociada*, *Estado*, *Usuario* y *Uso*. Si deciden editar estos campos se abre una nueva página en la que pueden modificar su nombre, eliminarlo o añadir nuevos registros a la lista. Un ejemplo de esta nueva página es el siguiente:

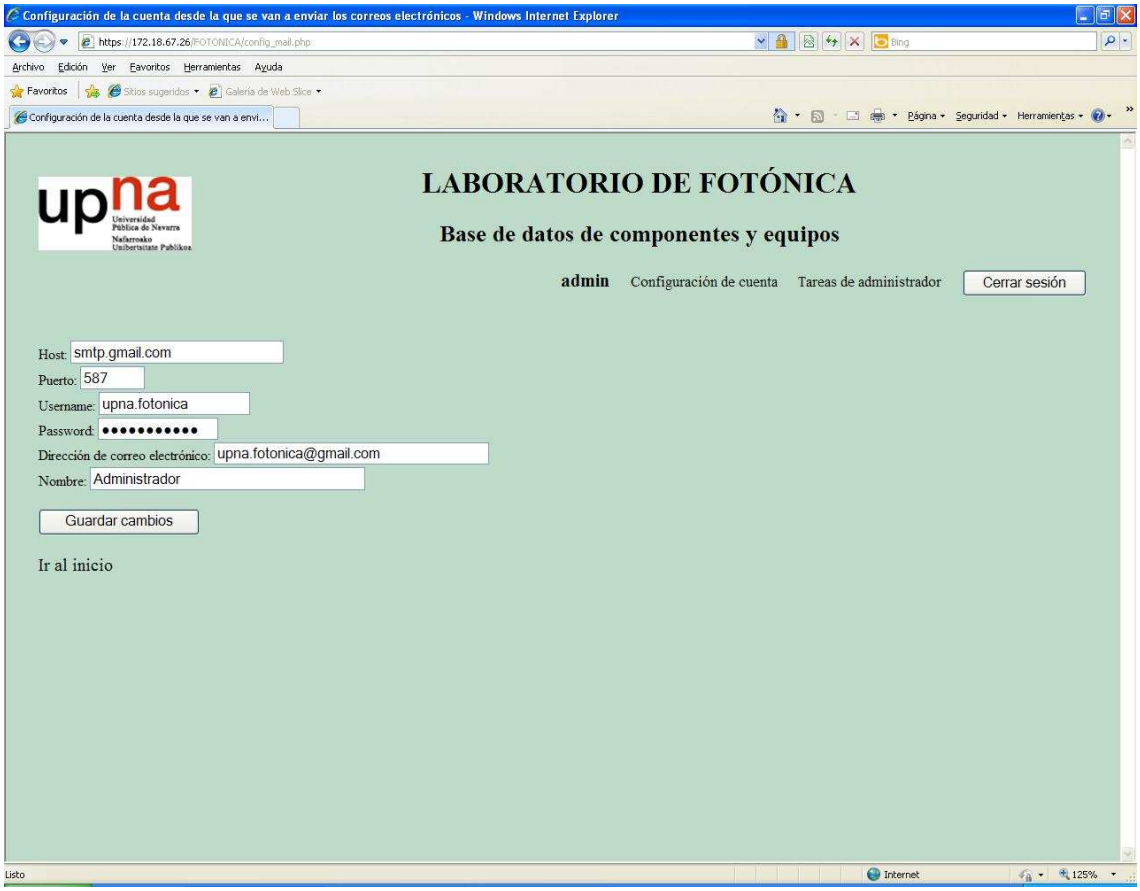


En este caso se ha abierto la lista de *Documentación asociada* y como ya se ha explicado, se puede editar el nombre de cada registro almacenado en ella, eliminar el registro o añadir uno nuevo.

Además este tipo de usuario tiene acceso a una página (*Tares de administrador*), a la que se tiene acceso a través del enlace situado al lado del botón de *Cerrar sesión*. Esta nueva página va a posibilitar a estos usuarios modificar la cuenta de correo electrónico desde la que se envían correos electrónicos, cambiar el directorio donde almacenar las copias de seguridad, crear nuevos usuario, editar la información relativa a los usuarios existentes, eliminar usuarios y crear y restaurar copias de seguridad de la base de datos, así como del material electrónico asociado a los componentes y equipos que la forman. El menú de esta página es el siguiente:



Si el usuario quiere cambiar la configuración de la cuenta para el envío de correos electrónicos (*Correo electrónico de administración*) se accede a la siguiente página:

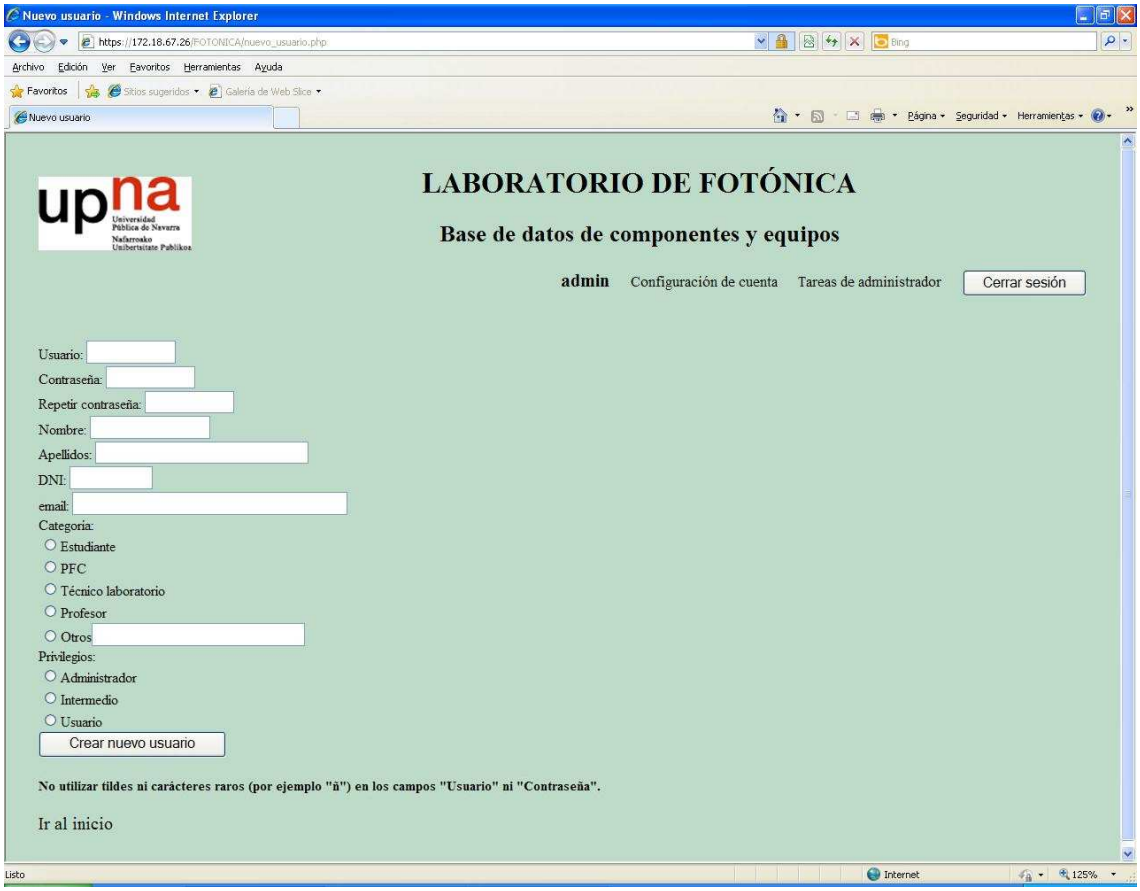


Como se puede observar por defecto se muestra la configuración actual.

Si se quiere modificar el directorio en el que almacenar las copias de seguridad (*Configuración de las copias de seguridad*) la página que se carga es la siguiente:

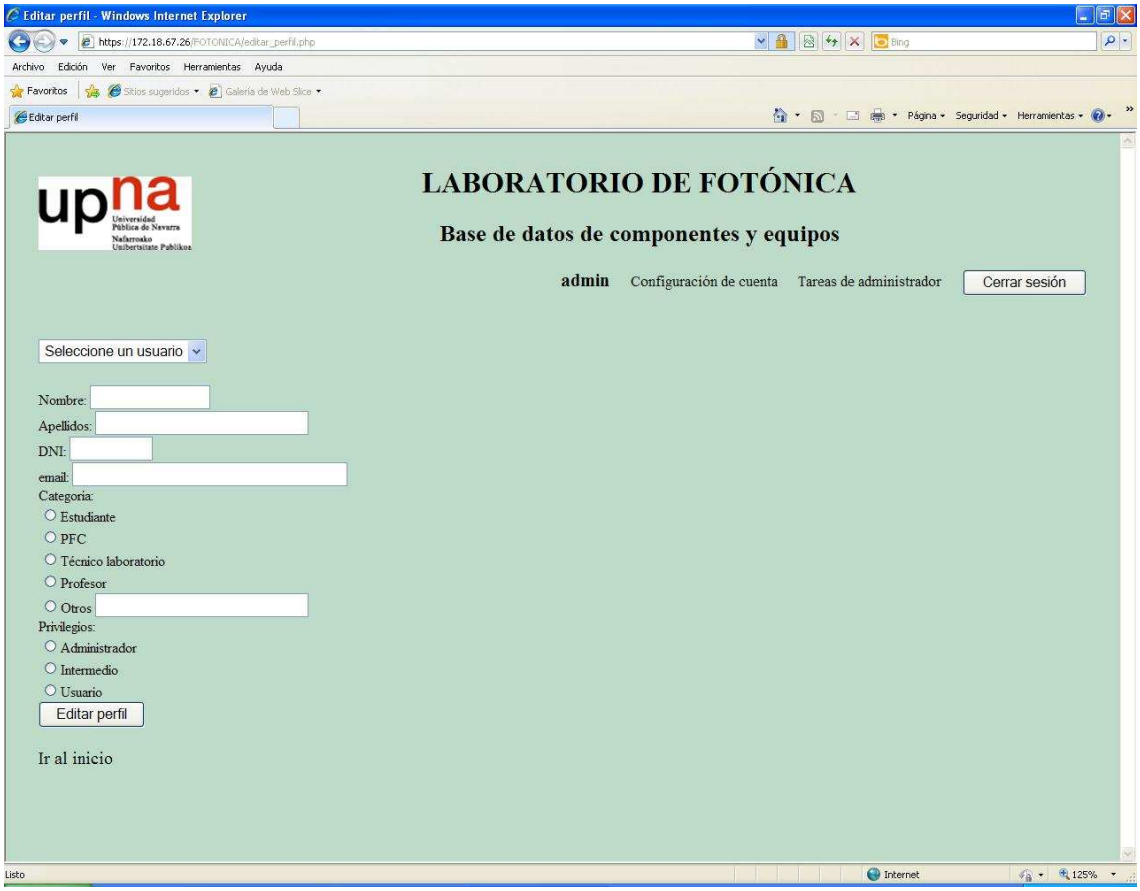
Por defecto se muestra el directorio actual donde se guardan las copias de seguridad, directorio que va a ser el utilizado para el almacenamiento de las copias de seguridad de la base de datos y del material electrónico asociado a los elementos que la componen.

Si se quiere crear un nuevo usuario (*Crear nuevo usuario*) se accede a la siguiente página:

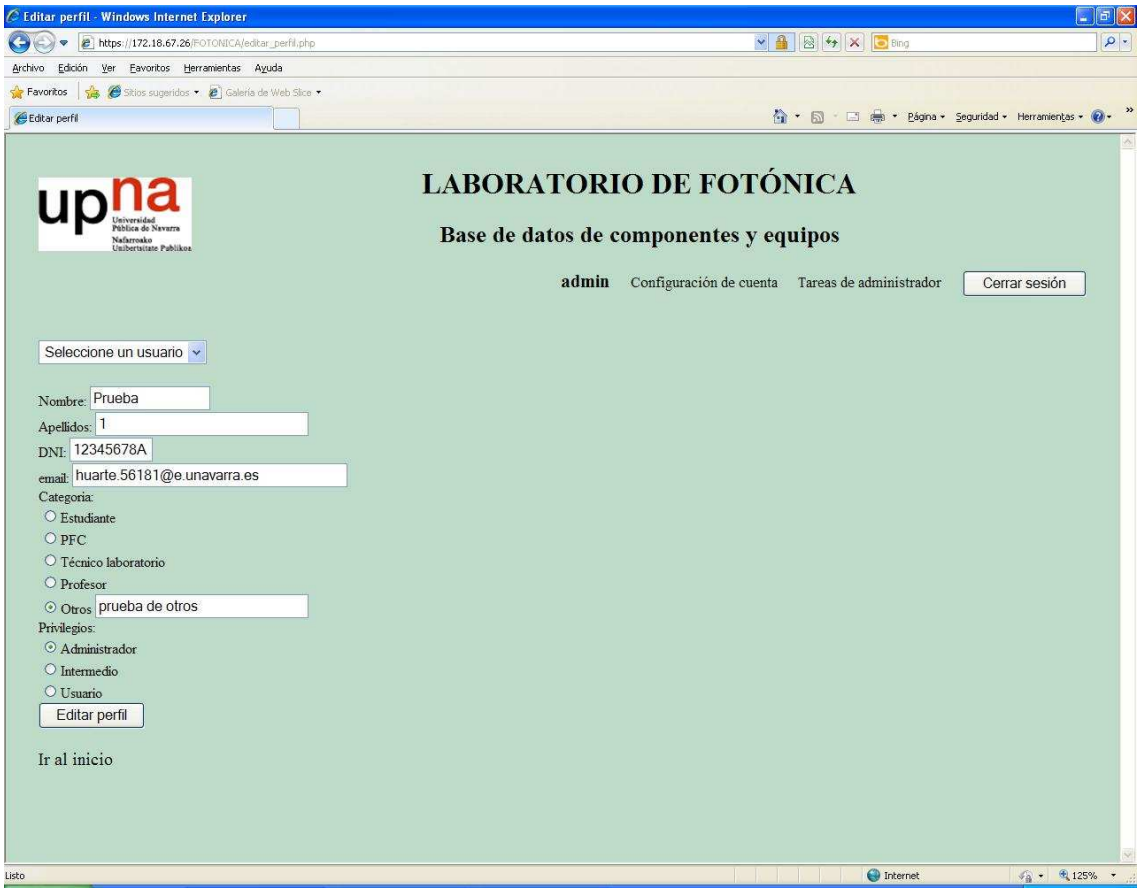


En él se solicita la información necesaria para la creación de un nuevo usuario. El nombre de usuario tiene que ser un nombre que no exista ya en la base de datos, ni tampoco puede ser una cadena vacía. Además la contraseña de acceso se solicita dos veces para evitar posibles errores humanos al introducirla.

Si se quiere editar la información relacionada con un usuario ya existente (*Editar perfil*) se muestra una página como la siguiente en la que se tiene que seleccionar un usuario de la lista de posibles usuarios.



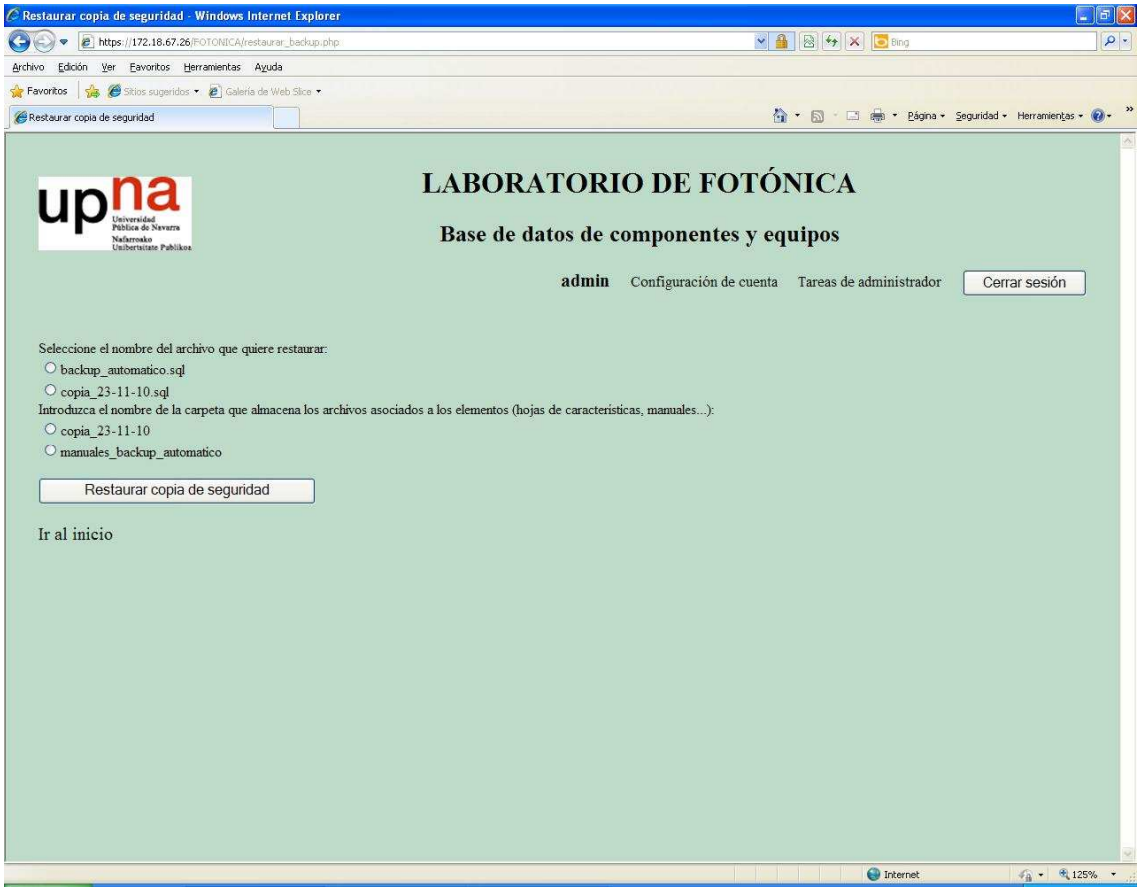
Una vez seleccionado el usuario se carga su información y el *administrador* puede modificar todos los campos relacionados con su perfil. El campo situado al lado de la opción Otros, perteneciente a la *Categoría* sólo se tendrá en cuenta si ésta es la opción seleccionada. Si la opción seleccionada en *Categoría* es otra da igual si se introduce o no información en dicho campo, ya que ésta no será almacenada en la base de datos.



Si lo que se quiere es eliminar un usuario (*Eliminar usuario*) se abre una página en la que se muestra un cuadro con información relevante de cada usuario que tiene acceso a la aplicación y seguidamente se solicita el nombre de ese usuario.

Si se quiere crear una copia de seguridad (*Crear copia de seguridad*) se solicita un nombre para el archivo (.sql) que va a almacenar la copia de seguridad de la base de datos, así como el nombre para una carpeta en la que se guardará el material electrónico asociado a los componentes y equipos de la base de datos. Si no se introduce ningún nombre en cualquiera de los campos solicitados se vuelve a cargar la página y hasta que ambos campos no dispongan de nombre no se realizará la copia de seguridad.

Si se quiere restaurar una copia de seguridad (*Restaurar copia de seguridad*) se muestra una página en la que se muestran los posibles archivos desde los que restaurar la base de datos así como las posibles carpetas que almacenan las copias del material electrónico.



Estas son las posibles acciones que puede llevar a cabo un usuario *administrador*.

4. Conclusiones

Durante la elaboración de este proyecto se ha estudiado en profundidad la programación en PHP y SQL y la configuración de una conexión segura entre cliente-servidor en Apache. Tras esto se han llegado a las siguientes conclusiones:

- Ésta es una aplicación segura debido al protocolo de transferencia de hipertexto utilizado entre cliente y servidor (HTTPS).
- Es una aplicación de acceso restringido.
- Con respecto a la aplicación anterior ésta tiene la posibilidad de crear y restaurar copias de seguridad de la base de datos desde la propia aplicación, lo cual es una mejora considerable, lo que permite la restauración de la base de datos a partir de un estado anterior de la misma de una manera muy sencilla.
- Gracias a los diferentes tipos de usuarios que tienen acceso a la aplicación, se puede controlar el uso que éstos le van a dar a la misma.
- La implantación de esta aplicación en otros laboratorios sería muy sencilla. Bastaría con crear una base de datos similar y realizar los cambios necesarios en los scripts para que la nueva aplicación se comunicara con esta nueva base de datos. Después se podrían albergar estos archivos en el mismo servidor en el que está almacenada esta aplicación en una nueva carpeta del directorio origen de Apache. Además la nueva base de datos se almacenaría en una nueva carpeta en el mismo directorio en el que se encuentra la base de datos de esta aplicación.
- En este momento la aplicación sólo está accesible desde ordenadores de la universidad debido a que la dirección IP del servidor es una dirección privada de la red de la universidad.
- Si algún día se cambia la dirección IP del servidor bastaría con cambiar en el archivo *config.php* la línea 172.18.67.26/FOTONICA por xxx.xxx.xxx.xxx/FOTONICA, siendo xxx.xxx.xxx.xxx la nueva dirección IP. Y en la barra de direcciones escribir:

  https://172.18.67.26/FOTONICA

  o

  https://172.18.67.26/FOTONICA/index.php

- La apariencia de la aplicación es fácilmente modificable, debido a que ha sido definida en archivos independientes (*index.css* y *estilo.css*).

5. Bibliografía

http://www.easyphp.org/

http://php.net/

http://es.php.net/manual/es/function.date.php

http://httpd.apache.org/

http://httpd.apache.org/docs/2.2/ssl/

http://dev.mysql.com/

http://sauce.pntic.mec.es/crer0052/apache/segura.htm

http://es.html.net/tutorials/css/lesson1.php

http://dev.mysql.com/doc/refman/5.0/es/backup.html

http://webscripts.softpedia.com/script/PHP-Clases/PHPMailer-Codeworx-Technologies-46188.html

http://www.webtaller.com/construccion/lenguajes/php/lessons/sesiones.php

http://www.programacionphp.net/recursos-manuales.html

http://dattatecayuda.com/%C2%BFcomo-creo-tablas-e-ingreso-datos-en-forma-manual-a-mis-bases-de-datos-mysql-con-phpmyadmin/3714

http://www.desarrolloweb.com/

http://www.desarrolloweb.com/manuales/12/

http://www.desarrolloweb.com/manuales/58

http://www.desarrolloweb.com/manuales/9/

http://www.desarrolloweb.com/manuales/34/

http://www.desarrolloweb.com/articulos/1181.php

http://www.desarrolloweb.com/articulos/1202.php

http://www.desarrolloweb.com/articulos/2022.php

6. Anexos

6.1. Anexo 1. Procedimiento de instalación

Para llevar a cabo la configuración necesaria para el correcto funcionamiento de esta aplicación seguir los siguientes pasos.

1. Instalar EasyPHP en C:\Archivos de programa.
2. Iniciar EasyPHP y pinchar con el botón derecho del ratón sobre el icono que aparece en la barra de tareas, “Configuration/EasyPHP”. Cuando se abre la ventana de configuración seleccionar “Start on Windows startup”, para que si por alguna razón el servidor se reinicia al iniciar Windows se inicie también EasyPHP. Seleccionar también “Start Apache and MySql as service (administrator only)”.
3. Para que el envío de correos desde la aplicación se lleve a cabo correctamente hay que añadir una extensión a PHP, para ello pinchar con el botón derecho del ratón sobre el icono que aparece en la barra de tareas, “Configuration/PHP Extension”. Seleccionar la extensión php_openssl.
4. Configurar HTTPS

    4.1. Copiar "openssl.cnf" de "C:\Archivos de programa\EasyPHP-5.3.2i\apache\conf" a "C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin"

    4.2. Copiar "libeay32.dll" y "ssleay32.dll" de "C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin" a "C:\WINDOWS\system32"

    4.3. Desde símbolo del sistema:

    4.4. Instalar Apache como servicio y que se inicie al iniciar Windows:

    C:\EasyPHP-5.3.2i\apache\bin>apache -k install -n Apache

    4.5. Ejecutar la aplicación OpenSSL:

    C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin>OpenSSL

    4.6. Generar la petición de certificado para el servidor:

    OpenSSL> req -config openssl.cnf -new -out apache.csr

    Enter PEM pass phrase: apachessl

    "Country Name" "ES"

    "State or Province Name" "Navarra"

    "Locality Name" "Pamplona"

    "Organization Name" "UPNA"

    "Organizational Unit Name" <ENTER>

"Common Name" "172.18.67.26"
(Dirección IP del equipo servidor)

"Email Address" "administrador@localhost"

"A challange password" <ENTER>

"An optional company name" <ENTER>

4.7. Generar una pareja de claves para el servidor :

rsa -in privkey.pem -out apache.key

Enter pass phrase for privkey.pem: apachessl

4.8. Generar el certificado de la clave pública del servidor firmado por la entidad certificadora:

x509 -in apache.csr -out apache.crt -req -signkey apache.key -days 3650

(El certificado caduca en 10 años)

4.9. exit

4.10. Pinchar con el botón derecho del ratón sobre el icono que aparece en la barra de tareas de EasyPHP, “Configuration/Apache”. Se abre el archivo httpd.conf. Modificarlo de manera que quede como se muestra en el Anexo 6.

4.11. Crear una nueva carpeta llamada "modssl" dentro de la carpeta de “C:\Archivos de programa\EasyPHP-5.3.2i\apache\conf", y copiamos los ficheros "apache.crt" y "apache.key" que se encuentran en "C:\Archivos de programa\EasyPHP-5.3.2i\apache\bin" a "C:\Archivos de programa\EasyPHP-5.3.2i\apache\conf\modssl".

4.12. Por último reiniciar el servicio "Apache", pulsando con el botón derecho del ratón sobre dicho servicio desde la ventana "Servicios" de las "Herramientas Administrativas" del "Panel de Control", y seleccionar la opción "Reiniciar".

5. Los scripts localizados en la carpeta scripts copiarlos en C:\Archivos de programa\EasyPHP-5.3.2i\www\FOTONICA

6. La carpeta fotonica es la DB, que se encuentra en C:\Archivos de programa\EasyPHP-5.3.2i\mysql\data

7. Crear una nueva tarea programa en Windows (por defecto se creó una los viernes a las 9 de la mañana pero se puede poner cualquier día, hora y cambiar la frecuencia). Asociar a esta tarea el archivo backup_automatizado.bat, que se encuentra en C:\Archivos de programa\EasyPHP-5.3.2i\www\FOTONICA.

8. Si se instala en un ordenador que no sea el servidor con dirección IP 172.18.67.26, cambiar en el archivo config.php esta dirección por la del nuevo equipo.

9. Para acceder a la aplicación habría que cambiar en la barra de direcciones 172.18.67.26 por la dirección nueva:

https://172.18.67.26/FOTONICA o https://172.18.67.26/FOTONICA/index.php

10. Reiniciar el equipo

6.2. Anexo 2. Cómo acceder a la aplicación si se elimina la tabla *usuarios* de la base de datos

Si la tabla *usuarios* de la base de datos *fotonica* se eliminara no se podría tener acceso a la aplicación. En el caso de que esto tuviera lugar habría que acceder desde el servidor a la aplicación PHPMyAdmin (https://127.0.0.1/home/mysql o https://localhost/home/mysql) y crear la tabla *usuarios* dentro de la base de datos *fotonica* y en ella crear un usuario *administrador*.

Recordad que la tabla *usuarios* debiera tener los siguientes campos:

- Usuario
- Password
- Nombre
- Apellidos
- DNI
- Email:
- Categoria
- Otros
- Privilegios

Siendo necesario poner en el campo Privilegios *Administrador*.

Una vez creado este usuario se podría volver a acceder a la aplicación y si fuera necesario realizar una restauración de la base de datos a partir de una copia de seguridad.

6.3. Anexo 3. Cómo restaurar una copia de seguridad si la tabla *config_backup* ha sido eliminada

Si la tabla *config_backup* o toda la base de datos se hubiera eliminado no se podría hacer la restauración debido a que no se sabría cuál es el directorio en el que están almacenadas las copias de seguridad. Para salvar este contratiempo acceder desde el servidor a la aplicación PHPMyAdmin (https://127.0.0.1/home/mysql o https://localhost/home/mysql) y crear una tabla llamada *config_backup* en la base de datos *fotonica* con un único campo llamado Directorio y en él escribir el directorio en el que se encuentran las copias de seguridad.

6.4. Anexo 4. Información del servidor

Servidor donde se encuentra almacenada la aplicación:

- Dirección IP: 172.18.67.26
- Nombre de usuario: apache
- Contraseña: laboratorio2010

Usuario y contraseña del gestor MySQL:

- Usuario: root
- Contraseña: upna

6.5. Anexo 5. Cuenta gmail

Para el envío de correos electrónicos de esta aplicación se ha creado una cuenta en gmail con los siguientes datos:

- Dirección de correo electrónico: upna.fotonica@gmail.com
- Contraseña: labfotonica
- Pregunta: What was your first phone number?
- Respuesta: 2010-1234567890

6.6. Anexo 6. Archivo de configuración *httpd.conf*

Para saber cómo debería quedar el archivo *httpd.conf* para que la comunicación entre el cliente y el servidor sea segura (HTTPS) ver una copia de éste (httpd.conf - HTTPS.txt), situada en el mismo directorio en el que se encuentra este archivo.