ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Desarrollo de una interfaz Web para la gestión remota de la central telefonica ERICCSON MD110

AUTOR: Antonio Garcia Melgar
DIRECTOR: Francesc Burrull i Mestres

Noviembre de 2004

Autor	Antonio Garcia Melgar
E-mail del Autor	agmelgar@fn.mde.es
Director(es)	Francesc Burrull i mestres
E-mail del Director	francesc.burrull@upct.es
Título del PFC	Desarrollo de una interfaz Web para la gestión remota de la central MD110
Descriptores	PHP, MD110,Interfaz Web

Resúmen

Se pretende implementar un nuevo modo de programación remota de la central a través WWW. Para ello se utilizará un PC con puerto serie y conexión a Internet. La cara correspondiente a puerto serie se encargará de entregar comandos y leer respuestas de la central, mientras que la cara correspondiente a WWW deberá tener funciones de seguridad incorporadas (SSL) y la creación dinámica de contenido (PHP).

Titulación	Ingeniero Técnico de Telecomunicación, especialidad en Telemática
Intensificación	
Departamento	Departamento de Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	Noviembre de 2004

Índice

Capítulo 1
Introducción

1.1 Antecedentes

Las centrales de telefonía modernas, disponen de numerosas posibilidades de configuración y a su vez de variados métodos de introducir dicha configuración. En el caso que nos ocupa, la central ERICCSON MD-110, se trata de una moderna central de disposición distribuida, destinada a grandes compañías, que permite su configuración mediante:

- Línea telefónica desde centrales de soporte al cliente que establecen las grandes operadoras.

- Control mediante un dispositivo específico fabricado por ERICCSON conectado en Red de Área Local con la central.

- Mediante conexión por puerto serie (protocolo V24) desde PC, mediante un programa de configuración propietario instalado en el PC.

1.2 Situación

El redactor del proyecto desempeña su actividad profesional como responsable entre otras cosas, de la gestión de la central en cuestión, y los modos de configuración reseñados en el párrafo anterior se concretan en este caso de dos maneras:

- La primera y más habitual forma de configuración de la central se materializa mediante la modalidad de conexión por puerto serie a un PC, utilizando un programa desarrollado para entorno Windows por la compañía Telefónica de España SA. (TESA), este módulo de configuración es utilizado por el personal supervisor de la central.

- La segunda manera se realiza en caso de avería, cuando la complejidad de la misma excede la calificación técnica o de medios disponibles por el personal supervisor. Este módulo de configuración consiste en establecer un enlace telefónico con la central desde las instalaciones de la compañía TESA

Según lo expuesto, se deduce que el modo habitual de configuración de la central obliga a los supervisores a configurar la central en modo local, es decir, desde PC mediante conexión por puerto serie, esto unido a la gran cantidad de cambios de configuración requeridos cada día, por dar servicio a un gran numero de usuarios,

supone en la practica la presencia de un operador en el local donde se ubica la central.

1.3 Necesidad del proyecto

Para permitir la configuración no presencial por parte de operador surge la idea de buscar el método de configurar la central de manera remota, sin tener que permanecer en el local de la misma, permitiendo de esta manera realizar simultáneamente otras acciones de mantenimiento necesarias por parte del personal supervisor.

A lo anterior hay que unir que dentro de la compañía existe una red corporativa de área extensa WAN con un alto grado de ramificación, que proporcionará soporte a la aplicación a desarrollar.

Capítulo 2
Objetivos

El objetivo es lograr un sistema sencillo, robusto y seguro para realizar la configuración de una central MD-110 de manera remota aprovechando la infraestructura disponible de una red WAN.

Este sistema deberá cumplir los siguientes requisitos:

- No debe requerir un programa específico en el ordenador desde el cual se realiza la configuración, ni instalación adicional alguna.
- Debe cumplir los requisitos de seguridad necesarios para evitar la manipulación de la central por personas no autorizadas.
- Debe permitir una política de seguridad, permitiendo distintos niveles de acceso según el usuario.
- Debe permitir introducir cualquiera de los comandos de configuración disponibles para la central MD-110.
- Debe presentar la respuesta de la central, ante la introducción de los comandos.
- Debe permitir abrir y cerrar el enlace con la central de manera remota.
- Debe permitir la configuración desde cualquier estación de la red WAN.
- Dejar el código abierto a posibles mejoras futuras.

Capítulo 3
Solución propuesta

3.1 Descripción

Teniendo en cuenta la necesidad expuesta en el capítulo 1 y los requerimientos establecidos en el capítulo 2, se decide estudiar la posibilidad de confeccionar una aplicación Web, que permita la configuración de la central, de manera remota.

Esta decisión permite ya a priori, se cumplan algunos de los requisitos establecidos:

- No requiere una aplicación específica del lado remoto, ya que se puede utilizar un navegador Web, del tipo Internet Explorer o Netscape, por poner un ejemplo, disponible en la mayor parte de los equipos conectados a la red corporativa que da soporte al sistema.

- Por otro lado, esta solución permite que la configuración pueda realizarse desde cualquier emplazamiento de la red WAN permitiendo de esta manera centralizar la gestión de las múltiples centrales que componen la red de telefonía de la compañía, redundando en aumento de la eficiencia del sistema.

El esquema de la **Figura 3.1** permite visualizar la implementación del acceso remoto a la configuración de la central, propuesto en este proyecto.

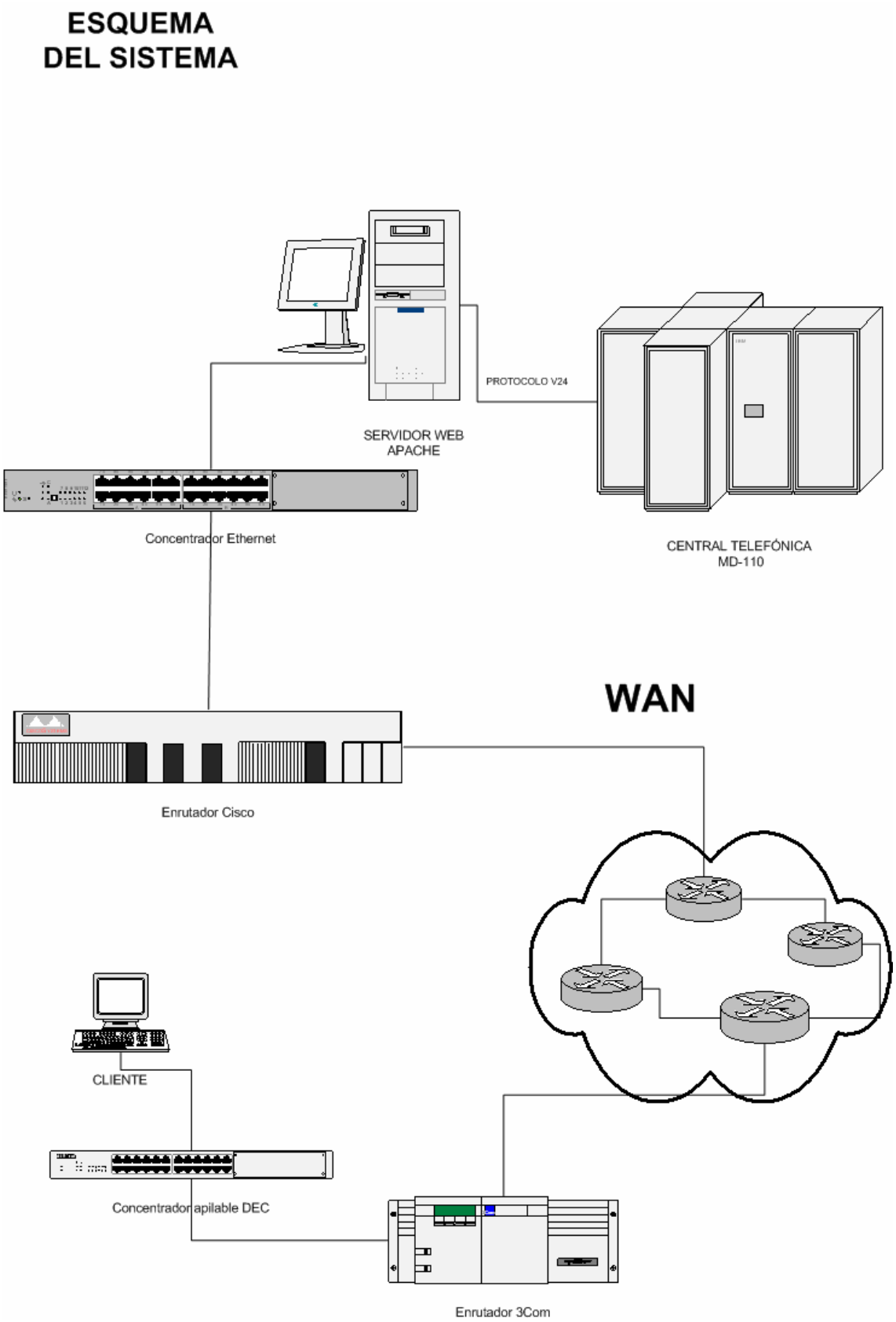


Figura 3.1 Esquema solución propuesta

3.2 Toma de decisiones.

Las siguientes decisiones a tomar son:

- Elección de la plataforma en la cual va a ejecutarse el servidor y la aplicación.

- Elección del servidor Web a utilizar en la plataforma en la cual se va a ejecutar la aplicación.

- Elección del lenguaje, y elección del modo de funcionamiento de la aplicación Web propiamente dicha.

- Diseño del sistema de seguridad a emplear, para evitar manipulaciones de la configuración de la central telefónica MD-110.

- Diseño del sistema de gestión de usuarios para dar distintos niveles de acceso a la central.

La primera decisión tomada es la de la elección de la plataforma, puesto que condiciona la toma de otras decisiones necesarias en el proceso de confección del sistema.

En este caso, se opta por el sistema operativo Linux sobre PC, por los siguientes motivos:

- El S.O. Linux permite establecer de manera muy clara los permisos de de los usuarios sobre los recursos del sistema, facilitando la gestión de la seguridad del sistema.

- El S.O. Linux gestiona todos los dispositivos del equipo, como si fueran ficheros facilitando enormemente, el manejo del puerto serie por parte de las aplicaciones.

- El S.O. Linux tiene una gran estabilidad de funcionamiento, permitiendo un funcionamiento ininterrumpido, con una gran fiabilidad.

- El S.O. Linux es de libre distribución, permitiendo un considerable ahorro económico, y es de código abierto, lo cual le da una mayor flexibilidad a la hora de adaptarlo a las necesidades.

La segunda decisión tomada es la de la elección del servidor Web a utilizar, sobre el cual se ejecutará nuestra aplicación Web.

En este caso, se opta por el servidor Web Apache 2.0, por los siguientes motivos:

- El servidor Web Apache, es de libre distribución permitiendo un considerable ahorro económico.

- El servidor Web Apache, está ampliamente probado y es de una gran fiabilidad como lo demuestra su alto nivel de implantación en Internet, con un porcentaje superior a cualquier otro modelo de servidores instalados.

- Esta además diseñado originalmente para ejecutarse en entrono Linux por lo que se adapta a la perfección a la plataforma elegida previamente.

La siguiente decisión está relacionada con el lenguaje, y elección del modo de funcionamiento de la aplicación Web propiamente dicha.

En cuanto a la elección del modo de funcionamiento, parece claro que debe utilizarse una aplicación que se ejecute del lado del servidor, puesto que esta debe de gestionar una central telefónica situada del lado del servidor, para realizar esto se pueden considerar varias opciones. En concreto existen dos métodos comúnmente usados:

- Utilizar el sistema de Common Gateway Interface (CGI, es una norma para establecer una comunicación entre un servidor Web y un programa, de tal modo que este último pueda interactuar con Internet. También se usa la palabra CGI para referirse al programa mismo, aunque lo correcto debería ser script) que utiliza un programa situado en el servidor, típicamente confeccionado en lenguaje **C** aunque no necesariamente y cuya salida estándar es un documento en código HTML que es servido al cliente.
- La otra opción es utilizar un lenguaje empotrado dentro del documento HTML, que es procesado por el servidor y una vez finalizada la ejecución se sirve al cliente, siendo el código del lenguaje empotrado transparente para el usuario.

De las dos posibilidades anteriores se ha optado por la segunda opción, utilizando lenguaje PHP, incluido como un módulo del servidor Apache, esta elección se produce por los siguientes motivos:

- Actualmente se ha producido un fuerte crecimiento en la utilización del lenguaje PHP, que permite la programación estilo procedimental, como el lenguaje **C**, y también permite el estilo de Programación Orientada a Objetos, dando así una gran flexibilidad al programador.

- Los desarrolladores del lenguaje PHP mantienen relación con los desarrolladores del servidor Apache, permitiendo una gran adaptación entre ambos productos.

- El Lenguaje PHP es de una gran sencillez de uso para los programadores noveles, que pueden realizar aplicaciones para Internet con muy poca experiencia, lo cual estimula a su aprendizaje y por otro lado permite a los programadores más avanzados confeccionar aplicaciones profesionales igualmente

- El manejo de ficheros, y por tanto del puerto serie, es muy sencillo en PHP, facilitando así la confección de la aplicación Web de la cual trata este proyecto.

En cuanto a la decisión de las medidas de seguridad a adoptar para evitar la manipulación de la central por parte de personal no autorizado, se decide utilizar los siguientes servicios de seguridad disponibles para el protocolo HTTP:

- El sistema de autentificación básica que proporciona el propio protocolo HTTP. Que permite que utilizando los encabezamientos adecuados se envíe al cliente una solicitud de Log-in que usado en combinación con las variables de entorno y otros mecanismos que describiremos con detalle en el Capítulo 5 de Implementación de la aplicación, permite restringir el acceso a la página Web, solo al personal autorizado.

- Por otro lado el sistema de seguridad debe permitir proteger el transvase de información entre el cliente y servidor tanto del nombre de usuario y password como de los diferentes comandos introducidos en la central. Para ello se utiliza el sistema Secure Socket Layer (SSL) que permite el enlace cifrado entre el cliente y servidor.

Finalmente en cuanto a la última decisión, relativa a la gestión de seguridad relativa a usuarios y niveles de acceso, inicialmente se tomó la decisión de realizarla mediante el uso de la base de datos relacional denominada MySql, de la cual se disponen funciones muy útiles en lenguaje PHP, pero una vez realizado un estudio de la documentación de la central MD-110, y puesto en contacto con el personal de mantenimiento técnico de la misma, se decide finalmente utilizar el propio sistema integrado en la central que permite almacenar los usuarios y su nivel de acceso. Aprovechando de esta manera los recursos que la propia central telefónica proporciona.

Así se facilita, la confección de la aplicación, que cuando debe decidir si da acceso a un usuario, únicamente necesita realizar el enlace con la central para comprobarlo.

3.3 **El concepto LAMPS**

Para el desarrollo de la aplicación y posterior puesta en funcionamiento se ha decidido utilizar un paquete software de libre distribución, disponible en Internet, que permite la instalación conjunta sobre Linux de:

- Apache
- MySQL
- PHP
- SSL

Este paquete facilita enormemente la instalación combinada de todas la anteriores aplicaciones respecto a la instalación de cada uno de dichos paquetes de software, compilando desde las fuentes y luego adaptarlos.

Capítulo 4
Implementación de la aplicación

4.1 Descripción

Se desarrolla una aplicación en lenguaje PHP, que resuelve los requerimientos establecidos en el capitulo 2, de acuerdo con la solución propuesta en el capítulo 3 del proyecto.

Antes de comenzar a desgranar el contenido de la implementación propuesta, en la **figura 4.1** se describe de manera gráfica y resumida la secuencia de funcionamiento normal.

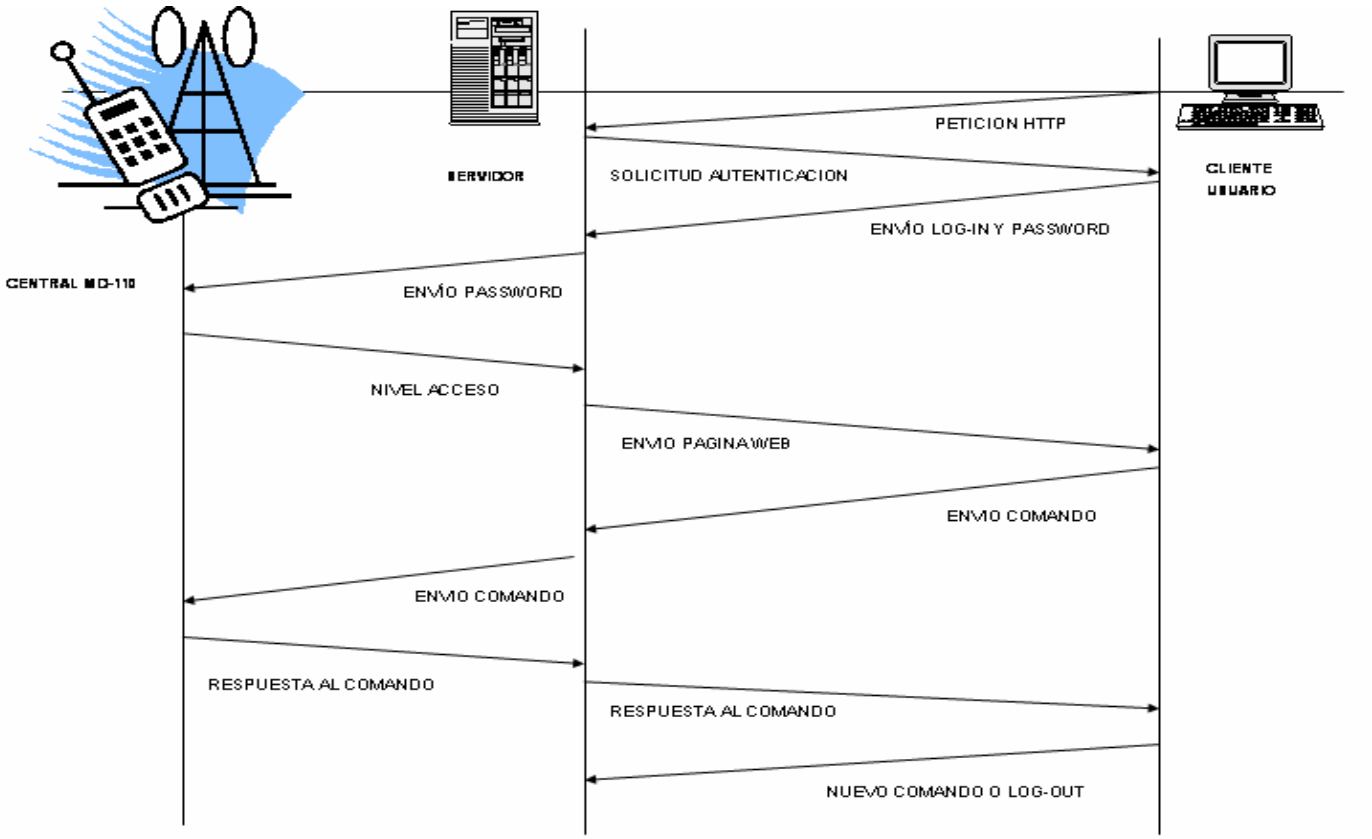


Figura 4.1 Secuencia de funcionamiento habitual de la aplicación.

4.2 Secuencia

La secuencia de la aplicación es la siguiente:

>>Las dos primeras líneas de la aplicación son las siguientes:

```
session_cache_limiter('nocache,private');
session_start();
```

Estas dos funciones están relacionadas con la sesión en PHP. Y se describen a continuación de acuerdo con el manual de PHP.

Descripción

string **session_cache_limiter** ([string limitador_del_cache])

session_cache_limiter() devuelve el nombre del limitador de caché actual. Si se especifica *limitador_del_cache*, el nombre del limitador de caché actual se cambia al nuevo valor.

El limitador de caché controla las cabeceras HTTP de control del caché enviadas al cliente. Estas cabeceras determinan las reglas por las que el contenido de la página puede ser guardado en el caché local del cliente. Cambiando el limitador de caché a *nocache*, impedirá cualquier tipo de almacenamiento en el caché por parte del cliente. Un valor de *public*, en cambio, permitiría el almacenamiento en el caché

Al comenzar la ejecución del script, el limitador de caché se reestablece al valor por defecto guardado en *session.cache_limiter*. De este modo, es necesario llamar a **session_cache_limiter()** en cada petición (y antes de llamar a **session_start()**)

Descripción

bool **session_start** (void)

session_start() crea una sesión (o la continúa basándose en el session id pasado por GET o mediante una cookie).

Esta función siempre devuelve **TRUE**.

Nota: Si está usando sesiones basadas en las cookies, debe llamar a **session_start()** antes de que haya ninguna salida al navegador.

>>A continuación se describe la función Authentícate(), la cual es llamada cada vez que deseamos enviar una solicitud de Log-in al cliente.

```
//funci?n que solicita la autenticaci?n al cliente*****************************

function authenticate() {
.       Header("WWW-Authenticate: Basic realm=AREA RESTRINGIDA");
.       Header("HTTP/1.0 401 Unauthorized");
.       echo 'AUTORIZACION REQUERIDA';//TEXTO QUE SE ENVIA SI PULSA CANCEL
.       exit;
}
```

Autentificación con PHP

Las características de autentificación HTTP en PHP solo están disponibles cuando se está ejecutando como un módulo en Apache y hasta ahora no lo están en la versión CGI.

En un script PHP como módulo de Apache, se puede usar la función **header()** para enviar un mensaje de "Autentificación requerida" al navegador cliente haciendo que muestre una ventana de entrada emergente con nombre de usuario y contraseña.

Una vez que el usuario ha rellenado el nombre y la contraseña, la URL que contiene el script PHP será llamada de nuevo con las variables predefinidas *PHP_AUTH_USER*, *PHP_AUTH_PW*, y *AUTH_TYPE* asignadas con el nombre de usuario, la contraseña y el tipo de autentificación respectivamente. Estas variables predefinidas se pueden encontrar en las matrices $_SERVER y *$HTTP_SERVER_VARS*.

Más adelante se podrá comprobar como se recupera de las variables predefinidas el loggin y el password del usuario.

Descripción

void **header** (string cadena [, bool reemplazar [, int cod_respuesta_http]])

La función **header()** es usada para enviar cabeceras HTTP puras. Para más información sobre las cabeceras HTTP se puede consultar la RFC 2616.

El parámetro opcional *reemplazar* indica si la cabecera debe reemplazar una cabecera previa semejante, o si debe agregar una segunda cabecera del mismo tipo. Por defecto esta función procede a reemplazar, pero si pasa **FALSE** como el segundo argumento, puede obligar a que se envíen múltiples cabeceras del mismo tipo.

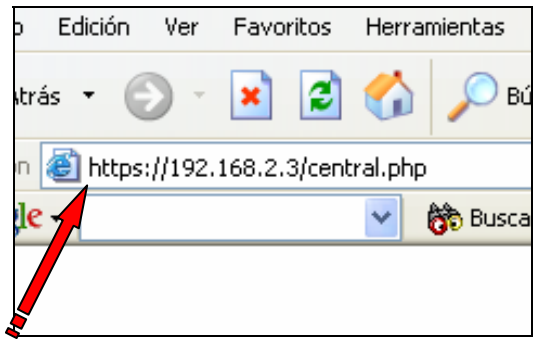
El segundo parámetro opcional, *cod_respuesta_http*, obliga a que el código de respuesta HTTP sea el valor especificado. (Este parámetro se encuentra disponible a partir de PHP 4.3.0.)

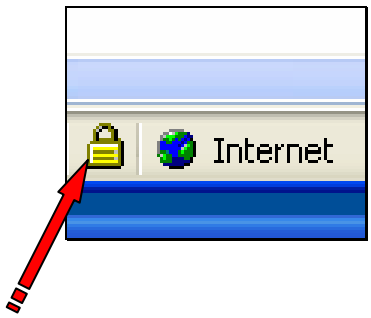
Existen dos llamadas de cabecera que son casos especiales. El primero es una cabecera que comience con la cadena "*HTTP/*" (no es importante la diferencia entre mayúsculas y minúsculas), la cual será usada para elegir el código de status HTTP a enviar.

El segundo caso es tratado cuando se expliquen las siguientes sentencias de código.

>>La siguiente sección de código corresponde a la comprobación de que se ha llamado a la página de configuración de la central utilizando el protocolo SSL, en caso negativo redirige la petición utilizando dicho protocolo.

```
//comprobacion de conexi?n segura SSl y en caso contrario redireccion **********

if (!isset($_SERVER['HTTPS']) || $_SERVER['HTTPS']!="on") {
        header("Location: https://$_SERVER[SERVER_NAME]".$_SERVER['REQUEST_URI']);
        exit;
}
```





isset

(PHP 3, PHP 4, PHP 5)

isset -- Determina si una variable está definida

Descripción

int **isset** (mixed var)

Devuelve **TRUE** si *var* existe; y **FALSE** en otro caso.

Si una variable ha sido destruida con **unset()**, ya no estará definida (no será **isset()**).

**

Variables Predefinidas
A partir de PHP 4.1.0, el método preferido para recuperar variables externas es mediante las superglobales mencionadas más adelante.

Variables de servidor: *$_SERVER*

Nota: Aparecieron en 4.1.0. En versiones anteriores, se utilizan *$HTTP_SERVER_VARS*.

$_SERVER es una matriz que contiene información tal como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz son creadas por el servidor web.

'*SERVER_NAME*'

El nombre del servidor anfitrión bajo el que está siendo ejecutado el script actual. Si el script está corriendo en un host virtual, éste será el valor definido para tal host virtual

'*REQUEST_URI*'

El URI que fue dado para acceder a esta página; por ejemplo, '*/index.html*'.

**

Este caso especial de la función header() es la cabecera "Location:". No solo envía esta cabecera de vuelta al navegador, sino que también devuelve un código de status *REDIRECT* (302) al navegador a menos que algún código de status *3xx* haya sido enviado ya.

Nota: HTTP/1.1 requiere una URI absoluta como argumento a Location: incluyendo el esquema, el nombre del host y una ruta absoluta, aunque algunos clientes aceptan URIs relativas. Usualmente puede usar *$_SERVER['HTTP_HOST']*, *$_SERVER['PHP_SELF']* y **dirname()** para construir una URI absoluta a partir de una relativa:

exit

(PHP 3, PHP 4, PHP 5)

exit -- Finaliza el script actual

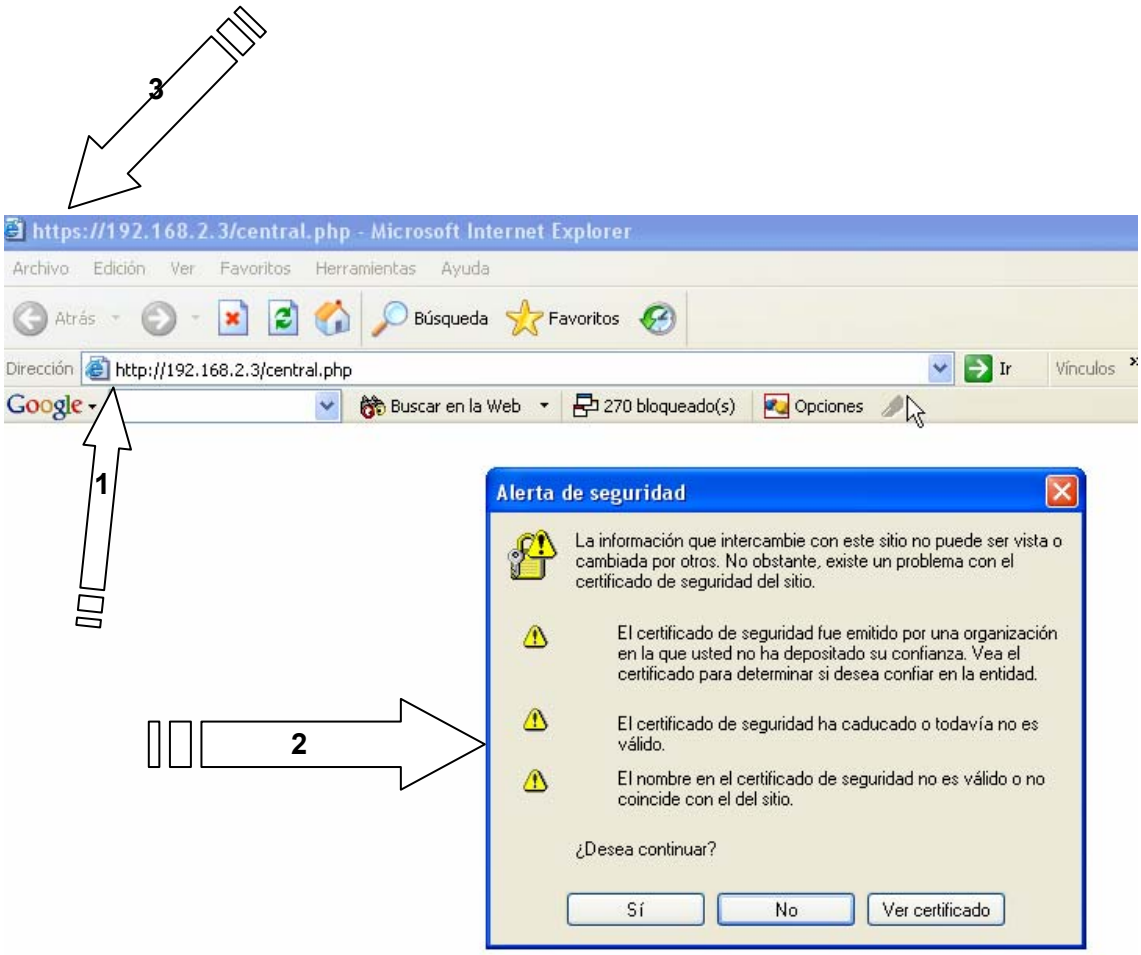
Descripción

void **exit** (void)

Esta construcción del lenguaje finaliza la ejecución del script. No devuelve nada.

Podemos describir con detalle la secuencia de esta redirección que consigue, que a pesar de que la página sea llamada siguiendo el protocolo HTTP esta es redirigida automáticamente mediante protocolo HTTPS obligando a enviar cifradas la clave, el nombre del usuario y todo el intercambio posterior de información.

1. En esta captura se puede observar que en la ventana de la dirección de la página se ha introducido, según el protocolo HTTP
2. A continuación aparece la ventana que indica que la comunicación va a ser cifrada a partir de ahora. (también indica el problema del certificado de seguridad)
3. finalmente se puede observar el la parte superior izquierda en la franja azul, que la página ya se está observando bajo el protocolo HTTPS hacia el cual ha sido dirigido por la propia aplicación.



>>La siguiente sección de código corresponde a la comprobación de que se ha llamado introducido el nombre de usuario y el password, en caso contrario llama a la función autentícate().

```
// comprueba autenticacion previa ***********************************************
// si no ha utilizado autenticacion basica llama a la funcion authenticate******

if ((!$_SERVER['PHP_AUTH_USER']) && (!$_SERVER['PHP_AUTH_PW'])) {
        $_SESSION['aceptado']="no";
        authenticate(); // envia cabecera de autenticacion
}
```

Variables de servidor: *$_SERVER*

'*PHP_AUTH_USER*'

Cuando se corre sobre Apache como módulo realizando autenticación HTTP, ésta variable es definida con el nombre de usuario definido por el cliente.

'*PHP_AUTH_PW*'

Cuando se corre sobre Apache como módulo realizando autenticación HTTP, ésta variable es definida con la contraseña entregada por el usuario.

Variables de sesión: *$_SESSION*

Nota: Introducidas en 4.1.0. En versiones anteriores, se usa *$HTTP_SESSION_VARS*.

Una matriz asociativa que contiene las variables de sesión disponibles en el script actual.

>>La siguiente sección de código corresponde a la comprobación de que se ha pulsado el botón de SALIR (Log-out) del formulario de la página de configuración de la central, en cuyo caso envía a la central el comando de fin de sesión y llama a la función authenticate(), forzando al posible nuevo utilizador a cerrar la ventana del navegador y volver a introducir su nombre y clave de usuario si quiere acceder a la página Web.

```
// accion si se pulsa el boton de salir ************************************

if (($_POST['USUARIO_ANTERIOR'] == 1) && ( $_POST['ANTIGUA_Aut'] ==
        $_SERVER['PHP_AUTH_USER'])) {
        $_SESSION['aceptado']="no";

        if($fout=fopen("/dev/ttyS0", 'a+')){
                $out="exit;";
                fwrite($fout, $out);
                fflush($fout);
                fclose($fout);
        }
        authenticate();
}
```

Variables HTTP POST: *$_POST*

Nota: Introducidas en 4.1.0. En versiones anteriores, se usa *$HTTP_POST_VARS*.

Una matriz asociativa de variables pasadas al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script. No necesita hacer **global $_POST;** para acceder a ella dentro de funciones o métodos, como se hace con *$HTTP_POST_VARS*.

**

Nota:

Tanto Netscape como Internet Explorer borrarán la caché de la ventana de autentificación en el navegador local después de recibir una respuesta 401 del servidor. Esto se está usando, para "desconectar" al usuario, forzándole a reintroducir su nombre y contraseña.

fopen

(PHP 3, PHP 4 , PHP 5)

fopen -- Abre un archivo o URL

Descripción

resource **fopen** (string nombre_archivo, string modo [, bool usar_ruta_inclusion [, resource contexto_z]])

fopen() asocia un recurso con nombre, especificado por *nombre_archivo*, a una secuencia.

Si PHP decide que *nombre_archivo* hace referencia a un archivo local, entonces intentará abrir una secuencia sobre ese archivo

El parámetro *modo* especifica el tipo de acceso que requiere para la secuencia. En este caso:

'a+'	Apertura para lectura y escritura; ubica el apuntador de archivo al final del mismo. Si el archivo no existe, intenta crearlo.

**

fwrite

(PHP 3, PHP 4 , PHP 5)

fwrite -- Escritura sobre archivos, segura con material binario

Descripción

int **fwrite** (resource gestor, string cadena [, int longitud])

fwrite() escribe los contenidos de *cadena* a la secuencia de archivo apuntada por *gestor*. Si el argumento *longitud* es entregado, la escritura se detendrá después de que *longitud* bytes hayan sido escritos, o al alcanzar el final de *cadena*, aquello que ocurra primero.

fwrite() devuelve el número de bytes escritos, o **FALSE** en caso de fallo.

**

fclose

(PHP 3, PHP 4 , PHP 5)

fclose -- Cierra el apuntador a un fichero abierto

Description

int **fclose** (int fp)

Se cierra el fichero apuntado por fp.

Devuelve **TRUE** en caso de éxito y **FALSE** en caso de fallo.

El apuntador al fichero debe ser válido y debe apuntarse a un fichero abierto con éxito con **fopen()** o con **fsockopen()**.

fflush

(PHP 4 >= 4.0.1, PHP 5)

fflush -- Vacía la salida hacia un archivo

Descripción

bool **fflush** (resource gestor)

Esta función obliga a que se produzca la escritura de la salida acumulada en búfer al recurso apuntado por el gestor de archivo *gestor*. Devuelve **TRUE** si todo se llevó a cabo correctamente, **FALSE** en caso de fallo.

El puntero de fichero debe de ser valido y debe de apuntar a un fichero abierto con éxito por **fopen()** o **fsockopen()**.

>>La siguiente sección de código corresponde a la comprobación de que se ha superado previamente el proceso de acceso a la central telefónica (Log-in), en cuyo caso no se repetirá el procedimiento de entrada. Si por el contrario no se ha superado el proceso de acceso a la central se forzará a la aplicación a realizar el proceso de Log-in.

```
//comprobaci?n si la sesion est? iniciada****************************************

if(($_SESSION['aceptado'])!="si"){
```

>>La siguiente sección de código corresponde a la primera acción del proceso de acceso a la central telefónica (Log-in), que es la recuperación de la clave introducida por el usuario por medio de la variable indicada.

```
        $passw=$_SERVER['PHP_AUTH_PW'];
```

>>La siguiente sección de código corresponde al envío a la central de una secuencia de caracteres (fijada en la documentación técnica de la central) al objeto de realizar el sincronismo, la central que detecta los caracteres, se adapta a la velocidad a la cual recibe los mismos desde el servidor Web. Si falla el proceso reinicia la conexión con el cliente volviéndole a solicitar que introduzca su nombre de usuario y su clave.

```
//envio sincronismo a la central*************************************************

        if($fdlog=fopen("/dev/ttyS0", 'a+')){
                $sincro="sssssssss;";
                if($plog=fwrite($fdlog, $sincro)){
                        fflush($fdlog);
                }
                else{
                        authenticate();
                }
```

>>La siguiente sección de código corresponde al proceso de escucha del puerto serie a la espera del siguiente mensaje enviado por la central, según el procedimiento de Log-in de la misma.

Para realizar el proceso de lectura del puerto serie se ha tenido que utilizar la función stream_select(), que permite detectar el momento en el cual se está recibiendo un carácter por dicho puerto, y permite también establecer un tiempo máximo de espera timeout, para poder gestionar convenientemente la recepción de información de la central.

Para establecer el tiempo se ha utilizado el método de prueba y error, por el cual se ha ido aproximando el valor hasta conseguir la completa recepción de caracteres procedentes del puerto serie.

Este sistema de recepción se utiliza de manera sistemática en cada una de las ocasiones en que se considera necesario recibir información de la central, por lo que no se volverá a describir.

```
//proceso de escucha del puerto serie ****************************************

                $leclog=array($fdlog);
                while(stream_select($leclog, $esclog=NULL,$exclog=NULL, 2, 20000)){
                        $datalog.=fread($fdlog, 1);
                        if(strlen($datalog)==0){
                                break;
                        }
                }
```

stream_select

(PHP 4 >= 4.3.0, PHP 5)

stream_select -- Ejecuta el equivalente al llamado de sistema select() en la matriz de secuencias dada, con un tiempo de espera especificado por tv_sec y tv_usec

Descripción

int **stream_select** (array &lectura, array &escritura, array &excepcional, int tv_sec [, int tv_usec])

La función **stream_select()** acepta una matriz de secuencias y espera a que éstas cambien su status. Su operación es equivalente a la de la función **socket_select()**, excepto que actúa sobre secuencias.

Las secuencias listadas en la matriz *lectura* serán vigiladas para ver si aparecen caracteres disponibles para lectura (o más precisamente, para ver si una operación de lectura no producirá un bloqueo - en particular, un recurso de secuencia se encuentra listo también al llegar al final del archivo, en cuyo caso un llamado a **fread()** devolverá una cadena de longitud cero).

Las secuencias listadas en la matriz *escritura* serán vigiladas para ver si una escritura no crea bloqueos.

Las secuencias listadas en la matriz *excepcional* serán vigiladas por la llegada de datos excepcionales ("out-of-band") de alta prioridad.

Nota: Cuando **stream_select()** devuelve un valor, las matrices *lectura*, *escritura* y *excepcional* son modificadas para indicar cuáles recursos de secuencia modificaron su status en realidad.

Los parámetros *tv_sec* y *tv_usec*, juntos forman el parámetro *tiempo de espera*, *tv_sec* especifica el número de segundos, mientras que *tv_usec* el número de microsegundos. El *tiempo de espera* es un límite superior sobre la cantidad de tiempo que **stream_select()** esperará antes de devolver un valor. Si tanto *tv_sec* como *tv_usec* son definidos como *0*, **stream_select()** no esperará por datos - en su lugar devolverá un valor inmediatamente, indicando el status actual de las secuencias. Si *tv_sec* es **NULL stream_select()** puede crear un bloqueo indefinidamente, y sólo devolverá un valor cuando ocurra un evento en alguna de las secuencias vigiladas (o si una señal interrumpe el llamado de sistema).

En caso de éxito, **stream_select()** devuelve el número de recursos de secuencia modificados contenidos en las matrices, que puede ser cero si el tiempo de espera expira antes de que algo interesante suceda. En caso de fallo, el valor **FALSE** es devuelto y se genera una advertencia (esto puede pasar si el llamado de sistema es interrumpido por una señal entrante).

Nota
El uso de un valor de tiempo de espera de *0* permite consultar el status de las secuencias de forma instantánea, sin embargo, NO es buena idea usar un valor de tiempo de espera de *0* en un ciclo, dado que causará que el script consuma mucho tiempo de CPU. Es mucho mejor especificar un valor de tiempo de espera de algunos pocos segundos, aunque si necesita hacer chequeos y ejecutar otro segmento de código concurrentemente, usar un valor de tiempo de espera de por lo menos *200000* microsegundos ayudará a reducir el uso de CPU del script.

El valor de tiempo de espera es el tiempo máximo que transcurrirá; **stream_select()** devolverá un valor tan pronto como las secuencias solicitadas se encuentren listas para su uso.

No se necesita pasar cada matriz a **stream_select()**. Se puede dejar de especificarlas y usar una matriz vacía o **NULL** en su lugar. Estas matrices son pasadas *por referencia* y serán modificadas después de que **stream_select()** devuelva un valor.

**

strlen

(PHP 3, PHP 4 , PHP 5)

strlen -- Obtiene la longitud de la cadena

Descripción

int **strlen** (string cad)

Devuelve la longitud de la *cadena*.

>>La siguiente sección de código se fuerza a la aplicación a comprobar si lo recibido desde la central corresponde a su secuencia estándar, en caso contrario fuerza la salida de la página Web.

```
//si falla secuencia login solicita nueva autenticacion************************

        if(!($datalog=="ENTER PASSWORD")){
                unset($_SERVER['PHP_AUTH_PW'],$PHP_AUTH_PW);
                unset($_SERVER['PHP_AUTH_USER'],$PHP_AUTH_USER);
                $_SESSION['aceptado']="no";
                authenticate();
        }
```

session_unset

(PHP 4 , PHP 5)

session_unset -- Elimina todas las variables de la sesión

Descripción

void **session_unset** (void)

La función **session_unset()** elimina y libera el espacio ocupado por todas las variables de la sesión actual registradas

>>La siguiente sección de código envía la clave del usuario a la central para verificar si tiene acceso la central.

```
//envia PASSWORD a la central*****************************************************

        if($plog2=fwrite($fdlog, $passw)){
                fflush($fdlog);
        }
```

>>La siguiente sección de código establece una variable de sesión en un valor, de tal manera que durante la sesión del usuario en cuestión, no se repita el proceso de Log-in una vez que ya ha sido autorizado. Al llegar a este punto finaliza el procedimiento de Log-in de la central.

```
//una vez completado el proceso de autorizacion por la central*****************

        $_SESSION['aceptado']="si";
        fclose($fdlog);
    }

}//*****************fin proceso login central***********************************
```

>>La siguiente sección de código envía el comando que el usuario autorizado ha enviado a la central, en caso de que no se envíe ninguno aparecerá el aviso indicado en la ventana de respuesta de la central.

```
//escritura puerto serie (envio de comando a la central)************************

$comando=$_POST['comando'];
if($fd=fopen("/dev/ttyS0", 'a+')){
    if($p=fwrite($fd, $comando)){
        fflush($fd);
    }
    else{
        $data="no se recibio comando en la central";
    }
```

>>La siguiente sección de código recibe la respuesta de la central al comando el usuario autorizado ha enviado a la central.

```
//escucha del puerto serie (respuesta de la central, al comando)***************

        $lec=array($fd);
        while(stream_select($lec, $esc=NULL,$exc=NULL, 2, 20000)){
                $data.=fread($fd, 1);
                if(strlen($data)==0){
                        break;
                }
        }
```

>>La siguiente sección de código cierra el puerto serie durante esta ejecución del script. También se muestra el aviso enviado al usuario en el caso de que el último comando enviado no hubiera podido ser enviado a la central por que no se pudo abrir el puerto serie.

```
//se cierra el fichero ttyS0 (pto serie COM1)**********************************

        fclose($fd);
}
else{
        $data="no se pudo enlazar con la central para envio de comando";
}
//*****************************************************************************
?>
```

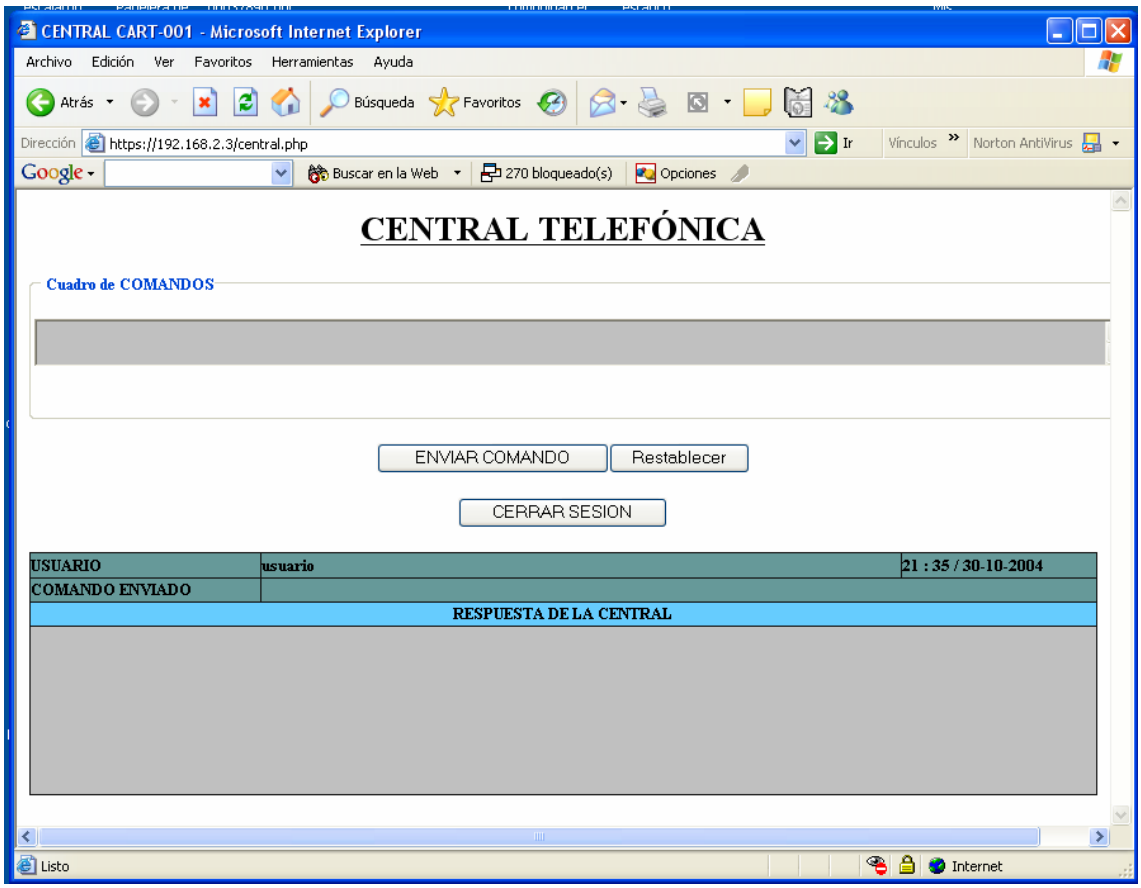
>>Una vez finalizada casi en su totalidad la parte de la página Web correspondiente al script PHP comienza ahora, la parte correspondiente a presentación de la información al cliente, incluyendo los formularios situados en la página. En concreto en la parte de código que se describe, se implementa en código HTML el formulario para escritura de comandos, por parte del usuario autorizado. Significando que esta página no sería descargada en el cliente en caso de no superar el proceso de Log-in de la central.

```
<!-- ********************comienza presentacion***************************** -->
<html>


<head>
<title>CENTRAL CART-001</title>
</head>

<body>

<!-- ********************formulario de entrada de comandos***************** -->

<p align="center"><u><font size="6"><b>CENTRAL TELEF?NICA</b></font></u></p>
<form method="POST" action="central.php">
<fieldset style="padding: 2; width:883; height:116">
  <legend><b>Cuadro de COMANDOS</b></legend>
  <p>
  <textarea rows="2" name="comando" cols="105" style="background-color: #C0C0C0">
        </textarea></p>
        </fieldset><p align="center">
        <input type="submit" value="ENVIAR COMANDO" name="B1">
        <input type="reset" value="Restablecer" name="B3"></p>
</form>


<div align="center">
<center>
```

Variables externas a PHP

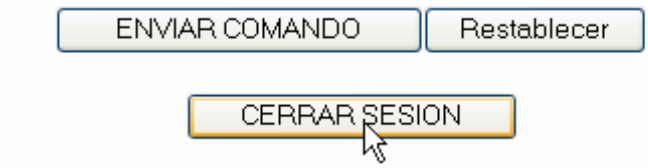
Formularios HTML (GET y POST)

Cuando se envía un formulario a un script PHP, las variables de dicho formulario pasan a estar automáticamente disponibles en el script gracias a PHP.

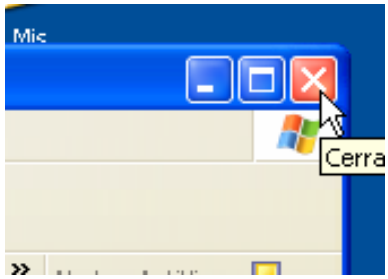


>>La siguiente sección de código corresponde al formulario del botón CERRAR SESIÓN que permite utilizar las características de autentificación de PHP para finalizar el acceso a la página..

```
<!-- **********************formulario de logout ****************************** -->

<p>
<form method="POST" action="central.php">
<input type='hidden' name='USUARIO_ANTERIOR' value='1'>
<?php
echo "<input type='hidden' name='ANTIGUA_Aut'
value='{$_SERVER['PHP_AUTH_USER']}'>\n";
?>
<input type="submit" value="CERRAR SESION" name="B4"></p>
</form>
 </p>
</center>
</div>
```



FINALMENTE EL USUARIO SE VE OBLIGADO A CERRAR EL NAVEGADOR SI QUIERE VOLVER A ACCEDER A LA PÁGINA DE CONFIGURACIÓN.



>>La siguiente sección de código corresponde al formato establecido para la presentación de los distintos datos recogidos en el script PHP, tanto la respuesta recibida de la central, cada vez que se procesa un comando, como el último comando enviado y la hora del envío del mismo.

```
<!-- **********************formato de tabla********************************** -->
<!-- **********************da forma a la salida de la central*************** -->


<div align="left">
  <table border="1" cellpadding="0" cellspacing="0"
  style="border-collapse: collapse" bordercolor="#111111" id="AutoNumber1"
  width="100%" height="100">
    <tr>
      <td width="186" height="19" bgcolor="#669999"><b>USUARIO</b></td>
      <td width="526" height="19" bgcolor="#669999">
        <?php echo "<b>{$_SERVER['PHP_AUTH_USER']}</b>"; ?></td>
      <td width="159" height="19" bgcolor="#669999"><b>
      <? echo  date("H : i "),"  /  ",date("j-n-Y")?></b></td>
    </tr>
    <tr>
      <td width="186" height="19" bgcolor="#669999"><b>COMANDO ENVIADO</b></td>
      <td width="682" colspan="2" height="19" bgcolor="#669999">
        <?php echo "<b>$comando</b>";?></td>
    </tr>
    <tr>
      <td width="868" colspan="3" height="19" bgcolor="#66CCFF">
      <p align="center"><b>RESPUESTA DE LA CENTRAL</b></td>
    </tr>
    <tr>
      <td width="868" colspan="3" bgcolor="#C0C0C0" height="133" align="left"
        valign="top"><?php echo "<b>$data</b>"; ?></td>
    </tr>
  </table>
</div>

</body>

</html>
```

Capítulo 5
Conclusiones y posible extensión

5.1 Conclusión

Una vez expuesto, el proyecto se considera que el resultado satisface la necesidad original de lograr un sistema de configuración remota de la central telefónica MD-110.

Además se consideran cumplidos los requerimientos establecidos para el sistema, principalmente por la facilidad que supone contar con una infraestructura de red corporativa y por supuesto, gracias a la potencia que confiere el uso combinado de las herramientas utilizadas LINUX+APACHE+PHP+SSL.

5.2 Beneficios del proyecto

En la práctica la realización del proyecto consigue facilitar la gestión de los recursos disponibles y lo hace de manera eficiente, ya que aprovecha los recursos que ya estaban disponibles sin apenas gasto, ya que no se producen:

- Costes adicionales en Hardware, ya que se aprovecha la infraestructura de telecomunicaciones de la red corporativa. Sin necesidad de comprar equipos específicos para esa función.
- Costes adicionales en software, ya que se ha utilizado software de libre distribución.

Por otro lado se ha trabajado con código abierto, lo cual permite futuras mejoras en el software sin dependencias externas, de aplicaciones propietarias.

La estructura de la aplicación diseñada permite, sin gran esfuerzo ser reutilizada para otras aplicaciones, ya que consiste en un interfaz Web entre un usuario remoto y una maquina conectada mediante puerto serie (protocolo V24).

De esta manera se podría variar su uso para gestionar otras máquinas conectadas mediante puerto serie, sin más que acomodar el procedimiento de Log-in, al protocolo propio de la máquina en cuestión.

También se considera relativamente sencillo el variar el código para adaptarlo a modelo de interconexión existente entre el servidor Web y la máquina a controlar, gracias a la facilidad que proporciona Linux de interactuar con los dispositivos de entrada y salida, al tratarlos como ficheros.

Aunque se pueda considerar como algo secundario dentro de los objetivos del proyecto, se considera muy positivo el haber podido trabajar con las herramientas LINUX+APACHE+PHP+SSL antes mencionadas.

Ello ha permitido al autor conocer y desarrollar habilidades en un campo que está de plena actualidad, las aplicaciones Web. Estas permiten que Internet sea algo más que un conjunto de documentos de acceso generalizado. Convirtiéndose en un atractivo campo donde se ejecutan aplicaciones que permiten interactividad y ofrecen un espectro muy amplio de oportunidades de trabajo.

5.3 Extensiones

Se considera interesante la posibilidad de poder efectuar un trabajo de extensión del proyecto, que consista en la realización de un completo interfaz gráfico adaptado a la maquina a ser configurada en cuestión.

Dicho interfaz podría ser “idéntico” desde el punto de vista del operador, al interfaz propietario disponible para la central, consiguiendo de esta manera evitar tiempo de formación

Esta máquina, puede tratarse de la propia central telefónica MD-110, cuya complejidad permite dar contenido por si solo a un completo proyecto, o puede tratarse de otro tipo de dispositivo cuyo control y configuración remoto se considere de interés.

Anexos

Anexo A
Manual de usuario

A.1 Instalación del sistema

La aplicación ha sido programada en el lenguaje PHP. Partiendo de la base de que disponemos de un punto de conexión a la red corporativa, a continuación se describen los elementos necesarios para utilizar el sistema:

A.1.a **Plataforma**

La siguiente configuración corresponde a la plataforma sobre la cual ha sido desarrollado y probado el sistema, lo cual no excluye que pueda ser utilizada una plataforma distinta de la aquí descrita, sobre la cual funcione correctamente.

El sistema operativo utilizado es el S.O. LINUX. Lo cual se considera un requerimiento imprescindible para el correcto funcionamiento de la aplicación, ya que esta utiliza características propias del mismo dentro del código.

Existen diversas distribuciones del mismo, en este caso se ha utilizado:

SUSE LINUX Professional 9.0

El anterior sistema operativo se encuentra disponible en la siguiente dirección:

http://www.suse.de/es/private/download/index.html

Para un funcionamiento correcto de SUSE LINUX Professional 9.0 se deben cumplir los siguientes requisitos:

- Procesador: Pentium® 1-4; AMD® Duron, Athlon™, Athlon XP, Athlon MP o Athlon 64; Intel® Celeron o EMT 64
- Memoria principal: Al menos 128 MB de RAM; se recomienda 256 MB
- Disco duro: Al menos 500 MB; se recomienda 2,5 GB para un sistema estándar

Además el sistema debe contar con el siguiente hardware:

- Tarjeta de red Ethernet
- Puerto serie (RS-232, V24)

A.2.b **Software**

Una vez dispongamos de la plataforma adecuada, de acuerdo con los requisitos descritos en el párrafo anterior, es necesario instalar cierto software para el correcto funcionamiento del sistema.

Lo primero es disponer de un servidor Web, este debe de ser capaz de servir documentos que contengan scripts PHP y además debe de implementar el protocolo SSL, de acuerdo con las especificaciones de la solución propuesta en el Capitulo 3.

Lo anterior se puede obtener e instalar de distintas maneras. En esta caso se ha desarrollado y probado utilizando un único paquete que contenía todos el software necesario:

XAMPP Linux 1.4.9a

Cuyo contenido es el siguiente:

Apache 2.0.52, MySQL 4.0.21, PHP 5.0.2 & 4.3.9 & PEAR + SQLite 2.8.9 + multibyte (mbstring) support, Perl 5.8.4, ProFTPD 1.2.10, phpMyAdmin 2.6.0-pl1, OpenSSL 0.9.7d, GD 2.0.1, Freetype2 2.1.7, libjpeg 6b, libpng 1.2.7, gdbm 1.8.0, zlib 1.1.4, expat 1.2, Sablotron 1.0, libxml 2.4.26, Ming 0.2a, Webalizer 2.01, pdf class 009e, ncurses 5.8, mod_perl 1.99_13, FreeTDS 0.62.4, gettext 0.11.5, IMAP C-Client 2002b, OpenLDAP (client) 2.2.13, mcrypt 2.5.7, mhash 0.8.18, Turck MMCache 2.4.6, cURL 7.10.7, libxslt 1.1.8, phpSQLiteAdmin 0.2

EL paquete de software anterior está disponible en la siguiente dirección:

http://www.apachefriends.org/en/xampp-linux.html

Una vez descargado el paquete software anterior procede su instalación siguiendo el siguiente procedimiento:

Una vez descargado el fichero es necesario disponer de los permisos de administrador del sistema (root), abrir un terminal de Linux y descomprimir el fichero descargado en el subdirectorio /opt tecleando el siguiente comando:

tar xvfz xampp-linux-1.4.9a.tar.gz -C /opt

Una vez realizado lo anterior el paquete ya está instalado solo resta ya arrancar la ejecución del mismo tecleando el siguiente comando:

/opt/lampp/lampp start

A continuación se debe observar algo así como esto en pantalla:

Starting XAMPP 1.4.9a...

LAMPP: Starting Apache...

LAMPP: Starting MySQL...

LAMPP started.

Entonces Apache ya se encontrará en funcionamiento con capacidad para interpretar documentos programados en PHP.

A continuación hay que poner en marcha la capacidad del servidor de implementar el protocolo SSL, que es el que proporciona el cifrado de los enlaces. Para ello debemos teclear el siguiente comando:

/opt/lampp/lampp startssl

Al teclear lo anterior la capacidad SSL se activa de manera permanente, cada vez que se arranque el servidor Web.

Para parar el servidor tecleamos:

/opt/lampp/lampp stop

Y para desinstalar el paquete software tan solo es necesario teclear:

rm -rf /opt/lampp

Una vez instalado el software y puesto en funcionamiento el servidor solo nos queda copiar el fichero de la aplicación desarrollada para el proyecto **central.php** en el siguiente subdirectorio para ser servido por el servidor:

/opt/lampp/htdocs/

A.1.c **Últimos ajustes**

Antes de poner en funcionamiento el sistema es necesario realizar los siguientes ajustes:

- Modificar los permisos del fichero /dev/ttyS0

  Para ello utilizaremos el comando

  Chmod 777 /dev/ttyS0

- Crear un fichero de texto llamándole serie para la configuración del puerto serie con el siguiente contenido:

  set modem type none

  set line /dev/ttyS0

  set carrier-watch off

  set speed 9600

  set stop-bits 1

  connect

- Abrir el puerto serie mediante la herramienta kermit disponible en el Sistema Operativo Linux, utilizando el archivo de configuración del puerto serie recién creado, mediante el siguiente comando:

  kermit + serie

- Finalmente conectaremos el PC a la red corporativa utilizando un latiguillo con conector RJ45 a la tarjeta de red ETHERNET.

A.2 Instrucciones de uso

Una vez efectuados los pasos descritos en el apartado de instalación, disponemos ya del sistema configurado y podemos utilizarlo de acuerdo con las instrucciones de uso que a continuación se describen.

Para poder acceder a la página Web de configuración de la central debemos contar con la debida autorización del supervisor de la central que nos proporcionará la clave de usuario.

Es misión del supervisor de la central administrar la lista de usuarios y otorgar los debidos permisos de acceso. Esta lista se introducirá en la central utilizando los debidos comandos de configuración.

Una vez dispongamos de la clave de usuario, necesitamos conocer la dirección IP del servidor en el cual reside la aplicación. Esta dirección la utilizaremos junto con el nombre del programa central.php para acceder a la página Web introduciéndola en la ventana correspondiente del navegador Web utilizado.

En la figura A.1 se muestra un ejemplo de lo dicho, para el caso de utilizar el navegador Internet Explorer y para la dirección IP 192.168.2.3

Figura A.1

El PC sobre el cual se ejecuta el servidor Web dispondrá de una dirección IP asignada de manera automática mediante DHCP, y dependerá de la política de direccionamiento de la compañía.

Una de las maneras que podemos utilizar para conocer la IP del servidor es teclear el siguiente comando desde un terminal abierto en el servidor:

ip –r addr

Una vez introducido el URL correspondiente en la ventana del navegador, aparecerá la ventana que solicita la introducción del nombre y la clave de usuario, tal como se muestra en la figura A.2

Figura A.2

Una vez introducido el nombre de usuario y la clave, se debe pulsar el botón de aceptar con el ratón o bien pulsar la tecla ENTER.

En caso de pulsar la tecla cancelar, no se podría acceder a la página Web de configuración y se mostraría la pantalla de la figura A.3

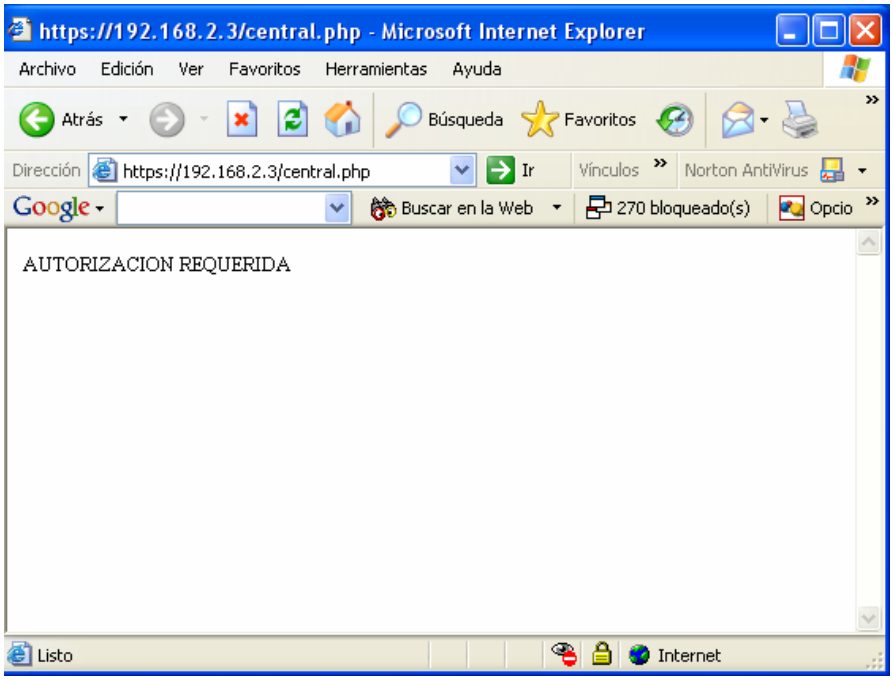


Figura A.3

A partir de ese momento se iniciarán los procesos de Log-in en la central, y una vez finalizados si se comprueba que el usuario tiene acceso aparecería la pantalla que se muestra en la Figura A.4 y en caso de no tener acceso se volvería a presentar la solicitud de nombre de usuario y clave tal como se muestra en la figura A.2

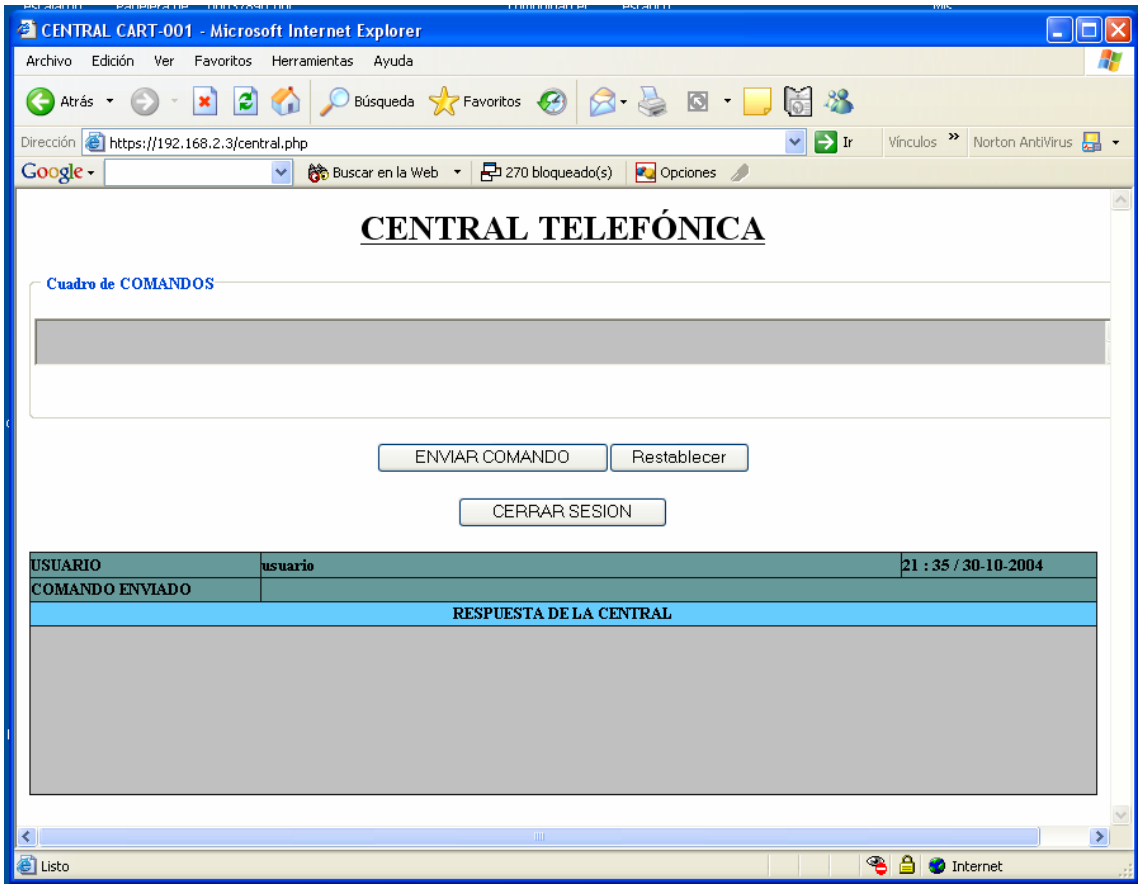


Figura A.4

Esta pantalla mostrada en la figura A.4 constituye la página de configuración de la central. En ella podemos distinguir dos partes bien diferenciadas.

Entradas de usuario:

Esta parte corresponde al lugar donde el usuario dispone de una ventana de texto en la cual puede escribir los comandos enviados a la central . La lista de comandos de la central se encuentra disponible al final del manual de usuario. Esta zona de la pantalla se muestra en la figura A.5

Figura A.5

Una vez introducido el comando en la ventana de texto, es necesario pulsar el botón de ENVIAR COMANDO mediante el ratón. Tal como se muestra en la figura A.6

Figura A.6

En caso de que nos equivocáramos a la hora de escribir el comando podemos corregirlo utilizando las teclas habituales del teclado o en caso de querer rescribir todo el comando se pulsaría la tecla Restablecer que también aparece en la figura A.7

Figura A.7

Finalmente en esta zona de la pantalla tenemos la oportunidad de cerrar la sesión. Tal como se muestra en la figura A.7

Figura A.7

Esto último es algo muy importante de cara a la seguridad, tal como explicamos a continuación, no debemos cerrar sin más la ventana tal como se muestra en la figura A.8

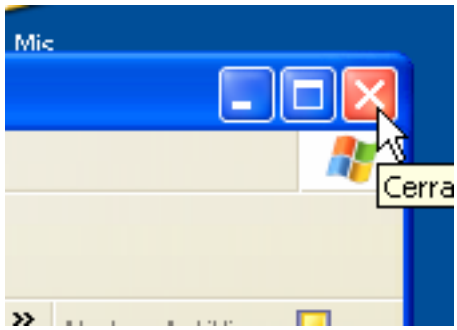


Figura A.8

Tampoco se debe seguir navegando sin más, utilizando la misma ventana después de haber accedido a la central pues en caso de que nos ausentáramos del PC en el cual estemos trabajando, otra persona podría acceder a la misma sin más que volver a la pantalla de configuración pulsando la flecha de retroceso del navegador.

La central dispone de un sistema de temporización, transcurrido el cual, si no se ha introducido ningún comando se desconecta no pudiendo acceder de nuevo a ella hasta que no se repita el proceso de Log-in. Pero si sucede el caso descrito en el párrafo anterior y no ha transcurrido dicho tiempo podría producirse un acceso indeseado.

Es por eso, que el usuario autorizado, debe pulsar la tecla CERRAR SESIÓN una vez que finalice su trabajo en la central.

Una vez pulsado este botón aparecerá de nuevo la ventana de solicitud de nombre de usuario y de clave mostrada en la figura A.2 pero no podrá acceder el usuario a la página Web de configuración aunque introduzca un nombre y clave autorizados hasta que no cierre la ventana de ese navegador y vuelva a realizar la operación desde el principio abriendo una nueva ventana de navegador.

Entradas de usuario:

Una vez descrita la zona de entradas del usuario se va a describir a continuación, la zona destinada a presentar al usuario, la información correspondiente a la sesión en curso. Esta zona se muestra en la figura A.9

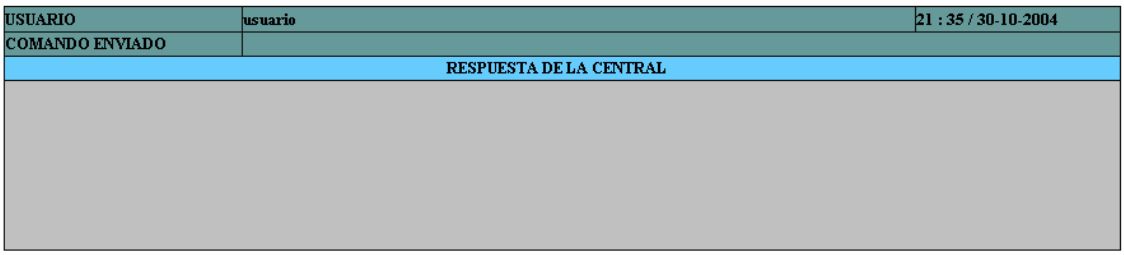


Figura A.9

Dentro de esta zona de la pantalla se cuenta con una parte en la cual se muestra el usuario que está trabajando en la central. Así como el último comando enviado por el usuario hacia la central. Tal como se muestra en la figura A.10

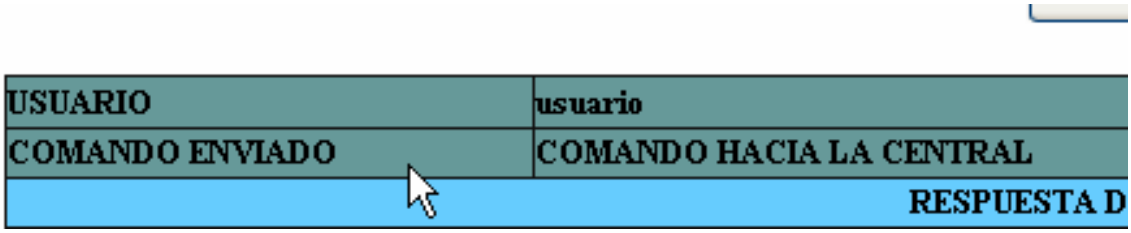


Figura A.10

También dentro de esta zona aparece una parte donde se muestra la hora del último comando enviado a la central. Tal como se muestra en la figura A.11

Figura A.11

Finalmente disponemos de una zona en la cual se muestra la respuesta recibida de la central correspondiente al último comando introducido por el usuario así como otro tipo de información que se quiera presentar al usuario, correspondiente a eventos sucedidos durante la ejecución del ejecución del script en PHP. Tal como se muestra en la figura A.12

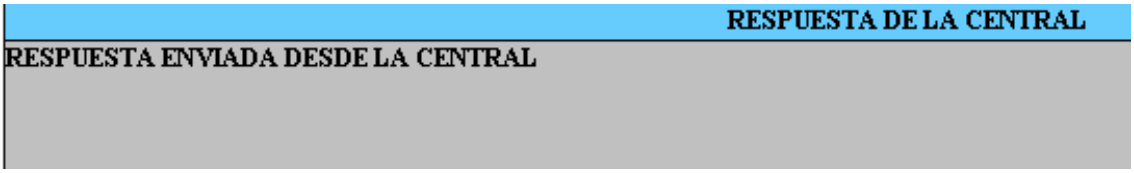


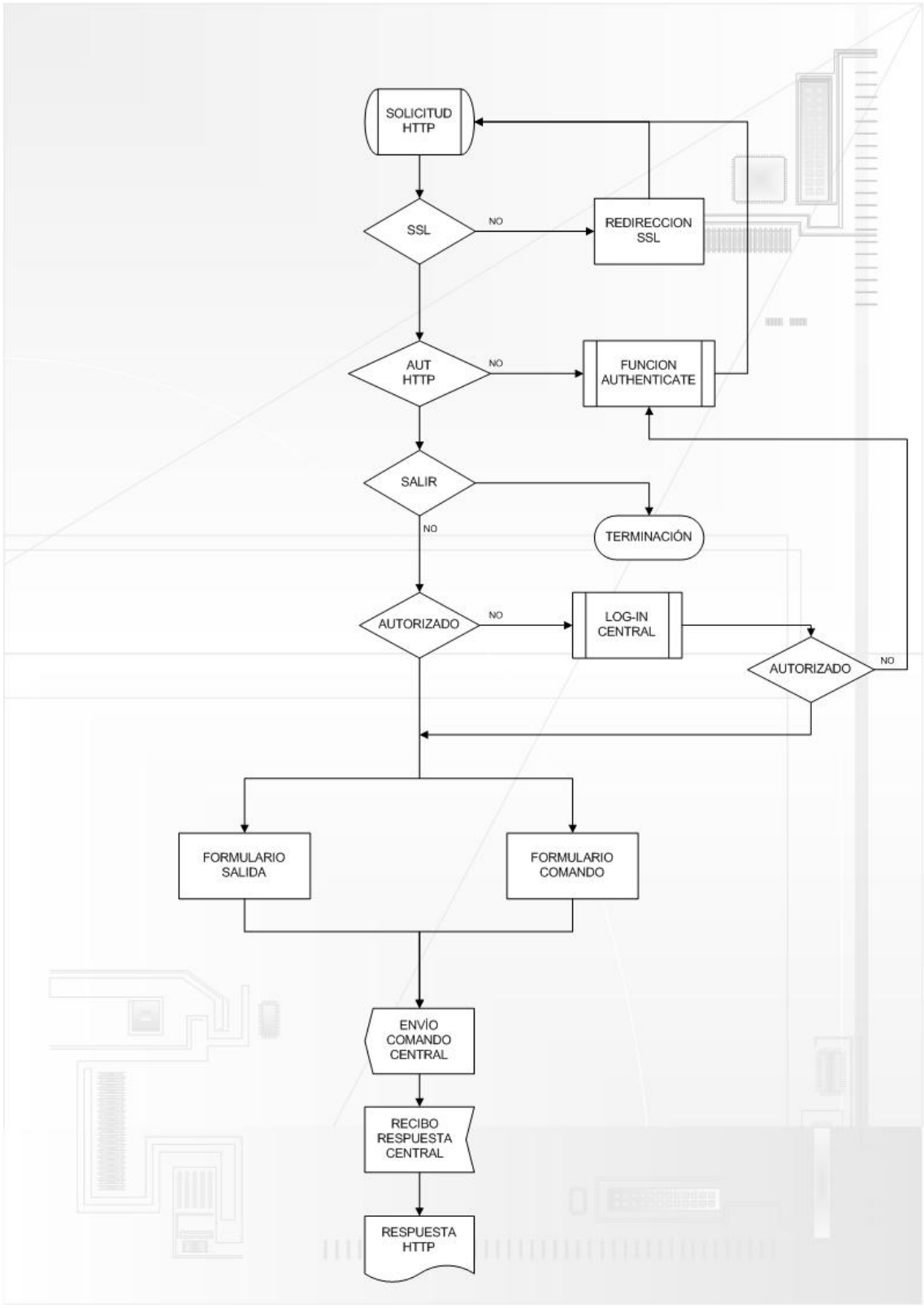
Figura A.12

Anexos

Anexo B
Código de la aplicación

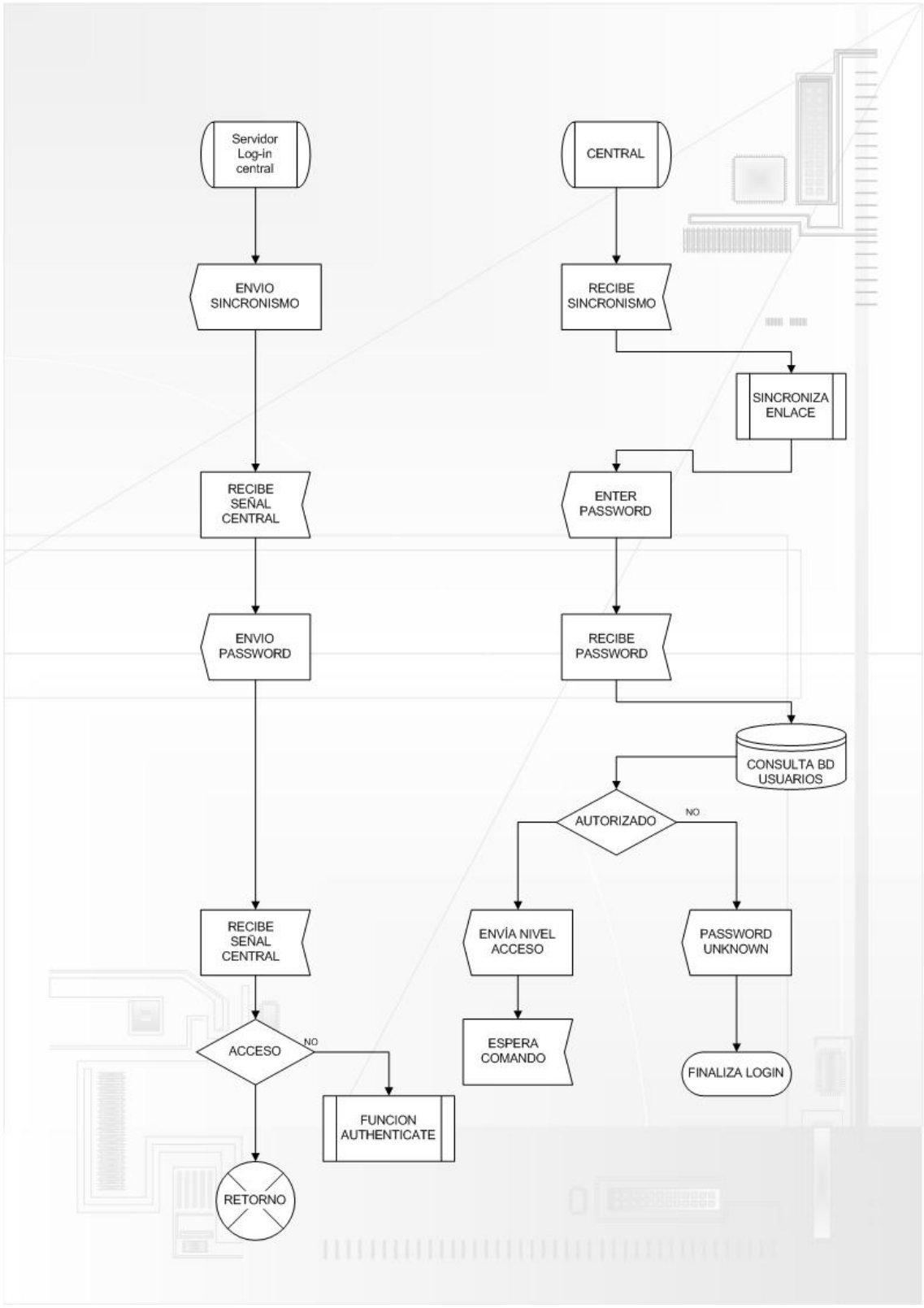
B.1(a) Algoritmo de la aplicación

La siguiente figura representa de algoritmo resumido de la secuencia principal de la aplicación.


SOLICITUD HTTP
SSL
NO
REDIRECCION SSL
AUT HTTP
NO
FUNCION AUTHENTICATE
SALIR
NO
TERMINACIÓN
AUTORIZADO
NO
LOG-IN CENTRAL
AUTORIZADO
NO
FORMULARIO SALIDA
FORMULARIO COMANDO
ENVÍO COMANDO CENTRAL
RECIBO RESPUESTA CENTRAL
RESPUESTA HTTP

B.1(b) Algoritmo de la aplicación

La siguiente figura representa de algoritmo resumido del proceso Log-in de la central.


Servidor Log-in central
ENVIO SINCRONISMO
RECIBE SEÑAL CENTRAL
ENVIO PASSWORD
RECIBE SEÑAL CENTRAL
ACCESO
NO
FUNCION AUTHENTICATE
RETORNO
CENTRAL
RECIBE SINCRONISMO
SINCRONIZA ENLACE
ENTER PASSWORD
RECIBE PASSWORD
CONSULTA BD USUARIOS
AUTORIZADO
NO
ENVÍA NIVEL ACCESO
ESPERA COMANDO
PASSWORD UNKNOWN
FINALIZA LOGIN

A.2 Código de la aplicación

A continuación se proporciona el código completo de la aplicación de configuración Web de la central MD-110.

```
<?php
session_cache_limiter('nocache,private');
session_start();


//*****************************************************************************
//*****************************************************************************
//
//
//
//
//.      .        lenguaje:.      .        PHP 5.0
//.      .        autor: ..       Antonio Garcia Melgar
//.      .        fecha: ..       noviembre 2004
//.      .        objeto:..       Configuracion web central telefonica MD-110
//.      .        s.o.:.  .       .        Suse Linux 9.0
//.      .        Servidor web:.  apache 2.0
//
//
//
//
//*****************************************************************************
//*****************************************************************************




//funci?n que solicita la autenticaci?n al cliente*****************************

function authenticate() {
.       Header("WWW-Authenticate: Basic realm=AREA RESTRINGIDA");
.       Header("HTTP/1.0 401 Unauthorized");
.       echo 'AUTORIZACION REQUERIDA';//TEXTO QUE SE ENVIA SI PULSA CANCEL
.       exit;
}

//comienzan las comprobaciones iniciales de seguridad
//antes de enlazar con la central
```

Esta sección del código corresponde a las comprobaciones iniciales de seguridad y Log-out

```
//comienzan las comprobaciones iniciales de seguridad
//antes de enlazar con la central


//comprobacion de conexi?n segura SSl y en caso contrario redireccion **********

if (!isset($_SERVER['HTTPS']) || $_SERVER['HTTPS']!="on") {
        header("Location: https://$_SERVER[SERVER_NAME]".$_SERVER['REQUEST_URI']);
        exit;
}


// comprueba autenticacion previa ***********************************************
// si no ha utilizado autenticacion basica llama a la funcion authenticate******

if ((!$_SERVER['PHP_AUTH_USER']) && (!$_SERVER['PHP_AUTH_PW'])) {
        $_SESSION['aceptado']="no";
        authenticate(); // envia cabecera de autenticacion
}


// accion si se pulsa el boton de salir ***************************************

if (($_POST['USUARIO_ANTERIOR'] == 1) &&( $_POST['ANTIGUA_Aut'] ==
        $_SERVER['PHP_AUTH_USER'])) {
        $_SESSION['aceptado']="no";

        if($fout=fopen("/dev/ttyS0", 'a+')){
                $out="exit;";
                fwrite($fout, $out);
                fflush($fout);
                fclose($fout);
        }
        authenticate();
}
```

En esta sección, se comprueba si se ha realizado login en la central, y en caso de que no sea así, comienza el proceso de Log-in en la central.

```
//comprobaci?n si la sesion est? iniciada**************************************

if(($_SESSION['aceptado'])!="si"){

//si la sesion no esta iniciada efectua proceso de login en la central *********

        $passw=$_SERVER['PHP_AUTH_PW'];

//envio sincronismo a la central**********************************************

        if($fdlog=fopen("/dev/ttyS0", 'a+')){
                $sincro="sssssssss;";
                if($plog=fwrite($fdlog, $sincro)){
                        fflush($fdlog);
                }
                else{
                        authenticate();
                }


//proceso de escucha del puerto serie ****************************************

                $leclog=array($fdlog);
                while(stream_select($leclog, $esclog=NULL,$exclog=NULL, 2, 20000)){
                        $datalog.=fread($fdlog, 1);
                        if(strlen($datalog)==0){
                                break;
                        }
                }

//si falla secuencia login solicita nueva autenticacion**************************

                if(!($datalog=="ENTER PASSWORD")){
                        unset($_SERVER['PHP_AUTH_PW'],$PHP_AUTH_PW);
                        unset($_SERVER['PHP_AUTH_USER'],$PHP_AUTH_USER);
                        $_SESSION['aceptado']="no";
                        authenticate();
                }
```

En esta sección continúa, el proceso de Log-in en la central.

```
//envia PASSWORD a la central*****************************************************

        if($plog2=fwrite($fdlog, $passw)){
                fflush($fdlog);
        }


//proceso de escucha del puerto serie ***************************************

        $leclog2=array($fdlog);
        while(stream_select($leclog2, $esclog2=NULL,$exclog2=NULL, 2, 20000)){
                $datalog2.=fread($fdlog, 1);
                if(strlen($datalog2)==0){
                        break;
                }
        }


//si el PASSWORD no es correcto solicita autenticacion**************************

        if($datalog2=="PASSWORD UNKNOWN"){
                unset($_SERVER['PHP_AUTH_PW']);
                unset($_SERVER['PHP_AUTH_USER']);
                $_SESSION['aceptado']="no";
                authenticate();
        }

//una vez completado el proceso de autorizacion por la central*****************

        $_SESSION['aceptado']="si";
        fclose($fdlog);
    }

}//*****************fin proceso login central************************************
```

Una vez efectuado el proceso de Log-in de la central se procede a tramitar el envío del comando introducido por el usuario en el formulario, hacia la central.

```
//escritura puerto serie (envio de comando a la central)************************

$comando=$_POST['comando'];
if($fd=fopen("/dev/ttyS0", 'a+')){
        if($p=fwrite($fd, $comando)){
                fflush($fd);
        }
        else{
                $data="no se recibio comando en la central";
        }


//escucha del puerto serie (respuesta de la central, al comando)****************

        $lec=array($fd);
        while(stream_select($lec, $esc=NULL,$exc=NULL, 2, 20000)){
                $data.=fread($fd, 1);
                if(strlen($data)==0){
                        break;
                }
        }

//se cierra el fichero ttyS0 (pto serie COM1)***********************************

        fclose($fd);
}
else{
        $data="no se pudo enlazar con la central para envio de comando";
}
//*******************************************************************************
?>
```

En esta sección, y una vez finalizado el cuerpo principal del programa en lenguaje PHP, comienza el código HTML, que conforma la presentación de la “Pagina Web”, en concreto en esta sección se implementa el formulario de envío de comandos.

```
<!-- **********comienza presentacion********** -->
<html>


<head>
<title>CENTRAL CART-001</title>
</head>

<body>

<!-- **********formulario de entrada de comandos********** -->

<p align="center"><u><font size="6"><b>CENTRAL TELEF?NICA</b></font></u></p>
<form method="POST" action="central.php">
<fieldset style="padding: 2; width:883; height:116">
  <legend><b>Cuadro de COMANDOS</b></legend>
  <p>
  <textarea rows="2" name="comando" cols="105" style="background-color: #C0C0C0">
        </textarea></p>
        </fieldset><p align="center">
        <input type="submit" value="ENVIAR COMANDO" name="B1">
        <input type="reset" value="Restablecer" name="B3"></p>
</form>


<div align="center">
<center>
```

En esta sección se implementa el formulario de envío de Log-out

```
<!-- ********************formulario de logout ****************************** -->

<p>
<form method="POST" action="central.php">
<input type='hidden' name='USUARIO_ANTERIOR' value='1'>
<?php
echo "<input type='hidden' name='ANTIGUA_Aut'
value='{$_SERVER['PHP_AUTH_USER']}'>\n";
?>
<input type="submit" value="CERRAR SESION" name="B4"></p>
</form>
 </p>
</center>
</div>
```

Finalmente, en esta sección se implementa la presentación de la salida que proviene de la central así como la identidad del usuario, el último comando enviado y la hora de envío del último comando

```
<!-- **********************formato de tabla*********************************** -->
<!-- **********************da forma a la salida de la central**************** -->


<div align="left">
  <table border="1" cellpadding="0" cellspacing="0"
  style="border-collapse: collapse" bordercolor="#111111" id="AutoNumber1"
  width="100%" height="100">
    <tr>
      <td width="186" height="19" bgcolor="#669999"><b>USUARIO</b></td>
      <td width="526" height="19" bgcolor="#669999">
        <?php echo "<b>{$_SERVER['PHP_AUTH_USER']}</b>"; ?></td>
      <td width="159" height="19" bgcolor="#669999"><b>
      <? echo  date("H : i "),"  /  ",date("j-n-Y")?></b></td>
    </tr>
    <tr>
      <td width="186" height="19" bgcolor="#669999"><b>COMANDO ENVIADO</b></td>
      <td width="682" colspan="2" height="19" bgcolor="#669999">
        <?php echo "<b>$comando</b>";?></td>
    </tr>
    <tr>
      <td width="868" colspan="3" height="19" bgcolor="#66CCFF">
      <p align="center"><b>RESPUESTA DE LA CENTRAL</b></td>
    </tr>
    <tr>
      <td width="868" colspan="3" bgcolor="#C0C0C0" height="133" align="left"
        valign="top"><?php echo "<b>$data</b>"; ?></td>
    </tr>
  </table>
</div>

</body>

</html>
```

Anexos

Anexo C
SSL

C.1 Introducción

SSL (Secure Socket Layer) es un protocolo propuesto por Netscape (http://developer.netscape.com/docs/manuals/security/sslin/contents.htm) para implementar criptografía y permitir una comunicación segura a través de la red. El IETF ha estandarizado un protocolo llamado Transport Layer Security (TLS) basado en SSL (http://www.ietf.org/html.charters/tls-charter.html). TLS es muy parecido y compatible con la versión 3 de SSL.

SSL propone una alternativa al uso de sockets TCP/IP en nuestras comunicaciones. SSL es una capa software que se encuentra por encima de TCP/IP y por debajo de los protocolos de aplicación (ver figura 1).

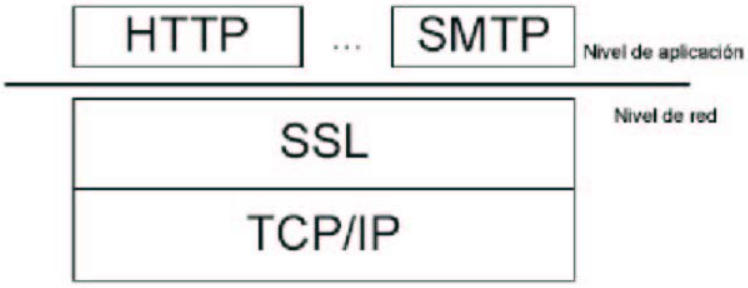


Figura 1

Las aplicaciones que se encuentran por encima hacen llamadas a las funciones de SSL en vez de TCP/IP. Esto implica que las aplicaciones deben ser preparadas para llamar a SSL en vez de TCP/IP, aunque las modificaciones que hay que hacer en ellas no son grandes. No se puede hacer una implementación transparente porque SSL necesita cierta información criptográfica adicional para establecer un socket.

Los objetivos de SSL son los siguientes:

• Permitir la autenticación tanto de cliente como servidor, usando claves públicas y

• Proporcionar una comunicación segura mediante el cifrado de la información entre emisor y receptor.

C.2 Funcionamiento de SSL

La conexión SSL es privada y fiable. El funcionamiento de SSL en breve es el siguiente:

Cliente y servidor se ponen de acuerdo en el tipo de cifrado que van a usar.

• Cliente y servidor intercambian información necesaria para establecer una clave secreta común.

• Cliente y servidor intercambian información cifrada usando un algoritmo secreto con una clave común.

Nótese que SSL usa cifrado simétrico para el intercambio de información, es decir, que usan una clave secreta conocida por ambos. Para ponerse de acuerdo en la clave, se usan algoritmos asimétricos de clave pública y privada.

De esta manera, SSL es capaz de ofrecer:

• Privacidad. La información se transmite cifrada.

• Integridad de datos. Se usan funciones *hash* para asegurar que no se modifica la información.

• Autenticidad. Se intercambian certificados digitales.

• No-repudio. El uso de firmas y certificados digitales asegura que no se pueda repudiar una transacción.

SSL se compone en realidad de varios protocolos:

• Record protocol. Este protocolo proporciona los servicios de seguridad básicos a las protocolos de aplicación, HTTP fundamentalmente.

• Handshake protocol. El protocolo que se usa para ponerse de acuerdo en la clave común.

• Change Cipher Spec y Alert Protocol. Protocolos de gestión de los intercambios SSL al igual que el Handshake Protocol.

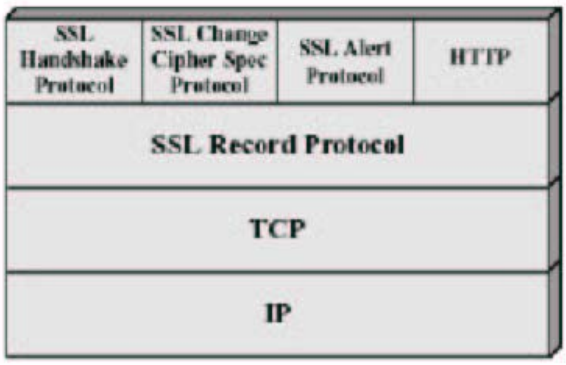


Fig. 2. Pila de protocolos completa SSL

Dos conceptos que define SSL son:

• Conexión. Un transporte (como se define en el modelo OSI) que proporciona un tipo de servicio apropiado. Las conexiones son transitorias y están asociadas a una sesión.

• Sesión: una asociación entre un cliente y un servidor. Las sesiones se crean mediante el protocolo de Handshake. Definen un conjunto de parámetros de seguridad criptográfica, que pueden ser compartidos por múltiples conexiones. Las sesiones se usan para evitar la costosa negociación de los parámetros para cada conexión.

3.1 SSL Record Protocol

El SSL Record Protocol proporciona dos servicios a las conexiones SSL:

• Confidencialidad: usa la clave secreta compartida generada durante el protocolo de Handshake para el cifrado convencional de los datos.

• Integridad del mensaje. El protocolo de Handshake también genera una clave secreta común que se usa para formar un código de autenticación de mensaje (*Message Authentication Code, MAC*), una función similar a un hash.

El proceso que sigue el Record Protocol se muestra en la figura 3:

• Los mensajes se fragmentan en bloques de 214 bytes o menos.

• Se aplica compresión opcionalmente.

• Se calcula una código de autenticación de mensaje (o una función hash) sobre los datos comprimidos.

• El resultado del MAC concatenado con el mensaje se cifra mediante un algoritmo simétrico.

• Finalmente se le añade una cabecera.

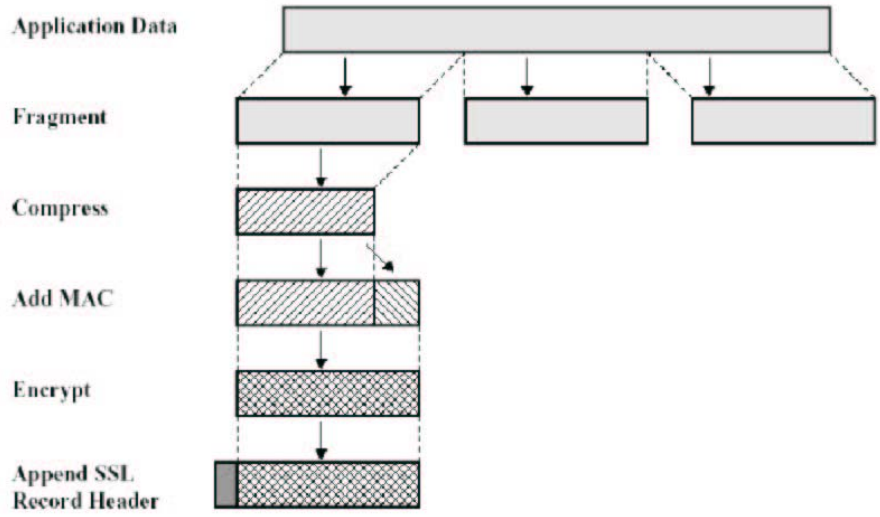


Fig. 3. Funcionamiento del SSL Record Protocol

Como se puede observar, en el funcionamiento del SSL Record Protocol intervienen mecanismos criptográficos, para los cuales son necesarios ciertos parámetros (la clave secreta sin ir más lejos). Estos parámetros se negocian y generan durante el protocolo de Hansdshake, que además permite la autenticación de cliente y servidor. Los otros dos protocolos (Change Cipher Spec y Alert) realizan funciones de control y de aviso de errores respectivamente. No se estudiarán.

3.2 Handshake Protocol

Es la parte más compleja de SSL. Permite la autenticación de cliente y servidor y la negociación de los algoritmos de cifrado y las claves. Se usa antes de enviar los datos de aplicación.

Un concepto que usado en el funcionamiento de este protocolo es la suite de cifrado (*CipherSuite*), que es una lista de las combinaciones de algoritmos criptográficos soportados por el cliente.

Hay una serie de suites de cifrado definidas en el estándar para proporcionar diferentes servicios criptográficos, con nombres asociados que describen su contenido. Por ejemplo, la suite llamada SSL_RSA_EXPORT_WITH_DES_40_MD5 usa:

• RSA para el cifrado con clave pública, con un módulo adecuado para la exportación fuera de los EE.UU.

• DES como algoritmo de cifrado simétrico, con una clave de 40 bits.

• MD5 como algoritmo de resumen o *hash*.

Entonces, una suite de cifrado determina:

• El algoritmo de clave pública usado para el intercambio de claves. Se puede usar

RSA, Fortezza y variaciones del algoritmo de Diffie-Hellman.

• El algoritmo de cifrado simétrico usado. Pueden ser RC4, RC2, DES, 3DES, DES40, IDEA o Fortezza.

• El algoritmo de resumen usado. MD5 o SHA-1.

• Si cumple las normas para la exportación fuera de los EE.UU

Una vez definidos los términos necesarios examinaremos el protocolo con un poco más de detenimiento. La figura 4 muestra el intercambio de mensajes del Handshake Protocol:

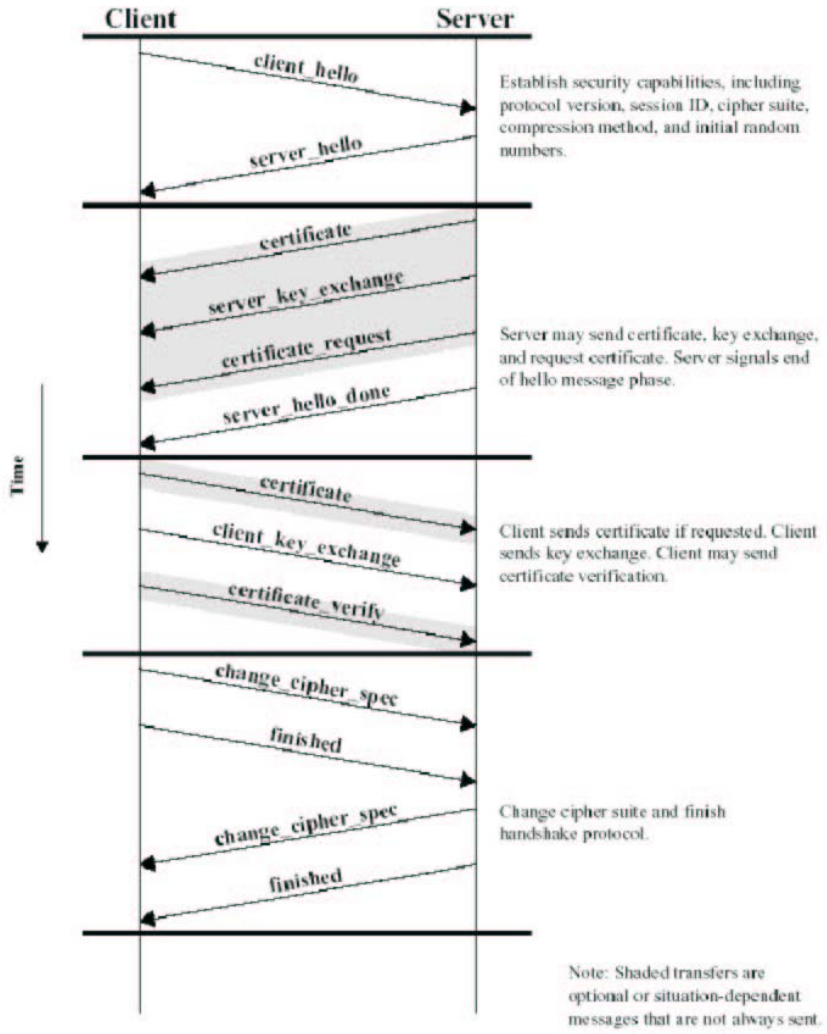


Fig**. 4. Intercambio de mensajes del Handshake Protocol**

• **Fase 1. Se establecen las capacidades de seguridad.** El cliente envía al servidor información necesaria para establecer una comunicación SSL, como, por ejemplo, las suites de cifrado que soporta.

• El servidor decide si soporta esa suite de cifrado u otras y comunica la suite de cifrado que va a utilizar, junto con otros parámetros.

• **Fase 2. Autenticación del servidor e intercambio de claves.** El servidor envía su certificado, que consiste en un único certificado X.509 o una cadena de estos. Nótese que hay suites de cifrado que no requieren la autenticación del servidor, como por ejemplo cuando se usa Diffie-Hellman Anónimo (*Anonymous Diffie-Hellman*).

• Con la información enviada en esta fase el cliente autentica el servidor. En el caso en que se esté usando RSA, el servidor envía una clave pública para que el cliente cifre con ella la información necesaria para generar una clave secreta común y la devuelva al servidor. De este modo, el servidor la descifrará con la clave privada y podrá generar la misma clave común.

• **Fase 3. Autenticación del cliente e intercambio de claves.** En el caso en que el haya que autenticar al cliente, éste envía la información necesaria (su certificado). Lo normal es que el cliente no se identifique. En ese caso, esta fase consiste en el envío de la información necesaria para generar la clave secreta común, bien cifrada con la clave pública del servidor (en el caso que se use RSA) o bien usando el algoritmo de Diffie-Hellman.

• Con la información recibida del cliente y descifrada con la clave privada del servidor, éste último genera una clave secreta. El cliente genera esa misma clave secreta. Esta clave se llama clave de sesión y será usada en el algoritmo simétrico.

• **Fase 4. Finalización.** El cliente envía un mensaje al servidor diciendo que los siguientes mensajes serán cifrados con la clave de sesión. Además envía un mensaje cifrado diciendo que la parte del cliente del protocolo Handshake ha finalizado.

• El servidor envía un mensaje diciendo que los siguientes mensajes serán cifrados con la clave de sesión. Además envía un mensaje cifrado diciendo que la parte del servidor del protocolo Handshake ha finalizado.

• Comienza el intercambio de información cifrada con la clave de sesión mediante un algoritmo simétrico.

Nótese, que la clave secreta generada, la clave de sesión, se utilizará durante toda la sesión, es decir, que el algoritmo de intercambio de claves sólo se realiza al principio. El resto de peticiones se realizará con la clave generada.

Nótese también que en ningún momento se intercambia la clave de sesión que se usará.

Lo que se intercambian son parámetros necesarios para generar la clave de sesión. Con esos parámetros, tanto el cliente como el servidor son capaces de generar la misma clave de sesión. Aunque un atacante pudiera interceptar esta información, sólo con esos parámetros no podría generar la clave de sesión.

ANEXO D

MD110: DESCRIPCION CENTRAL MD110

MD110

Los sistemas de comunicaciones para la empresa MD110 de Ericsson ofrecen una nueva arquitectura que hace posible una nueva forma de trabajar con una movilidad cada vez mayor. MD110 es el sistema PBX más avanzado del mundo y ofrece potentes soluciones para las redes de hoy y de mañana. Este sistema exclusivo combina la experiencia mundial de Ericsson en conmutación, redes y comunicaciones móviles con el fin de ofrecer los beneficios que demandan los usuarios.

Basado en una arquitectura modular y distribuida, MD110 sirve de base para una red capaz de sop-ortar aplicaciones de voz, datos y multimedia. A causa de su arquitectura distribuida, MD110 ofrece un alto grado de tolerancia a fallos y una escalabilidad excepcional, desde 100-200 a 20.000 o más usuarios en un entorno de oficina, campus o empresa. La movilidad se soporta en cualquier configuración y en múltiples emplazamientos.

MD110 de Ericsson constituye un sistema excepcional que le prepara para el futuro y, al mismo tiempo, proporciona una solución rentable para las necesidades de comunicación de las empresas actuales. Cualquiera que sean sus necesidades, hoy o mañana, habrá un sistema MD100 que sobrepase sus expectativas. En la actualidad más de 17 millones de usuarios empresariales confían en MD110 y han puesto su confianza en Ericsson para que proteja sus inversiones en el futuro mediante a través de la continua mejora del sistema.

ERICSSON

MD110

MD110 se puede construir utilizando de uno a cuatro módulos apilables, colocados uno al lado del otro o adosados por la parte de atrás..

El concepto distribuido de MD110

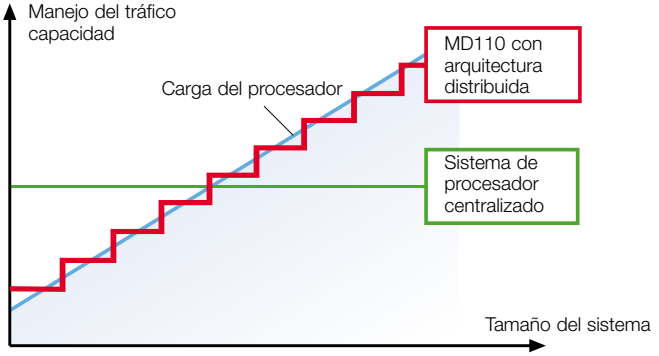
El elemento básico de MD110 es su arquitectura distribuida. La arquitectura del sistema se construye con módulos independientes, que se pueden interconectar para crear sistemas para más de 20.000 extensiones.

El módulo básico principal de MD110 es el Módulo de Interface de Línea, LIM. Cada LIM es totalmente autónomo, al estar equipa-do con sus propias unidades de procesador, software, conmutación y dispositivos, como por ejemplo extensiones y líneas de enlace. Un sistema MD110 puede estar formado por un LIM en una configuración autónoma o por varios LIM interconectados. MD110 opera con capacidad de procesamiento totalmente distribuida y todos los LIM tienen el mismo software. Los LIM cooperan de igual modo y forman un único sistema totalmente integrado.

Todas las funciones telefónicas y del sistema son transparentes y están disponibles para todo el sistema.

La conexión entre los LIM se realiza a través de un interface estándar de transmisión de 2 Mbits. Esto significa que se pueden instalar LIM adosados en la misma ubicación o cada uno en un lugar; por ejemplo dentro de un edificio o a través área geográfica dispersa. Con un solo sistema se presta servicio a toda la organización, con total transparencia de funciones y del sistema. Se emplea la misma administración y, de este modo, se simplifica; los movimientos y los cambios se realizan fácilmente.

Dado que cada LIM es independiente, continúa trabajando aunque se interrumpa la comunicación con el resto del sistema. Esto significa que prácticamente se elimina el riesgo de una parada total del sistema, convirtiéndolo en un sistema realmente resistente. Los servicios de telefonía de una oficina local, suministrados por un LIM distribuido remotamente, seguirán funcionando, aunque se interrumpa el contacto con la oficina principal. Se pueden seguir realizando llamadas internas y externas.



La arquitectura distribuida de MD110 proporciona una capacidad flexible de procesamiento, y permite dimensionar la capacidad de manejo de tráfico y el tamaño del sistema para que se adapten exactamente a sus necesidades. En sistemas centralizados de procesador, puede haber mucho o poco.

Por ejemplo, en un edificio, los LIM se pueden distribuir en varias plantas, reduciendo el riesgo de parada del sistema e incrementando la fiabilidad. Un fallo en un LIM no influye en el resto del sistema, reduciendo el tamaño de las unidades eliminadas. Las conexiones de línea de enlace con la red pública se pueden organizar en diferentes rutas de distintos LIM, reduciendo al mínimo el riesgo de no poder realizar llamadas salientes. La fiabilidad se puede incrementar aún más mediante la conexión de varias rutas a diferentes centrales locales o incluso a diferentes operadores públicos.

Los LIM se pueden instalar remotamente a través de enlaces de 2 Mbits, privados o públicos, pero constituyendo un módulo en un sistema MD110 completamente integrado. Se puede implantar un solo sistema de comunicación con total transparencia de funciones y del sistema para organizaciones dispersas geográficamente. De este modo, el concepto distribuido de MD110 proporciona la solución para redes más avanzada posible, con una transparencia más avanzada que la ofrecida por QSIG/ conexión de redes, ya que, tanto las funciones como el sistema son transparentes.

El LIM tiene capacidad para 640 extensiones. En instalaciones más grandes, se añaden más LIM. Se pueden mantener 20.000 extensiones o más en un emplazamiento. Dos LIM se pueden interconectar, adosándolos por la parte posterior. Para tres LIM o más, se necesita el segundo componente principal de MD110, el Selector de Grupo. El selector de grupo se controla mediante el LIM, el cual establece conexiones entre los LIM.

Características principales

Estos principios de diseño y la estructura modular del sistema confieren a MD110 características exclusivas y zofrecen a los clientes beneficios en relación con lo siguiente:

- escalabilidad
- flexibilidad
- fiabilidad y disponibilidad
- descentralización

Escalabilidad

El concepto modular y distribuido de MD110, lo convierte en extremadamente escalable. Se puede comenzar con un pequeño sistema e ir actualizándolo según sea necesario.

Los sistemas se pueden ampliar o reducir fácilmente. Gracias al concepto modular, el equipo se puede mover de un lugar a otro, desde donde se estén reduciendo las actividades hasta donde estén aumentando. Las inversiones ya realizadas se protegen y se evitan nuevos gastos.

Flexibilidad (incluso de carga de procesador)

Con procesamiento y conmutación distribuidos, siempre hay capacidad suficiente de ambos en el sistema para proporcionar el rendimiento necesario a los usuarios finales de forma óptima, independientemente del tamaño del sistema.

Fiabilidad y disponibilidad

Dado que la capacidad de procesamiento y conmutación se distribuye entre todos los LIM, un fallo en el hardware o el software sólo influye en los servicios del LIM que funciona incorrectamente, no en el sistema completo.

Sistema distribuido en el emplazamiento o el campus

Los módulos se pueden distribuir con flexibilidad por el área del campus o del escritorio, manteniendo una transparencia de funciones del 100%.

Características del sistema

Además de la arquitectura distribuida del sistema y de su construcción modular, el sistema MD110 más reciente ofrece nuevas funciones muy significativas:

- Movilidad con equipos inalámbricos
- Números Personales
- Posición libre (free seating)
- Correo de Voz Integrado
- Mensajes de alarma
- Telefonía a través de IP
- Conexión de redes
- Centro de Llamadas
- Interface Abierto
- Gestión de Redes SNMP

Movilidad con equipos inalámbricos

MD110 proporciona el primer PBX inalámbrico del mercado. Ericsson está demostrando su liderazgo en sistemas móviles a través de la integración de funciones DECT (telecomunicación inalámbrica digital mejorada) con el sistema de comunicación empresarial. Esta integración aprovecha al máximo la arquitectura distribuida de MD110 y soporta hasta 20.000 usuarios de equipos inalámbricos en un sistema.

De este modo, los usuarios tienen movilidad en cualquier lugar del área de cobertura, que puede incluir varios emplazamientos.

También se soporte el roaming integrado y el handover uniforme como software básico de MD110. No se necesita ningún servidor adicional. Con el servicio Número Personal, se incrementa aún más la movilidad, permitiendo a los usuarios responder a llamadas de negocios en cualquier lugar donde se encuentren.

Número Personal

El Número Personal para comunicaciones móviles combina distintos métodos de acceso (extensiones con cable, comunicaciones inalámbricas para la empresa, celulares, PSTN y buscadores) con una nueva generación de servicios personales que ayudan a los usuarios móviles a administrar sus llamadas.

Personal rastrea dónde se encuentra el usuario y cada usuario puede tener hasta cinco perfiles personales que se pueden activar según la situación (en la oficina, mientras viaja, en casa, etc.). El perfil determina lo que les ocurre a las llamadas entrantes, cuáles se desvían a diferentes teléfonos en un orden predefinido o se transfieren a un servicio de copia de seguridad.

Las llamadas entrantes a la persona y el departamento correctos también permiten organizar extensiones en grupos de llamadas, que se pueden encontrar en la misma oficina o en diferentes lugares. Dependiendo sus necesidades, hay varias opciones disponibles para supervisores de grupo de llamadas y agentes, incluidas las aplicaciones Call Center Manager y Agent Desktop.

Posición Libre (Free Seating)

Posición Libre (Free Seating) es un desarrollo de la Extensión Virtual y responde a las necesidades de compañías con empleados con movilidad. Si los usuarios de posición libre (free seating) necesitan trabajar fuera de la oficina, sólo tienen que conectarse a cualquier teléfono libre. Este teléfono tiene toll baring class$$, indicaciones de espera de mensajes y registro de llamadas de todas las llamadas salientes para dicho usuario. Cuando se realizan llamadas desde este teléfono, el nombre y el número del usuario se presentan a los abonados llamados. Las personas que llaman sólo ven el número y el nombre virtual (no el teléfono en el que se encuentra el usuario). Cuando el usuario sale de la oficina, simplemente desconecta.

Correo de Voz Integrado

PBX MD110 ofrece opcionalmente correo de voz en una tarjeta. Tarjeta admite hasta 300 buzones de correo y 16 sesiones simultáneas. Se pueden almacenar hasta 10 horas de mensajes de correo de voz en un disco duro independiente. Las funciones básicas son almacenamiento, búsqueda, recuperación y borrado de mensajes. Los usuarios pueden grabar saludos personales.

Mensajes de alarma

MD110 ofrece la posibilidad de crear SMS (Servicios de Mensajes Cortos) basados en DECT. Una primera aplicación transmite mensajes a terminales basados en eventos o alarmas. Se pueden realizar aplicaciones personalizadas para diferentes

Dialog 3413

Consumo de Energía	*Alimentación interna*	*Alimentación externa*
Solución alternativa para pila completamente equipada, incl. digitales	150 – 1,400 W	150 – 1,800 W
Disipación de calor para cada pila completamente equipada, incl.telephones teléfonos digitales	115 – 180 W	115 – 350 W

Dimensiones y peso

	1 módulo	*2 módulos*	*3 módulos*	*4 módulos*
Altura (mm/in)	630/24.8	1030/40.5	1430/56.3	1830/72.0
Anchura (mm/in)	598/23.5	598/23.5	598/23.5	598/23.5
Profundidad (mm/in)	355/14.0	355/14.0	355/14.0	355/14.0
Peso (kg/lbs)	45/99	85/187	125/275	165/363

conexiones, incluido Internet u otros interfaces externos; correo electrónico, entradas de alarma o contactos. Estas aplicaciones están destinadas a organizaciones que necesitan alertar a equipos especiales, por ejemplo, en caso de emergencia. Entornos típicos para estas aplicaciones son las plantas de producción, centrales de procesamiento y hospitales; en general, organizaciones en las que puede ser necesario avisar a los empleados en caso de emergencia. En hoteles, los huéspedes o directores pueden enviar mensajes urgentes a los empleados de servicio, cuando se requiera una respuesta inmediata.

Telefonía a través de IP

Con BC11, Ericsson introduce la Telefonía a través de IP como un modo idóneo de integrar a trabajadores dispersos a través de la red IP. Los trabajadores que se encuentran en pequeñas oficinas o trabajan en su casa ya pueden utilizar las abundantes funciones de MD110 de un modo rentable. Junto con el puerto integrado que se desliza en cualquier ranura de un plano posterior de MD110, también se introduce un teléfono IP que cumple con H.323. MD110 supervisa las extensiones y controla la configuración de llamadas, pero permite enrutar la corriente de medios directamente a la LAN, una vez que se ha establecido la llamada. De este modo se mantiene la Calidad de Servicio sin distorsiones y la corriente de medios deja de estar limitado a conversación y se puede utilizar para conexiones de vídeo.

La Extensión/Gatekeeper de IP cumple con H.323 para permitir la conexión de cualquier Cliente Estándar de PC que cumpla con H.323 o de teléfonos IP.

La arquitectura distribuida es la clave de la escalabilidad excepcional de MD110; puede empezar a introducir Telefonía a través de IP a pequeña escala en ubicaciones seleccionadas y crecer al mismo ritmo que las necesidades de su empresa. BC11 se adapta a un entorno de oficina donde predomine la telefonía IP. Sin embargo, los entornos principales son el trabajo en casa/trabajo remoto y las pequeñas sucursales.

Conexión de redes

MD110 está diseñado desde el principio para servir de base a una red unificada que soporte aplicaciones integradas de voz, datos y multimedia. La total transparencia a lo largo de toda la red y en múltiples emplazamientos para todos los servicios de voz, datos y multimedia ha sido siempre una característica distintiva de MD110.

Ahora el sistema dispone de la Asignación Dinámica de Rutas (DRA) que ofrece soporte a la transparencia de funciones a través de redes públicas conmutadas ISDN o IDN para interconexiones de PBX, eliminando de este modo la necesidad de costosas líneas alquiladas dedicadas.

Centro de Llamadas

MD110 con el software de Distribución Automática de Llamadas (ACD), constituye una plataforma ideal para grandes organizaciones con cientos o miles de llamadas diarias. Esta función no sólo enruta llamadas entrantes a la persona y el departamento correctos; también organiza extensiones en grupos de llamadas, que se pueden encontrar en la misma oficina o en diferentes lugares. Dependiendo de sus necesidades, hay varias opciones disponibles para supervisores de grupo de llamadas y agentes, incluidas las aplicaciones Call Cen-ter Manager y Agent Desktop.

Interfaces Abiertos

MD110 está basado en normas e interfaces abiertos. Los interfaces abiertos permiten complementar MD110 con numerosas aplicaciones, desarrolladas por nosotros mismos o por otros proveedores. Entre los ejemplos se encuentran las soluciones de Centros de Llamadas, Mensajerías Unificadas y desarrollos específicos para clientes. En la última versión, se puede añadir el envío de mensajes de alarma basados en SMS a la lista de aplicaciones. MD110 soporta todos los estándares generalmente disponibles, como CTI/CSTA, TAPI, TSAPI, SNMP. También hay varios interfaces abiertos disponibles para el desarrollo de aplicaciones por parte de terceros. Entre ellos, se encuentran sistemas de facturación, correo de voz y desarrollo de aplicaciones para la operadora. La conectividad se proporciona normalmente a través de IP/Ethernet; en algunos casos se utiliza V.24.

Naturalmente, MD110 soporta todos los estándares de telefonía generales, incluidos PSTN, DECT, ISDN. Ahora puede añadir IP a la lista, incrementando aún más la apertura.

Network Administration Administración de la Red

Se está introduciendo el sistema de gestión estandarizado para toda la red, basado en SNMP (Protocolo Simple de Gestión de Redes) estándar del mercado, utilizando TPC/IP y PPP (a través de LAN o módem). El DNA (Dynamic Network Administration) proporciona herramientas de gestión de redes que incluye el Gestor de Directorio, el Gestor de Extensiones, el Gestor de Rendimiento, el Gestor de Nodos y el Gestor de Eventos, así como el Servidor D.N.A., común para todas las aplicaciones y la aplicación Estación de Trabajo de Operadora.

Estructura Mecánica de MD110

MD110 presenta una estructura mecánica que permite un alto grado de flexibilidad. La estructura mecánica se basa en módulos apilables que se pueden montar de diversas formas, con el fin de adaptar una instalación al espacio disponible. Los módulos se pueden montar unos encima de otros, dependiendo de la altura del techo. Si se desea, se pueden montar en la pared, separados o unos al lado de otros. También se pueden montar los módulos en una pared, separados unos de otros por una distancia determinada, de modo que se puedan adaptar al interior de la sala.

ACM - Módulo Completo

MD110 preembalado para compañías medianas y pequeñas sucursales.

Dimensiones y peso
(con cubierta)

Altura (mm/pulgadas)	460/18.1
Anchura (mm/pulgadas)	598/23.5
Profundidad (mm/pulgadas)	355/14
Peso	
(excl. baterías)	35 kg (77 lbs.)
(excl. baterías)	55 kg (121 lbs.)

Consumo de Energía

1 armario	320 VA
2 armarios	640 VA
Disipación de calor	70–120 W por armario

ACM Módulo Completo

El Módulo Completo se adapta a las necesidades de las empresas medianas y pequeñas, o sucursales con 200 extensiones como máximo. El módulo ACM (Módulo Completo) tiene todas las funciones que se pueden esperar de un gran sistema de comunicaciones de última generación, pero con un tamaño adaptado a las necesidades de empresas más pequeñas o sucursales.

ACM es una versión compacta de MD110 para compañías medianas y pequeñas, o para sucursales. Consulte los ejemplos de configuración del siguiente cuadro. Está basado en la estructura mecánica estándar apilable de MD110, pero está preembalado y se complementa con una unidad de alimentación integrada y baterías integradas. El usuario se beneficia de todas las funciones de MD110, pero con un tamaño adaptado a las necesidades de empresas más pequeñas o sucursales. MD110 basado en ACM es la solución perfecta cuando no se dispone de mucho espacio; se adapta perfectamente a los espacios existentes.

Estructura y configuración del sistema

El Módulo Completo de MD110 se puede configurar

- como un conmutador autónomo del edificio principal para compañías medianas y pequeñas con necesidades de alto rendimiento o alta densidad. Abierto para futura integración con una configuración del sistema más grande.
- como un nodo secundario de un sistema más grande, utilizando conexión de redes basa da en QSIG más VPN, ISDN llamada a llamada o señalización de D-a través de-B.
- como un nodo de campus o hub de edificio, completamente integrado con MD110, proporcionando transparencia de funciones del 100%.

Configuraciones típicas del Módulo Completo de MD110 [1)]

Nodo autónomo o secundario	*1 armario*	*2 armarios*
Líneas de enlace digitales	30	60
Extensiones analógicas + digitales	80 + 16	192 + 16
Extensiones inalámbricas + analógicas + digitales	170 + 16 + 16	340 + 32 + 132
Líneas de enlace analógicas	16	40
Extensiones analógicas + digitales	80 + 16	170 + 16
Extensiones inalámbricas + analógicas + digitales	170 + 16 + 16	340 + 32 + 32
Nodo de campus o hub de edificio	*1 armario*	
Enlaces/canales de características	2/60	
Extensiones analógicas + digitales	80 + 16	
Extensiones inalámbricas + analógicas + digitales	160 + 16 + 16	

1) La configuración autónoma incluye tarjetas de interface para S&NM y una unidad de disco duro para copia de seguridad del software.

ACM - Datos Técnicos

PSM - Módulo Conmutador y Procesador

- Control común, incluido el procesamiento y la conmutación (1.024 intervalos de tiempo).
- Medios para copia de seguridad del software del disco duro.
- 10 posiciones de tarjeta accesibles para interfaces de cualquier tipo de acceso interno o externo.
- Interface para S&NM (NIU)
- Fuente de alimentación integrada (y repuesto de batería opcional)

IFM - Módulo de Interface

- Conmutación para 256 intervalos de tiempo
- 17 posiciones libres de tarjeta accesibles para interfaces de cualquier tipo de acceso interno o externo
- Fuente de alimentación integrada (y repuesto de batería opcional)

www.ericsson.com/enterprise

Datos Técnicos

Módulos apilables de MD110

PSM - Módulo Conmutador y Procesador

- Control común, incluido el procesamiento y la conmutación (1024 intervalos de tiempo), y OS/AS (Sistema Operativo/ Software de Aplicación)
- Disco para copia de seguridad del software del disco duro
- 10 posiciones libres de tarjeta universal accesibles para inter faces de cualquier tipo
- Interface para S&NM (IPU)

IFM - Módulo de Interface

- Conmutación para 256 interval os de tiempo
- 17 posiciones libres de tarjeta accesibles

PWM - Módulo de Alimentación

- 2 x 12,5 A x 48 V (nominal) 1.200 W
- Alimentación rentable para 600 extensiones digitales como máximo.
- Un PWM puede alimentar hasta cuatro módulos (PSM o IFM).
- Repuesto de batería opcional de 26 Ah incorporado

PBM - Módulo de Reserva de Alimentación

- 2 x 26 Ah

OAM - Módulo de Aplicaciones Opcionales

- módulo vacío para hardware opcional u otras aplicaciones, que utiliza una norma de mon taje de 19".

GSM - Módulo de Conmutación de Grupos

- Magazine incorporado para 31 enlaces como máximo de unión con MD110

PDM - Módulo de Distribución de Alimentación para Alimentación Externa

- Incluye unidades limitadoras para 10, 20 ó 30 módulos.

ACM - Módulo Completo

- Contiene el PSM, una fuente de alimentación y repuesto de batería. En ACM, la fuente de alimentación, el cable y las baterías están montados en la base para hacer un uso óptimo del espacio.

MDM - Módulo de Distribución Principal

- Un pequeño sistema MDF interno como alternativa al MDF externo. El módulo tiene capacidad para 416 líneas en el lado de la central (teléfono digital y/o analógico y/o extensiones de línea de enlace) y 520 líneas en el lado de la línea del MDF.

Conexión a la red

115–230 V AC, ±15%, 50–60 Hz

Compatible con IEC 950

Almacenamiento

DRAM. Disco duro integrado para copia de seguridad de la memoria. PC opcional para copia de seguridad.

Condiciones Ambientales

Durante el funcionamiento:

Temperatura +5°C a +40°C (41°F a 104°F)

Humedad relativa: 20–80%

No se necesita refrigeración forzada.

Protección de línea

Interface protegido mediante transformadores.

Datos de línea de extensión analógica

Resistencia de alimentación de corriente 2 x 400 ohms, 48V

Resistencia de bucle 1.800 ohmios, incluido el teléfono

Señalización de Botón de rellamada, emisión de impulso de corte temporizado o hilo de conversación con conexión a tierra

Datos de línea de extensión digital

Dos cables

Longitud de línea ≤ 1,000 m (3,280 pies)

Red de conmutación

Multiplexación por División en el Tiempo (TDM)

Conmutador de no bloqueo físico de etapa única

Datos de línea de enlace analógica

Resistencia de bucle

Líneas a central pública 1.800 ohmios

1,800 ohmios

Líneas de conexión 2,000 ohmios

Datos de transmisión

Impedancia y niveles relativos adaptables al mercado.

Codificación de PCM de Ley A de acuerdo con CCITT G.711.

Atenuación de diafonía De acuerdo con Q.517 de CCITT.

Conexión de redes multinodo.

Total transparencia de funciones utilizando enlaces de 2 Mb estándar, PCM, G.703.

Configuraciones de MD110

LIM formado por una pila típica de 4 módulos que puede soportar hasta 640 extensiones (inalámbricas, digitales o analógicas) y 256 líneas de enlace.

MD110 tiene una moderna arquitectura de sistema, capaz de soportar la conexión de servidores específicos para aplicaciones, tales como correo de voz, servicios de movilidad (Servicios de Comunicación Personal) y otros equipos para funciones de grupo.

En la ilustración se muestra un ejemplo de cómo se pueden colocar los módulos apilables para una central de 3 LIM.

Bibliografía

BIBLIOGRAFÍA

Creación de aplicaciones Web con PHP 4
Ratschiller, Tobias, Madrid, Prentice Hall , D.L. 2000

HTML dinámico a través de ejemplos
Bobadilla Sancho, Jesús, Madrid : Ra-Ma , D.L. 1999

Apuntes de la asignatura Arquitecturas distribuidas.
Tercer curso ITT esp Telemática

Apuntes de la asignatura Laboratorio de Arquitectura de Redes de Comunicación
Tercer curso ITT esp Telemática

Apuntes de la asignatura Software de comunicaciones.
Segundo curso ITT esp Telemática

Internet: Manual de PHP on line
http://www.php.net/manual/es/

Internet: varias de programación en PHP
http://www.rinconastur.com/php/

Internet: Kermit Manual UNIX
http://www.columbia.edu/kermit/ckututor.html

Internet: estandar RS232
http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

Internet: Apache
http://www.apache.org/

Internet: Linux+Apache+MySQL+SSL
http://www.apachefriends.org/en/xampp-linux.html

Manual de operacion de la central MD-110