3 . La plataforma

Una vez descritos cada uno de los elementos que componen la plataforma de gestión, en este capítulo describiremos las distintas funcionalidades que tiene para poder gestionar tanto la misma como los distintos dispositivos.

3.1 Esquema funcional de la plataforma

A continuación se muestra un esquema a alto nivel de cada una de las partes en que se divide nuestro sistema, en puntos sucesivos se describirán cada una de ellas con más detalles:

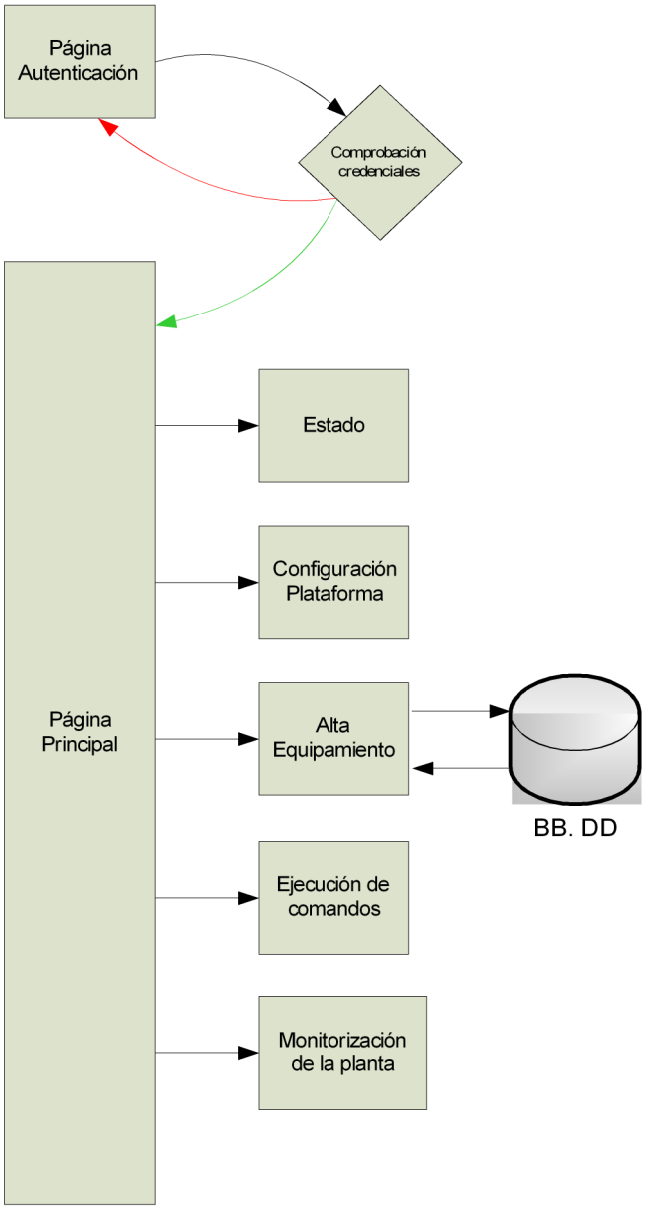

Ilustración 13: Esquema funcional del sistema

3.2 Partes de la plataforma

En este apartado se describen en detalle cada una de las partes en que se divide la misma.

3.2.1 Página de acceso

3.2.1.1 Descripción

Esta primera página será el punto de entrada, en ella el administrador deberá introducir sus credenciales para poder tener acceso a la gestión de los dispositivos.

3.2.1.1 Esquema

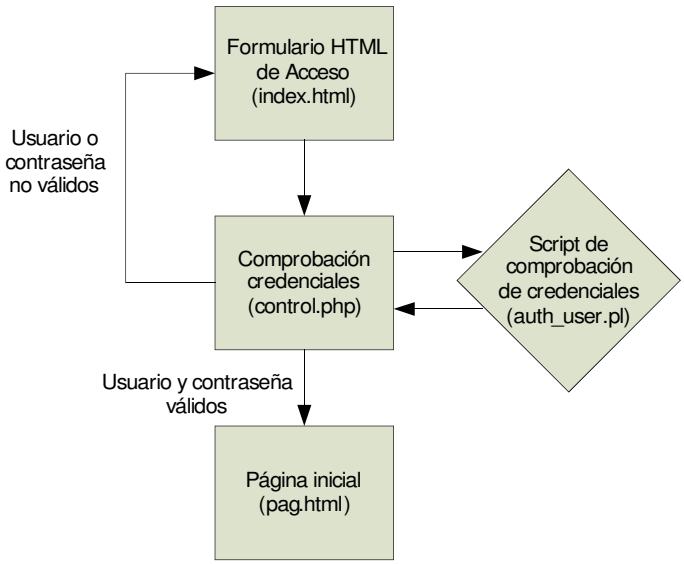

Ilustración 14: Esquema Página Acceso

3.2.1.2 Funcionamiento

Para poder acceder a la plataforma, el administrador ha de introducir sus datos de usuario y contraseña en el formulario que se le muestra (*index.html*). Este formulario enviará las credenciales a la página de autorización, la cual decidirá si permitir el acceso a la plataforma o redirigirlo a la página de inicio si las credenciales no son correctas. El código de esta página es el siguiente:

```
<?php

# Pagina que ejecuta un Script para
# comprobar las credenciales de acceso

    session_start();

#Tomamos los valores de usuario y contraseña
```

```
#que hemos introducido en el formulario

    $user=$_POST["usuario"];
    $pass=$_POST["contrasena"];

    $_SESSION['user']=$user;
    $_SESSION['pass']=$pass;
    $_SESSION['sesion']="OK";

# Ejecutamos el Script de comprobacion de datos

    system("./cgi/auth_user.pl $user $pass",$result);

#Comparamos el resultado y en función de éste
# accedemos a la web o volvemos a solicitar las
# credenciales

    if ($result == 1)
    {
        header ("Location: pag.html"); # Accedemos a la web
    } else
    {
        header ("Location: index.html?error=1"); # Volvemos a la
pagina de inicio
    }
?>
```

Como puede observarse, la página de autorización ejecuta un script en perl (*auth_user.pl*) que será el que realmente haga la comprobación de dichas credenciales. Éste script utiliza los módulos PAM de Linux para comprobar que las credenciales introducidas por el usuario se corresponden con algunas de las almacenadas en el sistema, es decir, se le permitirá el acceso al usuario siempre y cuando éste sea un usuario válido del sistema GNU/Linux que la sostiene pues dicha autenticación se produce contra el sistemas de cuentas del sistema. El código de este script es el siguiente:

```
#!/usr/bin/perl

use Authen::SimplePam;
use strict;

# Creamos las distintas variables que se utilizarán

my ($password, $user, $res, $service);
my $simple = new Authen::SimplePam;

# Recogemos las credenciales
```

```
$password = $ARGV[1];
$simple->password($password);

$service='proyect';
$simple->service($service);

$user = $ARGV[0];
$simple->user($user);

# Realizamos la autenticación

$res = $simple->auth_user;

# Si ha habido éxito en la comprobación
# devolvemos 1, si no, devolvemos 0

if ($res == 1)
{
    exit 1;
} else
{
    exit 0;
}
```

3.2.1 Página principal

3.2.1.1 Descripción

Esta página muestra información sobre el sistema, como son el nombre del equipo, la hora, la versión de sistema operativo o la versión de la base de datos instalada. De la misma forma, se presentan dos botones para el reinicio o el apagado de la plataforma.

3.2.1.2 Esquema

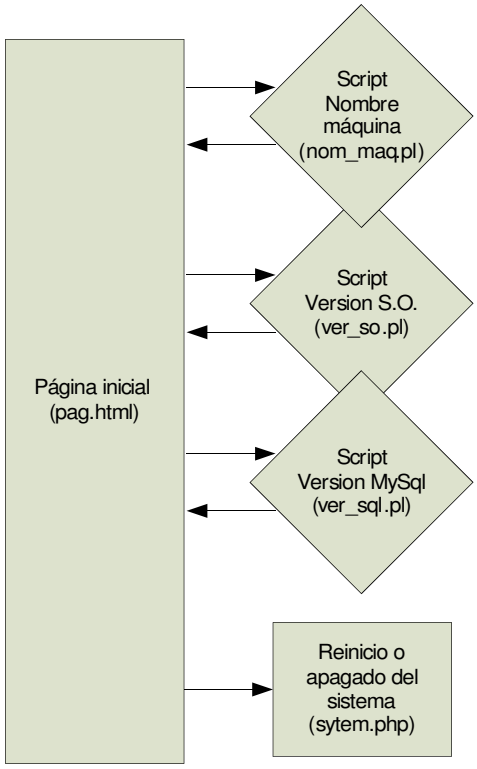

Ilustración 15: Esquema Página Inicio

3.2.1.3 Funcionamiento

Como hemos comentado anteriormente, esta primera página muestra cierta información sobre el sistema como es el nombre del mismo o la versión de sistema operativo. Para poder mostrar esta información, la página llama a distintos scripts que proporcionan la información que se pretende mostrar. A continuación, como ejemplo, se muestra el bloque de código que realiza la llamada al script para recoger el nombre de la máquina:

```
# Llamada al script para recoger el nombre de la máquina

<?php
    $maquina=exec("./cgi/nom_maquina.pl",$result);
    echo $maquina;
?>
```

Además de los datos comentados anteriormente, desde esta misma página se puede realizar tanto un apagado ordenado del sistema como un reinicio del mismo. Para

ello la página de inicio dispone de un formulario con dos botones que llaman a una segunda página (system.php) que ejecuta dichos comandos. El bloque de la página de inicio que realiza dicha llamada es:

```
# Bloque de código que realiza la llamada a la págian de reinicio
del sistema

<form name="System" action="system.php" method="POST">
    <td><input name="syscomand" type="submit" value="Reiniciar el
equipo"></td>
    <td><input name="syscomand" type="submit" value="Apagar el
equipo"></td>
</form>
```

La página *system.php*, ejecuta los comandos de reinicio o apagado del sistema dependiendo del valor de la variable *syscomand* pasada desde el formulario. El código de esta página es el siguiente:

```
<?php
# Página que ejecuta los comandos de reinicio o apagado

    session_start();

    if(!isset($_SESSION['sesion']))
    {
        header('location:log_out.php');
    }else
    {
        if($_POST["syscomand"]=="Reiniciar el equipo")
        {
            echo "<a align=\"center\" class=\"menu2\">El equipo
esta reiniciandose";
            echo "</a>";
            exec("sudo reboot",$result);

        }else if($_POST["syscomand"]=="Apagar el equipo")
        {
            echo "<a>El equipo se apagara en breves momentos";
            echo "</a>";
            exec("sudo shutdown -h now",$result);

        }
    }
?>
```

3.2.2 Página de configuración de la interfaz de red

3.2.2.1 Descripción

Mediante esta página podremos configurar todos los parámetros de red necesarios para poder prestar el servicio, como serán la IP de la interfaz, su máscara, su ruta por defecto para alcanzar a aquellos equipos que no se encuentren en su misma red y el servidor de DNS que será utilizado para alcanzar aquellos equipos de los que no se disponga su IP sino su nombre.

3.2.2.2 Esquema

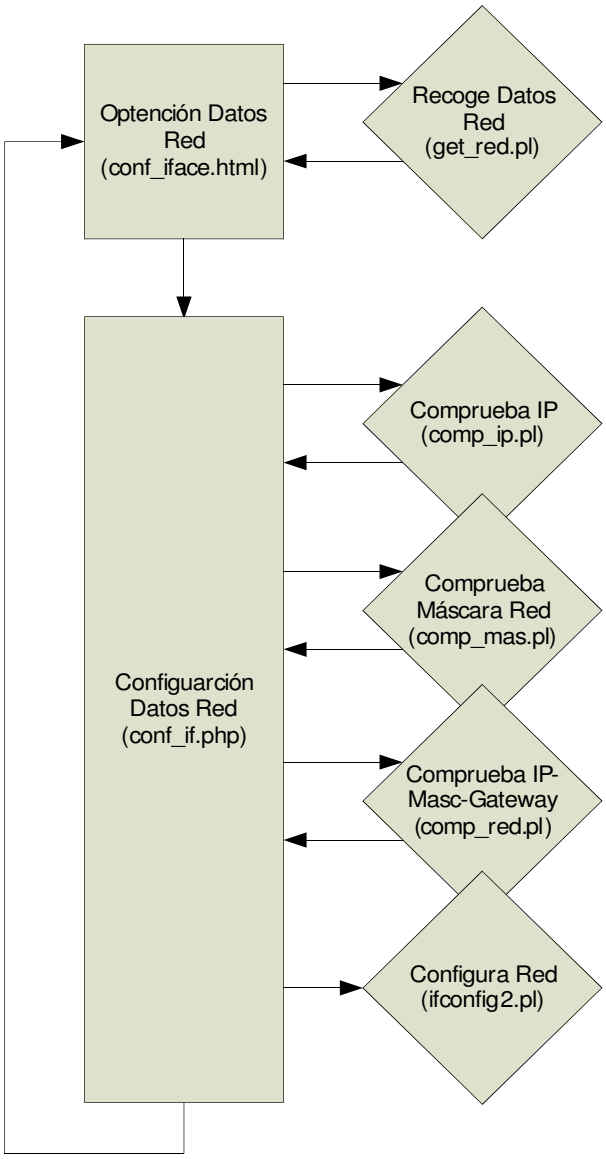

Ilustración 16: Página Configuración de la Interfaz de Red

3.2.2.3 Funcionamiento

En esta página es posible configurar los datos de red necesarios para poder acceder a la plataforma. En dicha página se muestra un formulario en el que se observan los valores que tiene en esos momentos la interfaz de red a través de la que se presta el servicio. Para poder mostrar estos datos, la página llama un script en perl (*get_red.pl*) el cuál devuelve todos los datos de red configurados en esos momentos. A continuación se muestra la parte de código que realiza esta llamada:

```
<?php
    $ip=exec("./cgi/get_red.pl",$result);
    $result=split(" ",$ip,4); #
?>
```

Este script realiza un tratamiento del archivo *"interfaces"*, es decir, recorre el archivo buscando los parámetros de IP, máscara, ruta por defecto y DNS configurados en el dispositivo. Si encuentra alguno de estos parámetros, separa la línea en sus distintas palabras y guarda el valor de la segunda en un vector, el cuál será devuelto a la página principal para su tratamiento y posterior visualización. El código de este script es el siguiente:

```
#!/usr/bin/perl -w
#Script que recoge los datos de la interfaz de red

use strict;

# Definición de las variables
my $linea;
my @salida=”0,0,0,0”;
my @aux;
my $i=0;

# Se abre el fichero “interfaces” para su tratamiento
open (FIENT, "/etc/network/interfaces");

# Se recorre el fichero completo
while($linea = <FIENT>)
{
    chop($linea);

    if($linea=~/address/) # Se guarda el valor de la dirección IP
    {
        @aux=split(" ",$linea);
          $salida[0]=$aux[1];
```

```
    }elsif($linea=~/netmask/) # Se guarda el valor de la máscara de
red
    {
        @aux=split(" ",$linea);
        $salida[1]=$aux[1];

    }elsif($linea=~/gateway/) # Se guarda el valor de la puerta de
enlace
    {
        @aux=split(" ",$linea);
        $salida[2]=$aux[1];

    }elsif($linea=~/dns-nameservers/) # Se guarda el valor del DNS
    {
        @aux=split(" ",$linea);
        $salida[3]=$aux[1];
    }
}

# Se cierra el fichero "interfaces"
close FIENT;

# Se devuelven cada uno de los valores obtenidos
print "@salida\n";
```

Una vez devuelto los datos, estos son tratados y mostrados en el formulario para su comprobación y modificación por parte del usuario. El código de este formulario es el siguiente:

```
# Formulario para modificar los valores de red de la plataforma
<form name="ConfIface" action="conf_if.php" method="POST" align="cen
ter">

        <table align="center">
        <tr>
        <td>

        <?php

        # Mostramos los errores que hay en los datos introducidos
                if($_GET['error']==1)
                {
                echo "<p class=\"menu\" style=\"color:red\">Los
datos introducidos no                    son validos. Por favor,
introdúzcalos de nuevo";

            }else if($_GET['error']==2)
```

```
                {
                echo "<p class=\"menu\" style=\"color:red\">La
puerta de enlace no                                    se encuentra en
la misma red que la ip";
                }
        ?>

        </td>
        </tr>

      # Formulario
        </table>
        <table align="center">
        <tr align="left">

        <th class="etiquetasform">IP:</th>
                <td><INPUT name="ip_iface" MAXLENGHT="13" value="<?
echo                                  $result[0];?>" type="text"</td>
        </tr>
        <tr align="left">
                <th class="etiquetasform">MASCARA:</th>

 <td><INPUT name="mask_iface" MAXLENGHT="15" value="<? echo
                        $result[1];?>" type="text"></td>
        </tr>
        <tr align="left">
                <th class="etiquetasform">GATEWAY:</th>

 <td><INPUT name="gate_iface" MAXLENGHT="15" value="<? echo
                        $result[2];?>" type="text"></td>
        </tr>
        <tr align="left">
                <th class="etiquetasform">DNS:</th>

 <td><INPUT name="dns_iface" MAXLENGHT="15" value="<? echo
                        $result[3];?>" type="text"></td>
        </tr>
        <tr align="center">

 <td colspan="2"><INPUT name="Enviar" type="submit"
                                value="Enviar"></td>
        </tr>
        </table>
    </form>
```

Para poder modificar estos valores, el usuario, una vez cambiados los datos del

formulario, tan solo deberá pulsar el botón de *"enviar"*, con este gesto, se provoca la llamada a una página en PHP que realizará los siguientes procesos:

a) Comprobar la validez de las IPs introducidas: Para ello se realiza una llamada a un script en perl (*comp_ip.pl*) el cual comprueba que cada uno de los octetos de la dirección IP son iguales o inferiores a 255. El código de dicho script es el siguiente:

```
#!/usr/bin/perl -w
# Script que comprueba las direcciones IP

use strict;

# Definición de las variables
my $ip=$ARGV[0];
my $expresion="25[0-5]|2[0-4]\\d|[01]?\\d?\\d";

# Comprobación de que la IP tiene el formato correcto
if($ip=~ /^($expresion)\.($expresion)\.($expresion)\.
($expresion)$/)
{
    print "0"; # se devuelve 0 si es distinto
}else
{
    print "1"; # Se devuelve 1 si es correcto el formato
}
```

b) Comprobar la máscara de red: Para ello se realiza una llamada a un script en perl (comp_masc.pl) que comprueba que los octetos que componen la máscara de red son valores iguales a 255, si existe alguno que no lo sea, comprueba que sean los valores 254, 252, 248, 240, 224, 192, 128 o 0, y que tras ellos tan solo existe octetos igual a 0, es decir que las máscaras 255.0.192.0.0 o 255.255.200.0 no serían válidas. El código de dicho script es el siguiente:

```
#!/usr/bin/perl -w
# Script que comprueba la máscara de red

use strict;

# Definición de las variables
my $masc=$ARGV[0];
my $error=0;
my @aux=split(/\./,$masc);
my $i=0;
```

```
# Se cuentan cuántos octetos son iguales a 255

while($aux[$i]=="255" && $i<3)
{
    $i++;
}

# Se comprueba que los octetos sólo pueden valer 255, 254,
# 252, 248, 240, 224, 192, 128 o 0

if(!($aux[$i] eq "255" || $aux[$i] eq "254" || $aux[$i] eq "
      252" || $aux[$i] eq "248" || $aux[$i] eq "240" || $au
      x[$i] eq "224" ||$aux[$i] eq "192" || $aux[$i] eq "128
      " || $aux[$i] eq "0"))
{
    $error=1; # Si algún octeto es distinto,
              #la máscara de red NO es válida

}else
{
    if($i==3)
    {
        $error=0; # Se han comprobado todos los octetos por
                  # lo tanto la  máscara de red es correcta
    }else
    {
        $i++;

      # El último octeto comprobado es distinto de 255 y
      #de 0 por lo que a partir de él, todos han de ser 0

     while($aux[$i] eq "0" && $i<3)
        {
            $i++;
        }

      # Se comprueba que el último octeto es 0
        if($aux[$i] eq "0")
        {
            $error=0; # Se han comprobado todos los octetos
                      # por lo tanto la máscara de red es
                      # correcta
        }else
        {
            $error=1; # El último octeto NO es 0 por lo que
                      # la máscara de red NO es correcta
        }
```

```
    }
}

# Se devuelve el valor de la comprobación
print $error;
```

c) Se comprueba que la IP y la ruta por defecto se encuentran en la misma red: Como en las comprobaciones anteriores, se realiza una llamada a un script en perl (*comp_red.pl*). Para ello, se descomponen cada uno de los valores en 4, 1 por cada octeto que compone una dirección IP o una máscara de red, y se pasan a binarios mediante una función de perl. Se vuelven a componer las direcciones IPs con los valores obtenidos y se hace un AND lógico con cada una de ellas y la máscara para obtener la dirección de red a la que pertenecen. Una vez obtenidas, se realiza otro AND entre ellas para comprobar que son iguales. Si todo está correcto se devuelve un 1, si no, se devuelve un 0. El código de este script es el siguiente:

```
#!/usr/bin/perl -w
# Script que comprueba que la IP de la interfaz y su ruta
# por defecto se encuentran en la misma red

use strict;
use integer;

# Definicion de variables

my $entrada='';
my $i=0;
my $ip = $ARGV[0];
my $masc = $ARGV[1];
my $gate = $ARGV[2];
my $salida='';
my $error;

# Seleccion de los obtetos de las ips

my @aux=split(/\./,$ip);
my @aux1=split(/\./,$masc);
my @aux2=split(/\./,$gate);

# Creación de las IPs en modo binario

my $ip_bin=binar($aux[0]).binar($aux[1]).binar($aux[2]).bina
      r($aux[3]);
my $masc_bin=binar($aux1[0]).binar($aux1[1]).binar($aux1[2])
      .binar($aux1[3]);
```

```
my $gate_bin=binar($aux2[0]).binar($aux2[1]).binar($aux2[2])
      .binar($aux2[3]);

# Cálculo de la red

my $red1 = $ip_bin & $masc_bin;
my $red2 = $gate_bin & $masc_bin;

# Comprobación de que la IP y la ruta por defecto se
# encuentran en la misma red

if($red1 eq $red2)
{
    $error=0;
}else
{
    $error=1;
}

print $error;

#############################################################
#
# Subrutina que calcula el valor en 8bits
# de un valor decimal
#############################################################
#

sub binar
{
    # Definición de las variables
    $entrada = $_[0];
    $i=0;
    $salida='';

    # Creación del número binario

    while ( $i < 8 )
    {

        # En $salida vamos agregando los restos (0 o 1) de
        # dividir la $entrada entre 2.
        # Esto es lo mismo que extraer el 1er bit de
        # $entrada.

        $salida   = ($entrada % 2) . $salida;
```

```
                # Como ya está analizado el bit menos significativo,
                # reducimos el valor de $entrada para la siguiente
                # vuelta del bucle.
                # Esto es lo mismo que desplazar el nº binario una
                # posición la derecha.

                $entrada /= 2;
                $i++;
            }

        return $salida;
        }
```

Una vez realizadas todas las comprobaciones y habiendo constatado que los valores introducidos son correctos, se procede a su modificación en el sistema. Para poder realizar esto, se ejecuta el siguiente trozo de código:

```
 # Modificación de la IP y de la máscara
exec("sudo /sbin/ifconfig lo $ip netmask $mask");

 # Modificación de la ruta por defecto
exec("sudo /sbin/route add default gw $gateway");
```

Para que estos cambios se mantengan cuando el equipo sea reiniciado, la página web ejecuta el script en perl "*ifconfig2.pl*", el cual modifica el fichero "interfaces" del sistema. El código de este script es el siguiente:

```
#!/usr/bin/perl -w
# Script que hace perennes los cambios realizados en las interfaces

use strict;

# Definición de las variables
my $ip=$ARGV[0];
my $masc=$ARGV[1];
my $gateway=$ARGV[2];
my $dns=$ARGV[3];
my $FICHERO="interfaces";
my $linea;

# Abrimos la plantilla y el fichero a crear
open (FIENT, "./fich/red");
open (FISAL, ">./fich/$FICHERO");

# Se recorre la plantilla completamente modificando
# los valores de la interfaz, de la máscara, de la ruta
```

```
# por defecto y la del DNS.

while($linea = <FIENT>)
{
    chop($linea);

    $_=$linea;

    s /IP/$ip/;
    s /MASC/$masc/;
    s /GATEWAY/$gateway/;
    s /DNS/$dns/;

    $linea=$_;
    print FISAL "$linea\n";
}

# Se cierran los ficheros utilizados
close FIENT;
close FISAL;

# Se sustituye el fichero actual por el que se ha creado
system("mv ./fich/$FICHERO /etc/network/interfaces");
```

3.2.3 Página de consulta del equipamiento

3.2.3.1 Descripción

Mediante esta página se puede realizar una consulta a la base de datos para conocer los datos de los distintos dispositivos alojados en ella, así como hacer modificaciones de los mismos.

3.2.3.2 Esquema

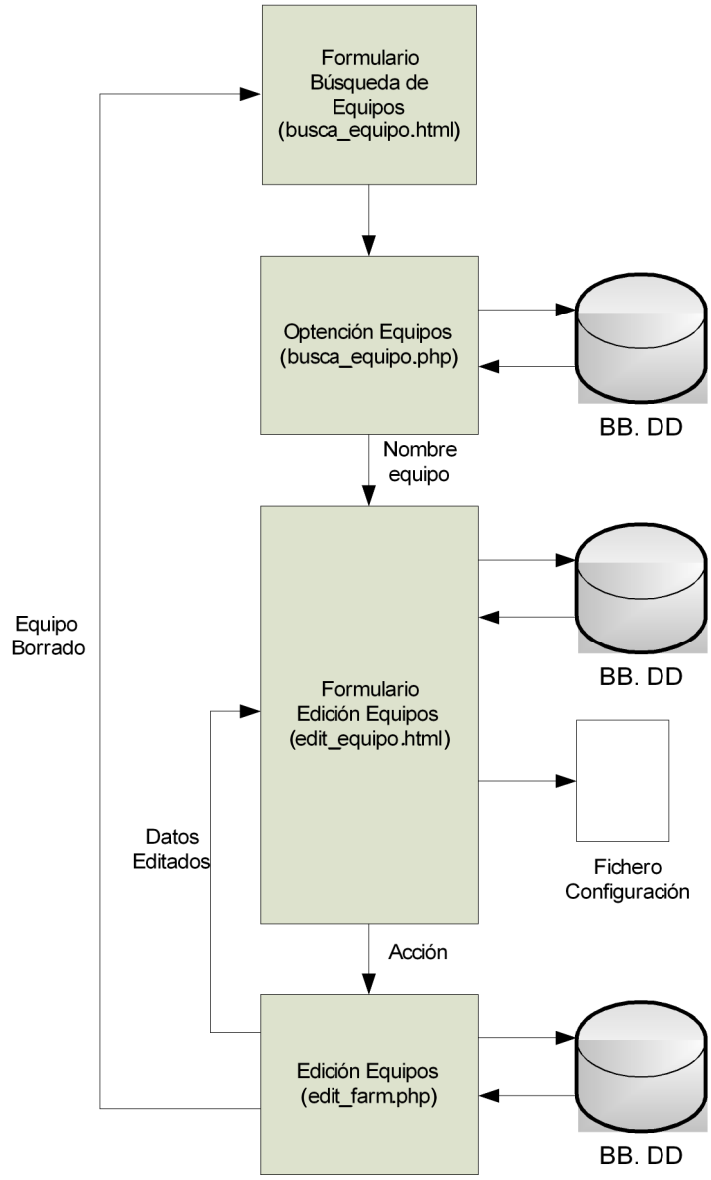

Ilustración 17: Esquema Página de Consulta del Equipamiento

3.2.3.3 Funcionamiento

Lo primero que se observa en esta parte de la plataforma, es un formulario (*busca_equipo.html*) a través del cual poder realizar la consulta a la base de datos para conocer los equipos guardados en ella. En este formulario es posible introducir distintos datos de los equipos como pueden ser: el nombre, la IP de la interfaz LAN, la IP de la interfaz WAN, localidad, provincia y/o teléfono.

Una vez introducidos los datos de búsqueda, y tras pulsar el botón *"consulta"*, se realiza una llamada a una página PHP (*busca_equipo.php*) que realizará una consulta a la base de datos con los distintos valores introducidos. Antes de poder lanzar esta consulta,

la plataforma ha de conectarse a la base de datos, las credenciales para acceder a ella serán las usadas por el administrador para poder gestionar el sistema, dichas credenciales están almacenadas en la sesión ***http***. El código que realiza la conexión a la base de datos es el siguiente:

```
# Conexión a la base de datos MySql
$db=mysql_connect($direccion,$_SESSION['user'],$_SESSION['pass']) OR
DIE (mysql_error());

# Uso de la base de datos propia del proyecto
mysql_query('USE proyecto') OR DIE (mysql_error());
```

Una vez que la plataforma se encuentra conectada a la base de datos, se procede a realizar la consulta. Esta consulta recoge todos aquellos equipos cuyos campos sean igual a los introducidos en el formulario. La parte de código que realiza esta acción es:

```
$result=mysql_query("SELECT id_gateway, nom_farm, soe, ip_wan, ip_lan FROM
      dat_farm,conf_farm WHERE id_gateway LIKE '$id' AND
      id_gateway_cf=id_gateway AND        ip_wan LIKE '$wan' AND ip_lan
      LIKE '$lan' AND soe LIKE '$num_soe' AND provincia        LIKE
      '$prov' AND soe LIKE '$num_soe'") OR DIE (mysql_error())
```

Tras obtener todos los dispositivos, la página PHP crea una tabla con los datos más significativos como son el nombre del equipo (*id_gateway*), el responsable del mismo (*nom*) y sus IPs (*ip_wan* e *ip_lan*). El campo "nombre del equipo" es un vínculo hacia el formulario de edición de dicho equipo, es decir, cuando se pincha sobre el nombre del equipo, el usuario es redirigido hacia la siguiente URL ***"./edit_equipo.html?farma=$nombre_equipo"***, donde *$nombre_equipo* es el nombre del dispositivo.

Una vez redireccionada la petición, la nueva página realizará una nueva consulta a la base de datos con el nombre del equipo y mostrará en la pantalla un formulario con todos sus datos. El código de la consulta es el siguiente:

```
$result=mysql_query("SELECT id_gateway, nom_farm, apell_farm, soe, ip_wan,
      masc_wan,    ip_lan, masc_lan, ciudad, provincia, dir, tlfno FROM
      dat_farm, conf_farm WHERE  id_gateway='$farma' AND
      id_gateway_cf=id_gateway") OR DIE (mysql_error());
```

En este formulario podrán ser modificados todos los valores a excepción del nombre del equipo. Para hacer que un valor de un formulario no pueda ser modificado, habrá que añadir la constante "*readonly*" en el código html:

```
<INPUT readonly name="Nombre" MAXLENGHT="13" type="text" value="VALOR">;
```

Para iniciar el proceso de la modificación de datos o borrado del equipo, el administrador tan solo deberá pulsar sobre el botón "actualizar" o "borrar" dependiendo de lo que se desee. Tras esta acción, el formulario pasará los datos a la página "*edit_farm.php*" que, dependiendo de la acción realizada, procederá a la actualización de los datos del equipo o al borrado del mismo de la base de datos. La parte del código PHP que realiza estas acciones es:

```
if($_POST["Accion"]=="Actualizar")
{
    $result=mysql_query("UPDATE conf_farm SET ip_wan='$ip_wan',
      masc_wan='$masc_wan', ip_lan='$ip_lan', masc_lan='$masc_lan'
      WHERE id_gateway_cf='$id'") OR DIE (mysql_error());

    $result=mysql_query("UPDATE dat_farm SET nom_farm='$nom',
      apell_farm='$apel', soe='$soe', ciudad='$localidad',
      provincia='$provincia', tlfno='$telef', dir='$calle',
      cod_post='$cp' WHERE id_gateway='$id'")
      OR DIE (mysql_error());

    header("Location:edit_equipo.html?farma=$id");

}else
{
    $result=mysql_query("DELETE FROM conf_farm WHERE
      id_gateway_cf='$id'")OR DIE  (mysql_error());

    $result=mysql_query("DELETE FROM dat_farm WHERE
      id_gateway='$id'") OR DIE (mysql_error());

    header("Location:busca_equipo.html");
    }
}
```

Una vez realizadas las modificaciones de los datos en la base de datos, la página redireccionará al usuario hacia la página del objeto modificado o a la página de búsqueda de equipos dependiendo de la opción tomada.

3.2.4 Alta de nuevos equipos

En la plataforma existen dos formas de dar de alta nuevos equipos: de forma individual o de forma masiva. A continuación de se describen ambas.

3.2.4.1 Alta de nuevos equipos de forma individual

3.2.4.1.1 Esquema

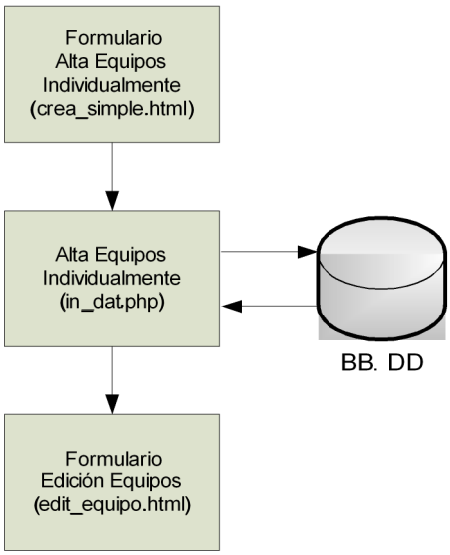

Ilustración 18: Esquema Alta Nuevos Equipos Individual

3.2.4.1.2 Funcionamiento

Para dar de alta un equipo de forma individual, el usuario deberá acceder a un formulario (*crea_simple.html*) en el que deberá introducir al menos los datos de nombre, tipo de equipo y teléfono para crear correctamente el objeto dentro de la base de datos.

Una vez rellenados al menos los datos obligatorios, y tras pulsar el botón de enviar, el formulario realiza una llamada a una página en PHP (*in_dat.php*) que se encarga tanto de comprobar que todos los datos obligatorios se han introducido de forma correcta como, tras ello, de dar de alta el equipo en la base de datos. El código PHP de esta página es el siguiente:

```
<?php
# Script que comprueba que los datos obligatorios han sido
# rellenados y da de alta los equipos de forma individual en la
# base de datos

# Comprobación de que los campos nombre, ip de la interfaz wan y
# teléfono no están vacíos
if(empty($_POST["farma"]) || empty($_POST["ip_wan"]) ||
                             empty($_POST["telefono"]) )
    {
        # Si están vacíos se vuelve a la página
```

```
            # de creación del equipo
            header ("Location: crea_equipo_simple.html?error=1");
        }else
        {
            # Definición de las variables
            $id=$_POST["farma"];
            $nom=$_POST["nom_farma"];
            $apel=$_POST["apel_farma"];
            $ip_wan=$_POST["ip_wan"];
            $masc_wan=$_POST["masc_wan"];
            $ip_lan=$_POST["ip_lan"];
            $masc_lan=$_POST["masc_lan"];
            $soe=$_POST["num_soe"];
            $provincia=$_POST["provincia"];
            $localidad=$_POST["localidad"];
            $telef=$_POST["telefono"];
            $calle=$_POST["calle"];
            $cp=$_POST["cp"];
            $tipo=$_POST["tipo"];

            $direccion = 'localhost';

            # Conexión a la base de datos
            $db=mysql_connect($direccion,$_SESSION['user'],
                $_SESSION['pass']);
            mysql_query('use pruebas') OR DIE (mysql_error());

            # Dar de alta los equipos en las dos tablas
            mysql_query("INSERT INTO dat_farm (id_gateway, nom_farm,
                apell_farm, soe, provincia, ciudad,dir,tlfno) VALUES
                '$id', '$nom', '$apel', '$soe', '$provincia',
                '$localidad', '$calle',
                '$telef')")OR DIE(mysql_error());

            mysql_query("INSERT INTO conf_farm (id_gateway_cf, ip_wan,
                masc_wan, ip_lan, masc_lan, tipo) VALUES ('$id',
                '$ip_wan',' $masc_wan', '$ip_lan', '$masc_lan',
                '$tipo')") OR DIE   (mysql_error());

            # Se cierra la conexión con la base de datos
            mysql_close($db);

            # Se redirige a la página de edición de los equipos
            header ("Location: edit_equipo.html?farma=$id");
        }
    }
?>
```

Como se puede observar, tras la creación del objeto en la base de datos, se redirige al usuario a la página de edición de equipos para que rellene o modifique, si se desea, los datos introducidos.

3.2.4.2 Alta de nuevos equipos de forma masiva

3.2.4.2.1 Esquema

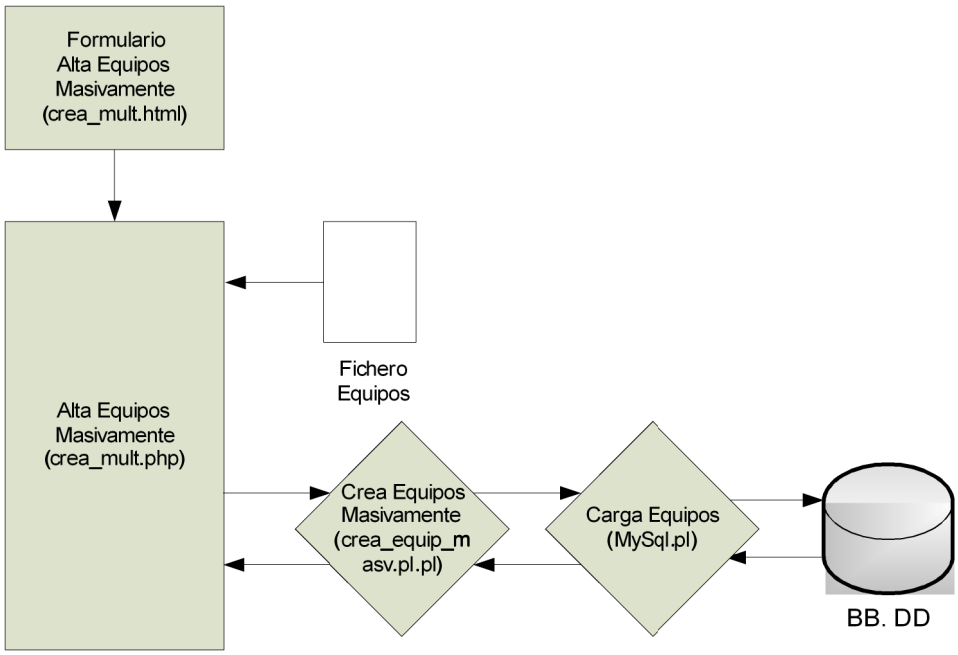

Ilustración 19: Esquema Alta Nuevos Equipos Masivo

3.2.4.2.2 Funcionamiento

Para poder hacer uso de esta funcionalidad, existe un enlace en la página de creación de equipos de forma individual, que nos remite a ella.

Para dar de alta los dispositivos tan solo se tendrá que subir un fichero con el formato indicado en dicha página web y pulsar sobre el botón "*Subir*", esta acción realizará una llamada a una página web (*crea_equipo_mult.php*) que realizará la carga del fichero y una llamada a un script en perl que realizará su almacenamiento en la base de datos. El código de dicha página es el siguiente:

```
<html>
<head>
    <title>Subir fichero</title>
    link rel="STYLESHEET" type="text/css" href="estilos_admin.css">
</head>
```

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

      <?php

          $path="./fich/"; # Directorio en el que se subirá el
                           # archivo

          # Datos del arhivo
          $nombre_archivo = $HTTP_POST_FILES['userfile']['name'];
          $tipo_archivo = $HTTP_POST_FILES['userfile']['type'];
          $tamano_archivo = $HTTP_POST_FILES['userfile']['size'];

          # Datos del usuario que ha creado la sesión
          $usuario=$_SESSION['user'];
          $pass=$_SESSION['pass'];

          # Se comprueba que el archivo es un fichero de texto y
          # tiene un tamaño menor de 100k
          if (!
      (strpos($tipo_archivo,"plain")) && ($tamano_archivo < 100000)
      )
          {

              echo "La extension o el tamano de los archivos no es
                   correcta. <br><br><table><tr><td><li>Solo se
                   permiten archivos .txt<br><li>de 100 Kb
                   maximo.</td></tr></table>";

          }else
          {

              # Se procede a subir el fichero
              if (move_uploaded_file($HTTP_POST_FILES['userfile']
      ['tmp_name'], $path.$nombre_archivo))
              {
                       $comand_out=exec("./cgi/crea_equipo_masivo.pl
                         $usuario $pass$nombre_archivo",$result);

                   # Se comprueba que el fichero tiene la sintaxis
                   # correcta
                   if($comand_out != 0)
                       {
                       echo "Imposible cargar los datos en la base
                         de datos, compruebe la sintaxis del
                         fichero";
```

```
                    }else
                    {
                        echo "Para poder visualizar si los equipos
                          han sido cargados correctamente pulse <a
                          href=\"./fich/alta_equipos.txt\">Aqui</a>";
                    }
                }else
                {
                        echo "Ocurrio un error al subir el fichero.
                          No pudo guardarse. Por favor intentelo mas
                          tarde.";
                }
            }
            ?>

        </div>
        </body>
        </html>
```

Como se observa en el código anterior, se realizan distintas comprobaciones antes de proceder a la carga del fichero como son que el tamaño y formato del archivo sea el correcto. Tras ello se realiza una llamada al script "*crea_equipo_masivo.pl*" el cual realizará la carga propiamente dicha de los datos.

El código de este archivo es el siguiente:

```
#!/usr/bin/perl -w
# Script que de alta equipos de forma masiva

use strict;

# Definición de las variables a utilizar
my $linea;
my $user=$ARGV[0];
my $pass=$ARGV[1];
my $error=0;
my $error2=0;

# Apertura de los ficheros que se utilizarán para la carga de los
# datos y su resultado
open (FIENT, "./fich/$ARGV[2]");
open (FISAL, ">./fich/alta_equipos.txt");

# Recorremos todo el fichero
while($linea = <FIENT>)
{
    #  Recogemos los datos de cada una de las líneas
```

```
            chop($linea);

 (my $id_gateway, my $user_eq, my $pass_eq, my $ipwan, my $mascwan,
my$iplan, my $masclan, my$tipo)=split(";",$linea);

    # Ejecutamos un script para realizar la carga de los datos en
    # la BB.DD.
    $error=`./cgi/mysql.pl 1 $user $pass $id_gateway $user_eq
      $pass_eq $ipwan $mascwan $iplan $masclan $tipo`;

    print FISAL "$id_gateway        $error\n";
}

close FIENT;
close FISAL;
```

Este script realiza el tratamiento del fichero que contiene los datos del equipo, ejecuta el script (*mysql.pl*) que carga dichos datos en la BB.DD. y crea un fichero de salida en el que guarda el resultado generado al cargar un nuevo dispositivo.

Por su parte, el script mysql, realiza más funciones que sólo la de cargar los datos de los nuevos dispositivos en la base de datos debido a que es llamado por otros scripts de la plataforma. Por este motivo, las funciones que realiza este script son:

a) Carga de nuevos equipos: El primer paso de este proceso es la comprobación de las IPs de los dispositivos. Tras ello, se crea el objeto en la tabla *dat_farm* y se comprueba que no ha habido error. Si todo ha ido bien, se procede a la creación del objeto en la tabla *conf_farm*. Si al proceder a crear éste, se produjese algún error, se procede al borrado del objeto creado en la tabla *dat_farm*. La parte del código del script que realiza esto es:

```
# Se prepara la creación del objeto en la tabla dat_farm
$sth = $dbh->prepare("INSERT INTO dat_farm
      id_gateway,user,passwd)     VALUES
      ('$ARGV[3]','$ARGV[4]','$ARGV[5]')") or die("Imposible
      preparar la sentencia");

# Se ejecuta la sentencia
$sth->execute();

# Se comprueba si ha habido ningún error
if($sth->err == 1062)
{

    $error=$sth->errstr;     # Si ha habido error, éste se
```

```
                                    # asocia a la variable a
                                    # devolver
}else
{

    # Si no ha habido error, se prepara y se ejecuta la
    # creación del objeto en la tabla conf_farm
    $sth = $dbh->prepare("INSERT INTO conf_farm
      (id_gateway_cf, ip_wan, masc_wan, ip_lan, masc_lan,
      tipo) VALUES ('$ARGV[3]', '$ARGV[6]', '$ARGV[7]',
      '$ARGV[8]', '$ARGV[9]', '$ARGV[10]')");

    $sth->execute();

    # Se comprueba que no ha habido ningún error
      if($sth->err == 1062)
       {
         $error=$sth->errstr;     # Si ha habido error, éste
                                  # se asocia a la variable a
                                  # devolver

         # Debido al error, se borra el objeto de
         # la tabla dat_farm
         $sth = $dbh->prepare("DELETE FROM dat_farm WHERE
             id_gateway = '$ARGV[3]'");
         $sth->execute();

    }else
    {
         # Si no ha existido ningún error, se da por correcta
         # la creación del objeto

         $error="Equipo dado de alta correctamente";
    }
}

$sth->finish();
```

b) Actualización del estado de los equipos apagados: Esta parte del script actualiza el estado de los dispositivos que la plataforma detecta que se encuentran apagados, para detectar el estado se realiza un intento de comunicación a través de un paquete ICMP. La parte del script que realiza esto es:

```
# Se prepara la sentencia que actualizará
# el estado de los equipos
```

```
$sth = $dbh->prepare("UPDATE conf_farm SET status='0' WHERE
      id_gateway_cf='$id_gateway'");

# Se ejecuta la sentencia
$sth->execute();
$sth->finish();
```

3.2.5 Ejecución de comandos de forma remota

3.2.5.1 Descripción

Mediante esta página, el usuario será capaz de ejecutar distintos tipos de comandos en los equipos alojados en la base de datos dependiendo del tipo de equipo que sea.

3.2.5.2 Esquema

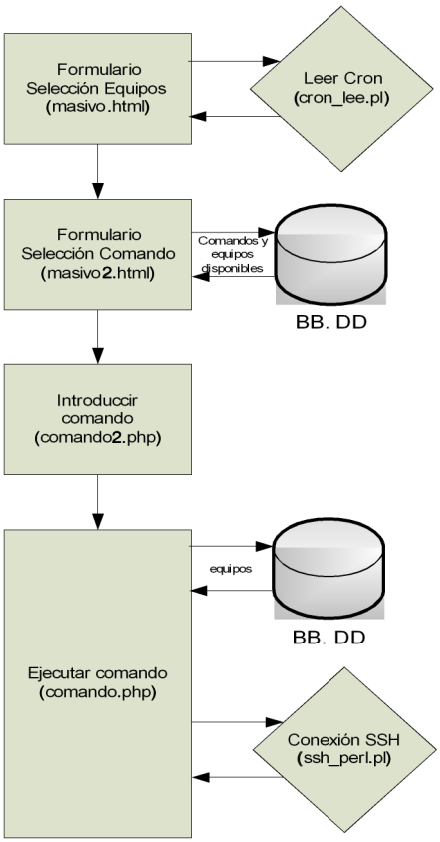

Ilustración 20: Esquema Ejecución de Comandos Remotamente

3.2.5.3 Funcionamiento

Este módulo de la plataforma comienza con un formulario (*masivo.html*) en el que

el administrador puede seleccionar distintos registros de la base de datos con los que poder filtrar los equipos sobre los que se va a ejecutar el comando. Los registros disponibles son el nombre del equipo, la localidad o la provincia en la que se encuentra, la IP de la interfaz externa del equipo o el tipo de equipo que es. Para poder mostrar este último registro, la página html realiza una consulta a la base de datos, lo que nos permite independizar la creación de la página de los distintos tipos de dispositivos que puede albergar la base de datos. La parte del código que realiza la consulta SQL y muestra los distintos tipos de equipos puede verse a continuación:

```
<?php

    # Conexión a la base de datos
    $db=mysql_connect($direccion,$_SESSION['user'],
      $_SESSION['pass']) OR DIE                (mysql_error());
      mysql_query($bbdd) OR DIE (mysql_error());

    # Selección de los distintos tipos de equipos
    $result=mysql_query("SELECT DISTINCT tipo FROM
      conf_farm") OR DIE                       (mysql_error());

    # Impresión por pantalla de los datos recogidos de la BBDD
    echo "<td class=\"etiquetasform\" align=\"top\">Tipo:</td>";
    echo "<td><SELECT name=\"tipo\">";
    while ($row = mysql_fetch_row($result))
    {
        echo "<OPTION VALUE=\"$row[0]\">$row[0]</OPTION>";
    }

    # Cierre de la conexión a la BBDD
    mysql_close($db);
?>
```

Una vez relleno el formulario con los datos con los que se desea realizar el filtrado de equipos sobre los que ejecutar el comando, el administrador debe pulsar en "*Continuar*" para seguir con el proceso. En el segundo formulario (*masivo2.html*) se recogen los datos introducidos en el primero y se utilizan para realizar una consulta SQL y así conocer sobre qué equipos se ejecutará el comando. La consulta SQL utilizada es la siguiente:

```
$result=mysql_query("SELECT DISTINCT id_gateway_cf FROM dat_farm,
      conf_farm WHERE dat_farm.id_gateway LIKE '$id' AND
      id_gateway_cf=id_gateway AND conf_farm.ip_wan LIKE '$wan' AND
      dat_farm.ciudad LIKE '$local' AND dat_farm.provincia LIKE
      '$prov' AND conf_farm.tipo='$tipo'");
```

Una vez realizada la consulta y mostrado su resultado en la página mediante una tabla, este mismo formulario realiza otra consulta SQL para conocer qué comandos pueden ser ejecutados sobre los equipos anteriormente mostrados. La sentencia SQL que

recoge dichos comandos es:

```
$result2=mysql_query("SELECT mnemo,comando FROM comando WHERE tipo
      LIKE '$tipo'");
```

El resultado de esta sentencia se muestra en forma de campo de selección múltiple para que el administrador pueda una páginaa elegir el comando que desee ejecutar.

Tras pulsar sobre el botón continuar, este segundo formulario llama a un tercero (*comando.php*) al cual se le pasan los datos referentes a los equipos sobre los que ejecutar el comando y el comando seleccionado por el administrador, con este último el tercer formulario realiza una consulta a la base de datos donde se encuentran todos los comandos y presenta en pantalla la estructura que se deberá introducir en el cuadro de texto anexo para poder ejecutar el comando de una forma correcta, indicando con letras minúsculas cuáles son los valores propios del comando y en mayúsculas aquellas variables que pueden ser modificadas para su ejecución.

Una vez introducido el comando en el cuadro de texto, para que éste sea ejecutado en los equipos, el administrador debe pulsar sobre el botón "*Ejecutar*" el cuál desencadenará una llamada a una página PHP (comando2.php) que recogerá los datos de los equipos en los que ejecutar el comando y el propio comando. Con los datos de los equipos, se realiza una consulta como la que se realiza en la página "*masivo2.html*" para recoger los equipos sobre los que ejecutar el comando.

Una vez obtenidos, se realiza una llamada al script en perl "*ssh_perl.pl*" con la IP del equipo, el usuario y la contraseña del mismo y el comando a ejecutar. Con estos datos, el script realiza la conexión al equipo mediante ssh, ejecuta el comando proporcionado por el administrador y devuelve el resultado del mismo. El código perl de este script es el siguiente:

```
#!/usr/bin/perl -w
# Script que ejecuta un comando en un equipo remoto

# Liberías a utilizar
use Net::SSH::Expect;

# Definición de las variables
my $result;
my $ssh_server=$ARGV[0];
my $ssh_user=$ARGV[1];
my $ssh_password=$ARGV[2];
my $ssh_comand=$ARGV[3];

# Creación de la variable SSH
my $ssh = Net::SSH::Expect->new (
    host => $ssh_server,
    password=> $ssh_password,
```

```
    user => $ssh_user,
    raw_pty => 1
);

# Autenticación en el equipo remoto
my $login_output = $ssh->login();

# Se comprueba que la autenticación NO ha fallado
if ($login_output =~ /denied/){

    print "Usuario o contrasenas NO son validos";
}else
{

    # Se ejecuta el comando
    $result = $ssh->exec("$ssh_comand");

    $ssh->close();

    # Se devuelve el resultado a la consola de gestión
    print "$result";

}
```

Una vez devuelto el resultado del comando a la página "*comando2.php*", ésta crea una tabla con el nombre de los equipos y el resultado obtenido al ejecutar el comando en cada uno de los equipos. El código PHP que realiza la llamada al script y a su vez crea la tabla con los resultados, es el siguiente:

```
## Código que realiza la llamada al script perl y muestra el
# resultado en una tabla

# Bucle para ejecutar el comando en cada uno
# de los dispositivos seleccionados
while ($row = mysql_fetch_row($result))
{
    $equipo[$i]=$row[0];

    # Se realiza la llamada al script y
    # se guarda su resultado en un vector
    $comand_out[$i]=`./cgi/ssh_perl.pl $row[1] $row[2] $row[3]
      $comand`;
    $i++;
}

# Se crea el encabezado de la tabla
echo "<table align=\"center\" border = '1'>";
```

```
echo "<tr>";
echo "<td align=\"center\"><b>Dispositivo</b></td>";
echo "<td><b>Resultado de ejecutar el comando</b></td>";
echo "</tr>";

# Se rellena la tabla con el nombre de cada uno de los dispositivos
# y el resultado de ejecutar el comando en cada uno de ellos
while($i!=0)
{
    echo "<tr>";
    echo "<td align=\"center\">$equipo[$j]";
    echo "<td align=\"center\">$comand_out[$j]";
    $i--;
    $j++;
}
```

3.2.6 Configuración del CRON del sistema

3.2.6.1 Descripción

En esta página de la plataforma será posible configurar el CRON del usuario que se encuentra ejecutando la misma con aquellos comandos que se consideren oportunos.

3.2.6.2 Esquema

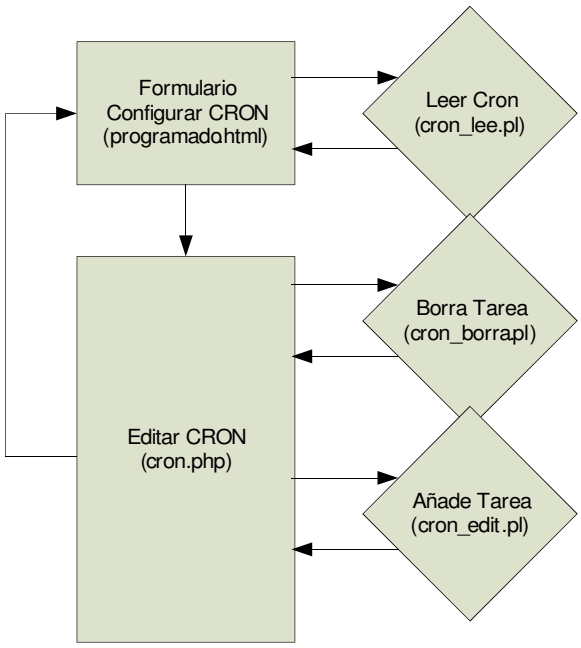

Ilustración 21: Esquema Configuración CRON sistema

3.2.6.3 Funcionamiento

La página a la que se accede para configurar el CRON se encuentra dividida en dos partes bien diferenciadas, una primera en la cual se muestra la configuración actual del CRON y otra segunda, un formulario en el cual dar de alta el comando o borrar aquel que se desee.

Para poder mostrar la configuración actual del CRON del usuario, la página ejecuta un script en perl, *cron_lee.pl*, que lee el cron del usuario y crea un nuevo archivo con la información obtenida de él. Este nuevo archivo es abierto en modo lectura por la página y utilizado para imprimir por pantalla el contenido del CRON del usuario. La parte del código de la página web que realiza esta función es la siguiente:

```
# Código PHP que ejecuta el script en perl y abre el fichero
# con la configuración del CRON del usuario
<?php
    # Ejecución del script en perl
    system("./cgi/cron_lee.pl");

    # Apertura del fichero auxiliar de
    # configuración del CRON del usuario
    $Descriptor1 = fopen("./fich/miguel","r");
    $i=1;
?>
```

El script *cron_lee.pl* ejecutado por la página hace uso del módulo Config::Crontab, este módulo es usado para la manipulación del CRON de cualquier usuario, con él es posible crear, borrar, leer o escribir un CRON de un usuario. El código de este script es el siguiente:

```
#!/usr/bin/perl -w
# Script que lee el fichero CRON de un usario

use Config::Crontab;
use strict;

# Difinición de la variable CRON
my $ct = new Config::Crontab(-owner =>'miguel');

# Lectura del fichero CRON
$ct->read;

# Creación del fichero auxiliar
$ct->file('./fich/miguel');

# Escritura en el fichero auxiliar de la configuración del usuario
```

```
$ct->write;
```

En la segunda parte de esta página se muestran las acciones que se pueden realizar para la manipulación del CRON del usuario:

a) Creación de una nueva tarea: Para poder dar de alta una nueva tarea en el CRON, el usuario deberá elegir la frecuencia con la que ejecutarla y escribir el correspondiente comando.

b) Borrado de una tarea: Para poder realizar el borrado de una tarea, el usuario tan solo deberá escribir en el espacio indicado la tarea que desea borrar y pulsar sobre "*Borrar Tarea*".

Una vez optado por una acción, la página web realizará una llamada a la página "*cron.php*" que será la encargada de la realización de dicha tarea. En primer lugar realizará la comprobación de la acción a tomar y dependiendo de ésta se realizará lo siguiente:

a) Borrado de una tarea: Al pulsar sobre el botón tarea, se realiza una llamada al script en perl "*cron_borra.pl*" pasándole como parámetro el usuario que se encuentra autenticado en la página web y la tarea a borrar. Por su parte, el script "cron_borra.pl" abrirá el CRON del usuario, leerá la parte del CRON que contiene la tarea a borrar y procederá a su borrado. El código del script "cron_borra.pl" es el siguiente:

```
#!/usr/bin/perl -w
#Script encargado de borrar una tarea en el CRON del usuario

use Config::Crontab;
use strict;

# Definición de las variables a usar
my $user=$ARGV[0];
my $sec=$ARGV[1];
$sec =~ tr/a/*/;

# Apertura del CRON del usuario
my $ct = new Config::Crontab(-owner =>$user);

# Lectura del CRON del usuario
$ct->read;

# Lectura del bloque concreto que contiene la tarea a borrar
my $oldblock = $ct->block($ct->select(-data_re => $sec));
```

```
# Borrado de la tarea
$ct->remove($oldblock);

# Escritura del nuevo CRON
$ct->write;

# Cierre del CRON
$ct->close;
```

b) Añadir tarea: Si por el contrario se pulsa sobre el botón añadir tarea, se realizará una llamada al script "*cron_edit.pl*" el cual, tras recoger los parámetros de nombre de usuario, tiempo a programar y comando a ejecutar, leerá el fichero CRON del usuario y añadirá al final de éste el bloque que deberá ejecutarse. El código perl de dicho script es:

```
#!/usr/bin/perl -w
# Script que añade una nueva tarea en el CRON del usuario

use Config::Crontab;
use strict;


# Definición de las variables
my $user=$ARGV[0];
my $horario=$ARGV[1];
my $comando=$ARGV[2];
my $fich=$ARGV[5];
my $comando_cron;
$horario =~ tr/a/*/;

$comando_cron=$horario." ".$comando;

# Se lee el CRON del usuario
my $ct = new Config::Crontab(-owner =>$user);
$ct->read;

# Se crea el la tarea a añadir
my $block = new Config::Crontab::Block(-data =
>$comando_cron);

# Se apunta al último bloque del fichero
$ct->last($block);

# Se escribe la tarea
$ct->write;
```

```
# Se cierra el fichero CRON
$ct->close;
```

3.2.7 Bloque dedicado a la monitorización

3.2.7.1 Introducción

Esta parte de la plataforma de gestión se encuentra dedicada a la monitorización de los equipos, en ella se comprueban que los equipos estén, al menos, encendidos, realizando un ping a cada uno de ellos, que no haya sido cambiado, comprobando que la MAC del equipo instalado es la misma que la del equipo almacenado en la base de datos, y que la licencia de uso de cada uno de ellos no ha sido modificada sin previo aviso.

3.2.7.2 Monitorización del estado del equipo

3.2.7.2.1 Descripción

En esta página se muestran aquellos equipos que se encuentran apagados y la fecha desde la cual no se tiene ningún tipo de contacto, es decir, aunque el equipo siga sin responder en sucesivos intentos de conexión, la fecha mostrada será la del primer error.

3.2.7.2.2 Funcionamiento

En esta página tan solo se realiza una consulta a la base de datos y se recogen los datos de nombre y fecha de última conexión de aquellos equipos cuyo estado es "apagado". El código de la página es el siguiente:

```
<html>
<head>
<title>Pagina en la que se muestra aquellos equipos que no
responden</title>

<link href="css/menu.css" media="all" rel="stylesheet" type="text/c
ss" />
<script type="text/javascript" src="javascript/Menu.js">
</script>
</head>
<body>
      <?php
        $direccion = 'localhost';

        # Conexión a la base de datos
        $db=mysql_connect($direccion,$_SESSION['user'],
            $_SESSION['pass'])OR DIE (mysql_error());
```

```
        mysql_query('use pruebas') OR DIE (mysql_error());

        # Recogida de los datos
        $result=mysql_query("SELECT id_gateway_cf, date_down FROM
            conf_farm WHERE status=1") OR DIE(mysql_error());

        # Impresión en pantalla de la tabla de equipos
        echo "<fieldset align=\"left\">";
        echo "<legend align=\"left\">Farmacias</legend>";
        echo "<table border = '1'>";
        echo "<tr>";
        echo "<td><b>id_gateway</b></td>";
        echo "<td><b>No contesta desde</b></td>";

        while ($row = mysql_fetch_row($result))
        {
            echo "<tr>";
                echo "<td><a id=\"inicio\"  href=edit_equipo.html?
                  farma=$row[0]> $row[0]</a></td>";
                echo "<td>$row[1]</td>";
        }

        echo "</table>";

        # Cierre de la base de datos
        mysql_close($db);
    ?>
</body>
```

Para poder obtener los equipos que no son accesibles, se ha desarrollado un script en perl de nombre "*status.pl*" que se ejecuta cada hora, el cuál realiza una conexión a la base de datos para conocer las IPs de los dispositivos almacenados, realiza un ping a la IP del equipo para comprobar si el equipo se encuentra activo, actualizando en la base de datos el valor del campo "*status*" a 0 si el equipo responde o a 1 si no es así, además actualiza el valor del campo "*date_down*" con la fecha en la que se ha comprobado que el equipo ha dejado de responder.

El código fuente de este script es:

```
#!/usr/bin/perl
# Script que comprueba si los equipos están activos
# y actualiza su estado en la base de datos

use Net::Ping;
use Net::MySQL;
use strict;
```

```
# Definición de las variables a utilizar

my @elementos;
my $mysql = Net::MySQL->new(
    hostname => 'localhost',    # Default use UNIX socket
    database => 'pruebas',
    user     => 'miguel',
    password => 'administrador'
);
my $record_set;
my $IP;
my $query;
my $AUX;
my $fecha;

# Se recogen los datos de la base de datos
$mysql->query(q{SELECT ip_lan, status FROM conf_farm});

# Se crea el índice con el que recorrer el vector resultado de la
# query anterior
$record_set = $mysql->create_record_iterator;

# Se comprueba el estado de cada uno de los equipos
while (my $record = $record_set->each)
{

    # Se comprueba si el equipo responde
    $AUX=Net::Ping->new("icmp",1,64); #Creamos un Objeto PING


    if($AUX->ping($record->[0],1))
    {

        # Se actualiza el estado de la farmacia
        # a RESPONDE (status=0)
        my $query="UPDATE conf_farm SET status=0 WHERE
            ip_lan=\"$record->[0]\"";
        $mysql->query($query);

    }else
        {

        # Se actualiza el estado de la farmacia
        # a NO RESPONDE (status=1)
        if($record->[1] == 0)
        {
```

```
                $fecha=localtime(time);
                my $query="UPDATE conf_farm SET status=1,
                      date_down=\"$fecha\" WHERE ip_lan=\"
                      $record->[0]\"";

                    $mysql->query($query);
            }
        }

        # Se elimina el Objeto ping
        $AUX->close();

    }

    # Se cierra la conexión con la base de datos
    $mysql->close;
```

3.2.7.3 Monitorización de los cambios de dispositivos y licencias

3.2.7.3.1 Descripción

Estas páginas muestran aquellos equipos que han sido reemplazados por encontrarse estropeados o aquellos a los que se las ha modificado su licencia de uso.

3.2.7.3.2 Funcionamiento

Al igual que la página anterior, se realiza una consulta a la base de datos y se recogen tan solo aquellos equipos que tengan el registro "*stat_mac*" igual a 1, si se ha cambiado el equipo, o el registro "lic_stat" igual a 1, si se ha cambiado la licencia de uso.

Para poder conocer si se han realizado estas modificaciones, se ha desarrollado un script en perl de nombre "*comprueba_si_cambio.pl*" el cual realiza estas comprobaciones. El primer paso que realiza este script es recoger las IPs de todos los equipos dados de alta en la base de datos. Posteriormente comprueba si los equipos se encuentran encendidos, si resulta que se encuentra encendido, realiza una conexión por ssh al dispositivo para recoger los datos de MAC y licencia y las comprueba con los obtenidos de la base de datos. Si alguno de ellos hubiese cambiado, se encargará de actualizar el estado de los registros "*stat_mac*", "*mac*" y "*mac_ant*" si se ha cambiado el equipo o los registros "*lic_stat*", "*lic*" y "*lic_ant*" si se ha modificado la licencia.

El código fuente de este script es:

```
#!/usr/bin/perl
# Script que comprueba el cambio de equipo o licencia
```

```
use Net::Ping;
use Net::MySQL;
use strict;
use Net::SSH::Expect;

# Definición de las variables
my @elementos;
my $mysql = Net::MySQL->new(
                hostname => 'localhost',# Default use UNIX socket
                database => 'pruebas',
                user     => 'miguel',
                password => 'administrador'
              );
my $record;
my $record_set;
my $IP;
my $query;
my $AUX;
my $fecha;
my $command_result;
my $ssh;
my $login_output;
my @mac;
my @AUX2;
my @users;
my @AUX3;
my $query;


# Recogida de las IPs, mac y licencia de los equipos
$mysql->query(q{SELECT ip_lan,mac,lic FROM conf_farm});
$record_set = $mysql->create_record_iterator;

# Comienzo del proceso de comprobación
while ($record = $record_set->each)
{
     # Se comprueba si el equipo responde
    $AUX=Net::Ping->new("icmp",1,64); #Creamos un Objeto PING

    if($AUX->ping($record->[0],1))
    {

        # Se abre la conexion SSH
        my $ssh = Net::SSH::Expect->new (
            host => $record->[0],
            password=> $pass,
            user => $user,
```

```
    raw_pty => 1,
    timeout => 25
);
my $login_output = $ssh->login();

# Se ejecuta el comando necesario para
# poder recoger la MAC y el numero de usuarios del equipo
$command_result = $ssh->exec("info device");


# Se cierra la conexion SSH
$ssh->close();


# Se obtiene la MAC
$AUX=$command_result;
@AUX2=split("MAC Address:",$AUX);
@mac=split(" ",$AUX2[1]);

# Se obtiene la licencia del equipo
@AUX3=split("Product Name:",$AUX2[1]);
@users=split(" ",$AUX3[1]);

# Se comprueba si la MAC es la misma
# que la registrada en la BBDD
if($mac[0] ne $record->[1])
{

    $query="UPDATE conf_farm SET stat_mac=1,
          mac=\"$mac[0]\", mac_ant=\"$record->[1]\"WHERE
          ip_lan=\"$record->[0]\"";
}else
{
    $query="UPDATE conf_farm SET stat_mac=0 WHERE
          ip_lan=\"$record->[0]\"";
}

# Se actualiza la MAC del equipo en la BB.DD.
    $mysql->query($query);

# Se comprueba si la licencia es la misma que la registrada
# en la BBDD

    if($users[2] ne $record->[2])
    {
```

```
                    $query="UPDATE conf_farm SET stat_lic=1,
                         lic=\"$users[2]\", lic_ant=\"$record->[2]\" WHERE
                         ip_lan=\"$record->[0]\"";

                    }else
                    {
                     $query="UPDATE conf_farm SET stat_lic=0 WHERE
                         ip_lan=\"$record->[0]\"";
                    }

                # Se actualiza la licencia del equipo en la BB.DD.
                $mysql->query($query);

        }
    }

    # Se cierra la conexión con la base de datos
    $mysql->close;
```