PROYECTO INTEGRADO

Autor: Salvador Fernández Morilla
Tutor: José Antonio Carrasco Díaz

I.E.S. Francisco Romero Vargas (Jerez de la Frontera)
Administración de Sistemas Informáticos en Red
Curso: 2017/18

Tabla de contenido

Introducción.

Introducción.

Crear un sistema que permita la administración a distancia de cámaras de vigilancia, a través de una página web con acceso limitado a usuarios registrados, o con instrucciones a un BOT de Telegram desde el móvil.

Finalidad.

Con este sistema se puede mantener un control a distancia de las cámaras de vigilancia instaladas en un lugar, como un local, un garaje o el propio hogar. Los usuarios registrados en el sistema podrán, según sus permisos, visualizar, gestionar o configurar los vídeos e imágenes creados por las cámaras (como por ejemplo el formato), programar grabaciones o capturas a determinadas horas, acceder a la retransmisión de las cámaras o solicitar capturas de imágenes.

Objetivos.

Este proyecto permitirá que un grupo de usuarios, registrados en una página web, administren las grabaciones de una cámara de vigilancia. La página web se creará usando PHP y CSS, estará alojada en un servidor HTTP Apache, y solo una cuenta de administrador podrá dar de alta a estos usuarios, que quedarán registrados, junto a sus permisos, en una base de datos MariaDB.

Los usuarios podrán realizar las acciones mencionadas en el apartado Finalidad dependiendo de los permisos otorgados por el administrador. También podrán realizar instrucciones desde un una cuenta del servicio de mensajería Telegram, usando un bot que se programará mediante PHP. Solo se permitirán instrucciones de cuentas asociadas a los usuarios registrados en el sistema, pudiendo el administrador asociarlas en cualquier momento, ya sea durante el registro o después.

A excepción de la visualización de imágenes o grabaciones almacenadas, el resto de instrucciones crearán scripts que se ejecutarán en el servidor para configurar o ejecutar comandos de Motion, un programa que monitorizará la señal de vídeo proveniente de la cámara de vigilancia, capaz de captar movimiento, conectada de forma inalámbrica al servidor.

Las capturas de imágenes podrán ser instantáneas o programadas, es decir, un usuario podrá solicitar una captura de imagen en cualquier momento o programar capturas a determinadas horas. También podrán ser privadas o públicas, es decir, el usuario podrá indicar si la captura se almacenará en una carpeta personal del servidor a la que solo tendrán acceso el mismo usuario o el administrador, o en una carpeta pública a la que podrán acceder todos los usuarios. Esto es

importante de cara a la visualización de capturas, pues solo podrán ver sus capturas personales y las públicas, a excepción del administrador.

En cuanto a la programación de grabaciones, solo podrá realizarlas el propio administrador aunque estas siempre serán públicas para el resto de usuarios, y podrán visualizarse desde la página web.

Medios necesarios.

Para la realización de este proyecto se necesita lo siguiente:

- Un ordenador con sistema operativo Debian.
- Una cámara PlayStation Eye.
- Una cuenta de usuario en Telegram.
- Motion, programa para monitorizar la señal de vídeo de la cámara.
- Servidor HTTP Apache.
- Sistema de Gestión de Base de Datos MariaDB.

Planificación.

- Instalación del sistema operativo Debian: **6 horas.**
- Instalación y configuración del servidor Apache: **1 hora.**
- Montaje de la cámara web: **30 minutos.**
- Estudio del programa Motion, para el control de movimiento: **32 horas.**
- Instalación y configuración del programa Motion: **30 minutos.**
- Comprobación del funcionamiento de la cámara con el programa Motion: **30 minutos.**
- Planificación de la base de datos MariaDB: **30 minutos.**
- Creación de la base de datos MariaDB: **30 minutos.**
- Planificación del código para la página web: **32 horas.**
- Programación de la página web: **112 horas.**
- Estudio de la programación de un bot en Telegram: **8 horas.**
- Creación y programación de un bot en Telegram: **32 horas.**
- Prueba de todo el proyecto en condiciones reales: **4 horas.**
- Preparación de la documentación del proyecto: **8 horas.**

Horas totales que se planifican para el proyecto: **237 horas y 30 minutos.**

Realización del Proyecto.

Aquí vamos a contar como nos ha ido el proyecto, lo que hemos hecho, los problemas que hemos encontrado, la documentación que hemos usado. Pondremos capturas de pantalla del proceso, fotos si las consideramos necesarias, ficheros de configuración, fuentes de los programas, etc.

Trabajos realizados.

CONFIGURACIÓN DEL SERVIDOR.

Para comenzar, se monta el servidor usando un PC (Placa K7VT4A Pro, Procesador AMD Sempron 2400 y 2GB RAM) donde se instala la última versión del sistema operativo Debian, la cual se montará en una memoria USB. Dicha versión se obtendrá descargando la imagen ISO desde el enlace: https://cdimage.debian.org/debian-cd/current/i386/iso-cd/debian-9.4.0-i386-xfce-CD-1.iso
El servidor es arrancado desde la memoria USB procediendo a instalar Debian.

Se seguirán las instrucciones del asistente de instalación e introducirá la información necesaria que solicite. En este caso, en la Partición de Discos, se necesitó hacerse manual debido a lo que se comenta más adelante en "*Problemas Encontrados*".

Cuando al acabar el proceso de instalación se accede al sistema operativo, se configura el archivo /etc/network/interfaces como se muestra a continuación.

```
  GNU nano 2.7.4                                  Fichero: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

allow-hotplug enp0s18
iface enp0s18 inet dhcp
```

En este archivo se indica que el servidor recibirá una IP del servidor DHCP, en este caso un router LiveBox de Orange, el cual configurará accediendo desde un navegador.

Al iniciar sesión desde el navegador en la página de configuración del router LiveBox, se accede a la pestaña Básica > Configuración LAN, sitio donde se asignará una dirección estática al servidor.

En este caso, el nombre del servidor es Siros, y se le asignó la dirección 192.168.1.110, la cual se compruebaunavezsereinicieelservidor.

```
root@Siros:/var/www/html# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 00:0b:6a:bb:17:dd brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.110/24 brd 192.168.1.255 scope global enp0s18
       valid_lft forever preferred_lft forever
    inet6 fe80::20b:6aff:febb:17dd/64 scope link
       valid_lft forever preferred_lft forever
```

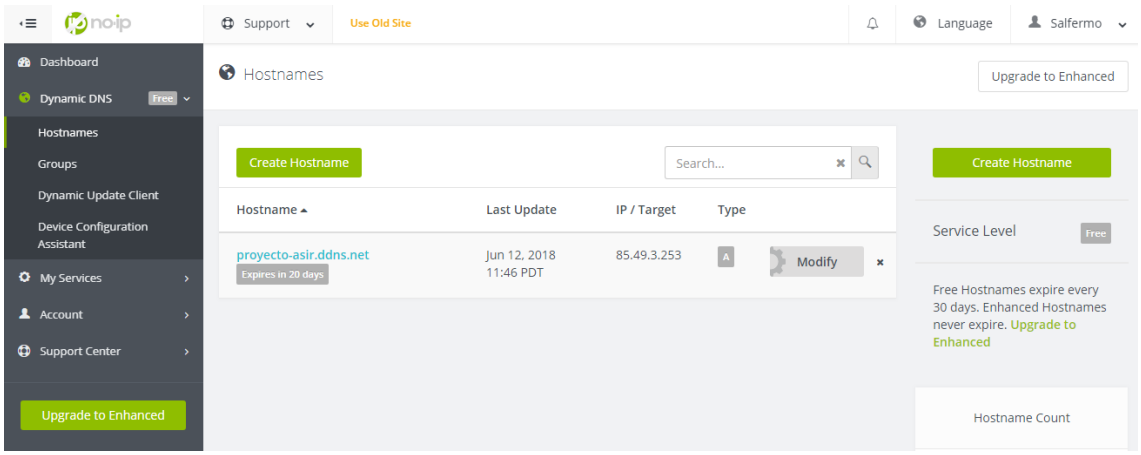
Cuando se comprueba, realizando ping al exterior o abriéndose un navegador desde el servidor, que se cuenta con acceso a internet, se actualiza la lista de paquetes con el comando 'apt-get update' y se instalan los necesarios para la realización de este proyecto con 'apt-get install apache2 php mysql-server php-mysql motion ffmpeg curl.

CONFIGURACIÓN NO-IP

Para acceder desde el exterior al servidor mencionado antes, se encontró un problema, no contar con dirección IP pública estática, sino una dinámica. Para solucionar esto se usará DNS Dinámico.

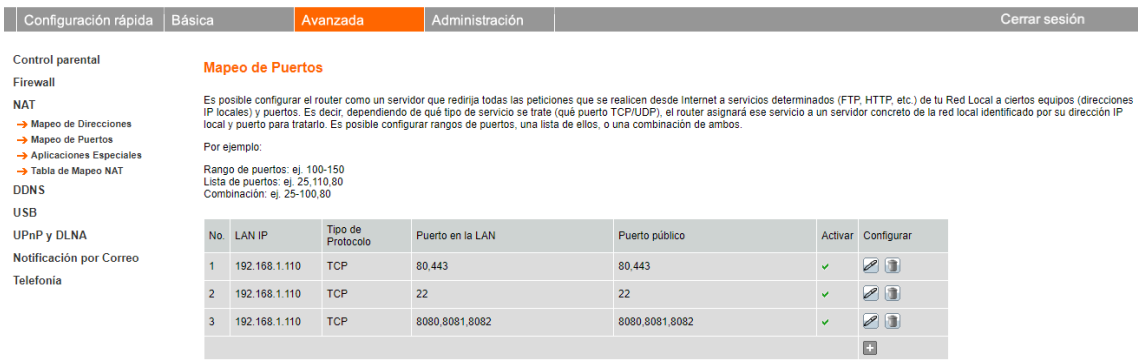
No-IP es un servicio de DNS Dinámico que permite identificar el servidor con un nombre de dominio, este se actualiza cuando cambia la dirección IP pública. Para utilizarse, se debe registrar en su página https://www.noip.com/ y, accediendo a la cuenta, se crea un nuevo Hostname en Dynamic DNS > Hostnames.

En este caso el dominio seleccionado es proyecto-asir.ddns.net, y aunque informe que expirará en 30 días, se puede volver a renovar sin coste alguno.



Para que No-IP funcione en el servidor Siros se debe, o bien instalar un paquete (el cuál no se consiguió que funcionase) o, al disponer de un router LiveBox de Orange, configurar con la información de la cuenta No-IP para usar el servicio. Así pues, se accede al router desde el navegador y, en Avanzada > DDNS se escribirá la información de la cuenta No-IP.

También será necesario redirigir los puertos necesarios al servidor, siendo en este caso el 80, 443, 8080, 8081 y, aunque no es necesario, el 22 se ha redirigido también durante esta práctica para poder trabajar con el servidor mediante SSH.



En la imagen mostrada anteriormente, aparece el puerto 8082 porque es el utilizado para pruebas.

MOTION

Este proyecto se usa una cámara PlayStation Eye, que se conecta mediante USB. En un principio se esperaba usar una cámara de vigilancia AXIS 2110, pero la imagen retransmitía era bastante mala.

La cámara PlayStation Eye no requiere instalaciones para su correcto funcionamiento en Debian, pero para poder detectar movimiento será necesaria una aplicación, en este caso Motion.

Motion es un software que permite realizar grabaciones y/o capturas de una cámara cuando detecte movimiento en esta. La configuración de parámetros de Motion se realiza en el archivo /etc/motion/motion.conf, y aunque son muchos parámetros, solo será necesario cambiar los siguientes:

-------- Parámetros --------

daemon on

width 640

height 480

framerate 100

target_dir /var/www/html/eventos

ffmpeg_video_codec mp4

stream_port 8081

stream_maxrate 100

stream_localhost off

output_pictures best

webcontrol 8080

webcontrol_localhost off

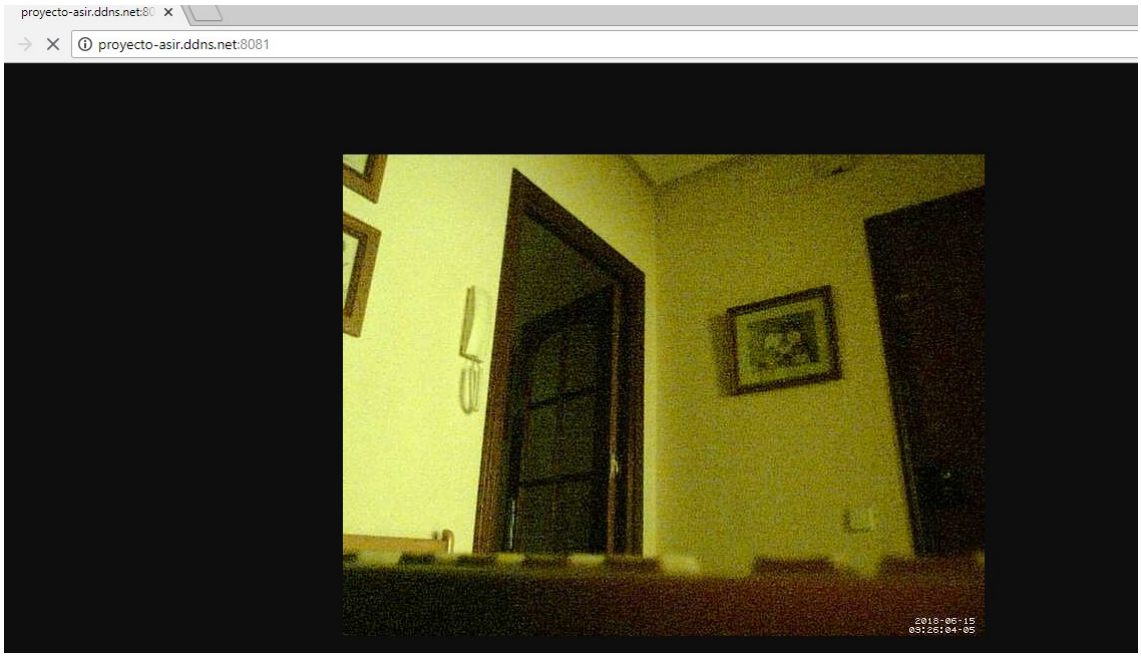
on_movie_end php /var/www/telegram/enviar_evento.php

En el parámetro target_dir se indica donde se guardarán las capturas o vídeos que Motion realice. El parámetro stream_port indica el puerto desde el cual ver la retransmisión de la cámara. La opción best de output_pictures indica que guardará la imagen que Motion considere mejor, en vez de guardar varias imágenes.

Webcontrol es un parámetro realmente útil para el proyecto, pues permite la administración de Motion desde una página web de la propia aplicación. Así, se podrá realizar capturas o cambiar parámetros de configuración desde el navegador. Se usará más adelante.

Por último, existen parámetros de eventos que activan scripts, uno de estos es on_movie_end, el cual se utilizará para ejecutar un archivo PHP cada vez que acabe de grabar un vídeo. Cuando Motion detecta movimiento, realiza capturas y una grabación de vídeo, así nos aseguramos que el

archivo PHP se ejecuta justo después de haberse detectado movimiento. Esto interesa para Telegram, pero se detallará más adelante.
Cuando se quiera iniciar Motion se usará el comando motion -b, y para comprobar que funciona correctamente se accederá a http://proyecto-asir.ddns.net:8081/, donde se verá lo que capta en estos momentos la cámara.



MySQL y APACHE

Una vez comprobado que se puede acceder a la retransmisión de la cámara, el siguiente paso es preparar MySQL y Apache. Con el comando 'mysql -u root' se conecta con MySQL, donde se creará una base de datos, que recibirá el nombre de 'vigilancia', y en esta una tabla donde se almacenará la información de los usuarios registrados en el sistema.

```
MariaDB [vigilancia]> CREATE TABLE usuarios (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> nombre VARCHAR(255),
    -> clave VARCHAR(255),
    -> correo VARCHAR(255),
    -> PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.27 sec)
```

Las claves se guardarán desde PHP encriptadas con SHA1.

Para la configuración de Apache no se cambiará nada por ahora, dejando la configuración predeterminada se comienza a programar las páginas web. Esto se realizará en el directorio por defecto para Apache, /var/www/html.

SITIO WEB

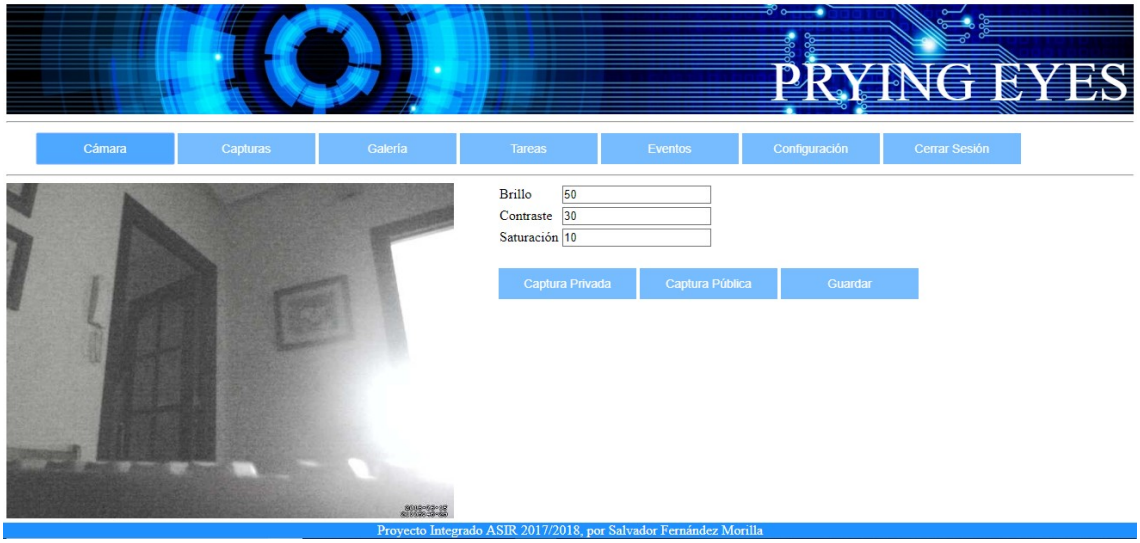
En la carpeta predeterminada /var/www/html, se eliminará el archivo index.html y se creará uno nuevo, cuya finalidad será mostrar una pantalla de inicio de sesión, redirigiendo la información a login.php, el cuál se encargará de comprobar que el usuario se encuentre en nuestra base de datos. Si es así, redirigirá a acceso.php, si no de nuevo a index.html.

```
<!-- index.html -->
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" type="text/css" href="estilo.css">
        <title>Identificación de Usuarios</title>
    </head>

    <body>
        <div class="bg"><h1>Prying Eyes</h1></div>

        <hr><br><br>

        <center>
            <form method="post" action="./login.php">
                <input type="text" name="nombre" placeholder="Nombre" required/>
                <br><br>
                <input type="password" name="clave" placeholder="Contraseña" required/>
                <br><br>
                <input class="button" type="submit" value="Acceder" name="acceder"/>
            </form>
        </center>
        <div class='footer'>Proyecto Integrado ASIR 2017/2018, por Salvador Fernández Morilla</div>
    </body>
</html>
```

```
<?php
    // login.php
    session_start();

    require_once("conexion.inc.php");
    $c = new mysqli($h,$u,$p,$bd);

    if ($c->connect_error){
        die("La conexión falló: ".$c->connect_error);
    }

    $nombre = $_POST['nombre'];
    $clave = sha1($_POST['clave']);

    $c->query("SET NAMES utf8");
    $sql = "SELECT id FROM usuarios WHERE nombre = '".$nombre."' AND clave = '".$clave."'";

    $resultado=$c->query($sql);
    if ($resultado->num_rows > 0) {
        $fila = $resultado->fetch_assoc();
        $_SESSION['id'] = $fila['id'];
        header("Location: acceso.php");
    }else{
        header("Location: index.html");
    }
    $c->close();
?>
```

El archivo acceso.php es el centro del sitio web, mostrando las opciones que podrán realizar los usuarios registrados: mostrar la retransmisión de la cámara, las capturas realizadas, la posibilidad de programar capturas a ciertas horas, mostrar los vídeos realizados en la detección de movimiento y, en caso del administrador, configurar la lista de usuarios registrados.

Cámara muestra la retransmisión de la cámara, ofreciendo la posibilidad de realizar capturas privadas o públicas instantáneas. Las capturas privadas solo se mostrarán al usuario que las realizó, mientras que las públicas estarán disponibles para cualquier usuario. Además, también se mostrarán los parámetros de configuración de la cámara Brillo, Contraste y Saturación, aunque solo podrá modificarlos el administrador.



Para la captura de imagen, acceso.php llamará a capturar.php, indicando si se requiere una captura privada o pública, dependiendo de lo cual se ejecutará un comando linux u otro. Para ejecutar comandos linux desde PHP usamos 'exec'.

```php
<?php
    // capturar.php
    session_start();
    if (isset($_SESSION['id'])){
        echo exec("curl http://proyecto-asir.ddns.net:8080/0/action/snapshot > /dev/null");
        $cadena = "cp ./eventos/lastsnap.jpg
        ./usuario".$_SESSION['id']."/".$_POST['captura']."/".$_SESSION['id']."-".date("Ymdhis").".jpg";
        echo exec($cadena);
    }else{
        echo "Usted no está autorizado para ver esta página.";
    }
?>
```

Los usuarios cuentan con una carpeta propia con el formato 'usuario'+ID, y dentro de esta dos carpetas, 'privadas' y 'publicas'. Así, cuando se requiere una captura privada, esta irá a 'privadas', y si es pública, a 'publicas'. Se ha realizado así para facilitar luego la búsqueda de capturas.

Antes, cuando se explicaron las modificaciones en el archivo de configuración de Motion, se señaló que activar el parámetro webcontrol permitía acceder a la página web de la aplicación, donde podrían realizarse capturas o modificar parámetros de configuración. La captura se realiza con una de las opciones de esta página http://proyecto-asir.ddns.net:8080/0/action/snapshot que realizará una captura y la guardará en el directorio que se indica en el archivo de configuración.

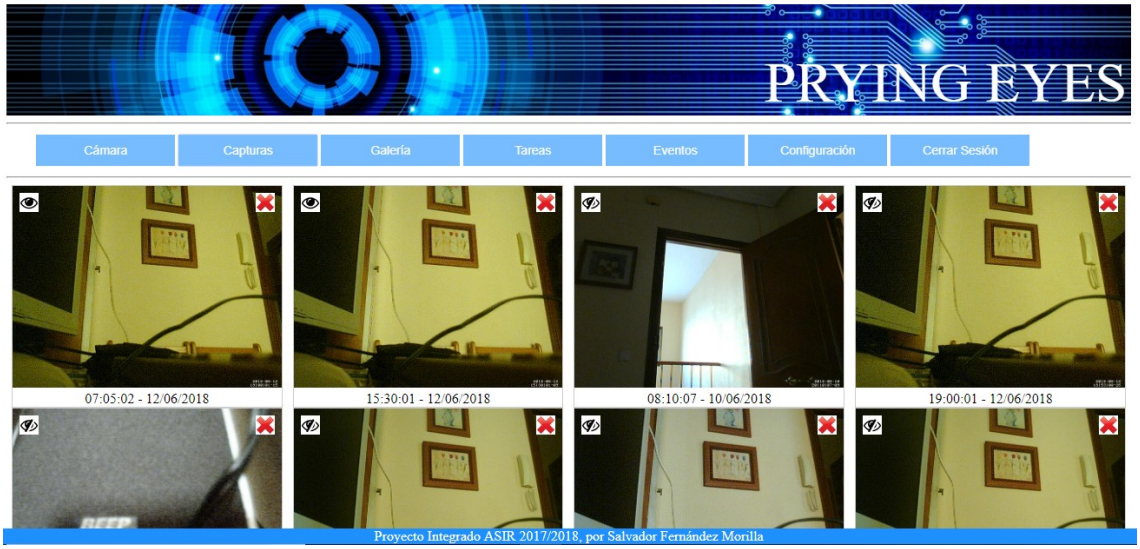
Para conseguir el Brillo, Contraste y Saturación también será necesaria esta página. El archivo acceso.php llamará a ver_configuracion.php para recuperar estos parámetros usando comandos de linux. En esencia, lo que hacemos es descargar la página resultante donde muestra el parámetro de configuración que interesa, buscar este y descartar lo demás.

```php
<?php
    // ver_configuracion.php
    session_start();
    if (isset($_SESSION['id'])){
        $output = array();

        exec("curl -s http://proyecto-asir.ddns.net:8080/0/config/get?query=brightness | grep brightness | cut -d'
        ' -f 3",$output[0]);
        exec("curl -s http://proyecto-asir.ddns.net:8080/0/config/get?query=contrast | grep contrast | cut -d' ' -f
        3",$output[1]);
        exec("curl -s http://proyecto-asir.ddns.net:8080/0/config/get?query=saturation | grep saturation | cut -d'
        ' -f 3",$output[2]);

        echo json_encode($output);
    }
?>
```

Si el usuario que inicia sesión es el administrador, podrá modificar los valores de Brillo, Contraste y Saturación. Para guardar esta configuración se usa el archivo guardar_configuracion.php, que ejecutará URLs que modifiquen estos valores desde la página web de Motion, guardando a continuación la configuración en el mismo archivo /etc/motion/motion.conf.

```php
<?php
    // guardar_configuracion.php
    session_start();
    if (isset($_SESSION['id'])){
        $brillo = $_POST['brillo'];
        $contraste = $_POST['contraste'];
        $saturacion = $_POST['saturacion'];

        exec("curl http://proyecto-asir.ddns.net:8080/0/config/set?brightness=$brillo");
        exec("curl http://proyecto-asir.ddns.net:8080/0/config/set?contrast=$contraste");
        exec("curl http://proyecto-asir.ddns.net:8080/0/config/set?saturation=$saturacion");

        exec("curl http://proyecto-asir.ddns.net:8080/0/config/writeyes");
    }
?>
```

Capturas muestra, como su nombre indica, las capturas privadas y públicas realizadas por el usuario que inicia sesión.



Este listado se obtiene con el archivo listado_imagenes.php que, al indicarle que se requieren las capturas propias, ejecutará una línea de comandos linux para encontrar las capturas del usuario que ha iniciado sesión, ya sean en la carpeta 'privadas' o 'publicas' de este usuario, mostrando además su ruta, la cuál se aprovechará en el código de acceso.php.

```php
<?php
    // listado_imagenes.php
    session_start();
    if (isset($_SESSION['id'])){
        if ($_POST["seleccion"] == "propias"){
            exec("find ./usuarios/usuario".$_SESSION['id']." -name '*.jpg'",$output);
        }else{
            exec("find ./usuarios/usuario*/publicas -name '*.jpg'",$output);
        }
        echo json_encode($output);
    }
?>
```

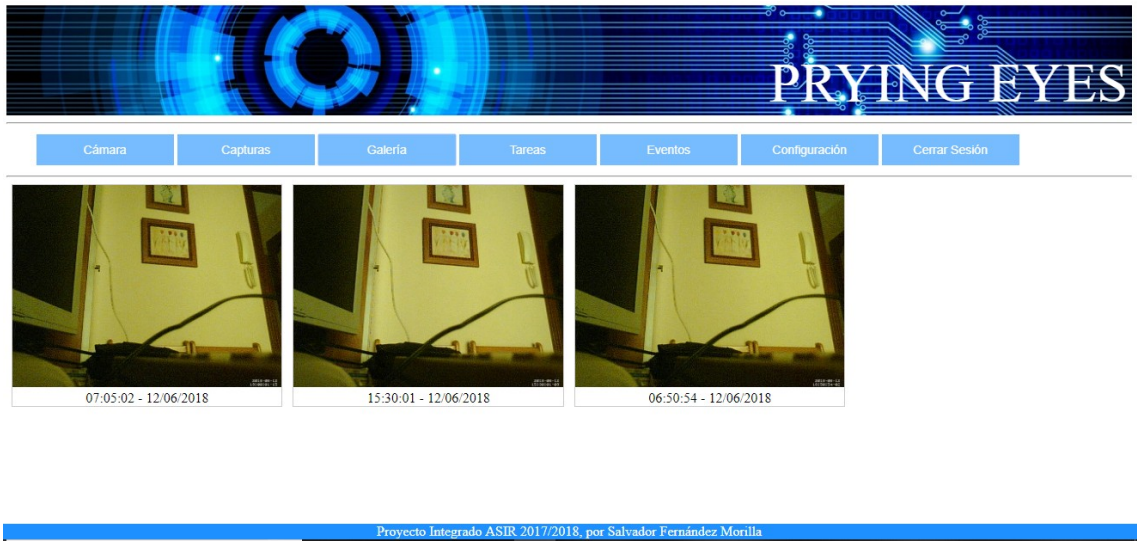
Además, cada captura mostrada incluirá dos botones, uno para eliminar la imagen usando el archivo eliminar_imagen.php, y otro para cambiarla de privada a pública o viceversa, con el archivo publicar_imagen.php. Aquí es cuando se aprovecha la ruta completa que se encontró anteriormente.

```php
<?php
    // eliminar_imagen.php
    session_start();
    if (isset($_SESSION['id'])){
        $ruta = $_POST["ruta"];
        exec("rm $ruta");
    }
?>
```

```php
<?php
    // publicar_imagen.php
    session_start();
    if (isset($_SESSION['id'])){
        $ruta = $_POST["ruta"];
        $nueva_ruta = $_POST["nueva_ruta"];
        exec("mv $ruta $nueva_ruta");
    }
?>
```

También se muestra, como descripción de cada imagen, la fecha y hora a la que se tomo, la cuál se consigue del mismo nombre del archivo.

Galería muestra las capturas públicas realizadas por todos los usuarios registrados, y para esto usamos, de nuevo, el archivo listado_imagenes.php pero indicando que no se requieren capturas propias, de esta manera ejecutará el comando find en linux para encontrar las capturas en cualquier carpeta 'publicas' dentro de las carpetas de los usuarios.

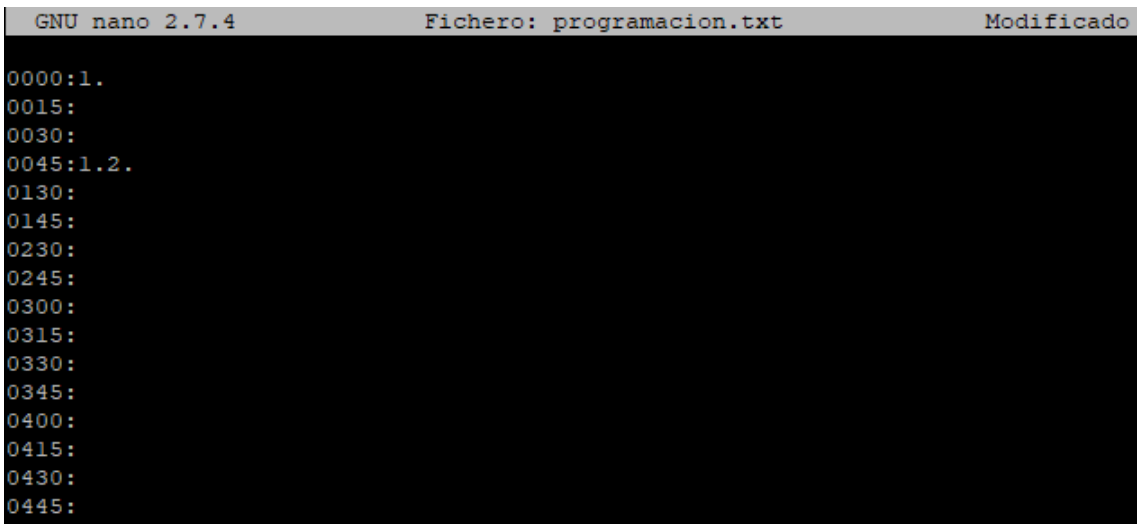


Como son capturas públicas de otros usuarios (aunque también aparezcan las del usuario que inicia sesión) no podrán eliminarse ni modificarse.

Tareas mostrará la programación del día cada 15 minutos, para que cada usuario pueda programar una captura a una hora del día, la cual se realizará cada día. Por ejemplo, podría programarse que la cámara realizase una captura a las 18:15 cada día, pero no a las 18:16, ya que el intervalo es cada 15 minutos.

Cuando un usuario pulsa en la celda correspondiente a la hora en la que desea solicitar una captura, se mostrará el icono de una cámara, indicativo de que existe una solicitud de captura de este usuario a esa hora.

La programación de capturas se registrará en el fichero /var/www/html/scripts/programacion.txt, donde cada línea corresponderá a un intervalo de 15 minutos de entre las 24 horas, seguido de ':' y los identificadores de los usuarios que han solicitado capturas a esa hora, separados por puntos.



Para la realización de capturas programadas se usará Crontab, una aplicación que ejecuta tareas cada ciertas horas indicadas en su fichero. Para configurar Crontab se usa el comando crontab -e, escribiendo lo siguiente:

```
  GNU nano 2.7.4        Fichero: /tmp/crontab.q573j3/crontab        Modificado

*/15 * * * * /var/www/html/scripts/programacion.sh
```

Cabe destacar que cada usuario del sistema linux cuenta con su propio Crontab, así que si se ejecuta crontab -e como root, las tareas las realizará con los permisos de root. En este caso lo haremos como root.

Se ha indicado en Crontab que, cada 15 minutos, ejecute el script programacion.sh, al que deberá darse permisos de ejecución con chmod ug+x.

```
  GNU nano 2.7.4                                          Fichero: programacion.sh

#!/bin/bash

# Tomamos la hora del sistema y, en programacion.txt, buscamos los usuarios correspondientes a esa hora.
HORA=$(date +"%H%M")
USUARIOS=$(grep $HORA /var/www/html/scripts/programacion.txt | cut -d ":" -f2 | sed -e "s/\./ /g" | sed -e "s/\s$//")

# Creamos una captura con la cámara.
curl http://proyecto-asir.ddns.net:8080/0/action/snapshot > /dev/null

FECHA=$(date +"%Y%m%d%H%M%S")

for ID in $USUARIOS; do
        # Copiamos la captura realizada para cada usuario que la solicitó en la hora programada.
                cp /var/www/html/eventos/lastsnap.jpg /var/www/html/usuario$ID/privadas/$ID-$FECHA.jpg
```

Este script comprueba en programacion.txt si, en la línea correspondiente a la hora actual, existen identificadores de usuarios. Si es así, realizará una captura de imagen y la copiará a cada carpeta 'privadas' correspondiente a cada uno de estos usuarios, aprovechando el identificador.

Eventos mostrará las grabaciones de vídeo realizadas cuando Motion detecte movimiento. Para mostrar estas grabaciones se usará el archivo eventos.php que buscará los archivos .avi que se encuentren en /var/www/html/eventos. Como el comando find muestra la ruta de cada archivo encontrado, esta ruta será aprovechada luego en acceso.php para mostrarlos.

```php
<?php
    // eventos.php
    session_start();
    if (isset($_SESSION['id'])){
        exec("find ./eventos -name '*.avi'",$output);
        echo json_encode($output);
    }
?>
```

Configuración solo estará disponible para el administrador, mostrando una tabla con los usuarios registrados en el sistema. Esta información puede modificarse desde la propia tabla, a excepción del identificador.

Para mostrar la lista de usuarios se usa el archivo listado_usuarios.php, que realizará un SELECT en la base de datos.

```php
<?php
    // guardar_usuarios.php
    session_start();
    if (isset($_SESSION['id'])){
        $id = $_POST['id'];

        require_once("conexion.inc.php");
        $conexion = new MySQLi($h,$u,$p,$bd);
        $conexion->query("SET NAMES utf8");

        $id = $_POST['id'];
        $nombre = $_POST['nombre'];
        $correo = $_POST['correo'];

        if ($id == "0"){
            $sql = "INSERT INTO usuarios (nombre,correo) VALUES ('$nombre','$correo')";
            $conexion->query($sql);
            $sql = "SELECT MAX(id) AS id FROM usuarios";
            $resultado=$conexion->query($sql);

            if ($resultado->num_rows > 0) {
                $fila = $resultado->fetch_assoc();
                $id = $fila['id'];
                exec("mkdir ./usuario/usuario".$id);
                exec("mkdir ./usuario/usuario".$id."/privadas");
                exec("mkdir ./usuario/usuario".$id."/publicas");
            }
        }else{
            $sql = "UPDATE usuarios SET nombre = '$nombre',correo = '$correo' WHERE id = $id";
            $conexion->query($sql);
        }
        $conexion->close();
    }
?>
```

Para actualizar los datos con las modificaciones o nuevos usuarios se usará guardar_usuarios.php.

Este archivo recibirá el listado de usuarios modificado y, línea por línea, comprobará si debe actualizar un registro o crear uno nuevo. Los usuarios con identificador '0' indican que es necesario crearlos. Para los nuevos usuarios se creará su carpeta de usuario y, en esta, las carpetas 'privadas' y 'publicas' para las capturas que realice.

```php
<?php
    // listado_usuarios.php
    session_start();
    if (isset($_SESSION['id'])){
        require_once("conexion.inc.php");
        $conexion = new MySQLi($h,$u,$p,$bd);

        $conexion->query("SET NAMES utf8");
        $sql = "SELECT * FROM usuarios ORDER BY id";
        $resultado = $conexion->query($sql);

        $data = array();
        $data = $resultado->fetch_all(MYSQLI_ASSOC);

        echo json_encode($data);

        $conexion->close();
    }
?>
```

Cambiar contraseña permite modificar la contraseña del usuario en cuestión, pero solo estará disponible para usuarios que estén guardados en la base de datos, es decir, cuando se añade un nuevo usuario, hasta que no se guarde, no podrá modificarse su contraseña.

```php
<?php
    // cambiar_pass.php
    session_start();
    if (isset($_SESSION['id'])){
        $id = $_POST['id'];

        require_once("conexion.inc.php");
        $conexion = new MySQLi($h,$u,$p,$bd);

        $conexion->query("SET NAMES utf8");

        $id = $_POST['id'];
        $pass = $_POST['pass'];

        $sql = "UPDATE usuarios SET clave = sha1('$pass') WHERE id = $id";
        exec("echo $sql > consola.log");
        $conexion->query($sql);

        $conexion->close();
    }
?>
```

Cerrar Sesión redirige al usuario a logout.php que, al eliminar la sesión, redirigirá a su vez a index.html.

```php
<?php
    // logout.php
    session_start();
    unset ($SESSION['username']);
    session_destroy();

    header('Location: index.html');

?>
```

Para acabar, se deberá otorgar permisos al usuario www-data sobre la carpeta /var/www/html usando el comando 'chown www-data.root /var/www/html -R', ya que es el usuario empleado por los servidores web, como Apache, y necesitará permisos para ejecutar, crear o modificar archivos.

TELEGRAM

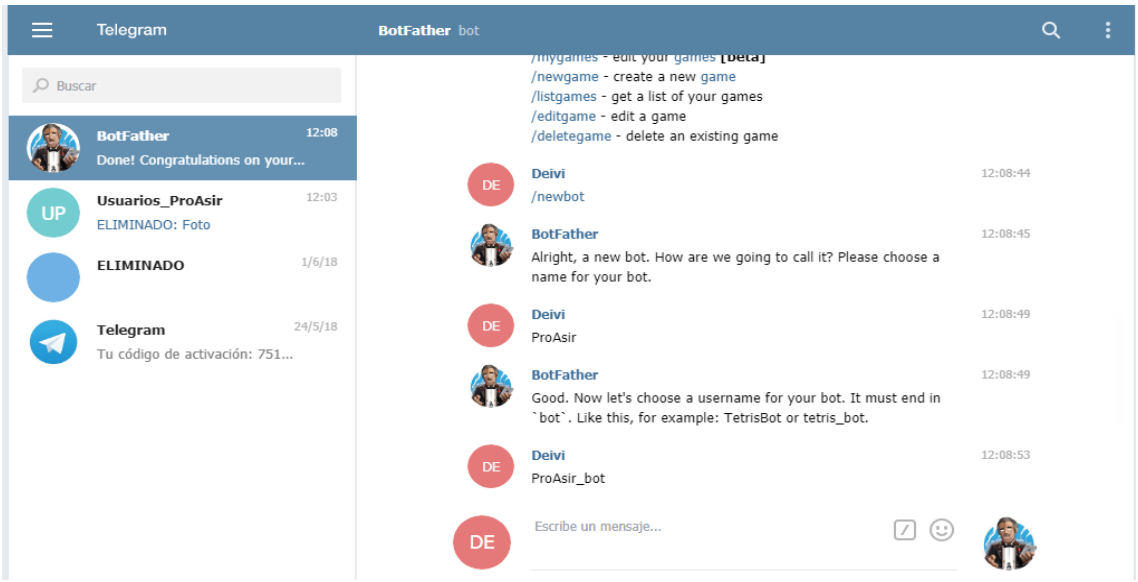
Los usuarios no solo podrán realizar capturas desde el navegador, sino también desde el móvil con Telegram. Esta aplicación permite crear bots de manera sencilla y programarlos para realizar tareas. En este proyecto usaremos un bot para que, al ejecutar un comando, devuelva una captura de la cámara.

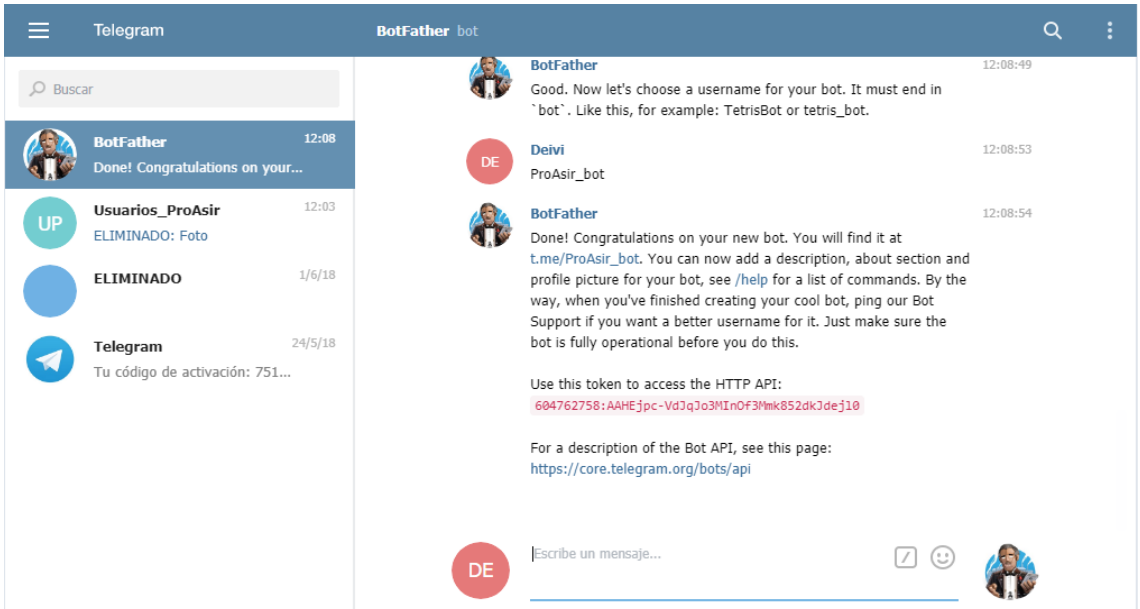
También se programará para que envíe capturas cuando Motion detecte movimiento. Como en el archivo de configuración de Motion seleccionamos que, al detectar movimiento, realizase una única captura, será esta la que el bot envíe.

Para empezar, será necesario descargar Telegram en el móvil, confirmar el número de teléfono y crear el perfil de usuario.

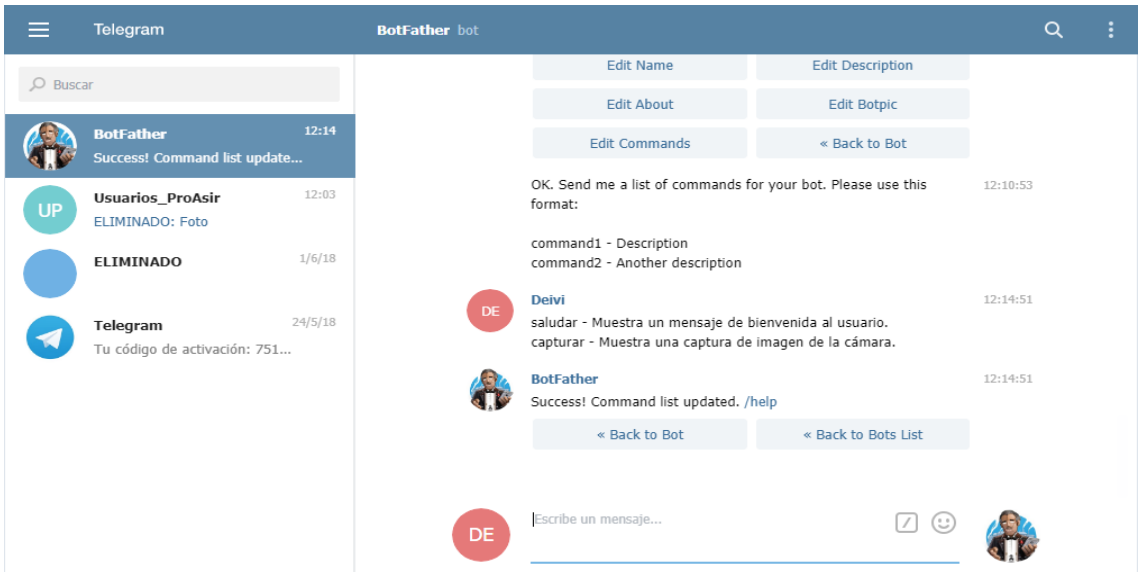
Por comodidad, es recomendable acceder a Telegram desde https://web.telegram.org/ en el navegador del ordenador, usando el asistente BotFather para crear y modificar bots.

Accediendo a la conversación con BotFather, se selecciona el comando /newbot, se escribe un nombre para este y se recibe un Token, que es el identificador que usa Telegram para diferenciar los bots. Será necesario este Token para programarlo.





Los comandos del Bot puede crearse con BotFather, seleccionando el Bot en cuestión y dirigiéndonos a Edit Bot > Edit Command.



Lo siguiente será indicar a Telegram que cualquier mensaje o comando recibido por el Bot lo envíe a nuestra página web.

En este punto es importante añadir que el sitio web requiere de un Certificado SSL, y por mucho que se mencione por internet que un certificado autofirmado es suficiente, durante la realización de este proyecto no se pudo realizar de esta manera. Fué necesario conseguir un certificado de CERBOT, https://certbot.eff.org/lets-encrypt/debianstretch-apache, siguiendo las indicaciones de la página, y crear un nuevo sitio web.

En el Servidor, en /etc/apache2/sites-available, creamos el fichero de configuración para el nuevo sitio web seguro, activándolo con el comando 'a2ensite'.

```
 GNU nano 2.7.4                                    Fichero: telegram-ssl.conf

<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
                ServerAdmin webmaster@localhost

                DocumentRoot /var/www/telegram

                SSLEngine on

                SSLCertificateFile /etc/letsencrypt/live/proyecto-asir.ddns.net/fullchain.pem
                SSLCertificateKeyFile /etc/letsencrypt/live/proyecto-asir.ddns.net/privkey.pem

        </VirtualHost>
</IfModule>
```

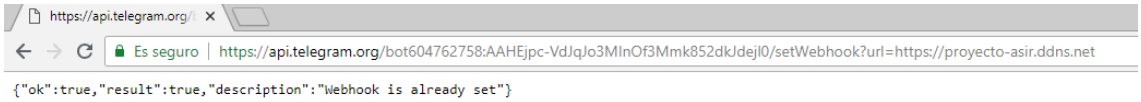
La programación del Bot en PHP requiere que antes se configure para envíe los mensajes o comandos que reciba a nuestro Servidor, así que necesitamos ejecutar la siguiente línea en un navegador:

https://api.telegram.org/botTOKEN_BOT/setWebhook?url=URL_NUESTRA_WEB

Para este proyecto, la URL quedaría así:

https://api.telegram.org/bot604762758:AAHEjpc-VdJqJo3MInOf3Mmk852dkJdejl0/setWebhook?url=https://proyecto-asir.ddns.net

De esta manera activamos Webhook, que permitirá programar el bot desde PHP.



Lo siguiente es la creación del archivo index.php que recibirá los mensajes del Bot. Este archivo se encontrará en /var/www/telegram/.

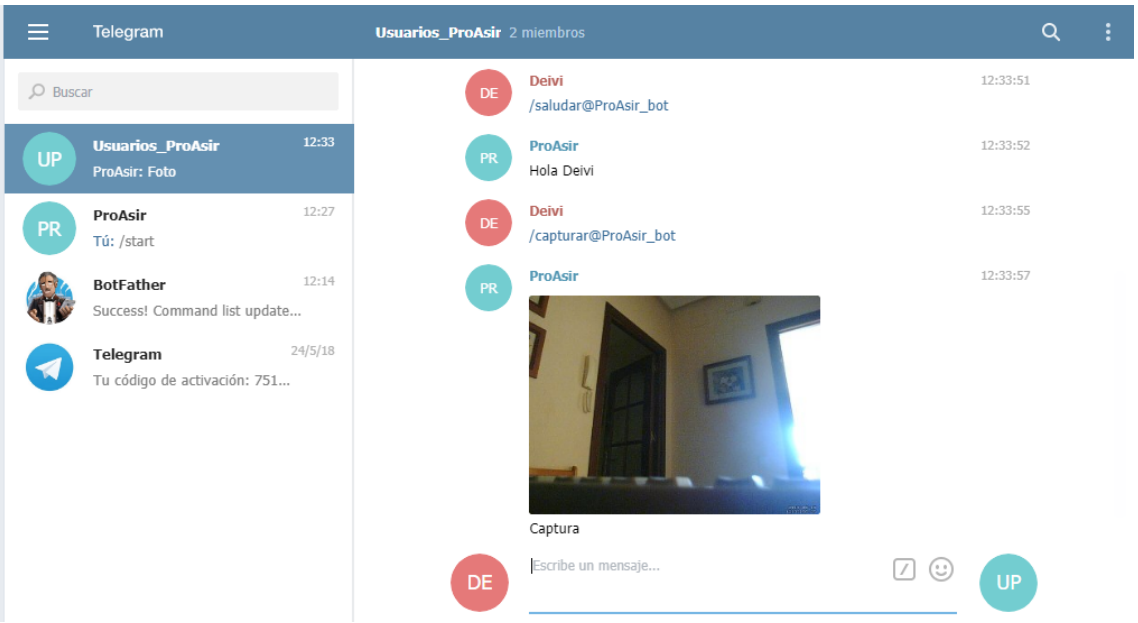
```php
<?php
    // index.php
    $botToken = "604762758:AAHEjpc-VdJqJo3MInOf3Mmk852dkJdejl0";
    $website = "https://api.telegram.org/bot".$botToken;
    $update = file_get_contents("php://input");
    $update = json_decode($update, TRUE);

    $chatId = $update["message"]["chat"]["id"];
    $chatType = $update["message"]["chat"]["type"];
    $message = $update["message"]["text"];
    $user = $update["message"]["from"]["first_name"];

    if ($chatId == "-302786092"){
        switch($message){
            case '/saludar@ProAsir_bot':
                $response = "Hola ".$user;
                sendMessage($chatId,$response);
                break;
            case '/capturar@ProAsir_bot':
                sendPhoto($chatId);
                break;
        }
    }
```

```php
    function sendMessage($chatId,$response){
        $url = $GLOBALS[website].'/sendMessage?chat_id='.$chatId.'&parse_mode=HTML&text='.urlencode($response);
        file_get_contents($url);
    }

    function sendPhoto($chatId){
        $caption = "Captura";
        echo exec("curl http://proyecto-asir.ddns.net:8080/0/action/snapshot > /dev/null");
        $output = exec("ls /var/www/html/eventos/*.jpg -t | head -1 | grep 'lastsnap' -v | rev | cut -d '/' -f1 |
        rev");
        $photo = "http://proyecto-asir.ddns.net/eventos/".$output;

        $url = $GLOBALS[website]."/sendPhoto?chat_id=".$chatId."&photo=".$photo."&caption=".$caption;

        file_get_contents($url);
    }

?>
```

En Telegram Web se crea un grupo al que solo podrán acceder aquellos usuarios que invite el administrador, consiguiendo con esto más privacidad. También se invita al Bot creado.

En el código PHP se comprueba la información que envía el Bot al Servidor y, si el identificador de la conversación no es la del grupo de Telegram, ignorará el mensaje. Si pertenece al grupo, realizará una acción u otra dependiendo del comando enviado. En este caso el Bot puede saludar al usuario o enviar una captura de la cámara en ese momento.

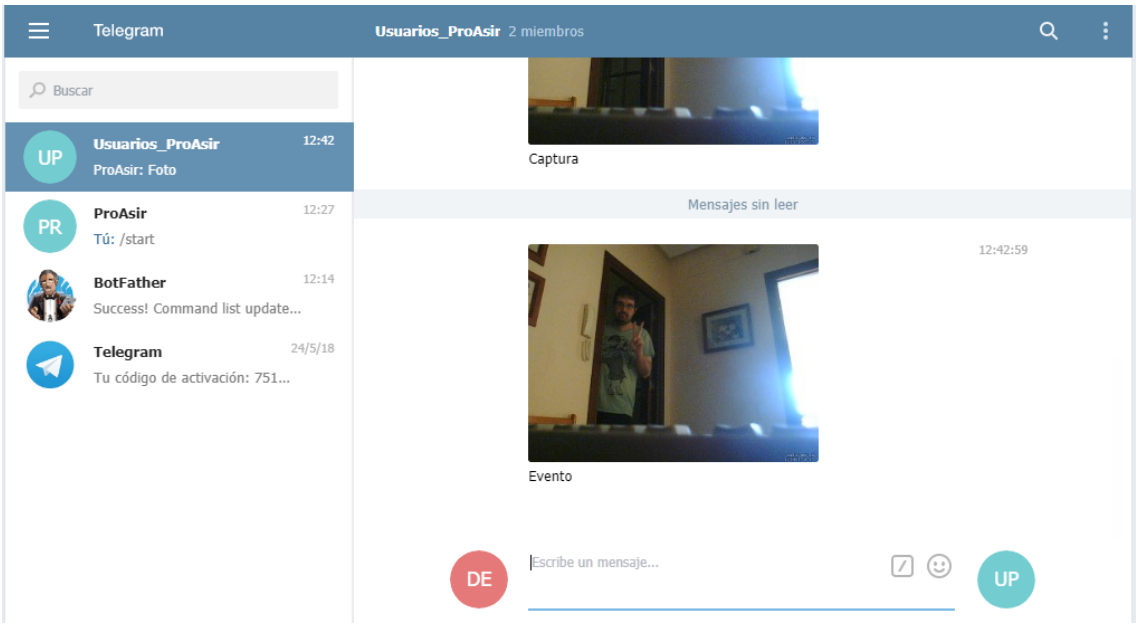
Cuando se realizan comprobaciones en el grupo el Bot responde correctamente a los comandos:



Otra de las tareas del Bot será la de enviar capturas al grupo creado anteriormente cuando Motion detecte movimiento. Antes, cuando se configuró el archivo Motion, se indicó que cuando se activase el evento de finalización de grabación de vídeo, se ejecutase un archivo PHP. Motion graba vídeo cuando detecta movimiento, así nos aseguramos que el archivo PHP se ejecuta entonces. El archivo en cuestión es enviar_evento.php.

```php
<?php
    // enviar_evento.php
    $botToken = "604762758:AAHEjpc-VdJqJo3MInOf3Mmk852dkJdejl0";
    $website = "https://api.telegram.org/bot".$botToken;
    $chatId = "-302786092";

    $caption = "Evento";
    $output = exec("ls /var/www/html/eventos/*.jpg -t | head -1 | grep 'lastsnap' -v | rev | cut -d '/' -f1 |
    rev");
    $photo = "http://proyecto-asir.ddns.net/eventos/".$output;

    $url = $GLOBALS[website]."/sendPhoto?chat_id=".$chatId."&photo=".$photo."&caption=".$caption;

    file_get_contents($url);
?>
```

En este código se indica a donde enviar la captura, el Token del Bot que recibirá la captura y el comando sendPhoto, usado para enviar imágenes. La captura se consigue mediante línea de comandos, buscando la última modificada en /var/www/html/eventos, que sería la captura realizada por Motion al detectar movimiento.



Problemas encontrados.

Uno de los primeros obstáculos durante la realización de este proyecto fue en la preparación del PC que actuaría como servidor, ya que no reconocía el monitor. Se consiguió solucionar al colocar bien la memoria RAM, ya que al parecer no estaba puesta correctamente, pero no lo suficiente mal como para que el PC diera el aviso.

La instalación del sistema operativo Debian no se realizó sin complicaciones. Los puertos USB del PC no funcionaban correctamente y, cuando se intentó usarlos para instalar Debian desde una memoria USB, durante el proceso de instalación daba error de lectura. Tampoco se pudo realizar desde el lector DVD, ya que producía el mismo fallo. Para solucionar esto se conectó un segundo disco duro con un sistema operativo Windows instalado, iniciando el instalador de Debian de la memoria USB desde aquí. Así, cuando el instalador reinició el equipo para continuar con la instalación, no mostró ningún error, instalándose Debian correctamente.

Por otro lado, aunque no formase parte íntegra del proyecto, se intentó instalar en Debian un Adaptador WiFi TP-Link para que no fuese necesario que el PC estuviese al lado del router. Se consiguió, pero la conexión se cortaba en ocasiones, y se consideró que esto era inaceptable para la realización del proyecto. Ahora, el servidor se encuentra conectado por cable de red al router.
PlayStation Eye no fue la primera opción como cámara. Se consiguió una cámara de vigilancia AXIS 2110 que podía conectarse al router por cable de red, pero la imagen que mostrada era borrosa, y se descartó en favor de la actual cámara.

Motion es una aplicación que detecta movimiento y permite realizar capturas de imagen, pero resultó imposible grabar vídeos. En principio se quería hacer capturas y grabaciones con la colección FFmpeg, pero no podía acceder a la cámara mientras Motion estuviese ejecutándose. Así resultó imposible que el administrador tuviera la opción de hacer grabaciones de vídeo.

Para acabar, Telegram también dió problemas al enviar capturas al Bot cuando Motion detectase movimiento. Cuando Motion realiza una captura por detección de movimiento, crea a su vez un archivo que enlaza a esta última imagen, así solo era necesario mirar ese enlace para conseguir la última captura de la aplicación. En la página funcionaba, pero con Telegram siempre se enviaba la misma imagen, como si estuviese almacenada en memoria. Para solucionar esto se buscó con líneas de comando linux la última imagen, en vez de mirar el enlace que creaba Motion.

Modificaciones sobre el proyecto planteado inicialmente.

No se ha podido usar una cámara de vigilancia inalámbrica, sino una cámara USB.

No se ha podido programar que el administrador realice grabaciones de vídeo.

Posibles mejoras al proyecto.

Motion permite configurar más de una cámara, sin importar si están conectadas directamente al servidor o no. Cada cámara puede contar con su propio archivo de configuración propio, mientras que el archivo motion.conf actuaría como configuración predeterminada. Sería interesante añadir más cámaras con las que interactuar.

Los usuarios que inicien sesión podrían filtrar las capturas que visualicen en la página por fecha u hora de creación, al igual que con los vídeos que se visualizan en la pestaña de eventos.

En este proyecto se pudo configurar parámetros de Motion desde la página y, siendo así, podrían añadirse más opciones de configuración que interesasen al administrador, como las áreas de detección de movimiento de la cámara.

Bibliografía.

http://www.lavrsen.dk/foswiki/bin/view/Motion/ConfigFileOptions

https://www.redeszone.net/2017/01/09/utilizar-cron-crontab-linux-programar-tareas/

https://www.w3schools.com/

https://www.youtube.com/watch?v=NRBXuvmGi8M

https://core.telegram.org/bots/api#getting-updates

https://certbot.eff.org/lets-encrypt/debianstretch-apache

https://www.noip.com/