

E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Backend sistema de detección de meteoritos por radio



Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Jaime Viscarret Aranzadi

Tutor: Mikel Izal Azcárate

Pamplona, 9 de septiembre de 2015

ÍNDICE

1. Resumen.....	pág. 3
2. Introducción	pág. 4
3. Objetivos	pág. 6
4. Estado del Arte	pág. 7
4.1 Radio Detección de meteoros.....	pág. 7
4.2 Detección por cámaras CCD	pág. 8
5. Metodología	pág. 10
5.1 Descripción de la base de datos	pág. 12
5.1.1 Administradores	pág. 13
5.1.2 Observador	pág. 14
5.1.3 Observación	pág. 14
5.1.4 Tipodispositivo	pág. 15
5.2 Programa JAVA para leer y enviar los datos al servidor	pág. 17
5.3 Archivo PHP para recoger los datos	pág. 20
5.4 Zona de administrador y zona pública	pág. 22
6. Resultados	pág. 33
6.1 Prueba con WIRESHARK.....	pág. 36
7. Conclusiones	pág. 40
8. Bibliografía	pág. 42

1. RESUMEN

En este proyecto se pretende hacer conjuntamente con otras personas un sistema de detección de meteoritos por radio. Concretamente, en esta parte se va realizar la creación de una base de datos y un backend para poder administrar y observar los datos obtenidos por el dispositivo receptor.

Para captar la aparición de meteoritos se utiliza un método que consiste en captar las ondas de radio que han sido reflejadas en las altas capas de la atmósfera. Una vez recibidos, son analizados por el software Colorgramme, el cual tiene como salida un archivo de texto .TXT.

Se ha creado un programa escrito en JAVA para buscar estos archivos dentro del directorio, y poder ser enviados al servidor de forma segura SSL, mediante una conexión HTTPS. Éste se ejecuta periódicamente de forma automática.

Para finalizar, se ha realizado una zona visible al público dentro de la aplicación web, donde se observan los datos detectados más recientes en forma de gráfica.

2. INTRODUCCIÓN

Desde hace muchos años se ha estudiado el paso de meteoroides cercanos a la Tierra, debido a la preocupación de las personas por posibles catástrofes o simplemente por el hecho de conocer más a fondo todo lo que nos rodea y nos afecta. En muchos lugares del mundo se están estudiando técnicas para poder detectar cuando estos meteoroides consiguen traspasar la atmósfera terrestre. Este trabajo se centra en el comportamiento que tienen las ondas de radio sobre la estela que dejan los meteoritos cuando llegan a la atmósfera.

Los meteoroides son restos pequeños de roca y metal que pueden proceder de asteroides y cometas [1]. Muchos de ellos son atraídos por la gravedad terrestre. La gran mayoría se vaporizan al llegar a nuestra atmósfera, dejando un rastro visible de polvo brillante al que popularmente se le atribuye el nombre de “estrella fugaz”. Entonces, los meteoritos son meteoroides que han conseguido alcanzar la atmósfera terrestre y llegan a la Tierra.

Concretamente, este trabajo se centra en los meteoritos. A continuación se explica brevemente el proceso que se produce al entrar en la atmósfera.

Cuando un meteoroide entra en la atmósfera, la estela luminosa que se forma se debe principalmente a la vaporización e ionización del cuerpo en el aire por la compresión de este. Al contacto con la capa de la atmósfera terrestre la energía cinética del cuerpo provoca el calentamiento e ionización (plasma), la cual se convierte en el haz de luz que alcanzamos a ver. Eso se produce a una altura de entre 80 y 120 km de altitud y a una velocidad de entre 11 y 72 km/s. El tamaño de estos meteoritos no llega a superar el peso de 1 gramo. Las propiedades de este plasma son las que condicionan su capacidad para reflejar una onda. [2]

El método para poder detectar estos meteoritos consiste en enviar una onda desde una antena transmisora la cual viaja hasta llegar a la atmósfera,

donde es reflejada por la estela de electrones que deja a su paso la entrada del meteorito. Es entonces cuando la onda es captada por la antena receptora.

Al ser un proyecto conjunto, otro departamento es el encargado de diseñar el sistema de antenas y detección. Esta parte se encarga en el envío y administración de los datos que han sido obtenidos y enviados a una máquina cliente por un puerto serie. A partir de ahí son analizados por un software libre que se ha obtenido en <http://radio.meteor.free.fr/fr/en/index.html>, llamado "Colorgramme Lab". Este proporciona unos archivos de texto de salida, los cuales se leen y envían automáticamente a la base de datos mediante un programa que se ha realizado.

En la zona de administrador es posible observar todos los datos que han sido obtenidos. Además se ha realizado una gráfica con los más recientes.

3. OBJETIVOS

El objetivo de este trabajo es crear un backend para administrar los datos que son recogidos y enviados desde el cliente. En él se podrá acceder a toda la información: tipo de antena, año, mes, día y hora concreta en la que se ha producido la detección.

Para ello, también es necesario crear un programa que corra en una máquina cliente, la cual debe estar conectada el tiempo que desee el que realice las muestras, podría ser una Raspberry en un futuro. También es necesario alojar en el servidor un php que se encargue de recoger los datos que han sido enviados a través del programa, y desde ese mismo php son enviados a la base de datos de forma directa.

Para realizar el proceso de envío de datos de cliente-servidor es necesario que se haga de forma segura. Para ello se enviarán vía SSL para que los datos estén encriptados, y para mayor seguridad, se pasará una clave que solo el servidor conozca. De esta forma no será posible que un usuario intruso envíe datos dañinos o capture datos.

4. ESTADO DEL ARTE

Existen aficionados a la astronomía y a la fotografía con ganas de poder observar las llamadas “lluvias de estrellas”. Existe una en especial de gran actividad, que se extiende desde el 16 de julio hasta el 24 de agosto cada año, dándose su máximo el día 10. Son meteoritos de alta velocidad (59km/s) que radian de la constelación de Perseo, y por ello reciben el nombre de Perseidas. Estos meteoritos provienen del cometa 109P/Swift-Tuttle, el cual orbita alrededor del Sol con un periodo de 135 años. [3]

Debido al interés astrofísico, o simplemente por el hecho de controlar más los acontecimientos que se producen en el firmamento, existe cada vez más una gran cantidad de sistemas de detección de meteoritos. Podemos encontrarnos con varios métodos para la detección, pero solo comentaremos los principales.

Algunos de ellos utilizan procesado de imágenes de cámaras, otros captan ondas de radiofrecuencia reflejadas, y otros, simplemente, se encargan de observar diariamente el firmamento en busca de posibles meteoroides.

4.1 Radio Detección de Meteoros

Estos sistemas utilizan las ondas, que se reflejan en la estela que dejan los meteoroides al entrar en la atmósfera, para detectar y contar el número de meteoritos que llegan a la Tierra. Como ya se ha explicado anteriormente, cuando un meteoroide llega a las capas altas de la atmósfera, la fricción hace que la temperatura del cuerpo aumente rápidamente, pudiendo alcanzar los 2000 K. A esa temperatura es cuando comienza la sublimación, dejando un rastro de electrones tras de sí. Este rastro (aire ionizado) es el que interesa ya que es capaz de reflejar las ondas de radio desde 30 Mhz hasta 500 Mhz.[4]

Las ionizaciones puedes ser de pocos segundos de duración (pings) o de hasta varios minutos (burst). Cuando se producen las lluvias de estrellas, es posible establecer conversaciones de radioaficionados a grandes distancias debido a la transmisión de las ondas por medio de la atmósfera.

Este método consiste en utilizar una antena transmisora que emite en ese rango de frecuencias. Las ondas se desplazan hasta la atmósfera donde son reflejadas por el rastro ionizado que ha dejado el meteorito, y se recibe por medio de una antena receptora. A la salida de este receptor se conecta un dispositivo como puede ser un ordenador, que tenga una tarjeta de sonido.

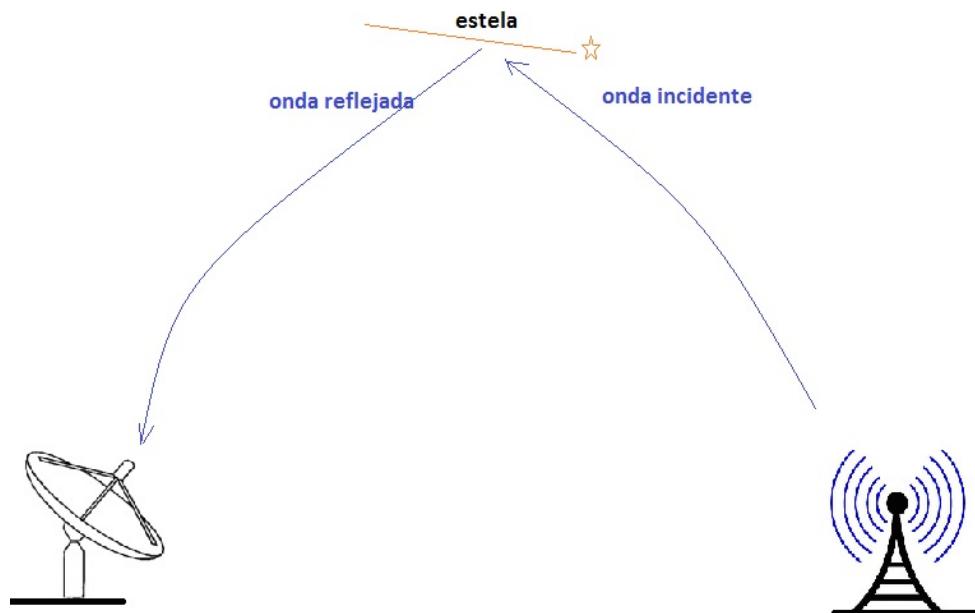


Figura 4.1 Funcionamiento de Radio detección

Para poder analizar los ecos del sonido existen software libres como por ejemplo el HROFF, y pueden ser analizados y representados por medio de otros software como el Colorgramme Lab V 2.5, el cual se pretende usar para llevar a cabo este proyecto global.

4.2 Detección por cámaras CCD

El dispositivo de carga acoplada (CCD) es un circuito integrado formado por un conjunto de condensadores acoplados. En la fotografía digital se trata

de un sensor con pequeñas células fotoeléctricas que registran cada imagen. Estos sensores se basan en transformar la luz que captan en corriente eléctrica. La fotografía CCD ha ayudado a la astronomía sustituyéndola por la fotografía convencional a partir de 1980. Puede alcanzar hasta un 70% de sensibilidad, en comparación con la típica que se encuentra en torno al 2%. [5]

La idea consiste en utilizar varias cámaras con gran angular u ojo de pez con detectores CCD. Estas cámaras monitorizan durante toda la noche el firmamento y permite automatizar la llegada a la atmósfera de meteoroides. [6]

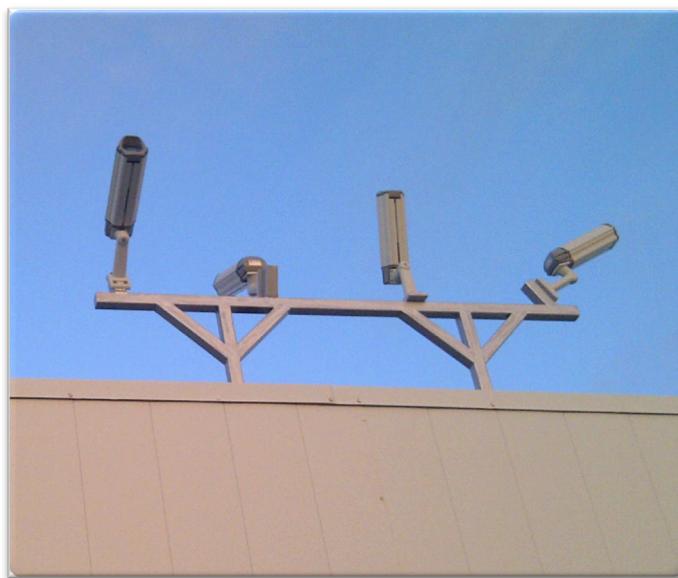


Figura 4.2 Cámaras CCD[7]

Estos equipos CCD pueden servir además para calcular la órbita que siguen, de manera que puede saberse de qué cuerpo del Sistema Solar procede (de un asteroide, planeta, etc.).

Se trata de un sistema de detección muy útil si se quiere estudiar la composición de una roca que haya caído en la superficie terrestre (si no llega a descomponerse) debido a que se conoce el lugar de impacto exacto al quedar recogido en las grabaciones.

5. METODOLOGÍA

El proyecto consta de dos partes claramente diferenciadas. Por una parte, se realiza la detección de los meteoritos por el método de reflexión de ondas de radio. Y por la otra, se ha realizado un script que se ejecuta periódicamente (cada día) para poder subir a la nube los datos que han sido obtenido y analizados por el software ColorGramme Lab 2.5.

Además, se ha creado una aplicación web donde se puede observar las detecciones más recientes con sus correspondientes gráficas. Esta aplicación web cuenta con un panel de administrador al cual solo tendrán acceso los usuarios que estén involucrados con el proyecto.

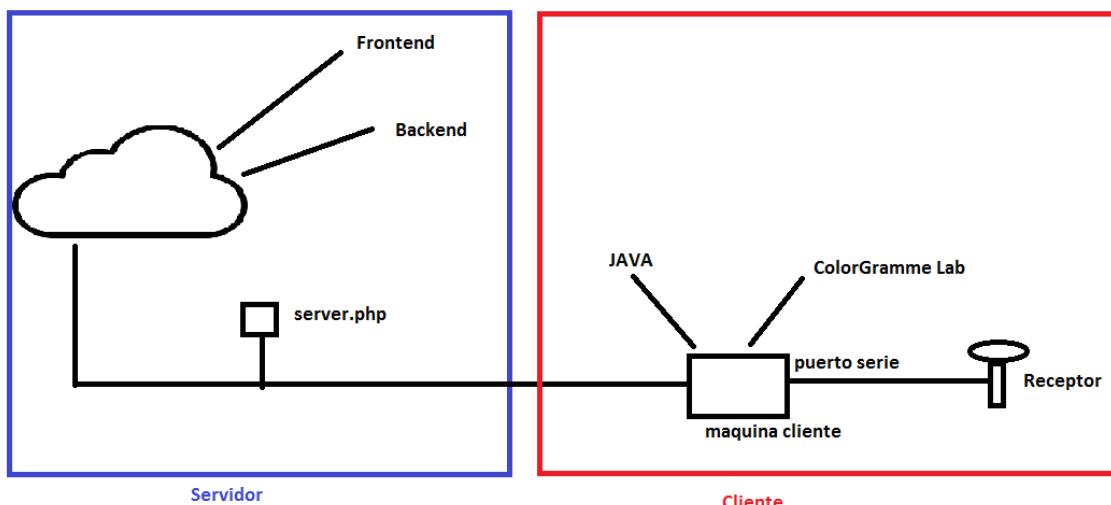


Figura 5.1 Esquema del Proyecto conjunto

Para realizar esta web se ha hecho uso de una plataforma llamada OpenShift. Se trata de un servicio de la empresa REDHAT cuyo objetivo es facilitar el trabajo del desarrollador sin tener que preocuparse de la infraestructura, con muchas opciones como PHP, JAVA, PHYTON, RUBY, etc.

En lugar de utilizar FTP para subir código como es lo habitual en Windows, este trabaja con una herramienta llamada GIT. Se trata de un software diseñado por Linus Torvalds pensado para controlar las versiones de los archivos que vamos añadiendo. [8] Puede dar detalles de la persona que ha tocado por última vez ese archivo, cuándo y qué. De esta manera es posible trabajar desde distintos dispositivos y lugares de forma conjunta.

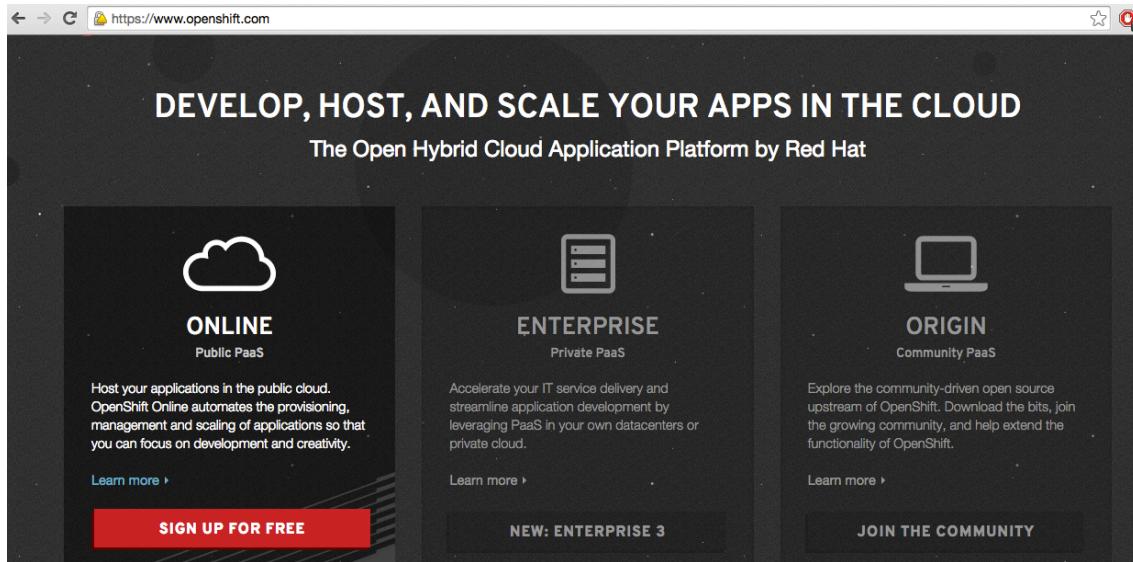


Figura 5.2 Openshift

OpenShift ofrece la posibilidad de crear 3 aplicaciones en cada cuenta de forma gratuita, con posibilidad de pagar para conseguir alguna ventajas (certificado propio para la página web, mayor capacidad de alojar datos, etc.).[9]

Una vez creada la aplicación se ha seleccionado el entorno de programación en el que se va a trabajar, eligiendo en este caso PHP 5.4, MySQL 5.5 como programa gestor de base de datos, y phpMyAdmin 4.0 para poder administrar la información que ha sido almacenada en la base de datos.

Una de las cualidades de OpenShift es que utiliza una clave pública para encriptar la conexión que se realiza entre tu máquina y el servidor, autorizándose solo a ti a poder subir y actualizar el código. Utiliza SSH para realizar las operaciones de GIT y dar acceso remoto a la aplicación desde cualquier lugar.

5.1 Descripción de la base de datos

La pieza clave de esta parte del proyecto consiste en administrar y observar los datos que han sido recogidos por el receptor y el software Colorgramme. Para ello ha sido necesario crear una base de datos con MySQL. Se trata de un sistema multiusuario (con la idea de poder trabajar de manera conjunta), libre y relacional.

El modelo relacional fue creado por el matemático Edgar Frank en 1970, acaparando gran atención por su simplicidad y funcionalidad. Se trata de un conjunto de tablas (o relaciones) con unas determinadas características que se encuentran relacionadas unas con otras. Cuando hablamos de atributos de una tabla nos referimos a las columnas de esa tabla.

La base de datos ha sido creada con el nombre “meteor” mediante la herramienta phpMyAdmin y posee las siguientes tablas:

- administradores
- observación
- observador
- tipodispositivo

A continuación se representa el modelo entidad-relación de las tablas anteriores.

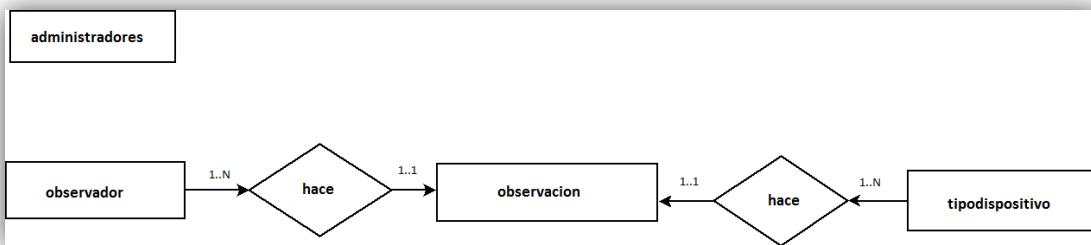


Figura 5.3 Modelo entidad-relación

En la relación existente tenemos que un observador realiza de 1 a N observaciones, y una observación solo es hecha por un único observador. Para que un observador esté registrado en el sistema debe haber hecho una observación y por eso es 1..N y no 0..N.

Además, una observación solo es hecha por un tipodispositivo (1..1), mientras que un dispositivo puede haber realizado de 1 a N observaciones. También debe haber hecho una observación para estar registrado como dispositivo, y por ello es 1..N. Los administradores no tienen ninguna relación con las demás tablas, meramente se utiliza para poder hacer login en el sistema como administrador.

Se describen a continuación las tablas que han sido creadas con sus atributos:

5.1.1 Administradores

Esta tabla ha sido creada para poder registrar y hacer login las personas que estén encargadas de observar y administrar los datos en el backend. Solo un administrador es capaz de crear un nuevo administrador, de borrar o de modificar.

Columna
idadministrador
username
password
nombre
apellido
mail

Figura 5.4 Tabla administradores

El idadministrador ha sido establecido como primary key (clave primaria) de la tabla. De esta manera podemos identificar cada tupla de la relación de manera inequívoca, ya que nadie puede tener el mismo idadministrador.

El password ha sido creado de manera que va encriptado en MD5, aunque existen otros métodos más seguros como Blowfish.

5.1.2 Observador

Al utilizar el script que lee los datos y los sube a la nube aparece el nombre de la persona que ha realizado esa determinada observación. Para poder listar bien las observaciones se ha decidido crear una tabla de observador con los siguientes atributos.

observador	
	Columna
	idobservador
	observador

Figura 5.5 Tabla observador

En este caso el idobservador es la primary key, la cual nos ayudará a relacionar esta tabla con la de la observación.

5.1.3 Observación

En esta tabla es donde se guardan todos los datos que se recogen de cada hora del día de cada mes (si se ha realizado correctamente la observación). Además aparecen datos que pueden ser más relevantes

Columna
idobservacion
idobservador
idtipodispositivo
mes
dia
detecciones
year

Figura 5.6 Tabla observación

En lugar de año se ha utilizado “year” debido a un problema que podía surgir por el uso de la ñ. De esta manera se puede saber cuántos meteoros se detectaron en una determinada hora del día. Se ha optado por utilizar como primary key el idobservacion, y como foreing key (o clave extranjera) idobservador e idtipodispositivo (los dos atributos son enteros).

De esta manera ha sido posible establecer unas reglas en cuanto a esos dos atributos. Si por casualidad el idobservador en la tabla de observador es actualizado, automáticamente la base de datos cambiará ese idobservador de la tabla observación. Como el idobservador y el idtipodispositivo son foreing key no es posible eliminar una tupla de observador o tipodispositivo si existen datos relacionados con estos. Si por casualidad esos datos son borrados, entonces sí se podría llegar a eliminar las tuplas que quedaban ligadas a esta.

5.1.4 Tipodispositivo

Se trata del dispositivo receptor que se ha utilizado para recoger las ondas reflejadas. Todos estos datos son utilizados para el análisis de los datos por el Colorramme.

tipodispositivo	
	Columna
	idtipodispositivo
	dispositivo

Figura 5.7 Tabla tipodispositivo

Como en los casos anteriores el idtipodispositivo es la primary key de la tabla e identifica a todas las tuplas como únicas.

5.2 Programa JAVA para leer y enviar los datos al servidor

El programa se ha realizado en JAVA con la ayuda de un par de librerías de JSON. La función que realiza principalmente es la búsqueda de archivos .TXT dentro del directorio que se encuentra (así aparecen los datos guardados por el Colorgramme), y va leyendo uno a uno extrayendo toda la información relevante.

Una vez que se ha guardado todo, se establece una conexión HTTPS (Hyper Text Transfer Protocol) con un archivo PHP alojado en el servidor, y se envían por el método POST. De esta forma los datos se envían encriptados y de forma segura. Lo que ocurre es que la dirección en la que se encuentra el php en el servidor posee un certificado de seguridad (SSL).

Además del protocolo HTTPS se ha implantado otra medida de seguridad para proporcionar al sistema. Se ha utilizado una clave que solo conoce el cliente y el servidor. Una vez que el servidor recibe los datos comprueba si la contraseña es correcta y, de ser así, comienza a subir las medidas a la base de datos.

Para ejecutar el programa de forma automática cada día se ha utilizado el demonio cron que mira en el archivo de texto crontab la lista de comandos deseados a ejecutar. Crontab mira la fecha y hora además de los permisos de ejecución y lo realiza en el background. Una vez que los datos son enviados correctamente los archivos pasan a ser eliminados del directorio.

A continuación se explica con algo de detalle las partes principales del código implementado.

El archivo de texto que lee el programa tiene la siguiente forma (se ha obtenido como ejemplo de la página <http://www.rmob.org/download.php?Ing=en>):

bug	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	20h	21h	22h	23h
01	40	21	34	12	22	12	14	10	20	10	29	22	16	11	16	20	27	35	22	13	12	13	12	10
02	10	14	19	23	25	35	23	22	35	27	57	52	18	18	12	13	25	29	33	15	8	7	7	15
03	17	5	26	15	14	16	13	14	27	28	27	25	23	24	16	18	19	23	11	12	13	8	12	14
04	27	22	22	17	20	9	18	19	30	18	31	27	18	17	16	22	24	19	14	22	20	???	20	10
05	10	10	16	14	23	15	14	11	24	25	27	27	15	23	16	26	32	15	34	9	17	11	12	8
06	15	17	14	18	21	19	44	18	31	34	24	32	21	20	16	30	20	17	26	13	9	9	16	14
07	11	31	8	24	13	21	24	19	30	28	27	27	20	19	16	22	36	31	14	16	14	23	23	23

Figura 5.8 Archivo.TXT

En primer lugar, una vez que se ha seleccionado el archivo .TXT se va leyendo línea a línea hasta obtener y guardar todos los datos. A continuación, se crea un objeto JSON mediante la librería que ha sido importada previamente, y se le introducen los datos deseados con su clave correspondiente para poder ser obtenidos posteriormente en el servidor. Como por ejemplo la clave que se envía al servidor para comprobar que somos el cliente.

```
JSONObject jsonObj = new JSONObject();
jsonObj.put("clave", clave);
```

Figura 5.9

Los datos son concatenados y enviados en una sola cadena para disminuir las conexiones que se realizan con el servidor. Una vez allí pasan a ser obtenidos de forma individual para poder ser subidos a la base de datos.

```
jsonObj.put("datos", datos);
```

Figura 5.10

40,21,34,12,22,12,14,10,20,10,29,22,16,11,16,20,27,35,22,13,12,13,12,10,11,10,14,19,23,25,35,23,22,35,27,57,52,18,18,12,

Figura 5.11

Se genera la URL a la que se va a enviar, se establece la conexión HTTPS y se indica el método que utilizamos para enviar los datos, en este caso POST.

```

String url = "https://lol-jvisca.rhcloud.com/rest/server.php";
// creamos el objeto url para mandar el json
URL obj = new URL(url);
// establecemos la conexión
HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();

// añadimos cabecera
con.setRequestMethod("POST");
    
```

Figura 5.12

Por último, se envían los datos por el método POST y se cierra la conexión.

```

con.setDoOutput(true); //
DataOutputStream wr = new DataOutputStream(con.getOutputStream());
wr.writeBytes(urlParameters);
wr.flush(); //obligamos a que mande todo y se quede el buffer vacío.
wr.close();
// para comprobar que se manda todo
int responseCode = con.getResponseCode();
    
```

Figura 5.13

El proceso se repite de manera iterada tantas veces como archivos .TXT haya en el directorio en el que se encuentra. Finalmente, los archivos son eliminados.

Existen una gran cantidad de lenguajes de programación con el que poder realizar este programa. Posiblemente sea más adecuado utilizar PHYTON, pero nos hemos decantado por JAVA.

```

[Observer]Grahame Booth
[Country]Canada
[City]Calgary, Alberta
[Longitude]114°13'15" W
[Latitude]051°09'07" N
[Longitude GMAP]-114.2207169
[Latitude GMAP]51.1518709
[Frequencies]61.240 MHz
[Antenna]Horizontal Full Wave Loop
[Azimut Antenna]0
[Elevation Antenna]0
[Pre-Amplifier]None
[Receiver]ICOM IC-PCR1000
    
```

Figura 5.14 Final del archivo TXT

5.3 Archivo PHP para recoger los datos

Se ha creado un archivo server.php el cual se encuentra alojado en el servidor (dentro de la aplicación que hemos creado previamente). Este se encarga de coger los datos que se envían por POST, darle el formato deseado para ser introducidos en las tablas y hacer la conexión con la base de datos.

Para poder subir los datos a la tabla observación es necesario hacer varias consultas previas a esta. Como se ha explicado anteriormente, el observador y el tipodispositivo deben ser registrados en la base de datos con el fin de establecer la relación entre las tablas (idobservador y idtipodispositivo).

Por ello, en primer lugar, se inserta en la tabla observador el nombre del observador. Al estar configurado el idobservador como autoincrement, este irá aumentando en 1 automáticamente cada vez que se inserte uno nuevo. Se realiza la misma acción con el tipo de dispositivo.

Una vez hecho esto, se hacen dos consultas más, esta vez para saber el idobservador y el idtipodispositivo que han ocupado en sus respectivas tablas.

```

if($rep1 ==0){
    $queryaux = "insert into observador(observador) values ('$observador')";
    $quoro = $aux ->query($queryaux);
    $idobservador= $aux ->insert_id;
}else{
    $row = $rep->fetch_array(MYSQLI_ASSOC);
    $idobservador = $row['idobservador'];
}

if($rep3 ==0){
    $queryaux2 = "insert into tipodispositivo(dispositivo) values ('$antena')";
    $quoro2 = $aux ->query($queryaux2);
    $idtipodispositivo = $aux ->insert_id;
}else{
    $row2 = $rep2->fetch_array(MYSQLI_ASSOC);
    $idtipodispositivo = $row2['idtipodispositivo'];
}

```

Figura 5.15 Obtención del idtipodispositivo y idobservador

Cuando ya se conocen todos los datos, se realiza un bucle que recorre todos los datos y se van enviando uno a uno en un bucle “for”.

Se muestra a continuación la parte del código donde se suben todos los datos:

```
for($i=0;$i <count($datosArr);$i++){
    $d = $datosArr[$i];
    if($d == "???"){
        $d="0";
    }
    if($d == "|||"){
        $cont= $cont+1;
    }
    if($d != "|||"){
        $query = "insert into observacion(idobservador,idtipodispositivo,mes,dia,detecciones,year) values ('$idobservador','$idtipodispositivo','$mes','$dia','$detecciones','$year')";
        $resultado = $aux ->query($query);
        echo $resultado;
    }
}
```

Figura 5.16 Parte del código de server.php

5.4 Zona de administrador y zona pública

En principio la idea era solo realizar una zona de administrador donde poder tratar y manejar los datos de las detecciones, nos pareció apropiado dejar una parte pública con gráficas donde poder observar las últimas novedades de los datos captados. Toda la zona web se ha realizado con HTML, PHP, JAVASCRIPT y CSS para darle algo de estilo. El enlace es: <https://lol-jvisca.rhcloud.com>

Utilizando herramientas como Bootstrap se ha logrado dar estilo de forma sencilla utilizando los css que vienen predefinidos. Además, se ha realizado para versión web y para móvil.



Datos más recientes

Aquí se muestran los datos referidos a las últimas observaciones obtenidas a lo largo del mes.

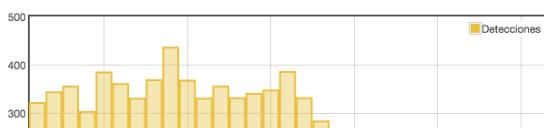




Figura 5.18 Captura de pantalla versión móvil

En el Frontend de la página nos encontramos con dos pestañas en el menú superior: una de inicio donde se muestra una breve introducción del proyecto, además de una gráfica que muestra las últimas detecciones que han sido guardadas en la base de datos durante el último mes, y otra pestaña de información donde se habla muy brevemente sobre la teoría de detección de meteoritos por radio.

Para la realización de la gráfica se ha utilizado la herramienta flot chart. Simplemente es una página web en la que están disponibles unas hojas de estilo y unos archivos js para poder realizar gráficas de forma sencilla. El resultado es bueno y no tiene gran dificultad para ser utilizado.



Figura 5.19 Número de detecciones a lo largo del último mes

En esta zona pública prácticamente no se hacen llamadas a la base de datos, simplemente se recogen los últimos datos que se han obtenido para ser representados en la gráfica que aparece en la parte inferior del inicio.

Para representar todos los datos se utiliza una función de flotchart.com que ha sido descargada previamente. En `##1##,##2##`, etc. se introducen los datos deseados desde el `vista.php`. (figura 5.20). Cabe decir que se ha pretendido usar el método MVC (modelo vista controlador), utilizando este modelo para realizar la página web, se puede dividir el trabajo de forma clara.

El controlador es el encargado de llamar a la vista y al modelo. El `modelo.php` se encarga de hacer las consultadas a la base de datos y devolver en una variable los datos obtenidos a la vista. El `vista.php` se encarga de

recoger esos datos, tratar con ellos e insertarlos en la plantilla (en este caso un archivo HTML).

Es un método muy eficaz para hacer código ya que si se desea cambiar el entorno de trabajo de web a móvil u otro lugar, el modelo nos serviría para cualquiera tipo. Solo sería necesario sustituir el vista con las plantillas deseadas e introducir los datos en ellas.

```
<script type="text/javascript" src="/js/flot/jquery.js"></script>
<script type="text/javascript" src="/js/flot/jquery.flot.js"></script>
<script type="text/javascript">
var data = [[1, ##1##], [2, ##2##], [3, ##3##], [4, ##4##], [5, ##5##], [6, ##6##],
            [12, ##12##], [13, ##13##], [14, ##14##], [15, ##15##], [16, ##16##], [17, ##17##],
            [22##], [23, ##23##], [24, ##24##], [25, ##25##], [26, ##26##], [27, ##27##], [28, ##28##]];
var dataset = [{label: "Detecciones", data: data}];
var options = {
    series: {
        lines: { show: false, steps: false },
        bars: {show: true, barWidth: 0.9, align: 'center'}
    }
};
$(document).ready(function () {
    $.plot($("#flot-placeholder"), dataset, options);
});
</script>
```

Figura 5.20 Código para dibujar la gráfica con flotchart

Este hace una llamada a la base de datos con el máximo idobservacion para ver cuál ha sido el último dato introducido. Una vez que se conoce, se mira cuál es el observador, el mes y el año para realizar otra consulta donde se recogen todos los datos de ese mes. El lugar para poder acceder a la zona de administración es: <https://lol-jvisca.rhcloud.com/jjadmin>

En cuanto a la zona de administrador, para acceder a ella es necesario hacer login, y por lo tanto estar registrado. Si el usuario y la contraseña son correctos, se guardará la sesión en una variable `$_SESSION()` para poder navegar por la zona de administrador sin que expire.



Iniciar sesión

The login form consists of several input fields and controls. It includes a 'Username' field with a light blue border, a 'Password' field with a light grey border, and a 'Recordarme' checkbox followed by the text 'Recordarme'. At the bottom is a large blue rectangular button labeled 'Login'.

Figura 5.21 Login de la zona de administración

Hay que tener precaución a la hora de realizar los inputs del formulario HTML debido a que los campos que se introducen van a realizar una llamada a la base de datos, y esto puede dar problemas si no se protege. Pueden aparecer los ataques de SQL INJECTION. Para evitar esto lo único que hay que hacer es filtrar bien las variables utilizadas en la consulta. Los atacantes pueden llegar a obtener cualquier información que se encuentre en la base de datos, así como usuarios, contraseñas, datos privados, etc. Puede utilizarse el comando `mysql_real_escape_string` para proteger las consultas.

Una vez que se ha iniciado sesión tenemos el siguiente menú de opciones disponibles.

METEOR Inicio Información

Administradores

- [Nombrar nuevo administrador](#)
- [Modificar / Eliminar administradores](#)
- [Listado de administradores](#)

Observadores

- [Modificar observadores](#)
- [Listado de observadores](#)

Observaciones

- [Listado de Observaciones](#)

Tipos de dispositivos

- [Modificar tipo de dispositivo](#)
- [Listado de dispositivos](#)

Figura 5.22 Menú de administración

Es necesario rellenar los campos que aparecen en la figura 5.23 para poder registrar al nuevo administrador (nombre y apellido son opcionales). Se comprueba que el Username no esté cogido y que las 2 password sean la misma para poder insertar en la base de datos al administrador.

Si pinchamos en la pestaña Modificar / Eliminar administradores, nos aparece una lista con los administradores que existen en el sistema, además de las dos opciones posibles (figura 5.24).



Alta de Administrador

Username	<input type="text"/>
Nombre	<input type="text"/>
Apellido	<input type="text"/>
Password:	<input type="text"/>
Repeat password:	<input type="text"/>
Mail:	<input type="text"/>
<input type="button" value="Submit"/>	

Figura 5.23 Alta de administrador

Administradores

Username	Nombre	Apellido	Mail	Modificar	Eliminar
jaime	Jaime	Viscarret	jaime_visca@hotmail.com	Modificar	Eliminar

[Volver](#)

Figura 5.24 Modificar/Eliminar Administradores

En modificar nos aparecerá un formulario relleno con los datos del administrador seleccionado para poder ser modificado y enviado.

Modificación de administradores

Username:

Nombre:

Apellido:

Correo:

Escribe una nueva contraseña si deseas cambiarla
En caso contrario no pongas nada

Password:

Figura 5.25

Como puede verse en la Figura 5.25 si no introducimos ninguna contraseña, se conservará la que ya existía.

En el caso de darle a eliminar nos aparecerán los datos del administrador y un botón para borrar, que una vez pulsado, será eliminado de la base de datos y ya no tendrá acceso al panel de administración.

En la parte de Observadores tendremos una zona muy similar a la de administradores y a la de dispositivos. Una vez que se ha realizado una detección completa de todo el mes el archivo es subido a la nube, y es ahí cuando se registra un nuevo observador (si no existía) o no se realiza ninguna acción, ya que los observadores no están repetidos.

A diferencia del apartado anterior de administradores, se ha suprimido el botón de eliminar, ya que como se ha explicado anteriormente al ser el idobservador clave extranjera de la tabla de observaciones no puede ser borrado un observador que relaciona una observación con su idobservador.

Un ejemplo sería tener una tupla de la tabla observación cuyo atributo idobservador sea igual a 1. En ese caso si se eliminara de la tabla observador el que corresponde con el idobservador 1 se daría un error al no existir un idobservador 1 correspondiente a la tupla de observación.

Modificación de observadores

Observador	Modificar
Grahame Booth	Modificar
Franky Dubois	Modificar

Figura 5.26

Al hacer click en el listado de observaciones aparece una lista con los diferentes observadores que han subido datos.

Observaciones

Observador	Mas detalles
Grahame Booth	Mas detalles
Franky Dubois	Mas detalles

Figura 5.27

Dando a “más detalles” se obtiene una lista de los años en los que ha hecho observaciones el observador determinado. Dándole de nuevo a “más detalles” se pueden ver los meses en los que se han recogido muestras de datos a lo largo de ese año y de esa observación concreta.

Observador : Franky Dubois

Year : 2014

Mes	
mar	Mas informacion
apr	Mas informacion
may	Mas informacion
jun	Mas informacion
Jul	Mas informacion
aug	Mas informacion

Figura 5.28

Por último, se selecciona un mes y se observa una tabla con todos los detalles de la recogida de datos. Se trata del número de detecciones que se han registrado en cada hora de cada día del mes, en ese año y con ese observador. Esta es la consulta que se realiza a la base de datos para poder obtener los datos correspondientes (figura 5.29). Se hablará de los resultados y de las tablas en el apartado siguiente.

```
$cnx = mConexion();
$id = $_GET["idobservador"];
$year = $_GET["year"];
$mes = $_GET["mes"];
$query = 'select detecciones from observacion where idobservador = "'.$id.'" AND year = "'.$year.'" AND mes = "'.$mes.'"' ;
$resultado = mysql_query($query);

return $resultado;
```

Figura 5.29

Cabe decir que la zona de los dispositivos es muy similar a la de los observadores y no se va a explicar a fondo. Solamente dejar una imagen de algunos de los dispositivos registrados.

Dispositivos

Nombre de Dispositivo

ICOM IC-PCR1000

HRO MRX-50

Icom IC-R75

Figura 5.30

6.RESULTADOS

En este apartado se muestran los resultados más representativos de la subida de datos, de las tablas de observaciones y de la base de datos. En primer lugar, se muestra en la figura 6.1 el formato para enviar los datos al servidor. Se ha creado un objeto de tipo JSON con todos los campos que interesaban (dispositivo, observador, mes, año, y los datos de ese mes), para ello se han ido concatenando los datos de cada hora en una sola cadena, poniendo “||” para saber que se trata de un cambio de día. De esta forma solo se envían datos una vez por cada archivo .TXT.

Para comprobarlo se ha implementado una línea en el código para imprimir por pantalla los datos que van a ser enviados. El resultado es el de la Figura 6.1

```
Main (3) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home/bin/java (29/8/2015 10:28:13)
Naoya Saito
datos : 8,11,16,1,5,4,1,???,???,3,2,7,8,12,8,9,11,15,10,24,18,15,11,18,||,14,9,6,5,5,7,6,3,3,4,7,7,13
feb
Naoya Saito
datos : 13,6,14,7,3,2,2,1,1,1,5,8,4,3,17,14,15,17,20,21,23,13,11,15,||,7,11,10,7,4,4,3,6,3,2,???,4,9,
mar
```

Figura 6.1 Envío de datos al servidor

Se puede observar que el resultado es de 24 datos concatenados con comas, y que aparece “||” para poder diferenciar el cambio de día. Una vez en el servidor se realiza una separación por comas para formar un array con un dato en cada posición.

```
$datos = $dec->datos ;
$datosArr = preg_split("/[\s,]+/", $datos);
```

Figura 6.2

En la imagen 6.2 tenemos un fragmento del código de server.php, donde se guarda en la variable \$datos la cadena mostrada en la figura 6.1, y en \$datosArr el array con los datos individuales. Una vez que se tiene el array podemos obtener un dato de cada posición.

Para demostrar que los datos han sido subidos al servidor correctamente tenemos dos opciones. Una es mirar en la base de datos y otra en la zona de administración gracias al apartado de observaciones.

En la figura 6.3 se muestra el número de observaciones que se han realizado en el mes de febrero del año 2014 por el observador Naoya Saito. Se puede comprobar que son los mismos datos que aparecían en la figura 6.1

METEOR	Inicio	Información	Administracion	Salir																				
Observador : Naoya Saito																								
Year : 2014																								
feb	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	10h	11h	12h	13h	14h	15h	16h	17h	18h	19h	20h	21h	22h	23h
1	13	6	14	7	3	2	2	1	1	1	5	8	4	3	17	14	15	17	20	21	23	13	11	15
2	7	11	10	7	4	4	3	6	3	2	0	4	9	6	9	15	19	20	22	15	16	12	6	10
3	10	16	8	4	4	4	1	3	2	0	3	4	10	7	13	4	14	12	19	12	16	21	8	13
4	7	8	11	5	3	5	0	1	0	2	2	1	7	12	16	16	11	17	27	16	13	19	12	19
5	13	12	8	6	5	1	0	2	3	4	1	5	8	17	17	11	9	22	19	21	15	12	10	21
6	9	2	6	3	12	3	3	2	0	5	8	5	16	12	9	29	23	19	23	21	18	15	17	18
7	6	8	4	5	1	3	0	0	3	2	2	11	7	16	21	10	10	14	13	24	29	9	12	19
8	20	21	11	13	6	3	3	1	16	5	0	0	0	16	9	8	13	18	11	17	9	11	4	
9	14	9	6	7	6	3	1	1	1	2	4	2	4	9	13	5	7	12	8	10	11	12	5	4
10	16	10	2	6	6	4	1	1	1	17	2	2	9	9	13	10	10	8	24	13	9	8	11	7
11	8	7	5	11	1	2	2	0	4	15	1	2	8	9	9	7	7	13	8	17	8	9	9	
12	8	9	3	5	1	2	8	4	1	4	0	6	6	20	11	15	8	12	10	22	13	8	10	19

Figura 6.3 Tabla de datos

Otra forma para verlo, como ya hemos explicado, es realizar una consulta SQL en el phpMyAdmin.

```
SELECT *
FROM observacion
WHERE idobservador = '5'
AND mes = "feb"
AND YEAR = "2014"
LIMIT 0 , 30
```

Figura 6.4 Consulta SQL

El resultado de la consulta se muestra en la Figura 6.5

idobservacion	idobservador	idtipodispositivo	mes	dia	detecciones	year
11921	5 [->]	19 [->]	feb	1	13	2014
11922	5 [->]	19 [->]	feb	1	6	2014
11923	5 [->]	19 [->]	feb	1	14	2014
11924	5 [->]	19 [->]	feb	1	7	2014
11925	5 [->]	19 [->]	feb	1	3	2014
11926	5 [->]	19 [->]	feb	1	2	2014
11927	5 [->]	19 [->]	feb	1	2	2014
11928	5 [->]	19 [->]	feb	1	1	2014
11929	5 [->]	19 [->]	feb	1	1	2014
11930	5 [->]	19 [->]	feb	1	1	2014
11931	5 [->]	19 [->]	feb	1	5	2014
11932	5 [->]	19 [->]	feb	1	8	2014
11933	5 [->]	19 [->]	feb	1	4	2014
11934	5 [->]	19 [->]	feb	1	3	2014
11935	5 [->]	19 [->]	feb	1	17	2014
11936	5 [->]	19 [->]	feb	1	14	2014
11937	5 [->]	19 [->]	feb	1	15	2014

Figura 6.5 Fragmento de la consulta

También aparece el tiempo que ha costado realizarla y el número total de valores que se han obtenido.

✓ Mostrando filas 0 - 29 (total de 745, La consulta tardó 0.0022 seg)

Figura 6.6 Tiempo de consulta

Con todo ello puede demostrarse que el método funciona y que queda claramente recogido en la base de datos.

6.1 Prueba con WIRESHARK

Otro de los objetivos que se planteaban era el envío de los datos al servidor de forma totalmente segura. Para ello utilizábamos el protocolo HTTPS estableciendo la conexión con el server.php, ya que rhcloud dispone de certificados de seguridad.

Para comprobar que realmente los datos viajan encriptados se ha realizado capturas del tráfico en red con la herramienta Wireshark. En un principio se utilizó el protocolo HTTP y se observó como los datos podían ser vistos si se conocía mínimamente el Wireshark.

En la figura 6.7 se muestra el tráfico observado de la conexión entre cliente y servidor.

No.	Time	Source	Destination	Protocol	Length	Info
59	10:45:59.000	66.10.0.1.230	192.168.1.34	TCP	46	standard query response okteto CHANNEL_CX_320.indd521.pdf.html
60	7.824094000	192.168.1.34	54.172.250.123	TCP	78	62236-443 [SYN] Seq=0 Win=65535 MSS=1460 WS=32 TSval=324990857
61	7.934662000	54.172.250.123	192.168.1.34	TCP	74	443-62236 [SYN, ACK] Seq=0 Ack=1 Win=17898 Len=0 MSS=1452 SAC
62	7.934741000	192.168.1.34	54.172.250.123	TCP	66	62236-443 [ACK] Seq=1 Ack=1 Win=132480 Len=0 TSval=324990857
63	8.097623000	192.168.1.34	54.172.250.123	TLSv1.2	309	Client Hello
64	8.144284000	192.168.1.34	255.255.255.255	DB-LSP-DI	256	Dropbox LAN sync Discovery Protocol
65	8.144501000	192.168.1.34	192.168.1.255	DB-LSP-DI	256	Dropbox LAN sync Discovery Protocol
66	8.207576000	54.172.250.123	192.168.1.34	TCP	66	443-62236 [ACK] Seq=1 Ack=244 Win=19072 Len=0 TSval=137820029
67	8.215084000	54.172.250.123	192.168.1.34	TLSv1.2	1506	Server Hello
68	8.216184000	54.172.250.123	192.168.1.34	TLSv1.2	1506	Certificate
69	8.216188000	54.172.250.123	192.168.1.34	TLSv1.2	214	Server Key Exchange
70	8.216269000	192.168.1.34	54.172.250.123	TCP	66	62236-443 [ACK] Seq=244 Ack=2881 Win=129600 Len=0 TSval=32499
71	8.216269000	192.168.1.34	54.172.250.123	TCP	66	62236-443 [ACK] Seq=244 Ack=3029 Win=129440 Len=0 TSval=32499
72	8.527607000	192.168.1.34	54.172.250.123	TLSv1.2	141	Client Key Exchange
73	8.534538000	192.168.1.34	54.172.250.123	TLSv1.2	72	Change Cipher Spec
74	8.535576000	192.168.1.34	54.172.250.123	TLSv1.2	111	Hello Request, Hello Request
75	8.644440000	54.172.250.123	192.168.1.34	TCP	66	443-62236 [ACK] Seq=3029 Ack=325 Win=19072 Len=0 TSval=137820
76	8.645537000	54.172.250.123	192.168.1.34	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
77	8.645693000	192.168.1.34	54.172.250.123	TCP	66	62236-443 [ACK] Seq=370 Ack=3080 Win=131008 Len=0 TSval=32499
78	8.665209000	192.168.1.34	54.172.250.123	TLSv1.2	343	Application Data
79	8.670147000	192.168.1.34	54.172.250.123	TCP	1506	[TCP segment of a reassembled PDU]
80	8.670148000	192.168.1.34	54.172.250.123	TLSv1.2	516	Application Data
81	8.782893000	54.172.250.123	192.168.1.34	TCP	66	443-62236 [ACK] Seq=3080 Ack=2087 Win=38016 Len=0 TSval=13782
82	8.821273000	54.172.250.123	192.168.1.34	TCP	66	443-62236 [ACK] Seq=3080 Ack=2537 Win=40832 Len=0 TSval=13782

Figura 6.7 Captura de tráfico cliente-servidor

Se puede observar el establecimiento de la conexión TCP en la línea numero 60, donde aparece el SYN, SYN +ACK y ACK propios de este protocolo. Así queda establecida la conexión entre cliente y servidor, y ya pueden comenzar a mandar datos. En la línea 63 aparece un tipo de protocolo llamado TLSv1.2 (Transport Layer Security),[10] se trata de un protocolo criptográfico que proporciona seguridad a la conexión y se basa en especificaciones previas de SSL. El servidor es el que tiene una autentificación y no el cliente.

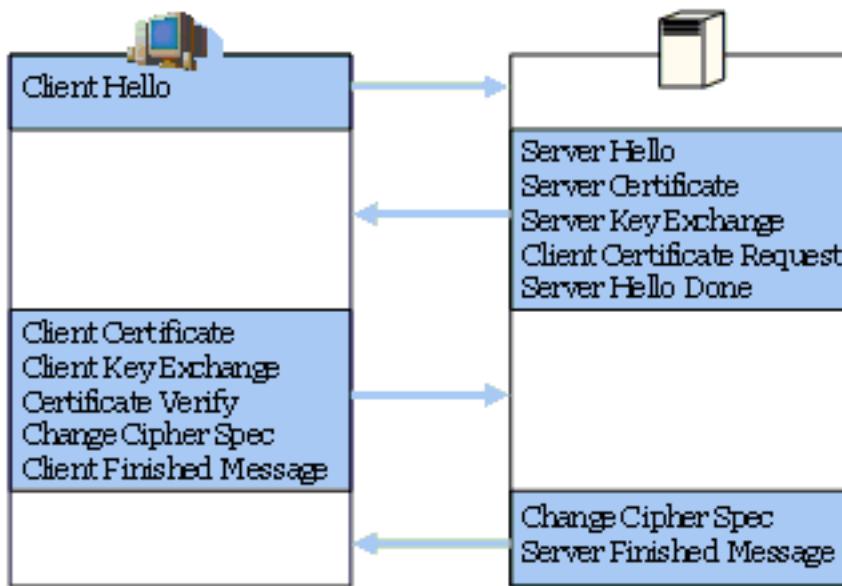


Figura 6.8 Esquema del protocolo TLS¹

Como puede verse el cliente envía un mensaje Client Hello al servidor donde se especifican los algoritmos, métodos de compresión y versión más alta de SSL que puede darse durante la comunicación.[11] A continuación, se recibe un Server Hello por parte del servidor, donde han sido elegidos los parámetros de conexión en función de lo que ofrece el cliente. En la siguiente figura puede verse la forma de cifrar los datos que se establecen.

```

> Random
Session ID Length: 32
Session ID: aa8decae877063124763f53fff208e7b3e802c9157a9cc76...
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Compression Method: null (0)
Extensions Length: 17
  
```

Figura 6.9

RSA es el método de encriptación más conocido en el que se utilizan la clave pública y la clave privada para cifrar y descifrar los datos. Es aquí donde el servidor envía su certificado de autentificación (línea 68 de la Figura 6.7), además de enviar el Server Key Exchange. A continuación el cliente es el que envía el Client Key Exchange, intercambiándose ya las claves entre cliente y servidor y estableciendo así un canal seguro.

¹ [https://technet.microsoft.com/en-us/library/cc785811\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc785811(v=ws.10).aspx)

```

Length: 329
▽ Handshake Protocol: Server Key Exchange
  Handshake Type: Server Key Exchange (12)
  Length: 329
  ▽ EC Diffie-Hellman Server Params
    Curve Type: named_curve (0x03)
    Named Curve: secp256r1 (0x0017)
    Pubkey Length: 65
    Pubkey: 04aeb286c2fd9461f7aaea0f17d67220b413f1c24d184374...
  ▷ Signature Hash Algorithm: 0x0601
    Signature Length: 256
    Signature: 53f51c9d4e492edd8aec90a8f8bda7e12134ff37a9076ab...
  Secure Sockets Layer

```

Figura 6.10 Server Key Exchange

Se muestra el contenido del paquete en el protocolo SSL del Server Key Exchange y del Client Key Exchange.

```

Secure Sockets Layer
▽ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
  Content Type: Handshake (22)
  Version: TLS 1.2 (0x0303)
  Length: 70
  ▽ Handshake Protocol: Client Key Exchange
    Handshake Type: Client Key Exchange (16)
    Length: 66
    ▽ EC Diffie-Hellman Client Params
      Pubkey Length: 65
      Pubkey: 046a69efd99cbacae0f9a34882b98d393832145b323f3620...

```

Figura 6.11 Client Key Exchange

En ambos aparece la clave pública que se utiliza para encriptar y desencriptar el mensaje por medio de la clave privada de cada uno (servidor y cliente). En la Figura 6.12 se puede ver el certificado que tiene el servidor, perteneciente a rhcloud.com, mostrado en el paquete número 68 de la traza de Wireshark.

De esta forma queda comprobado que la conexión es segura entre cliente y servidor y, gracias a la clave que creamos en el programa JAVA, solo el servidor aceptará la conexión con Nosotros (nuestro cliente).

```
› Certificates (2571 bytes)
  Certificate Length: 1360
  ▷ Certificate (id-at-commonName=*.rhcloud.com,id-at-organizationName=Red Hat Inc.,id-at
    Certificate Length: 1205
  ▷ Certificate (id-at-commonName=DigiCert SHA2 High Assurance Server CA,id-at-organizati
```

Figura 6.12

7. CONCLUSIONES

En este trabajo se ha realizado una de las dos partes de las que consta este proyecto conjunto. Esta se trataba de recoger los datos que eran analizados por el ColorGramme y conseguir subirlos a la nube de forma segura con una clave de autentificación del cliente.

Ha quedado comprobado que la comunicación era totalmente cifrada. Si los datos llegaran a ser interceptados por un intruso, no podrían ser descifrados.

Además de ello, se pretendía realizar una zona de administración de los datos almacenados en la nube. En un futuro, si se realizara la otra parte del proyecto (la de la obtención y detección de los meteoritos con una antena receptora), se podría considerar la idea de ampliar las funciones de administración, que no estaban muy definidas.

La parte pública de la web deja visibles algunos de los datos en forma de gráficas a cualquier usuario que desee curiosear sobre ello. Estas funciones del Frontend se podrían ampliar también en función del número de personas que esté interesada o que se vea que ha entrado en la web.

8. BIBLIOGRAFÍA

- [1] <http://www.nationalgeographic.es/science/space/meteoroides-meteoros-y-meteoritos>
- [2] <http://radio.meteor.free.fr/fr/en/index.html>
- [3] <https://es.wikipedia.org/wiki/Perseidas>
- [4] <http://www.astrosurf.com/blazar/meteoros/Radio%20Observacion%20de%20Meteoros.html>
- [5] https://es.wikipedia.org/wiki/Dispositivo_de_carga_acoplada
- [6] http://www.aam.org.es/index.php?option=com_content&view=article&id=423:nuevo-sistema-de-deteccion-de-bolidos-de-calar-alto&catid=27&Itemid=677
- [7] http://www.caha.es/new-fireball-detection-station-and-related-web-page_es.html
- [8] <http://wpmallorca.com/2013/02/12/pero-que-es-github/>
- [9] <https://www.openshift.com/>
- [10] https://es.wikipedia.org/wiki/Transport_Layer_Security
- [11] <https://es.wikipedia.org/wiki/X.509>