ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Grado

# ACCESO WI-FI MEDIANTE SERVIDOR CENTRALIZADO IMPLEMENTANDO PUNTOS DE ACCESO BASADOS EN OPEN-WRT


**Autor:** Cristian Martinez Candel
**Director:** Francesc Burrull i Mestres
Septiembre/2015

# TABLA DE CONTENIDO

# TABLA DE FIGURAS

# 1. Introducción

Con el paso del tiempo, hemos podido observar como el tráfico de datos móviles se ha ido incrementando exponencialmente debido a las nuevas tecnologías emergentes (servicios Push, voz sobre IP, video bajo demanda, entre otras) . Por ello, se hace imprescindible una conexión móvil en gran parte de los dispositivos (portátiles, Smartphone, Tablet, etc). En la actualidad, la mayor parte de los usuarios se ven obligados a utilizar las redes inalámbricas (Wi-Fi) para poder hacer uso de los servicios anteriormente señalados. Esta actividad viene motivada por la insuficiencia de datos o el precio de las tarifas móviles contratadas. Dicha necesidad despertó nuestro interés en facilitar el acceso a Internet, haciendo uso de las redes inalámbricas de los hogares. Es así como surge el proyecto en cuestión.

## 1.1 Motivación

Como hemos señalado con anterioridad, es imprescindible el uso de las redes inalámbricas Wi-Fi como apoyo para la conexión de los dispositivos a los servicios con mayor demanda de tráfico. Según el Instituto Nacional de Estadística (INE) en el año 2014, el 74% de los hogares españoles tenían acceso a Internet (la cifra aumenta hasta el 81% en los hogares de la Unión Europea). Con esto, podríamos considerar que aproximadamente el 74% de los hogares disponen de una red inalámbrica Wi-Fi.
Si tuviéramos acceso a todas estas redes, obtendríamos acceso en gran parte del territorio. Por ello, surgió la idea de hacer accesible la red inalámbrica de todos los hogares mediante la compartición de una parte del ancho de banda de ésta. Dicha compartición sería solo entre aquellos usuarios que acepten compartir su red.

## 1.2 Objetivos

El objetivo principal del presente trabajo fin de grado es el desarrollo de un servicio que gestione el acceso de los usuarios a Internet en los distintos puntos de acceso. Para alcanzar dicho objetivo, será necesario alcanzar los siguientes objetivos específicos:

- Buscar, Analizar y evaluar la información necesario para la implementación del servicio.
- Configurar el hardware necesario para el desarrollo del servicio (servidor y puntos de acceso).
- Testear el funcionamiento de la aplicación.

Por otro lado, los propósitos del servicio en cuestión serán:

- Compartir la red inalámbrica de casa con otros usuarios
- Reducir el consumo de datos móviles
- Hacer posible el uso de aplicaciones con gran demanda de tráfico en los dispositivos móviles
- Conectar a Internet dispositivos que no tengan la opción de acceder a la red móvil
- Propiciar conexión a Internet en caso de caída de la red móvil

## 1.3 Recursos utilizados

Para llevar a cabo el proyecto, hemos utilizado distintos recursos entre los que podemos distinguir:

### 1.3.1 Recursos hardware

- **Raspberry Pi:** Para implementar el servidor central, encargado de la gestión de los usuarios, hemos utilizado el dispositivo Raspberry Pi Model B.
- **TP-Link MR3020:** Hemos utilizado los puntos de acceso TP-Link MR3020 para dar acceso a los usuarios mediante conexión Wi-Fi.

### 1.3.2 Recursos software

- **Raspbian:** Distribución Linux basada en Debian optimizada para Raspberry Pi.

- **OpenWrt:** Distribución Linux basada en firmware usada en routers inalámbricos que añade funciones adicionales al firmware original. Dicho firmware ha sido instalado en nuestro punto de acceso TP-Link MR3020.

## 1.4 Estructura de la memoria

La memoria se distribuye de la siguiente forma:

- **Introducción del proyecto:** Es el apartado actual, en el que se presenta el proyecto especificando los motivos, objetivos y recursos utilizados.
- **Marco teórico:** Capítulo que recoge toda la información necesaria para la implementación del servicio. En él se estudiará el protocolo RADIUS, el entorno Linux y el campo de la criptografía.
- **Descripción de Hardware y Software:** En este apartado se detallará tanto el Hardware y Software utilizado en el servidor como el utilizado en el cliente para llevar a cabo el proyecto.
- **Desarrollo del servicio WeFree:** Este capítulo abarca todo el proceso de desarrollo de nuestro servicio, en el que podremos diferenciar los dos dispositivos implementados: Servidor y cliente.
- **Conexión con el servicio WeFree:** Una vez finalizado el desarrollo del servicio, comprobaremos la conexión desde varios sistemas.
- **Conclusiones y trabajos futuros:** Apartado en el que se detallan tanto las conclusiones obtenidas tras finalizar el proceso de desarrollo como los posibles trabajos de futuro.
- **Bibliografía empleada.**
- **Anexos de scripts utilizados.**

# 2 Marco teórico

La finalidad de este capítulo es analizar la teoría que impulsa el desarrollo de nuestro proyecto. Para ello, estudiaremos los siguientes campos:

- **Sistema Operativo GNU/Linux**
- **Criptografía**
- **Autenticación**
- **Firma Digital**
- **Protocolo RADIUS**

## 2.1 Sistema Operativo GNU/Linux

### 2.1.1 Introducción

Linux tiene su origen en Unix. Éste apareció en los años sesenta, desarrollado por los investigadores Dennis Ritchie y Ken Thompson, de los Laboratorios Telefónicos Bell.
Andrew Tanenbaum desarrolló un sistema operativo parecido a Unix (llamado Minix) para enseñar a sus alumnos el diseño de un sistema operativo. Debido al enfoque docente de Minix, Tanenbaum nunca permitió que éste fuera modificado, ya que podrían introducirse complicaciones en el sistema para sus alumnos.
Un estudiante finlandés llamado Linus Torvalds, constatando que no era posible extender Minix, decidió escribir su propio sistema operativo compatible con Unix.
En aquellos momentos el proyecto GNU, que Richard Stallman había iniciado hacía ya casi diez años, comprendía un sistema básico casi completo. La excepción más importante era el kernel o núcleo, que controla el hardware.
Torvalds decidió aprovechar el sistema GNU y completarlo con su propio núcleo, que bautizó como Linux. El sistema conjunto (herramientas GNU y núcleo Linux) forma lo que llamamos GNU/Linux.

### 2.1.2 Tareas en Linux

En **GNU/Linux** las tareas pueden configurarse para ejecutarse de forma automática en un período de tiempo concreto y en las fechas indicadas. Un administrador del sistema

puede utilizar las tareas automáticas para realizar copias de seguridad periódicas o controlar el sistema y ejecutar scripts personalizados, entre otras tareas.

**Cron** es un demonio que sirve para ejecutar tareas programadas según una combinación de la hora, día del mes, mes, día de la semana y semana. Cron asume que el sistema está activo de forma continua. Si el sistema no está activo cuando está programada una tarea, Cron no se ejecuta.

El fichero de configuración principal de cron, **/etc/crontab**, tiene la siguiente forma:

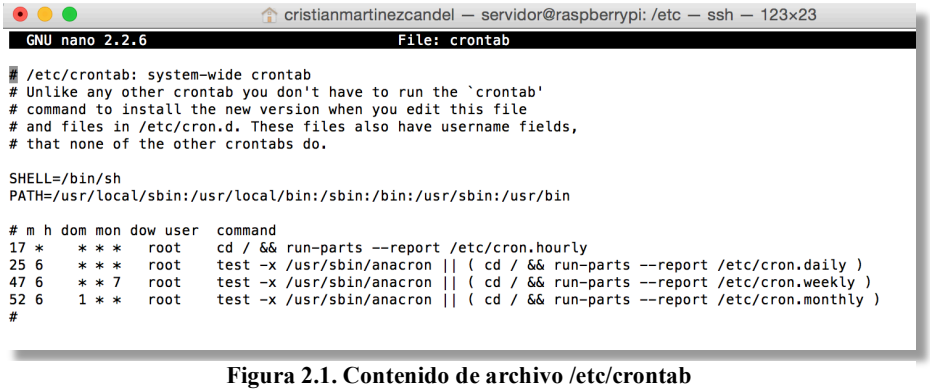



**Figura 2.1. Contenido de archivo /etc/crontab**

La variable **SHELL** indica al sistema el entorno de shell que deberá utilizarse y la variable **PATH** define las rutas usadas para ejecutar los comandos y el orden de búsqueda.

Cada línea del fichero **/etc/crontab** tiene el formato siguiente:

**[minuto] [hora] [día] [mes] [díadelasemana] [comando]**

- **minuto** - número entero entre 0 y 59.
- **hora** - número entero entre 0 y 23.
- **día** - número entero entre 1 y 31 (debe ser un día válido si se especifica un mes).
- **mes** - número entero entre 1 y 12 (o nombre corto del mes, por ejemplo, ene, feb, etc.).
- **díadelasemana** - número entero entre 0 y 7, donde 0 o 7 corresponde a Domingo, o bien el nombre corto del día de la semana, por ejemplo, Sun (Sunday, domingo), Mon, etc.).
- **comando** - comando que debe ejecutarse.

En cualquiera de los valores antes indicados, se puede utilizar un asterisco (*) para especificar todos los valores válidos.
Un guión (-) entre los números enteros indica un intervalo de números enteros.
Para especificar una lista se utilizan valores separados por comas.
La barra (/) puede utilizarse para especificar valores de pasos.
Las líneas que empiezan por almohadilla (#) son comentarios y no se procesan.

Por otro lado, existen palabras reservadas para la programación del cron. Estas son:

**@reboot:** Ejecutar al reinicio.
**@yearly:** Ejecutar una vez al año.
**@monthly:** Ejecutar una vez al mes.
**@weekly:** Ejecutar una vez a la semana.
**@daily:** Ejecutar una vez al día.
**@hourly:** Ejecutar cada hora.

### 2.1.3 Distribuciones GNU/Linux

Una distribución es una recopilación de programas y ficheros, organizados y preparados para su instalación. Estas distribuciones se pueden obtener a través de Internet de forma gratuita.
Existen números distribuciones GNU/Linux, entre la que destacaremos Debian y OpenWrt.
Debian cuenta con más de 43.000 paquetes y más de 1.000 desarrolladores detrás de este sistema. Por otra parte, existen distribuciones derivadas de esta, como Raspbian (distribución optimizada para placas Raspberry Pi) la cual utilizaremos en nuestro trabajo.
En el caso de OpenWrt, estamos ante una distribución GNU/Linux basada en firmware que optimizada para dispositivos router domésticos. En el siguiente capítulo, analizaremos detenidamente ambas distribuciones.

## 2.2 Criptografía

En este apartado, repasaremos las bases de la criptografía, profundizando en los algoritmos empleados en nuestro trabajo.

### 2.2.1 ¿Qué es la criptografía?

La **criptografía** se define como el arte de escribir en clave secreta o de un modo enigmático. Es un conjunto de técnicas que tratan sobre la protección de la información. La finalidad principal del uso de estas técnicas es conseguir que un mensaje sea confidencial.
En en lado opuesto encontramos el **criptoanálisis**, el cual se ocupa de conseguir la información de un mensaje confidencial sin tener permiso para ello.

Podemos clasificar los sistemas criptográficos en función de:

- Tipo de operación: Sustitución, trasposición.
- Número de claves: Simétrico, asimétrico.
- Modo de procesar el texto plano: Bloque, flujo.

En nuestro caso, haremos hincapié en los sistemas atendiendo al número de claves.

### 2.2.2 Algoritmos simétricos

Algoritmo en el que existe una única clave compartida por emisor y receptor. Dicha clave se utiliza para cifrar y descifrar el texto plano y el texto cifrado, respectivamente. Para ello, emisor y receptor deben ponerse de acuerdo en la clave que será utilizada. Cuanto mayor sea la clave, mayor dificultad para descifrar el mensaje cifrado. En la siguiente imagen, podemos observar el funcionamiento del cifrado simétrico.

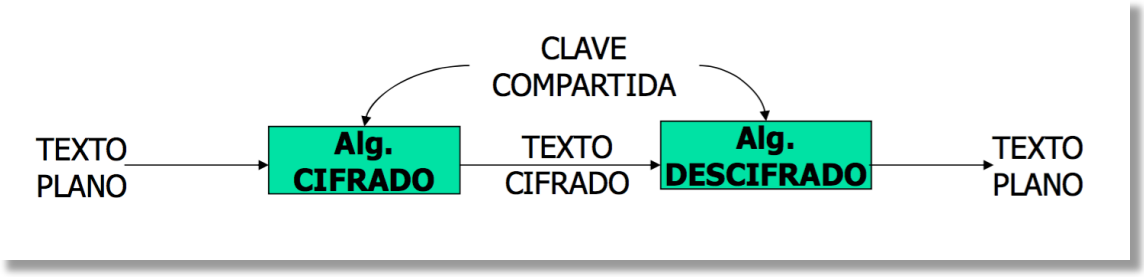



**Figura 2.2. Esquema de algoritmo simétrico**

Funcionamiento:

El texto plano es cifrado con la clave compartida. El texto cifrado se envía por el canal, de modo que si alguien obtiene dicho texto no podrá tener acceso al archivo original al menos que tenga la clave de cifrado. Una vez que el texto cifrado llega al receptor, éste lo descifra utilizando la clave previamente compartida.
Los algoritmos simétricos más importantes son:

### DES

Dicho algoritmo codifica bloques de datos de 64 bits. El tamaño de la clave es de 56 bits donde los 8 bits restantes son de paridad. Debido a la escasa longitud de la clave, este algoritmo ha quedado obsoleto.

### Triple DES

Este algoritmo surgió como remplazo del algoritmo DES. En este caso, el tamaño del bloque de datos sigue siendo de 64 bits mientras que la clave pasa a ser de 128 bits. Para el cifrado(Y) de un bloque de datos (X), se aplica:

**Y=EK1 [DK2 [EK1 (X)]]**

Mientras que para descifrar el texto plano (X):

**X=DK1 [EK2 [DK1 (Y)]]**

**AES**

Este algoritmo tiene un ancho de bloque de tamaño fijo de 128 bits, utilizando claves de 128,192 o 256 bits para el cifrado/descifrado. Se aplican operaciones a nivel de bytes. Este algoritmo es más complejo y por tanto más seguro que los citados anteriormente.

### 2.2.3 Algoritmos asimétricos

Estos algoritmos están basados en funciones matemáticas. Utilizan dos claves (clave pública y clave privada). Dichas claves poseen una longitud considerable. Estos algoritmos son más lentos que los algoritmos simétricos.

Mediante estos algoritmos podemos conseguir dos propósitos:

**Confidencialidad:**

Se cifra el texto plano con la clave pública del receptor en el emisor. Cuando el receptor reciba el texto cifrado, descifrará con su clave privada. De este modo, únicamente el receptor podrá descifrar el texto (único poseedor de su clave privada). Este el caso del algoritmo **RSA**.

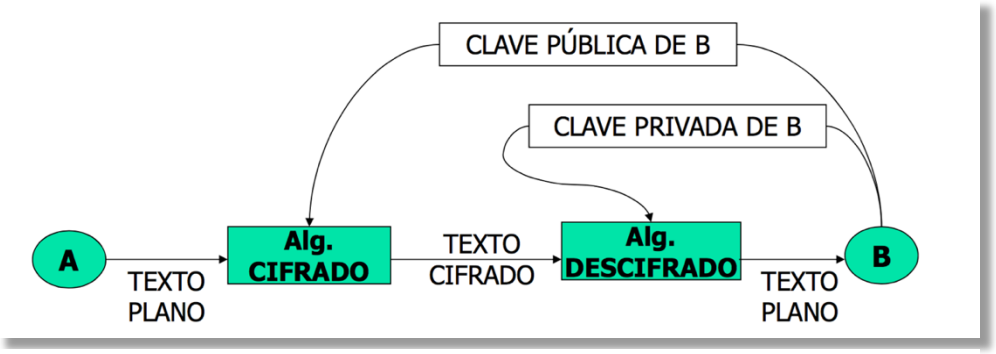



**Figura 2.3. Esquema de algoritmo asimétrico (Confidencialidad)**

**Autenticación:**

Se cifra el texto plano en el emisor con la clave privada del emisor. Para obtener el texto plano, el receptor podrá descifrar el texto cifrado mediante el uso de la clave púbica del

emisor. Con esto conseguimos **autenticación** e **integridad de datos** pero no confidencialidad.

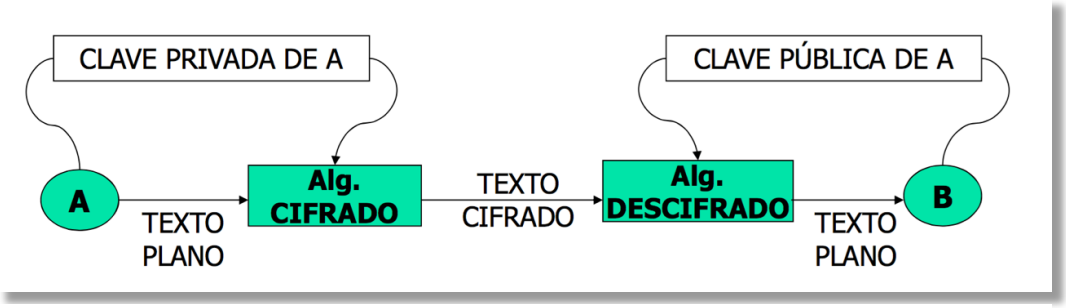



**Figura 2.4. Esquema de algoritmo asimétrico (Autenticación)**

Si queremos conseguir **autenticación** y **confidencialidad**, el emisor debe cifrar el texto plano con su clave privada y volver a cifrar el texto resultante con la clave pública del receptor. En el lado del receptor, se descifrará el texto obtenido del canal con su clave privada y volver a descifrar el mensaje obtenido mediante la clave pública del emisor.

Este tipo de algoritmos se utiliza para firma digital, intercambio de claves y cifrado/descifrado de mensajes.

Los algoritmos asimétricos más importantes son:

- **RSA**
- **Diffie-Hellman**
- **Curvas elípticas**

### RSA

Como hemos mencionado anteriormente, se dispone de dos claves:

Clave pública:

$$\mathbf{KP = \{e,n\}}$$

Clave privada:

$$\mathbf{Kp = \{d,n\}}$$

Ambas claves se usan para cifrar/descifrar y/o autenticar.

Cifrado:

$$\mathbf{Y=X^e \bmod n}$$

Descifrado:

$$\mathbf{X = Y^d \bmod n}$$

Se debe cumplir:

- Posibilidad de encontrar valores de e,d y n tales que se cumpla la relación anterior.
- Facilidad del cálculo de $X^e$ e $Y^d$ para todos los valores de X <n.
- Imposible calcular d aunque e y n sean conocidos.

Además, se debe satisfacer la siguiente función:

$$\mathbf{X=Y^d \bmod n=X^{ed} \bmod n}$$

El algoritmo de RSA consta de 3 partes:

1. **Generación de claves**

1.1. Cada usuario elige dos números primos distintos **p** y **q**.

1.2. Se calcula **n = p*q**.

1.3. Se calcula **φ (n) = (p-1)(q-1)** donde φ es la función φ de Euler.

1.4. Se escoge un entero positivo e menor que φ (n), que sea **coprimo** con φ (n).

1.5. Se determina un d satisfaciendo la congruencia **ed = 1 mod(φ(n))**.

2. **Cifrado**

   Para cifrar, utilizamos la ecuación mencionada anteriormente utilizando las claves previamente calculadas:

$$\mathbf{Y=X^e \bmod n}$$

3. **Descifrado**

   En este caso, debemos utilizar la ecuación descrita con anterioridad:

$$\mathbf{X = Y^d \bmod n}$$

La efectividad de este algoritmo radica en la factorización de números enteros. Para que el algoritmo RSA sea seguro es recomendable que n sea como mínimo de 2048 bits de longitud.

## Diffie-Hellman

Su efectividad radica en la dificultad de calcular logaritmos discretos. Este algoritmo se utiliza principalmente para el intercambio de claves. Para ello, existen dos valores públicos:

- Un número primo **q**
- Un número entero **α** que es raíz primitiva de q

Para el intercambio de claves entre los usuarios A y B se procede de la siguiente forma:

1. A selecciona un número aleatorio $x_A$ ($x_A$<q) y calcula:

$$\mathbf{y_A = \alpha^{x_A} \bmod q}$$

2. B selecciona un número aleatorio $x_B$ ($x_B$<q) y calcula:

$$\mathbf{y_B = \alpha^{x_B} \bmod q}$$

3. Cada usuario mantiene en secreto x ($x_A$ y $x_B$) y publica y ($y_A$ e $y_B$).

4. Usuario A calcula su clave secreta:

$$K_A=( y_B)^{xA} \bmod q$$

5. Usuario B calcula su clave secreta:

$$K_B=( y_A)^{xB} \bmod q$$

$K_A$ y $K_B$ comparten el mismo valor. De este modo, se ha conseguido intercambiar una clave secreta de modo seguro.

### Curvas elípticas

Dado que RSA emplea cada vez claves más largas, el tiempo de procesado también se ve incrementado. Por ello, surge el algoritmo de **Curvas Elípticas**.
Este algoritmo es capaz de igualar la seguridad de RSA utilizando claves mucho menores.

## 2.3 Autenticación

Existen numerosos sistemas de autenticación (Kerberos, EAP, PEAP,etc). En nuestro caso nos centraremos en el sistema utilizado en nuestro servidor Radius, **EAP**.

### 2.3.1 EAP

**EAP PPP** (Extensible Authentication Protocol) es un protocolo general de autenticación en PPP (Point to Point Protocol) que soporta múltiples mecanismos de autenticación.

Funcionamiento:

- Tras la fase de establecimiento del enlace, el autenticador envía una o más peticiones (Request) para autenticar al otro extremo.

- El otro extremo responde a cada petición (Response).
- El autenticador finaliza la fase de autenticación con un paquete de éxito (Success) o de fallo (Failure).

La trama de dicho protocolo tiene la siguiente forma:

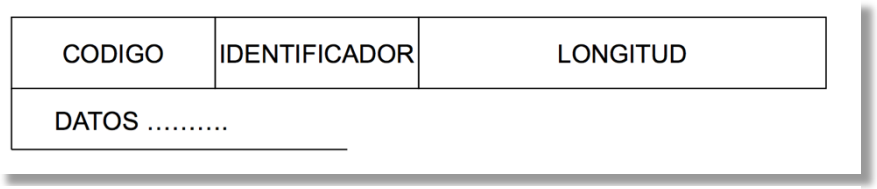



**Figura 2.5. Formato de trama EAP**

- **CÓDIGO** (1 byte): Identifica el tipo de paquete.
  1: Request
  2: Response
  3: Success
  4: Failure
- **IDENTIFICADOR** (1 byte): Empareja respuestas con peticiones.
- **LONGITUD** (2 byte): Longitud del paquete EAP incluyendo todos los campos.
- **DATOS** (0 o más bytes): El formato de este campo viene determinado por el código.

Paquete **Request**:

- Paquete Request lo envía el autenticador al otro extremo
- Cada Request tiene campo TIPO (1 byte) indicando qué se solicita
- Contenido de campo datos varíable

Paquete **Response**:

- Sólo se envía paquete Response en respuesta a un paquete Request
- Cada Response tiene campo TIPO que normalmente coincide con el del paquete Request El valor del campo IDENTIFICADOR debe ser el mismo que el del paquete Request

Tipos de **Resquest/Response**:

- 1 : Identity, solicita identidad del otro extremo
- 2 : Notification, mensaje a mostrar en el otro extremo
- 3 : NAK, (sólo en Response) tipo de autenticación deseada es inaceptable
- 4 : MD5 Challenge
- 5 : One time password
- 6 : Generic Token Card
- 13 : Transport Layer Security

En la siguiente imagen podemos apreciar el formato de la trama **Request/Response**:

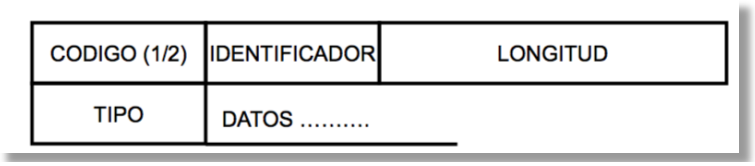



**Figura 2.6. Formato de trama Request/Response**

Paquete **Success/Failure**:

- El paquete Success reconoce una autenticación satisfactoria
- Si el autenticador no puede autenticar al otro extremo envía un paquete Failure

En la siguiente imagen podemos apreciar el formato de la trama **Success/Failure**:

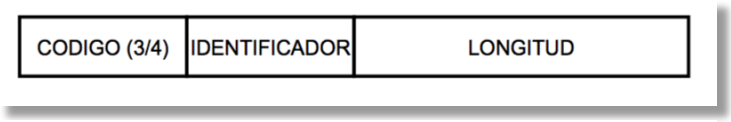



**Figura 2.7. Formato de trama Success/Failure**

## 802.1x

Es un estándar de autenticación donde se definen:

- Suplicante
- Autenticador
- Servidor de Autenticación

En la siguiente imagen se muestra el esquema del estándar 802.1x:

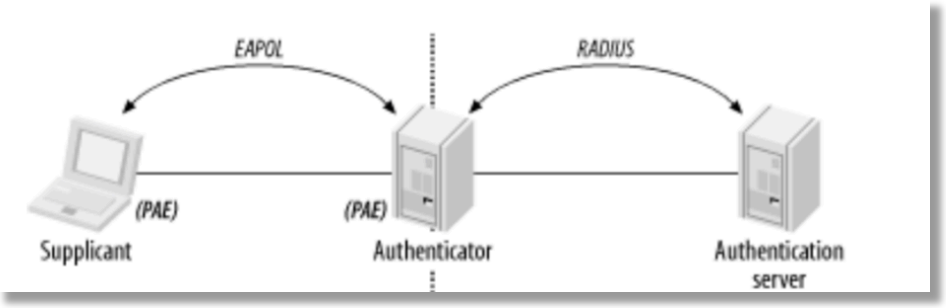



**Figura 2.8. Esquema del estándar 802.1X**

Como podemos observar, existe un intercambio de autenticación entre suplicante y servidor de autenticación pasando por el autenticador que hace la función de puente de comunicación.

El esquema de funcionamiento de 802.1x es el siguiente:

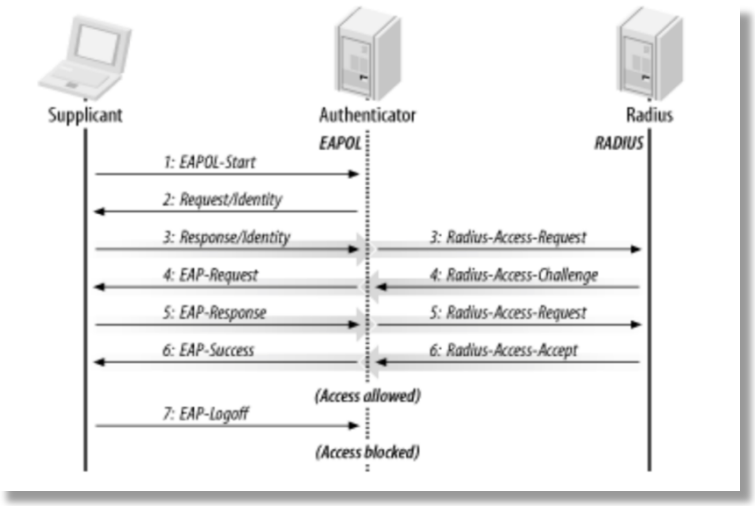



**Figura 2.9. Esquema de funcionamiento 802.1X**

En dicho esquema podemos observar el intercambio de paquetes para llevar a cabo la autenticación del suplicante.

## 2.4 Firma Digital

La firma digital se basa en criptografía de clave pública o de clave asimétrica (como ya hemos visto, tiene una clave pública y otra privada). La firma digital debe cumplir las siguientes propiedades:

- Poder verificar autor, fecha y hora de la firma.

- Poder autenticar el contenido del mensaje a la hora en la que se firmó.
- Debe estar verificada por un tercero para evitar disputas.

Por otro lado, una firma digital:

- Tendrá una firma que corresponde con un patrón de bits dependientes del mensaje firmado.
- Utilizará información única del emisor para evitar denegación y falsificación.
- Será sencilla de crear.
- Será sencilla de reconocer y verificar.
- Falsificarla debe ser computacionalmente no factible.

En la siguiente imagen podemos observar el esquema de una firma digital:

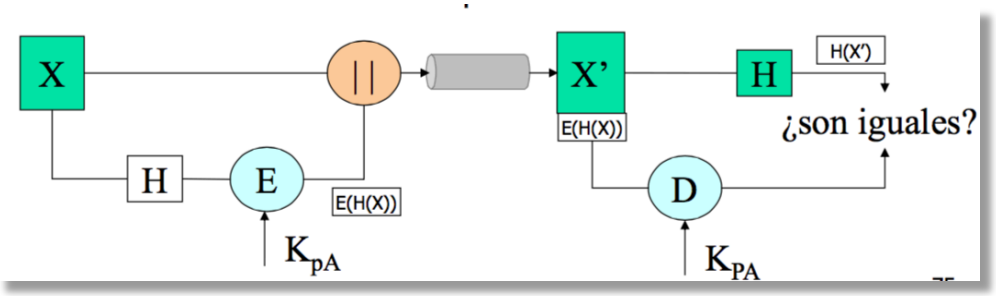



**Figura 2.10. Creación y comprobación de firma digital**

Existen 2 tipos de firma digital:

**Firma Digital Directa**

- Sólo intervienen los dos comunicantes.
- El destino conoce la clave pública del emisor.
- Firmamos el mensaje completo ó hash del mensaje con la clave privada del emisor.
- Problema: Seguridad de la clave secreta.

**Firma Digital Arbitrada**

- Una tercera entidad actúa como árbitro.

- Funcionamiento general: Todos los mensajes pasan por el árbitro que será el encargado de comprobar la validez de origen y contenido.
- Confiabilidad total en el árbitro.

### 2.4.1 Certificado digital

Un Certificado Electrónico es un conjunto de datos que permiten la identificación del titular del Certificado, intercambiar información con otras personas y entidades, de manera segura, y firmar electrónicamente los datos que se envían de tal forma que se pueda comprobar su integridad y procedencia. El certificado digital se utiliza para autenticar a un usuario o web por lo que es necesaria la colaboración de una tercera entidad de confianza (**Autoridad certificadora**).

Para la creación de un certificado digital se sigue el siguiente esquema:

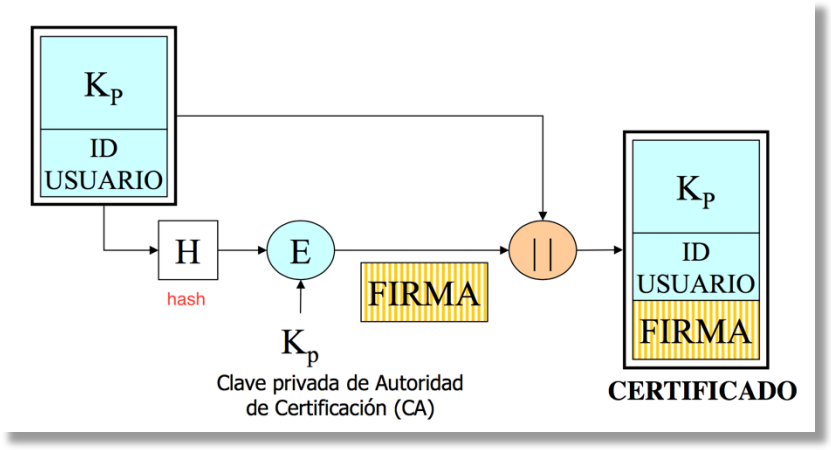



**Figura 2.11. Creación de un Certificado Digital**

Inicialmente se crea una pareja de claves, una clave privada (KpA) y una clave pública (KPA).

El usuario A solicita a la Autoridad Certificadora un certificado digital mediante su KPA y su ID de usuario.

La autoridad certificadora aplica la función HASH sobre estos datos y posteriormente el algoritmo de cifrado con su clave privada KpCA. Con esto obtenemos la FIRMA.

Para finalizar, se obtiene el CERTIFICADO del usuario A mediante la concatenación de la clave pública del usuario A, su ID de usuario y la FIRMA obtenida.

Para comprobar que el certificado es correcto, se sigue el siguiente proceso:

Por un lado, se aplica a la pareja KP/ID USUARIO al la función HASH aplicada anteriormente. Por otro lado, se aplica el algoritmo de Descifrado a la FIRMA con la clave pública de la autoridad certificadora (KPCA). Si el resultado de ambos procesos es el mismo, podemos deducir que el certificado es correcto.

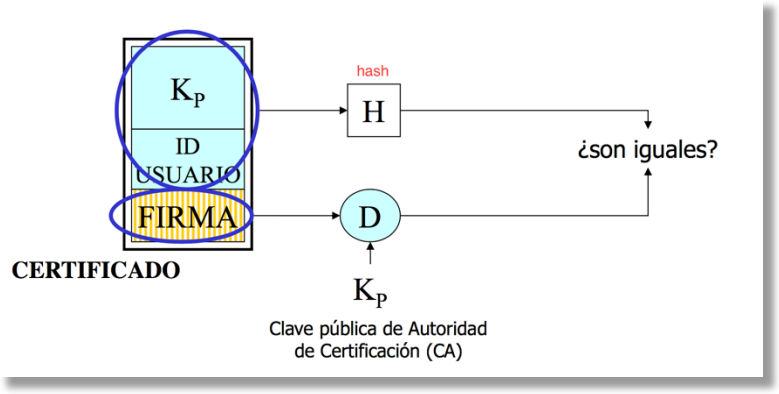



**Figura 2.12. Comprobación de Certificado Digital**

Los certificados tienen un periodo de validez que dependerá de la autoridad certificadora. Por ello, existe una lista de revocación de certificados (CRL) en la autoridad certificadora que contendrá los certificados no válidos (revocados). Los certificados se pueden revocar por distintos motivos (Un usuario cambia de autoridad certificadora, la clave privada del usuario se ve comprometida, certificado caducado, etc).

**Beneficios de los certificados digitales**

- Elimina la necesidad de recordar login y password
- Sirve como prueba de pertenecer a una organización
- Permite implementar comunicaciones cifradas
- Restringe el acceso a sitios web

## 2.5 Protocolo RADIUS

El Protocolo de servicio de usuario de acceso telefónico de autenticación remota (RADIUS) fue desarrollado por Livingston Enterprises, Inc., como un protocolo de autenticación del servidor de acceso y de contabilidad.

RADIUS es un protocolo cliente/servidor. El cliente RADIUS es típicamente un NAS y el servidor de RADIUS es generalmente un proceso de daemon (demonio) que se ejecuta en UNIX o una máquina del Windows NT. El cliente pasa la información del usuario a los servidores RADIUS designados. Los servidores de RADIUS reciben las peticiones de conexión del usuario, autentican al usuario, y después devuelven la información de la configuración necesaria para que el cliente entregue el servicio al usuario. Un servidor RADIUS puede funcionar como cliente proxy para otros servidores RADIUS u otro tipo de servidores de autenticación.

En la siguiente figura se muestra el esquema de funcionamiento RADIUS:

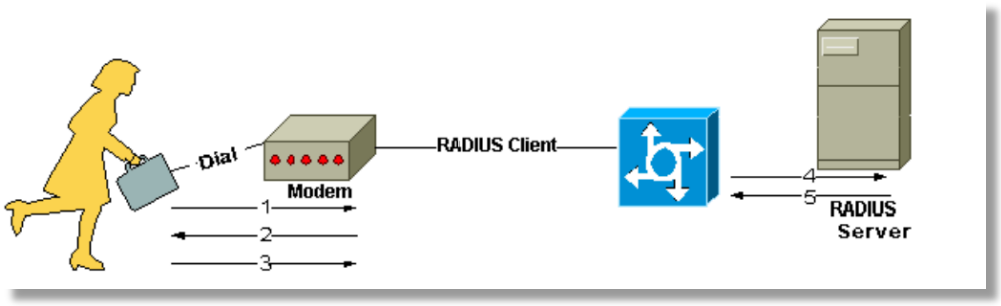



**Figura 2.13. Protocolo RADIUS**

Funcionamiento:

1. El usuario inicia la autenticación PPP (Point to Point Protocol) al NAS.
2. El NAS le pedirá que ingrese el nombre de usuario y la contraseña (en caso de Protocolo de autenticación de contraseña [PAP]) o la integración (en caso de Protocolo de confirmación de aceptación de la contraseña [CHAP]).
3. El usuario contesta la petición.
4. El cliente RADIUS envía el nombre de usuario y la contraseña encriptada al servidor de RADIUS.
5. El servidor RADIUS responde con Aceptar, Rechazar o Impugnar.
6. El cliente RADIUS actúa dependiendo de los servicios y de los parámetros de servicios agrupados con Aceptar o Rechazar.

**Autenticación y autorización**

El servidor RADIUS puede soportar varios métodos para autenticar un usuario. Cuando se proporciona el nombre de usuario y la contraseña original dados por el usuario, puede soportar el login PPP, PAP o de la GRIETA, de UNIX, y otros mecanismos de autenticación. Comúnmente, el ingreso de un usuario al sistema consiste en una petición (Solicitud de acceso) desde el NAS hacia el servidor RADIUS y de una correspondiente respuesta (Aceptación de acceso o Rechazo de acceso) desde el servidor. El paquete access-request contiene el nombre de usuario, la contraseña encriptada, la dirección IP NAS, y el puerto. El Early Deployment del RADIUS fue hecho usando el número del puerto 1645 UDP, que está en conflicto con el servicio del "datametrics". Debido a este conflicto, el RFC 2865 asignó oficialmente el número del puerto 1812 para el RADIUS. El formato de la petición proporciona asimismo información sobre el tipo de sesión que el usuario desea iniciar.

Cuando el servidor de RADIUS recibe la petición de acceso del NAS, busca una base de datos para el nombre de usuario enumerado. Si el nombre de usuario no existe en la base de datos, se carga un perfil predeterminado o el servidor RADIUS inmediatamente envía un mensaje Access-Reject (acceso denegado). Este mensaje de acceso denegado puede estar acompañado de un mensaje de texto que indique el motivo del rechazo. En RADIUS, la autenticación y la autorización están unidas. Si se encuentra el nombre de usuario y la contraseña es correcta, el servidor RADIUS devuelve una respuesta de Access-Accept (acceso aceptado) e incluye una lista de pares de atributo-valor que describe los parámetros que deben usarse en esta sesión. Los parámetros comunes incluyen el tipo de servicio (shell o entramado), el tipo de protocolo, la dirección IP para asignar el usuario (estática o dinámica), la lista de acceso a aplicar o una ruta estática para instalar en la tabla de ruteo de NAS. La información de configuración en el servidor RADIUS define qué se instalará en el NAS.

La siguiente imagen ilustra la secuencia de autenticación y autorización de RADIUS:

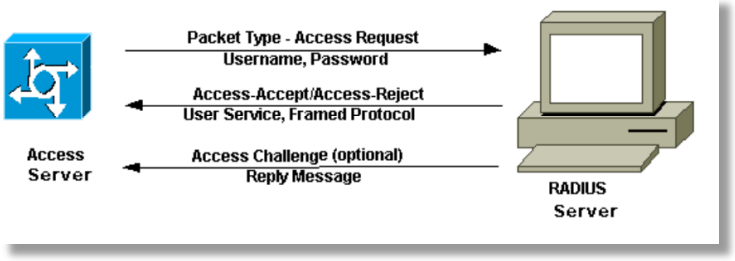



**Figura 2.14. Secuencia de autenticación y autorización**

**Contabilidad**

La funciones de contabilidad del protocolo RADIUS pueden emplearse independientemente de la autenticación o autorización de RADIUS. Las funciones de contabilidad de RADIUS permiten que los datos sean enviados al inicio y al final de las sesiones, indicando la cantidad de recursos (por ejemplo, tiempo, paquetes, bytes y otros) utilizados durante la sesión. Un Proveedor de servicios de Internet (ISP) puede usar RADIUS para el control de acceso y un software de contabilidad para satisfacer necesidades especiales de seguridad y facturación. El puerto de contabilidad para el RADIUS es el 1813. Las transacciones entre el cliente y el servidor RADIUS son autenticadas mediante el uso de un secreto compartido, que nunca se envía por la red. Además, las contraseñas del usuario se envían cifradas entre el cliente y el servidor de RADIUS para eliminar la posibilidad que alguien se hiciera con la contraseña de usuario en una red insegura.

# 3 Descripción de Hardware y Software

En este capitulo estudiaremos los recursos empleados para llevar a cabo el proyecto. Dado que nuestro servicio está compuesto por dos dispositivos distintos (servidor y cliente), podemos dividir los recursos en dos grandes grupos. Por un lado los recursos empleados en el servidor y por otro los recursos empleados en el cliente.

## 3.1 Recursos empleados en el servidor

En este apartado podemos distinguir dos tipos de recursos:

- **Recursos Hardware**
- **Recursos Software**

### 3.1.1 Recursos Hardware

Para llevar a cabo la implementación del servidor, se ha optado por el dispositivo Raspberry Pi Model B. La elección de este dispositivo viene motivada por la necesidad de disponer de un servidor central de bajo consumo, bajo coste y dimensiones reducidas, características técnicas que cumple el dispositivo. Por otro lado, cabe destacar la flexibilidad de configuración y el amplio abanico de opciones que ofrece ya que trabaja sobre Linux.

#### Raspberry Pi Model B

Como hemos comentado anteriormente, Raspberry Pi es un sistema embebido basado en Linux. Sus principales características son:

- **Tamaño reducido:** Sus medidas son 85,6 x 53,98 x 17 mm
- **Bajo consumo:** Consumo de 1W
- **Coste reducido:** El precio de venta es de 35$, a los que habría que añadir el coste de la tarjeta SD y la alimentación USB.

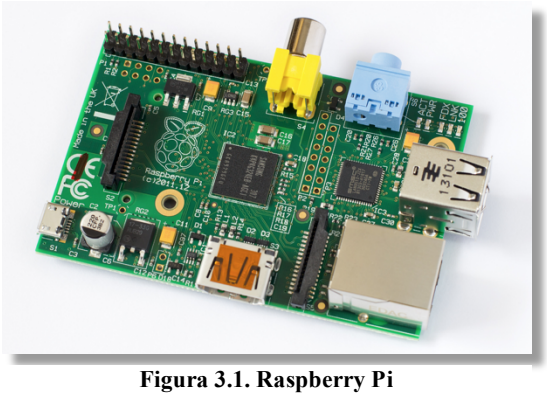


**Figura 3.1. Raspberry Pi**

En la siguiente imagen se recogen las especificaciones técnicas:

| | Raspberry Pi Model B |
|---|---|
| SoC | Broadcom BCM2835 |
| CPU | ARM 1176JZFS a 700 MHz |
| GPU | Videocore 4 |
| RAM | 256 MB |
| Video | HDMI y RCA |
| Resolución | 1080p |
| Audio | HDMI y 3.5 mm |
| USB | 2 x USB 2.0 |
| Redes | Ethernet 10/100 |
| Electricidad | micro USB |

**Figura 3.2. Especificaciones Raspberry Pi**

Como observamos en la imagen, las características técnicas se encuentran lejos de las características técnicas que podríamos encontrar en cualquier ordenador personal. Sin embargo, se obtiene un gran rendimiento.

En nuestro caso, el servidor no tendrá sobrecargas ya que las tareas que llevará a cabo serán bastante sencillas.

### 3.1.2 Recursos Software

#### Raspbian

Para implementar el servidor, se ha empleado el sistema operativo Raspbian basado en la distribución GNU/Linux Debian. Este sistema operativo está optimizado para su uso en los dispositivos Raspberry Pi. Además, dispone de más de 35.0000 paquetes en su

repositorio, lo que hace posible el abanico de opciones que mencionamos con anterioridad. Raspbian es un sistema operativo libre, lo que provoca que exista una gran comunidad de desarrollo que ayuda a mejorar el sistema.

## Aplicaciones necesarias para implementar el servidor

Como hemos mencionado, en el repositorio de Raspbian podremos encontrar infinidad de aplicaciones. Para nuestro proyecto, serán necesarias las siguientes:

### Servidor MySQL

MySQL es el sistema de gestión de base de datos más polular.
Una base de datos es una colección estructurada de datos. Para agregar, acceder y procesar los datos almacenados en una base de datos, es necesario un sistema de gestión de base de datos como MySQL Server.

MySQL es un sistema de gestión de base de datos relacional. Almacena los datos en tablas separadas en lugar de guardar todos los datos en una misma tabla. La lógica de MySQL está formada por objetos tales como bases de datos, tablas, vistas, filas y columnas, ofreciendo un entorno de programación flexible. Es posible configurar las reglas que rigen las relaciones entre los diferentes campos de datos, como uno-a-uno, uno-a-muchos, único, obligatorio u opcional, y "punteros" entre diferentes tablas. Para el acceso a las bases de datos, MySQL utiliza el lenguaje SQL.
SQL es el lenguaje estandarizado más común utilizado para el acceso a bases de datos.

El Software MySQL es Open Source lo que significa que es posible que cualquiera pueda usar y modificar el software. Es posible descargar el software MySQL desde internet y usarlo sin pagar nada.
MySQL es un servidor multi-hilo SQL que soporta el acceso de numerosos clientes a la vez.

La base de datos del servidor MySQL es muy rápida, fiable, escalable y fácil de usar. Esto permite la implementación de MySQL en cualquier parte, desde dispositivos embebidos hasta dispositivos con grandes características. Por otro lado, MySQL es un

sistema multi-hilo y multiusuario que soporta el acceso de numerosos usuarios de forma simultánea.

En nuestro caso, haremos uso de MySQL para generar una base de datos con los clientes del servicio y los usuarios que tendrán acceso a la red.

**Servidor Apache**

Apache es un servidor web HTTP potente y flexible que implementa los últimos protocolos, incluyendo HTTP / 1.1 (RFC2616).
Además es altamente configurable y extensible con módulos de terceros pudiéndose personalizar escribiendo estos mediante el módulo API de Apache. Se proporciona el código fuente completo y viene con una licencia sin restricciones.
Es posible implementarlo en Windows, Netware, OS, y la mayoría de las versiones de Unix, así como en otros sistemas operativos.
Incluye bases de datos relacionales y autenticación LDAP que permite configurar fácilmente las páginas protegidas con contraseña posibilitando el acceso a un enorme número de usuarios autorizados, sin empantanamiento del servidor.
Apache no tiene límite en el número de alias y redirecciones que pueden declararse en los archivos de configuración. Además se puede utilizar un potente motor de reescritura para resolver la mayoría de problemas de manipulación de URL.
El servidor Apache es capaz de distinguir entre las solicitudes realizadas desde diferentes direcciones IP o nombres.

En nuestro servidor web se alojará una base de datos, una página principal de nuestro servicio y el archivo de los usuarios que tendrán acceso a la red. La base de datos se actualizará mediante accesos HTTP desde los clientes, utilizando PHP para la actualización de la misma.

**Cliente DDNS**

DDNS (DNS dinámico) es un servicio utilizado para la actualización en tiempo real de la información sobre nombres de dominio en un servidor de nombres. Esto es comúnmente utilizado para asignar un nombre de dominio a una IP dinámica. Por ello, existe una relación IP dinámica – Nombre Dominio que se actualizada en tiempo real.
Esto posibilita el acceso remoto a al IP dinámica mediante el nombre de dominio (estático).
En nuestro caso, se emplea para hacer uso de un servidor dentro de una red personal que tiene asignada una IP dinámica por el proveedor de servicio de Internet (ISP).

**OpenSSL**
OpenSSL es un proyecto de Software libre basado en la librería SSleay desarrollada por Eric Young y Tim Hudson. Consiste en un conjunto de herramientas de código abierto que implementa los protocolos criptográficos TLS y SSL (Trasport Layer Security y Secure Socket Layer) para proporcionar comunicaciones seguras en una red.
OpenSSL también permite crear certificados digitales, tarea que realizaremos durante el desarrollo del cliente.
Utilizaremos OpenSSL para cifrar el archivo de los usuarios de la base de datos que será alojado en el servidor web.

## 3.2 Recursos empleados en el cliente

Al igual que en la sección anterior, podemos distinguir entre dos tipos de recursos:

- **Recursos Hardware**
- **Recursos Software/Firmware**

### 3.2.1 Recursos Hardware

Para la implementación de los puntos de acceso como clientes, hemos seleccionado el router TP-Link MR3020 teniendo en cuenta su compatibilidad con el firmware OpenWrt.

#### TP-Link MR3020

Este router cuenta con un tamaño reducido, lo que permite su portabilidad (6.7 x 7.4 x 2.2 cm). Ofrece velocidades de hasta 150Mbps en redes inalámbricas. Sin embargo, cuenta con una memoria interna reducida (4MB). Por otro lado, su puerto USB posibilita la extensión de su memoria interna.

**Figura 3.3. TP-Link WR3020**

Especificaciones técnicas:

| CARACTERÍSTICAS DE HARDWARE | |
|---|---|
| Interfaces | 1 Puerto WAN/LAN 10/100Mbps<br>1 Puerto USB 2.0 para conexión de módem 3G/4G,<br>1 Puerto mini USB para carga de energía |
| Botón | Botón de Seguridad Rápida<br>Botón Reset<br>Botón de Modo |
| Fuente de Alimentación Externa | 5VDC/1.0A |
| Dimensiones | 74 x 67 x22 mm (2.9 x 2.6 x 0.9 in.) |
| Tipo de Antena | Antena Interna |

**Figura 3.4. Características Hardware TP-Link WR3020**

| CARACTERÍSTICAS DE SOFTWARE | |
|---|---|
| Securidad | Firewall, MAC filtering, Denial of Service (DoS) |
| DHCP | Servidor, Lista de Clientes DHCP, Reserva de Direcciones |
| Reenvío de Puertos | Virtual Server, Port Triggering, DMZ, UPnP |
| Control de Acceso | Parental Control, Host List, Access Schedule, Rule Management |

**Figura 3.5. Características Software TP-Link WR3020**

| CARACTERÍSTICAS INALÁMBRICAS | |
|---|---|
| Estándares Inalámbricos | IEEE 802.11n, IEEE 802.11g, IEEE 802.11b |
| Frecuencia | 2.4-2.4835GHz |
| Potencia de Transmisión | <20dBm |
| Modo Inalámbrico | 3G Router, Travel Router (AP), WISP Client Router |
| Securidad Inalámbrica | Soporta Encriptaciones WEP de 64/128 bit y WPA-PSK/WPA2-PSK, Filtrado MAC Inalámbrico |

**Figura 3.6. Características Inalámbricas TP-Link WR3020**

### 3.2.2 Recursos Software/Firmware

Como hemos indicado anteriormente, emplearemos el dispositivo TP-Link MR3020. Dicho dispositivo trae consigo una versión firmware bastante limitada. Por ello, se empleará para el desarrollo del proyecto el Firmware OpenWrt.

#### OpenWrt

OpenWrt es una distribución Linux para dispositivos embebidos (normalmente routers). Este Firmware hace posible la instalación de numerosas aplicaciones y/o librerías, ampliando las posibilidades de los dispositivos inalámbricos. Este Firmware está optimizado para poder ser alojado en los router domésticos. Para la gestión y

configuración del dispositivo, OpenWrt dispone de configuración mediante interfaz web (LUCI) y configuración mediante línea de comandos (ash).
Además, OpenWrt tiene acceso a más de 2000 paquetes mediante su sistema de gestión de paquetes opkg. Estos paquetes posibilitan nuevas funciones que son inaccesibles con el firmware original. Con ello se consigue una libertad total por parte del usuario a la hora de personalizar su dispositivo router.

Las características principales de este Firmware son:

- **Software Libre:** OpenWrt es un proyecto de Software libre impulsado por la licencia GPL, que obliga a aquellas personas que mejoran el código a redistribuirlo para dar continuidad al proyecto.
- **Amplia comunidad:** Detrás de este proyecto hay numerosas personas que trabajan para mejorar el proyecto, formándose así una comunidad de desarrolladores.
- **Proyecto abierto:** Cualquier desarrollador puede formar parte de este proyecto, tomando como filosofía la mejora y continuidad del proyecto.
- **Sistema de gestión de paquetes:** Como ya hemos mencionado, OpenWrt dispone de un sistema de gestión de paquetes (**opkg**) que permite la instalación, eliminación y actualización de las distintas aplicaciones que pueden encontrarse en el repositorio.
- **Aplicar técnicas de calidad de servicio o filtrado de paquetes:**
  OpenWrt permite asignar técnicas de calidad de servicio, administrar colas de datos o gestionar el filtrado de paquetes mediante firewall.
- **Interfaz Web:** El proyecto **LUCI** es el encargado de gestionar la interfaz web que permite la gestión del dispositivo de una forma más sencilla y vistosa.
- **Actualizaciones regulares:** OpenWrt es un sistema que se se actualiza periódicamente gracias al trabajo de los desarrolladores del mismo.
- **Otras aplicaciones:** OpenWrt puede emplearse para el desarrollo de otras aplicaciones como redes de nodos, servidores de archivos, firewall o puertas VPN.

Para llevar a cabo el proyecto, haremos uso de distintas aplicaciones obtenidas mediante el sistema de gestión de paquetes **opkg**. Estas se verán con detalle en el siguiente apartado.

## Aplicaciones Software instaladas en OpenWrt

### Paquetes para ampliar memoria mediante puerto USB

Como hemos visto en el apartado de características técnicas de nuestro dispositivo TP-Link MR3020, la memoria interna de éste es demasiado pequeña para nuestro propósito. Por ello, aparece la necesidad de ampliar esta memoria mediante el puerto USB. Para llevar a cabo dicha ampliación de memoria, son necesarios los siguientes paquetes:

- **block-mount:** Gestiona el montaje de bloques en OpenWrt.
- **kmod-usb-storage:** Soporte del kernel para dispositivos USB.
- **kmod-usb-uhci:** Soporte del kernel para los controladores UHCI.
- **kmod-fs-ext4:** Necesario para reconocer las particiones con formato ext4.
- **mountd:** Pequeño programa para automontaje de los dispositivos USB.
- **mount-utils:** Utilidades de montaje de dispositivos.
- **libext2fs:** Posibilita la manipulación de sistema de archivos ext2.

### OpenSSL

OpenSSL es un proyecto de Software libre basado en la librería SSleay desarrollada por Eric Young y Tim Hudson. Consiste en un conjunto de herramientas de código abierto

que implementa los protocolos criptográficos TLS y SSL (Trasport Layer Security y Secure Socket Layer) para proporcionar comunicaciones seguras en una red.
OpenSSL también permite crear certificados digitales, tarea que realizaremos durante el desarrollo del cliente.
Además de para crear el certificado de nuestro servidor Radius, utilizaremos OpenSSL para descifrar el fichero de usuarios obtenido del servidor central en los distintos puntos de acceso (clientes).

**FreeRADIUS**

RADIUS es un protocolo de red de autenticación y autorización para usuario. Comúnmente utilizado por Proveedores de Servicio de Internet (ISP), el protocolo RADIUS tiene tres funciones principales :

• Autentica usuarios o dispositivos antes de permitirles el acceso a una red.
• Autoriza el uso de servicios de red específicos a usuarios o dispositivos.

FreeRADIUS es el servidor RADIUS de código abierto más popular y más utilizado en el mundo.
Abastece las necesidades de autenticación, autorización, y contabilización (AAA) de muchas compañías ISP . También es muy utilizado por la comunidad académica (eduroam utiliza software FreeRADIUS).
FreeRADIUS se inició en agosto de 1999 por Alan DeKok y Miquel van Smoorenburg.
Fue desarrollado utilizando un diseño modular para fomentar la participación de la comunidad.

La mayoría de tipos de autenticación son compatibles con FreeRADIUS. Por otro lado, también soporta servidores virtuales, lo que provoca una reducción de costes de mantenimiento.

**Modularidad**

La interfaz modular simplifica la adición o eliminación de módulos, sin afectar al rendimiento del servidor. Esta flexibilidad permite que el servidor pueda implementarse desde dispositivos empotrados hasta dispositivos más complejos.

**Escalabilidad**

Un único servidor Radius puede pasar de manejar una petición por segundo a manejar miles de peticiones por segundo simplemente mediante una reconfiguración de los ajustes por defecto. Por ello, muchas grandes organizaciones hacen uso de FreeRADIUS para abastecer sus necesidades de autenticación, autorización, y contabilización.

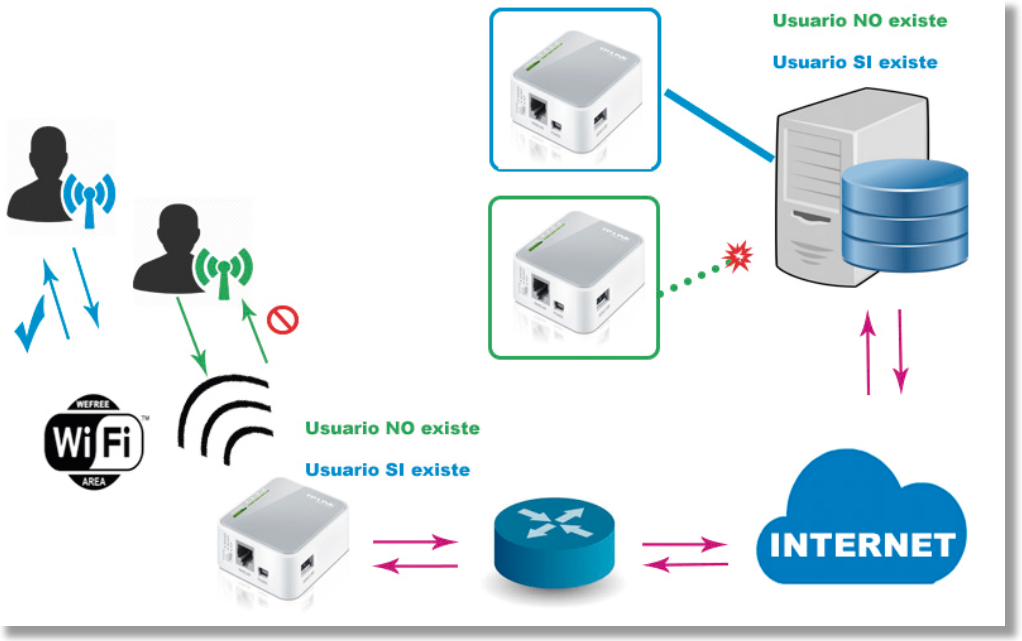
# 4 Desarrollo del servicio WeFree

En este capítulo, se llevará a cabo el desarrollo del servicio.

Como ya hemos mencionado anteriormente, la idea principal es compartir la conexión de casa, para más tarde poder disfrutar de las redes inalámbricas que disfruten de este servicio. Para ello, cada punto de acceso configurado tendrá asociado unas credenciales (usuario/contraseña) con el que podrá obtener acceso en cualquier red inalámbrica de nuestro servicio. Las credenciales tienen dos estados posibles dependiendo del estado de la conexión del punto de acceso asociado:

- **Activo:** Si el punto de acceso se encuentra conectado a la red, el usuario estará activo.
- **Inactivo:** Si el punto de acceso se encuentra desconectado, el usuario estará inactivo hasta el momento en el que el punto de acceso vuelva a estar conectado.

Para gestionar la conexión de los puntos de acceso y el estado de los usuarios, se hará uso de un servidor central. Este servidor recibirá los datos de los puntos de acceso periódicamente, actualizando la información de los mismos. Si dicha información no se encuentra actualizada, se deduce que el punto de acceso no está conectado y pasaremos a desactivar el usuario asociado. Por otro lado, si la información del punto de acceso se encuentra actualizada, el usuario estará activo. En el siguiente esquema, podemos observar el funcionamiento general.



## 4.1 Servidor web con base de datos MySQL

### 4.1.1 Instalación de Raspbian

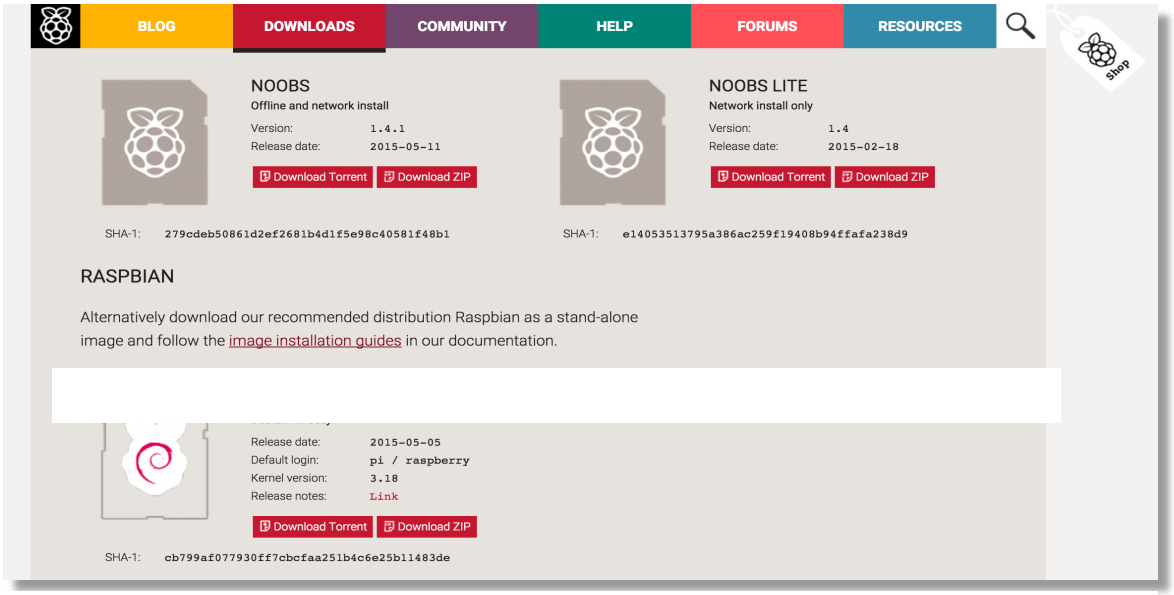



**Figura 4.1. Web oficial de Raspberry Pi**

Para crear nuestro servidor web hemos utilizado una Raspberry Pi, en la cual instalaremos la distribución Raspbian. Para ello, accedemos a la página de Raspberry Pi https://www.raspberrypi.org/downloads/ y descargamos el archivo "**RASPBIAN**".

Una vez descargado, debemos dar formato a nuestra tarjeta SD. Dicho formato será **FAT32**. EN nuestro caso lo hemos realizado con la utilidad de Discos de MAC. Más tarde, descomprimimos el fichero descargado, obteniendo una imagen de la distribución. Para instalar dicha imagen hemos seguido los siguientes pasos:

1.- Con la tarjeta SD desconectada, ejecutamos y observamos el resultado del comando:

***df -h***

2.- Insertamos nuestra SD en nuestro MAC/PC y volvemos a ejecutar el comando:

***df -h***

3.- Comparamos ambas salidas, observando la aparición de un nuevo dispositivo (nuestra tarjeta SD).Guardamos el nombre de nuestra tarjeta (disk3s1)

4.- Desmontamos la partición de la tarjeta SD ejecutando:

***sudo diskutil unmount /dev/disk3s1***

5.- Copiamos el contenido de la imagen descargada a nuestra tarjeta SD mediante el comando:

***sudo dd bs=1m if=Downloads/2015-05-05-raspbian-wheezy.img of=/dev/rdisk3***

En el anterior comando,

**if=/Ruta_Imagen/**

**of=/Ruta_Tarjeta_SD/**

Para asignar la ruta de la tarjeta SD hemos tenido en cuenta la salida del comando df -h (disk3s1) del que podemos extraer que nuestra tarjeta es disk3.

Una vez terminada la transferencia, extraemos la tarjeta SD y la insertamos en la Raspberry Pi.

### 4.1.2 Configuración inicial Raspberry Pi

Para configurar nuestra Raspberry, debemos acceder a ella mediante SSH. Para ello, necesitamos saber la IP asignada por el router mediante DHCP. En nuestro caso, dicha IP es 192.168.1.12 por tanto, el comando en cuestión sería:

***ssh pi@192.168.1.12***

Debemos tener en cuenta que los datos acceso por defecto son:

| | |
|---|---|
| Usuario: | **pi** |
| Contraseña: | **raspberry** |

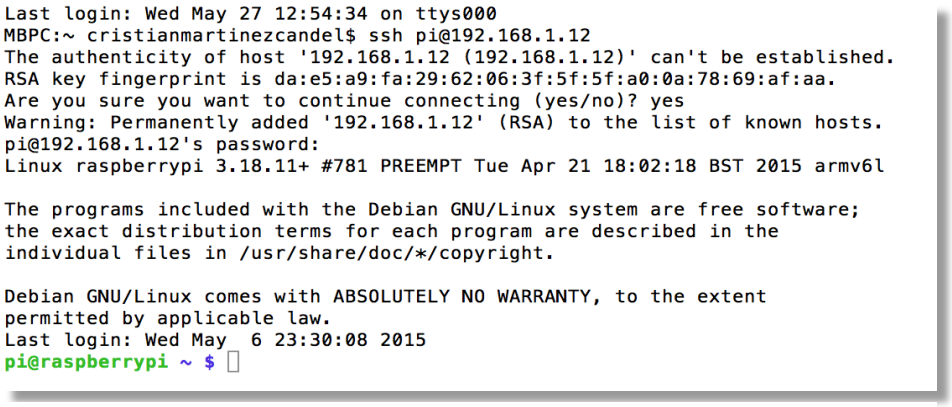



**Figura 4.2. Conexión a Raspberry Pi mediante ssh**

### Configuramos interfaces

Debemos asignar una ip fija a nuestra Raspberry que servirá como interfaz de servidor. Para ello, editamos el fichero de configuración **/etc/networks/interfaces** con el siguiente comando:

***sudo nano /etc/network/interfaces***

Editamos la interfaz eth0 con los siguientes datos:

| | |
|---|---|
| IP estática: | **192.168.1.254** |
| Máscara de red: | **255.255.255.0** |
| Puerta de enlace: | **192.168.1.1** |

En la siguiente imagen podemos observar el formato del fichero de configuración /**interfaces** tras la edición del mismo.

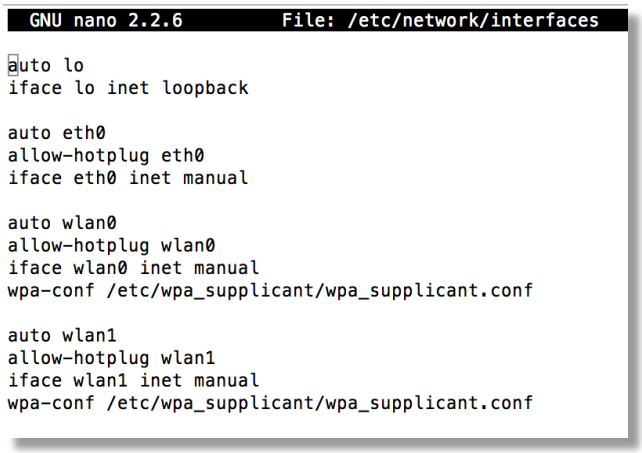



**Figura 4.3. Configuración de red por defecto**

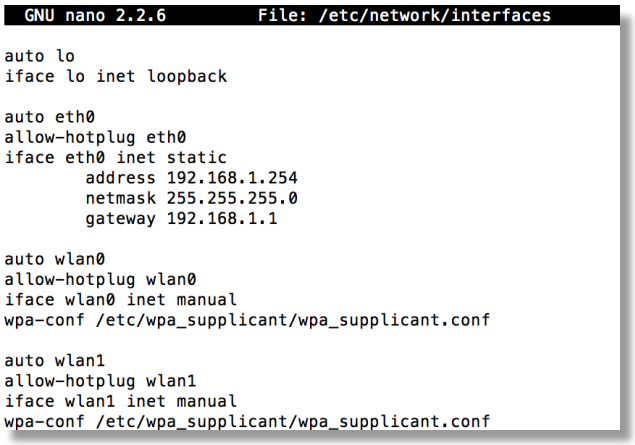



**Figura 4.4. Configuración de red editada**

Para que los cambios tengan efecto, es necesario reiniciar la Raspberry. Para ello, ejecutamos:

***sudo reboot***

Una vez reiniciada, podemos comprobar que los cambios se han realizado correctamente. Para ello, accedemos de nuevo a nuestra Raspberry con el comando:

***ssh pi@192.168.1.254***

| | |
|---|---|
| Usuario: | **pi** |
| Contraseña: | **raspberry** |

Ejecutamos ***ifconfig*** el cual nos mostrará la dirección IP actual.

```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:bc:07:48
          inet addr:192.168.1.254  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:153 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19656 (19.1 KiB)  TX bytes:18752 (18.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

**Figura 4.5. Configuración asignada a interfaz ethernet**

Una vez configurada la interfaz de red eth0, debemos cambiar los datos de acceso ya que sería un problema de seguridad dejar las credenciales asignadas por defecto. Para ello, debemos usar el usuario root. Ejecutamos el comando ***sudo passwd root*** e introducimos la contraseña para root (En nuestro caso hemos elegido root)**.** Una vez asignada la contraseña al usuario root, salimos de SSH y volvemos a entrar utilizando usuario y contraseña asignadas a root.

| | |
|---|---|
| Usuario: | **root** |
| Contraseña: | **root** |

***ssh root@192.168.1.254***

Para cambiar el nombre de usuario, ejecutamos el siguiente comando:
***usermod -l servidor pi -md /home/servidor***

Dicho comando cambiará el usuario por defecto **pi** por el usuario servidor. Nuevamente, volvemos a cerrar la conexión establecida con nuestra Raspberry para volver a establecer una nueva conexión con el usuario creado.

| | |
|---|---|
| Usuario: | **servidor** |
| Contraseña: | **raspberry** |

***ssh servidor@192.168.1.254***

Una vez establecida la conexión, por motivos de seguridad, eliminamos la contraseña generada para el usuario root

***sudo passwd -d root***

Para cambiar la contraseña por defecto, ejecutamos **sudo passwd servidor** e introducimos la contraseña elegida, en nuestro caso **servidor_wefree**

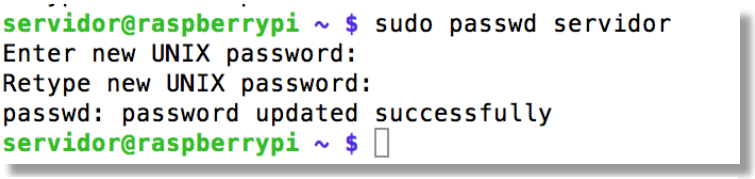



**Figura 4.6. Actualización de contraseña en servidor**

Para comprobar que se han efectuado los cambios, cerramos la conexión y comprobamos que podemos acceder con los nuevos datos de usuario (**servidor/servidor_wefree**) mediante ssh.

### 4.1.3 Instalación MySQL

Para mejorar la estructura de los usuarios que pueden acceder al servicio y de los clientes conectados, decidimos instalar un sistema de gestión de bases de datos (MySQL) en el cual gestionaremos los datos de cada usuario.

Para ello, instalamos en nuestra Raspberry el servidor MySQL y el complemento MySQL de FreeRADIUS como sigue:

***sudo apt-get install mysql-server***
***sudo apt-get install freeradius-mysql***

Durante la instalación nos pedirá una contraseña para el usuario root, en nuestro caso **servidor_wefree**

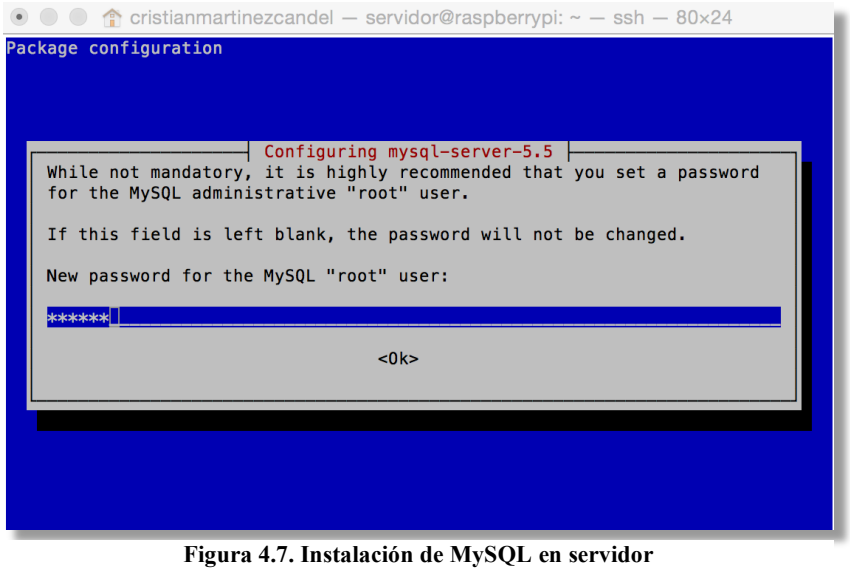



**Figura 4.7. Instalación de MySQL en servidor**

Ahora crearemos la base de datos que usaremos para nuestro servidor radius. Para ello accedemos a mysql con el usuario root configurado en la instalación:

***mysql -uroot –pservidor_wefree***

Una vez dentro, crearemos la base de datos que se llamará **wfradius**:

***create database wfradius;***

```
mysql> create database wfradius
    -> ;
Query OK, 1 row affected (0.01 sec)
```

**Figura 4.8. Creación de la base de datos**

Más tarde, crearemos un usuario llamado **wfradius** con todos los permisos para acceder a la base de datos creada anteriormente. Para ello, ejecutaremos:

***grant all on wfradius.* to wfradius@localhost identified by 'wfradius';***

```
mysql> grant all on wfradius.* to wfradius@localhost identified by 'wfradius';
Query OK, 0 rows affected (0.00 sec)
```

**Figura 4.9. Creación de usuario**

Accedemos mysql utilizando el usuario y la contraseña creados anteriormente:

***mysql -uwfradius -pwfradius***

Para acceder a la base de datos creada, utilizaremos el comando:

***use wfradius***

Para crear las tablas que usará el servidor Radius, ejecutamos el comando:

***source /etc/freeradius/sql/mysql/schema.sql***

Podemos comprobar las tablas creadas mediante el siguiente comando:

***show tables;***

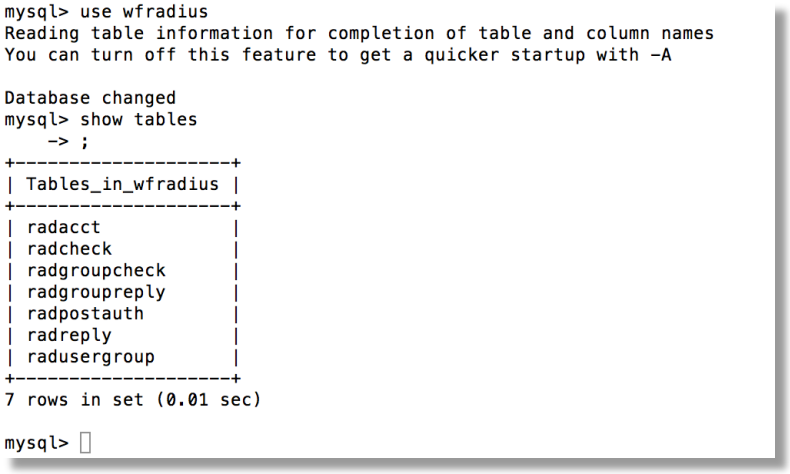



**Figura 4.10. Tablas creadas en base de datos**

En este momento, ya podremos añadir usuarios a nuestra base de datos. Para ello, accedemos a nuestra base de datos mediante el comando:

***mysql -uwfradius -pwfradius***

Los datos de los usuarios deberán introducirse en la tabla **radcheck**. Para ver la forma de ésta, ejecutamos:

***use wfradius***

***describe radcheck;***

```
mysql> describe radcheck;
+-----------+------------------+------+-----+---------+----------------+
| Field     | Type             | Null | Key | Default | Extra          |
+-----------+------------------+------+-----+---------+----------------+
| id        | int(11) unsigned | NO   | PRI | NULL    | auto_increment |
| username  | varchar(64)      | NO   | MUL |         |                |
| attribute | varchar(64)      | NO   |     |         |                |
| op        | char(2)          | NO   |     | ==      |                |
| value     | varchar(253)     | NO   |     |         |                |
+-----------+------------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

**Figura 4.11. Formato de tabla de usuarios**

Para introducir un usuario, seguiremos el esquema definido por MySQL. Observamos que el campo **id** es del tipo int auto-incrementable, es decir, cada vez que se inserte un usuario se le asignará un id de forma ordenada. El campo **username** será el nombre de nuestro usuario. En el campo **attribute** debemos definir que atributo le pasaremos. El campo **op** por defecto será el signo de asignación de valor (==) . Finalmente, el campo **value** será el valor del atributo asignado anteriormente. En nuestro caso, crearemos un usuario de prueba como sigue:

***INSERT INTO radcheck(username,attribute,value) VALUES ('wefree','Password','wefree2');***

```
mysql> INSERT INTO radcheck(username,attribute,value) VALUES ('wefree','Password','wef
ree2');
Query OK, 1 row affected (0.01 sec)
```

**Figura 4.12. Introducir usuario de prueba en base de datos**

Para comprobar el contenido de la tabla, podemos ejecutar:

***select * from radcheck;***

```
mysql> select * from radcheck;
+----+----------+-----------+----+---------+
| id | username | attribute | op | value   |
+----+----------+-----------+----+---------+
|  1 | wefree   | Password  | == | wefree2 |
+----+----------+-----------+----+---------+
1 row in set (0.01 sec)
```

**Figura 4.13. Comprobación de la correcta inserción**

En ella comprobamos que los datos son los que introducimos anteriormente.

## Clientes remotos

Para introducir los clientes remotos (puntos de acceso), debemos crear las tablas NAS. Para ello, entramos en mysql con nuestro usuario y contraseña:

***mysql -uwfradius -pwfradius***
Accedemos a nuestra base de datos:
***use wfradius***

Generamos las tablas con el comando:
***source /etc/freeradius/sql/mysql/nas.sql***

Mediante el comando ***show tables*** comprobamos que se ha generado una nueva tabla, la tabla **nas**:

```
mysql> show tables;
+--------------------+
| Tables_in_wfradius |
+--------------------+
| nas                |
| radacct            |
| radcheck           |
| radgroupcheck      |
| radgroupreply      |
| radpostauth        |
| radreply           |
| radusergroup       |
+--------------------+
8 rows in set (0.01 sec)
```

**Figura 4.14. Creación de la tabla nas**

Tal y como hemos visto con anterioridad, podemos comprobar la forma de la tabla mediante el comando ***describe nas***, el cual genera la siguiente salida:

```
mysql> describe nas;
+-------------+--------------+------+-----+---------------+----------------+
| Field       | Type         | Null | Key | Default       | Extra          |
+-------------+--------------+------+-----+---------------+----------------+
| id          | int(10)      | NO   | PRI | NULL          | auto_increment |
| nasname     | varchar(128) | NO   | MUL | NULL          |                |
| shortname   | varchar(32)  | YES  |     | NULL          |                |
| type        | varchar(30)  | YES  |     | other         |                |
| ports       | int(5)       | YES  |     | NULL          |                |
| secret      | varchar(60)  | NO   |     | secret        |                |
| server      | varchar(64)  | YES  |     | NULL          |                |
| community   | varchar(50)  | YES  |     | NULL          |                |
| description | varchar(200) | YES  |     | RADIUS Client |                |
+-------------+--------------+------+-----+---------------+----------------+
9 rows in set (0.01 sec)
```

**Figura 4.15. Estructura de tabla nas**

Como podemos observar en la imagen anterior, cada entrada se compone, entre otros, de los siguientes campos:

**nasname:** IP de cada cliente (Punto de acceso)
**shortname:** nombre de cada punto de acceso
**type:** tipo de punto de acceso (en nuestro caso other)
**secret:** Clave compartida entre el servidor y cada cliente (en nuestro caso wfradius)

Para insertar nuestros puntos de acceso en dicha tabla, emplearemos la siguiente sentencia:

***INSERT INTO nas (nasname, shortname, type, secret) VALUES ('192.168.1.X', 'AP1', 'other', 'wfradius');***

Si ejecutamos el comando ***select * from nas*** podremos comprobar el contenido de nuestra tabla nas:

```
mysql> select * from nas;
+----+--------------+-----------+-------+-------+----------+--------+-----------+---------------+
| id | nasname      | shortname | type  | ports | secret   | server | community | description   |
+----+--------------+-----------+-------+-------+----------+--------+-----------+---------------+
|  1 | 192.168.1.15 | AP1       | other |  NULL | wfradius | NULL   | NULL      | RADIUS Client |
+----+--------------+-----------+-------+-------+----------+--------+-----------+---------------+
1 row in set (0.00 sec)
```

**Figura 4.16. Contenido de tabla nas**

Para añadir otros puntos de acceso a nuestra tabla de clientes radius solo deberemos repetir los pasos anteriores.

### 4.1.4 Instalación de DaloRADIUS

Para mejorar el acceso a las bases de datos de una forma más rápida y sencilla, se ha decidido añadir al servidor la aplicación DaloRADIUS. Dicha aplicación nos permite ver y editar el contenido de nuestra base de datos de manera flexible, mediante una interfaz gráfica accesible mediante conexión HTTP. De este modo, podremos acceder desde cualquier parte a nuestra base de datos.
Para instalar dicha aplicación en nuestro servidor debemos:

Acceder al directorio **/usr/local/src**:
***cd /usr/local/src***
Descargar la aplicación mediante el comando ***wget***:

***wget http://sourceforge.net/projects/daloradius/files/daloradius/daloradius0.9-9/daloradius-0.9-9.tar.gz/download***

Descargamos los siguientes servicios adicionales (servicio web, servicio de base de datos) del repositorio Raspbian:
***apt-get install apache2 php5 php5-gd php-pear php-db libapache2-mod-php5 php-mail php5-mysql***

Accedemos al directorio **/var/www**

***cd /var/www***

Descomprimimos el archivo descargado con anterioridad mediante el comando:
***tar xzf /usr/local/src/download***

Movemos el archivo descomprimido a la raíz de nuestro servidor:

***mv -f /var/www/daloradius-0.9-9 daloradius/***

Cambiamos el propietario del directorio **daloradius/**:

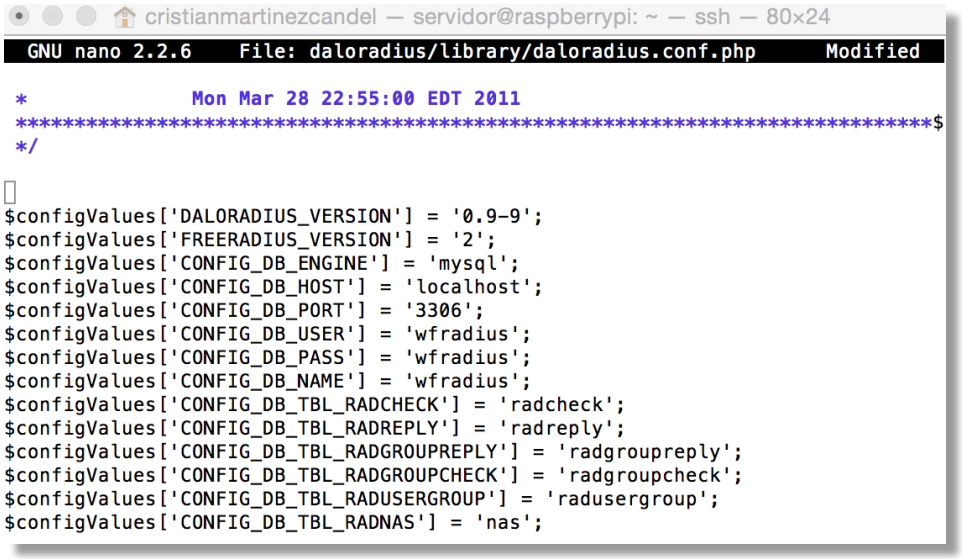
***chown -R www-data:www-data daloradius***

Cambiamos los permisos (propietario: leer y escribir, resto: Solo lectura) del archivo **daloradius/library/daloradius.conf.php:**

***chmod 644 daloradius/library/daloradius.conf.php***

Editamos los datos de nuestra base de datos (**Usuario, Contraseña, Nombre de la base de datos**) en el fichero de configuración:

***nano daloradius/library/daloradius.conf.php***




Accedemos a MySQL con nuestro usuario y contraseña:

***mysql -uwfradius -pwfradius***

Accedemos a nuestra base de datos:

***use wfradius***

Generamos las tablas de DaloRADIUS con el comando:

***source /var/www/daloradius/contrib/db/fr2-mysql-daloradius-and-freeradius.sql***

Reiniciamos el servidor http apache2.
***/etc/init.d/apache2 restart***

Una vez reiniciado el servicio apache2, nos logueamos mediante nuestro navegador en la dirección web 192.168.1.254/daloradius con las credenciales por defecto:

| | |
|---|---|
| Usuario: | **administrator** |
| Contraseña: | **radius** |

Para cambiar los datos de acceso a DaloRADIUS accedemos a **Config > Operators** y añadimos un nuevo operador (**New Operator**) con las siguientes credenciales:

| | |
|---|---|
| Operator Username: | **wefree** |
| Operator PAssword: | **servidor_wefree** |

Para eliminar el operador por defecto, seleccionamos **Remove Operator** e introducimos el nombre de usuario que deseamos eliminar: **administrator.**

### 4.1.5 Configuración de base de datos

#### Registro de usuarios

Para registrar los usuarios que tendrán acceso a Internet en los distintos puntos de acceso debemos conectarnos a nuestra base de datos introduciendo usuario y contraseña. Una vez conectados, en la sección **Management > Users** añadiremos los usuarios mediante la opción **New User**.

Dado que la autenticación en el servidor será mediante nombre de usuario, seleccionaremos **Username Authentication**. En dicha opción encontraremos los siguientes campos:

**Username:** Nombre de usuario.

**Password:** Contraseña asociada al usuario.

**Password Type:** Tipo de contraseña. En nuestro caso "Cleartext-Password".

**Group:** Grupo al que pertenecerá el usuario (Por defecto, "Desactivados"),

Para guardar el usuario, pulsamos en **Apply**.

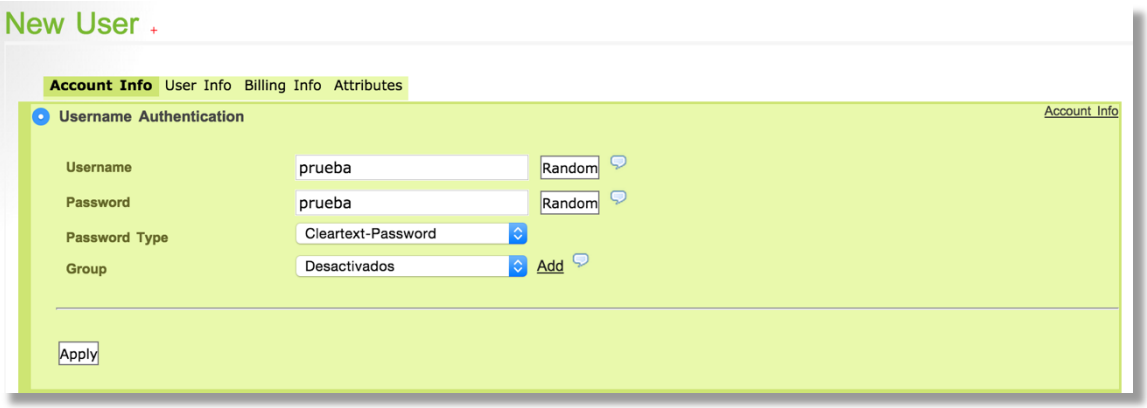



**Figura 4.17. Registro de usuarios mediante DaloRADIUS**

## Registros de puntos de acceso

Para llevar un control de los puntos de acceso que tendrán accesibilidad al servicio será necesario registrar dichos puntos de acceso en la tabla **hotspot**. En la sección **Management > Hotspots** podremos añadir los puntos de acceso mediante la opción **New Hotspot**.

Los campos necesarios son:

**Hotspot Name:** Nombre del punto de acceso (Debe coincidir con el nombre de usuario).
**MAC Address:** MAC del punto de acceso.

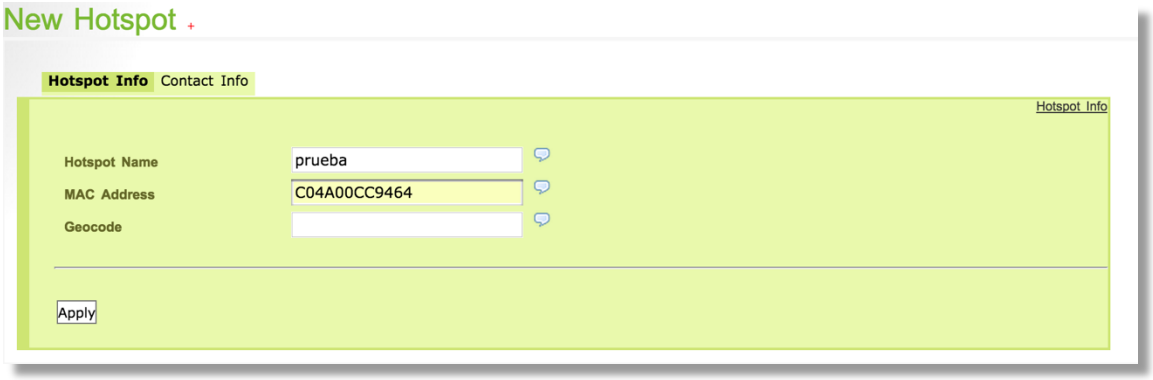



**Figura 4.18. Registro de puntos de acceso**

## Control de usuarios

Para deshabilitar los usuarios es necesario crear un nuevo perfil, denominado "Desactivados".

Para ello, accedemos a la página de configuración de nuestro servidor radius (Daloradius), nos dirigimos al apartado **Management** en el que encontraremos **Profile**. Creamos el nuevo perfil haciendo click en **New Profile**.
Pulsamos sobre **Quickly Locate attribute with autocomplete input**. En **Custom attribute** insertamos el valor **Auth-Type** y hacemos click en **Add Attribute** el cual abrirá un desplegable.

Completamos la siguiente información:

| | |
|---|---|
| **Value:** | Reject |
| **OP:** | := |
| **Target:** | check |

Asignamos el nombre a nuestro perfil en el campo **Profile Name** en este caso **Desactivados** y guardamos la información pulsando en **Apply**.

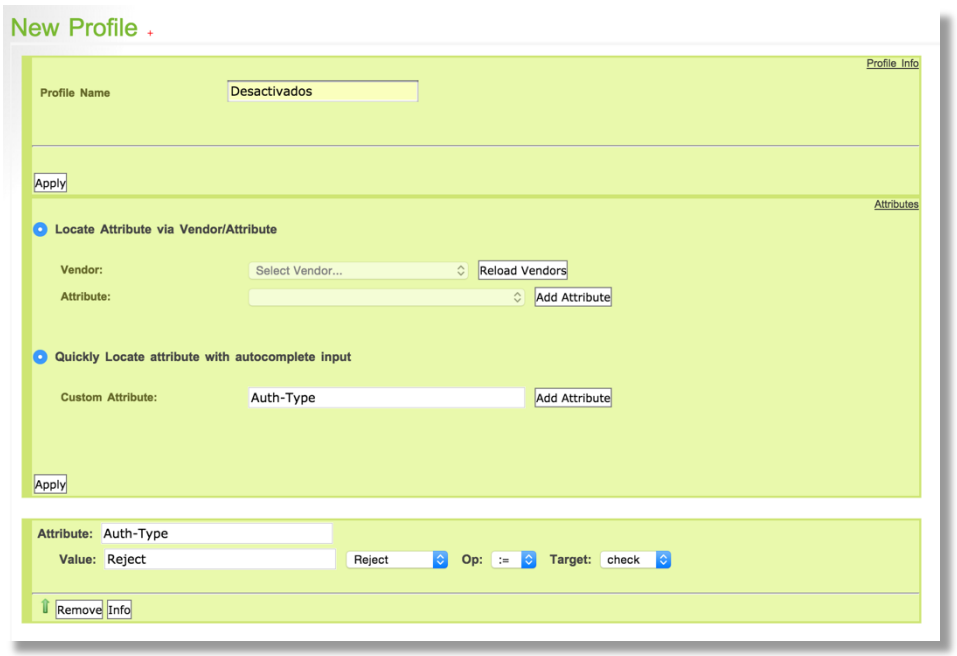



**Figura 4.19. Creación de perfiles en DaloRADIUS**

Para agregar un nuevo perfil para los usuarios **Activos** seguimos el mismo procedimiento exceptuando el valor del campo **Value** en el que para este perfil deberemos asignar el valor **Accept**.

Una vez realizado el proceso, si pulsamos sobre **List profiles** situado en la parte izquierda de nuestra pantalla, observaremos los perfiles guardados como se muestra en la siguiente imagen.

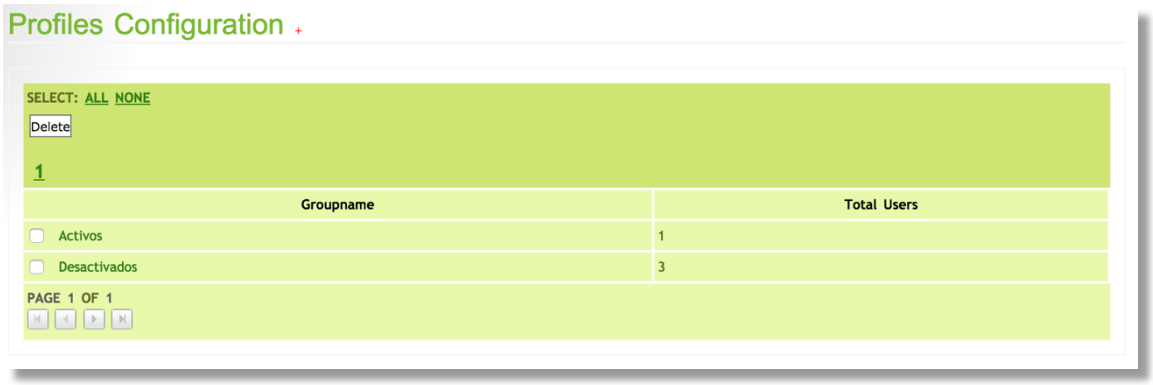



**Figura 4.20. Perfiles creados en DaloRADIUS**

En este momento, si queremos desactivar un usuario, debemos acceder al sub-apartado de **Management** denominado **User-Groups**. En él, agregamos un usuario al grupo **Desactivados** haciendo click en **New User-group Mappings**. Aquí debemos completar los siguientes apartados:

**Username:** Nombre de usuario a desactivar
**Groupname:** Desactivados
**Priority:** Prioridad (en nuestro caso 0)

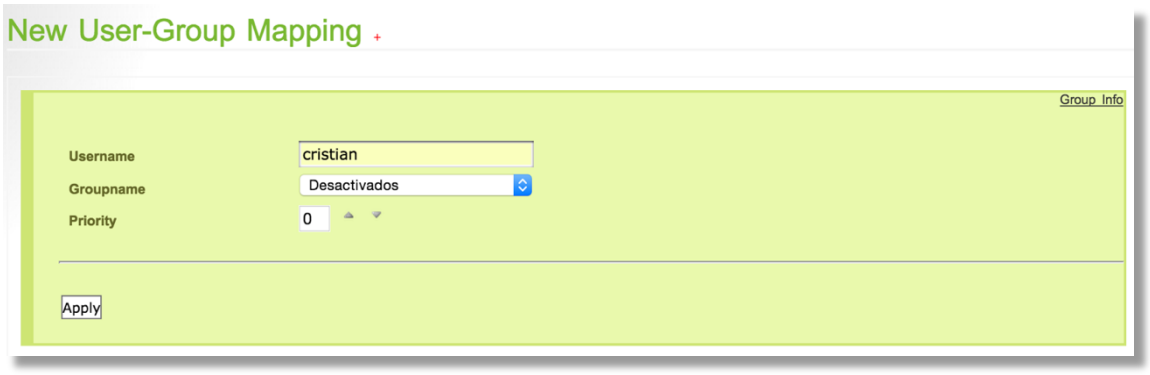



**Figura 4.21. Desactivar usuarios en DaloRADIUS**

Por otro lado, para activar un usuario tan solo tendremos que cambiar el campo **Groupname** al valor **Activos**.

Dada la escalabilidad de nuestro proyecto, hacer este proceso manualmente sería inviable. Por este motivo, para activar o desactivar usuarios realizaremos un script que haga esta función de forma autónoma.

### 4.1.6 Comprobación autónoma de la conexión con clientes

Para comprobar la conexión de los puntos de acceso con nuestro servidor, es necesario un script que compruebe el estado de éstas de forma autónoma cada cierto tiempo.
Para ello, comprobamos la fecha y hora de la última actualización de cada punto de acceso (datos en tabla NAS). Si el campo **hora** coincide, el punto de acceso estará conectado a la red. Si dicha condición se cumple, comprobamos que el usuario no se encuentre en la tabla **radusergroup** con nombre **Desactivados** y si es así eliminamos dicha entrada y agregamos el usuario a la tabla **radusergroup** con nombre **Activos**.
Por otro lado, si el campo "hora" no coincide, comprobamos si el usuario asignado al punto de acceso se encuentra en la tabla **radusergroup** con nombre **Desactivados** y si no es el caso, lo añadimos a la tabla **radusergroup** con nombre **Desactivados** y lo eliminamos de la tabla **radusergroup** con nombre **Activos**. Para hacer estas actualizaciones emplearemos sentencia MySQL como veremos a continuación.
Para llevar a cabo el proceso mencionado, crearemos el archivo bash en el directorio **/etc/script_wefree** que comprobará las conexiones. Para ello, ejecutamos:

***nano /etc/script_wefree/comprueba_con.sh***

Ahora editamos el fichero bash creado, añadiendo el siguiente código:

```
#!/bin/bash

###SCRIPT para ACTIVAR/DESACTIVAR usuarios según su conexión

echo 'ULTIMA COMPROBACIÓN: '  `date +"%d-%m-%Y %H:%M:%S"`;

host_sql="localhost"
usuario_sql="wfradius"
pass_sql="wfradius"
basedatos_sql="wfradius"
sql_con="-h $host_sql -u $usuario_sql -p$pass_sql -D $basedatos_sql -s -e"


#Declaramos la variable nombreus como un array

declare -a nombreus;

nombreus=(`mysql $sql_con "SELECT username FROM radcheck"`);
nusuarios=${#nombreus[@]};




i=0
while [ "$i" -le "$(($nusuarios-1))" ];
do

nombre=${nombreus[i]};

#Obtenemos la fecha de actualización de cada punto de acceso

fecha_update=$(mysql $sql_con "SELECT updatedate FROM hotspots WHERE name='$nombre'" | cut -c 1-13);
```

```
fecha_actual=$(date +"%Y-%m-%d %H");


#Si la hora de actualizacion del AP es la misma que la hora actual, el punto de acceso está conectado
if [ "$fecha_actual" == "$fecha_update" ]
then

echo "Punto de acceso conectado";

#Si la hora difiere, el AP no está conectado (Pasamos a desactivar la entrada en la tabla nas)

else

echo "Punto de acceso desconectado";


mysql $sql_con "DELETE FROM nas WHERE shortname='$nombre';" > /dev/null;

fi

res=$(mysql $sql_con "SELECT count(*) FROM nas WHERE shortname='$nombre'");

#Si el punto de acceso está conectado, añadimos el usuario a la tabla Activos

if [ "$res" -ne 0 ]
then

compact=$(mysql $sql_con "SELECT count(*) FROM radusergroup WHERE username='$nombre' AND groupname='Activos';");

    if [ "$compact" -eq 0 ]
    then
```

```
        echo 'Activando Usuario ' $nombre '...';
        mysql $sql_con "INSERT INTO radusergroup (username,groupname,priority)
VALUES ('$nombre','Activos','0');" > /dev/null;
        mysql $sql_con "DELETE FROM radusergroup WHERE username='$nombre'
AND groupname='Desactivados';" > /dev/null;
    else

        echo 'Usuario' $nombre ' ACTIVO';
       #mysql $sql_con "UPDATE radusergroup SET username='$nombre',
groupname='Desactivados', priority='0';" > /dev/null;


    fi

#Si no se cumple dicha condición, añadimos el usuario a la tabla Desactivados

else

compdes=$(mysql $sql_con "SELECT count(*) FROM radusergroup WHERE
username='$nombre' AND groupname='Desactivados';");


    if [ "$compdes" -eq 0 ]
        then
        echo 'Desactivando Usuario ' $nombre '...';
        mysql $sql_con "INSERT INTO radusergroup (username,groupname,priority)
VALUES ('$nombre','Desactivados','0');" > /dev/null;
        mysql $sql_con "DELETE FROM radusergroup WHERE username='$nombre'
AND groupname='Activos';" > /dev/null;
    else

     echo 'Usuario' $nombre ' INACTIVO';
#           mysql $sql_con "UPDATE radusergroup SET username='$nombre',
groupname='Desactivados', priority='0';" > /dev/null;
```

```
        fi
fi

let i++;


#Obtenemos los datos de los usuarios almacenados en nuestra base de datos y los pasamos a un fichero de texto

mysql $sql_con "SELECT username,attribute,op,value FROM radcheck WHERE username IN (SELECT username FROM radusergroup WHERE groupname='Activos')" > /etc/script_wefree/usuariosNE.txt

#Encriptamos el fichero de texto y lo pasamos al servidor web

openssl enc -aes-256-cbc -in /etc/script_wefree/usuariosNE.txt -pass pass:LSEMI -out /var/www/usuariosEN.txt;
done
```

Una vez editado nuestro script, damos permisos de ejecución al propietario (en nuestro caso, el servidor) mediante el siguiente comando:

***chmod u+x comprueba_con.sh***

### 4.1.7 Comprobar estado de los puntos de acceso

Para dar acceso a los usuarios, es necesario tener un registro de los puntos de acceso conectados a Internet. Para ello, en nuestro servidor hemos creado una página PHP que recoge los datos de los puntos de acceso mediante el paso de los datos de estos a través

de la URL de la página PHP. Los puntos de acceso ejecutarán la petición al servidor de forma autónoma periódicamente. Al recibir la petición, el archivo PHP comprobará si los datos del punto de acceso (IP y MAC) están asociados a algún punto de acceso de nuestro servicio. Más tarde, se comprueba si el punto de acceso está dado de alta. Si ya se encuentra dado de alta, se actualizan sus datos (IP y nombre de usuario asociado). Si no se encuentra dado de alta, creamos un nuevo NAS con los datos obtenidos (dirección IP, nombre de usuario y contraseña RADIUS).
Para crear el archivo PHP ejecutamos:

***nano /var/www/CompIPMAC.php***

**(Contenido del archivo CompIPMAC.php)**

```
<?php


$con=mysql_connect("localhost","wfradius","wfradius") or die ("Imposible conectar:".mysql_error());

mysql_select_db("wfradius") or die("No ha sido posible seleccionar la base de datos");

$ip=$_GET["IP"];
$mac=$_GET["MAC"];

//Actualizamos fecha de comprobación del punto de acceso

$fecha=strftime("%Y-%m-%d %H:%M:%S");

mysql_query("UPDATE hotspots SET updatedate='".$fecha."' WHERE mac='".$mac."';");

$comp_ap=mysql_query ("SELECT * FROM hotspots WHERE mac='".$mac."'");
```

```
while ($fila=mysql_fetch_assoc($comp_ap)){
  $nombreap=$fila["name"];

  $comp_ip=mysql_query("SELECT * FROM nas WHERE shortname='".$nombreap."'");


  //ACTUALIZAMOS DATOS DE LA TABLA NAS

  $existe=mysql_num_rows($comp_ip);

  //SI EXISTE EL NAS CORRESPONDIENTE A DICHO USUARIO ACTUALIZA LOS DATOS

      if ($existe > 0){

          mysql_query("UPDATE nas SET nasname='".$ip."',shortname='".$nombreap."',type='other',secret='wfradius';");

      //SI NO EXISTE, CREA UN NUEVO NAS
      }else{

          mysql_query("ALTER TABLE nas AUTO_INCREMENT = 0;");
          mysql_query("INSERT INTO nas(nasname,shortname,type,ports,secret) VALUES ('$ip','$nombreap','other','0','wfradius')");

       }
  }

?>
```

Debemos comprobar periódicamente que los puntos de acceso están conectados a Internet, ya que si no es así deberemos desactivar el usuario asignado a dicho punto de acceso.

Para dicha comprobación, debemos ejecutar el script cada 10 minutos. Para llevar a cabo este propósito, debemos añadir nuestro archivo bash al Cron (demonio) de nuestro servidor.

La sintaxis de una tarea del cron tiene la siguiente forma:

**[minuto] [hora] [día] [mes] [díadelasemana] [comando]**

- **minuto** - número entero entre 0 y 59.
- **hora** - número entero entre 0 y 23.
- **día** - número entero entre 1 y 31 (debe ser un día válido si se especifica un mes).
- **mes** - número entero entre 1 y 12 (o nombre corto del mes, por ejemplo, ene, feb, etc.).
- **díadelasemana** - número entero entre 0 y 7, donde 0 o 7 corresponde a Domingo, o bien el nombre corto del día de la semana, por ejemplo, Sun (Sunday, domingo), Mon, etc.).
- **comando** - comando que debe ejecutarse.

Por otro lado, existen palabras reservadas para la programación del cron. Estas son:

**@reboot:** Ejecutar al reinicio.
**@yearly:** Ejecutar una vez al año.
**@monthly:** Ejecutar una vez al mes.
**@weekly:** Ejecutar una vez a la semana.
**@daily:** Ejecutar una vez al día.
**@hourly:** Ejecutar cada hora.

En nuestro caso, para que el script se ejecute todos los días del año, cada 10 minutos, deberá tener la siguiente forma:

***/10 * * * * root /etc/script_wefree/comprueba_con.sh > /etc/prueba.out**

Para ello, creamos un archivo denominado **actualiza_auto** en el directorio **/etc/cron.d** mediante el comando:

***nano /etc/cron.d/actualiza_auto***

Editamos dicho archivo, añadiendo la sentencia anterior al archivo creado.

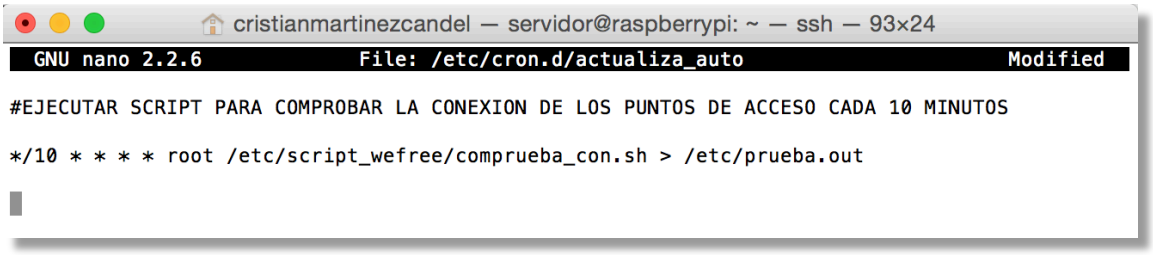



**Figura 4.22. Contenido del archivo /etc/cron.d/actualiza_auto**

Una vez editado, guardamos y reiniciamos el servidor mediante el comando **reboot**.
Como podemos observar, se ha re direccionado la salida de la ejecución del script al archivo **prueba.out** para poder comprobar la periodicidad de la actualización.

### 4.1.8 Solucionar problema IP dinámica

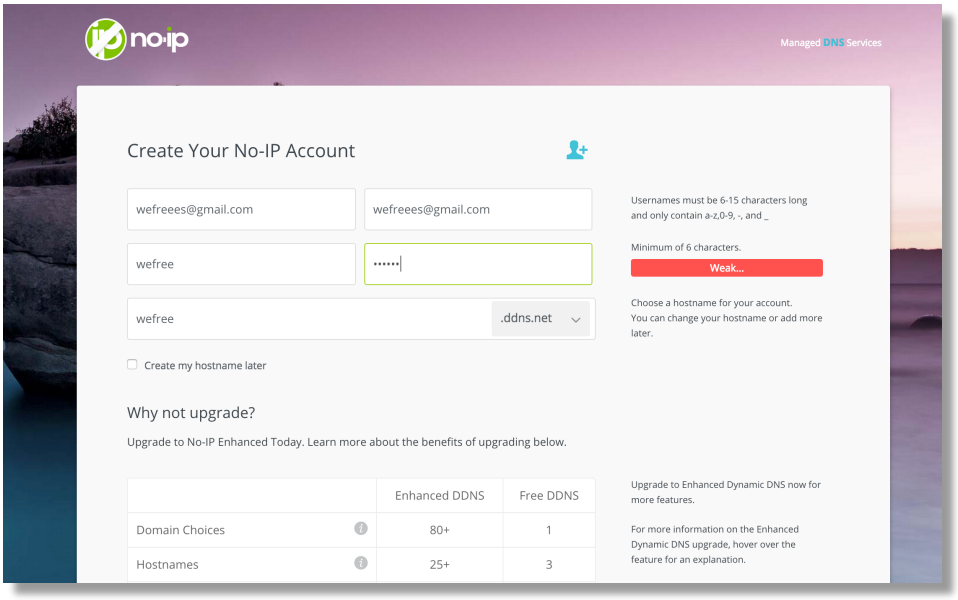
Nuestro proveedor de servicio nos asigna una IP pública dinámica la cual cambia su valor de forma periódica. Esto conlleva a que el acceso a nuestro servidor desde fuera de nuestra red sea casi imposible dado la dificultad tener actualizado el valor de nuestra ip.
Para resolver este inconveniente, podemos utilizar un servidor de DNS dinámico. Dicho servidor es el encargado de relacionar nuestra ip pública dinámica con una dirección web

estática de forma automática. Con esto logramos que aunque nuestra IP pública cambie, nuestro servidor siga estando accesible desde el exterior mediante la dirección web.
En nuestro caso, hemos optado por el servicio de www.noip.com ofrecido de forma gratuita.

## Instalación del servicio NO-IP

Primeramente nos registramos en www.no-ip.com

| | |
|---|---|
| **E-mail:** | wefreees@gmail.com |
| **Usuario:** | wefree |
| **Contraseña:** | XXXXXX |



Una vez registrados, creamos un nuevo host. Para ello, en la sección **Hosts/Redirects** elegimos **Manage Hosts** y hacemos click en **Add A Host**. Ahora, rellenamos el campo **Hostname** con el nombre **wefree**. Los demás campos podemos dejarlos por defecto y hacemos click en **Add Host**.

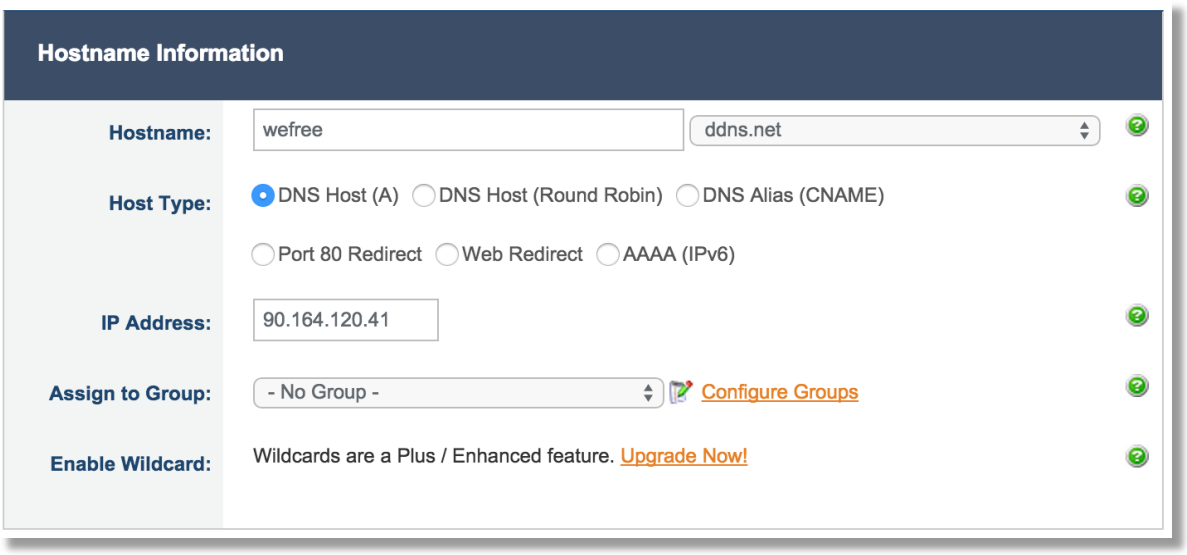



**Figura 4.23. Configuración del servidor en no-ip**

Una vez concluido el registro, instalamos la aplicación no-ip en nuestro servidor (Raspberry Pi). Para ello, ejecutaremos:

***mkdir noip***
***cd noip***
***wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz***
***tar -zxf noip-duc-linux.tar.gz***
***cd noip-2.1.9-1/***
***make***
***sudo make install***

Durante la instalación, deberemos ingresar **usuario**, **contraseña** y el **hostname** que hemos creado para nuestro servidor.

Para ejecutar el cliente utilzaremos el comando:
***sudo /usr/local/bin/noip2***

De este modo, se ejecutará el servicio noip2. Si el servidor se apaga o reinicia por cualquier motivo, este servicio no se ejecuta en el arranque de nuestro servidor.

Para ello, en el fichero descargado tenemos un script en el interior del archivo de texto **LEEME.PRIMERO**

Copiamos el contenido del script y creamos un archivo en el directorio **/etc/init.d** denominado **noip2** con el contenido del script.

***sudo nano /etc/init.d/noip2***

```
#!/bin/sh
#./etc/rc.d/init.d/functions    #descomente/modifique para su
killproc
case "$1" in
 start)
   echo "Iniciando noip2..."
   /usr/local/bin/noip2
 ;;
 stop)
   echo -n "Apagando noip2..."
   killproc -TERM /usr/local/bin/noip2
 ;;
 *)
   echo "Uso: $0 {start|stop}"
   exit 1
esac
```

Guardamos el archivo creado y hacemos el script ejecutable editando los permisos del fichero:

***sudo chmod +x /etc/init.d/noip2***

Más tarde,convertimos el script en demonio (daemon)

***sudo update-rc.d noip2 defaults***

Ahora podemos reiniciar nuestro servidor y comprobar la ejecución de nuestro script al iniciar el servidor.

### 4.1.9 Apertura de puertos

Para poder acceder al servidor desde Internet, es necesario abrir los puertos del router/firewall intermedio entre Internet y servidor.

Para llevar a cabo esta tarea, debemos acceder a la configuración NAT del router. En nuestro caso el router en cuestión es el aportado por el proveedor de servicios Orange. Para abrir los puertos, debemos ir a la Sección **Avanzado > NAT**. En ella encontraremos la pestaña **Asignación de puertos** en la que deberemos indicar el tipo de protocolo del servicio, puerto externo, equipo interno (en nuestro caso el servidor 192.168.1.254), puerto interno y nombre.

Una vez agregados los servicios **Servidor Web** y **ssh** la tabla tendrá una vista como muestra la siguiente imagen:


### 4.1.10 Página de inicio del servidor Web

Hemos decidido crear una página principal en la que informar del servicio a aquellos que estén interesados.

Para crear la página mencionada utilizaremos lenguaje HTML.

Esta página por el momento será meramente informativa, pudiendo añadir en un futuro otros servicios (ayuda para la instalación, mapa de localización de los puntos de acceso, etc).

Para crear la página, ejecutamos el comando:

***nano /var/www/index.html***

A continuación se muestra un ejemplo sencillo de página web sin formato. Para darle un formato más vistoso, podríamos utilizar páginas de estilos CSS.

cristianmartinezcandel — servidor@raspberrypi: ~ — ssh — 94×36

GNU nano 2.2.6 File: /var/www/index.html

```
<!DOCTYPE html>
<html lang="es">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">


<body>
<center>
<div id=cabecera>
<h1> WeFree </h1>

</div>
<br>

<hr>
<div id=primero>
<h2> Quien comparte, tiene Internet en cualquier parte </h2>
</div>
<hr>
<br>
<h3>Pasos a seguir: </h3></center>


<div align=center>
<ul>
<li> Cambiar el firmware de tu punto de acceso </li>
<li> Solicitar usuario y contraseña</li>
<li> Conectar el punto de acceso al router de casa </li>
<li> ¡Comenzar a navegar en cualquier red WeFree!</li>
</ul>
</div>
```

**Figura 4.24. Código HTML de página web principal**

Si visitamos la dirección web del servidor, obtendremos la página creada anteriormente.


**Figura 4.25. Vista de la página web principal**

## 4.2 Instalación de OpenWrt

Se ha decidido cambiar el firmware de fábrica del punto de acceso TP-Link WR3020 por el firmware de uso libre OpenWrt debido a las numerosas ventajas que aporta. En nuestro caso, debemos instalar un servidor RADIUS en el punto de acceso, algo que no sería posible con el firmware por defecto.

### 4.2.1 Actualización de Firmware en punto de acceso

Para obtener el firmware debemos ir a la página oficial de OpenWrt y descargar el firmware de nuestro punto de acceso:
**http://downloads.openwrt.org/attitude_adjustment/12.09/ar71xx/generic/openwrt-ar71xx-generic-tl-mr3020-v1-squashfs-factory.bin**

Para llevar a cabo el cambio, debemos conectarnos al punto de acceso mediante Ethernet y configurar nuestra IP dentro de la red 192.168.0.0 (en nuestro caso, hemos seleccionado la IP 192.168.0.10).

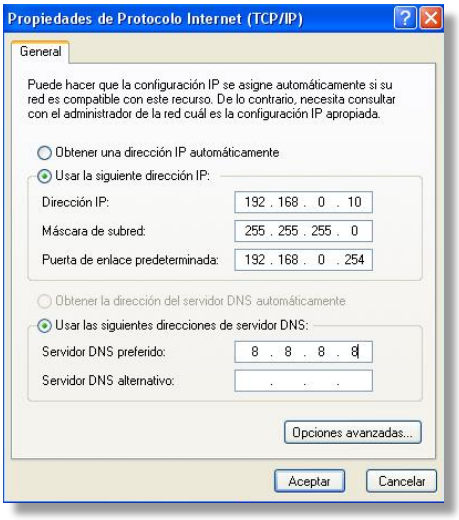



**Figura 4.26. Asignación de IP estática a PC (I)**

Accedemos a la configuración del punto de acceso insertando la IP del mismo en la barra de navegación de nuestro navegador de internet e insertamos los valores de usuario y contraseña por defecto (**admin/admin**). Nos dirigimos a **System Tools** y seleccionamos

**Firmware Upgrade**. En el campo **File** seleccionamos el firmware descargado anteriormente y pulsamos en **Upgrade**.

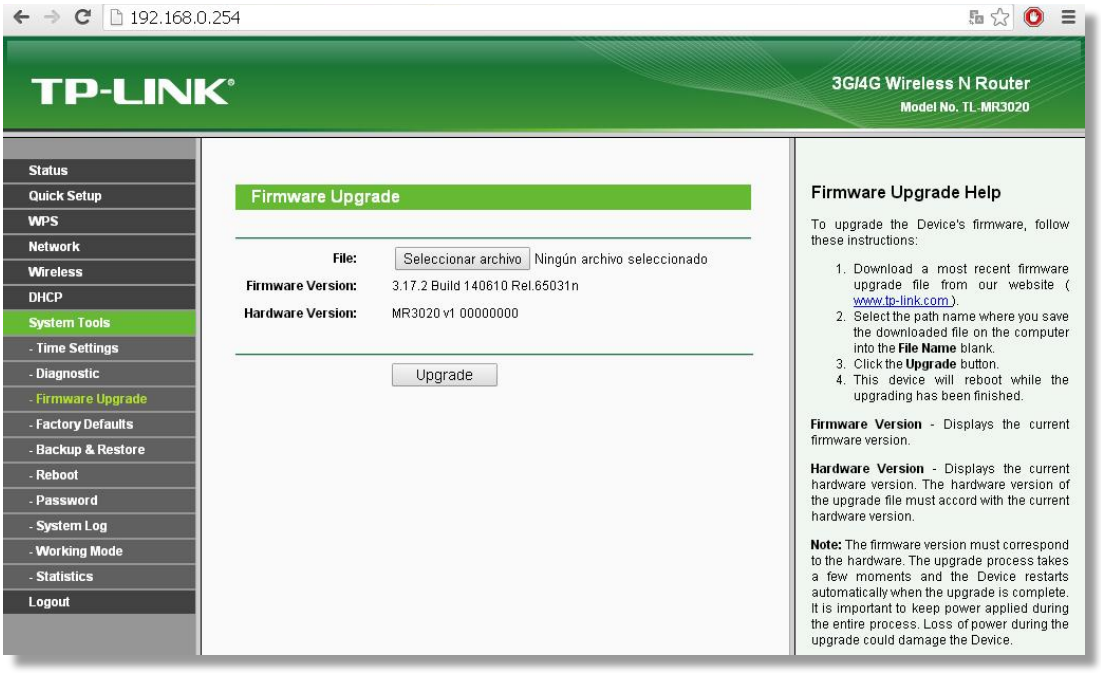



**Figura 4.27. Página de actualización de Firmware de TP-Link WR3020**

El proceso de actualización comenzará y cuando finalice el punto de acceso se reiniciará automáticamente.

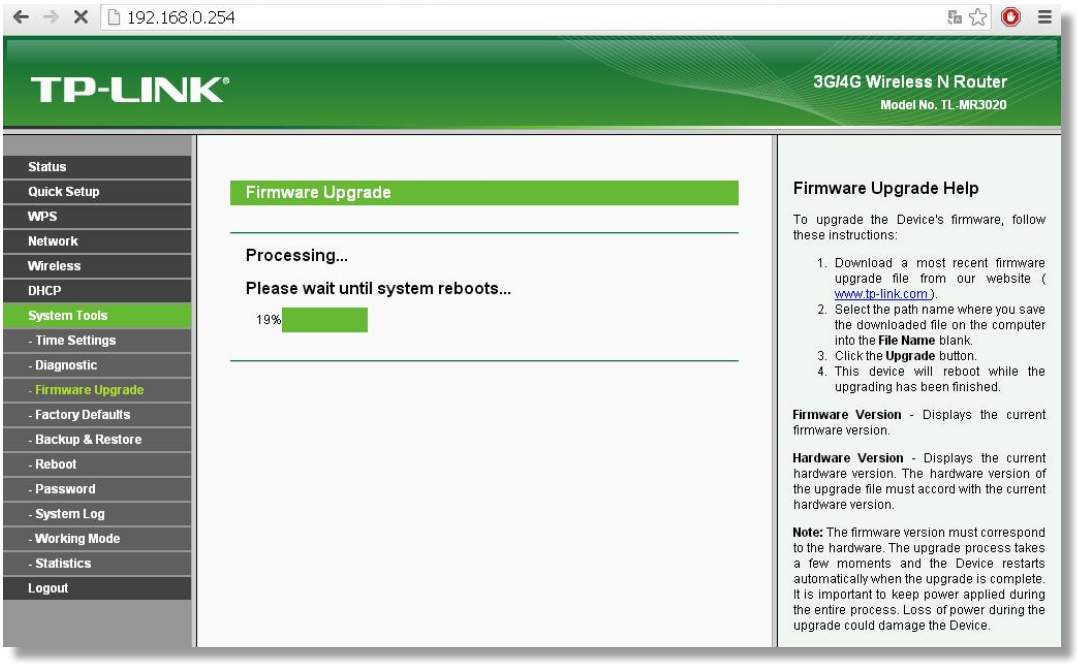



**Figura 4.28. Proceso de actualización de Firmware de TP-Link WR3020**

### 4.2.2 Configuración básica de OpenWrt

Una vez reiniciado, será necesario cambiar la IP de nuestro PC para poder acceder a la configuración del punto de acceso. En este caso debemos seleccionar una IP que pertenezca a la red 192.168.1.0 (en nuestro caso, 192.168.1.10).

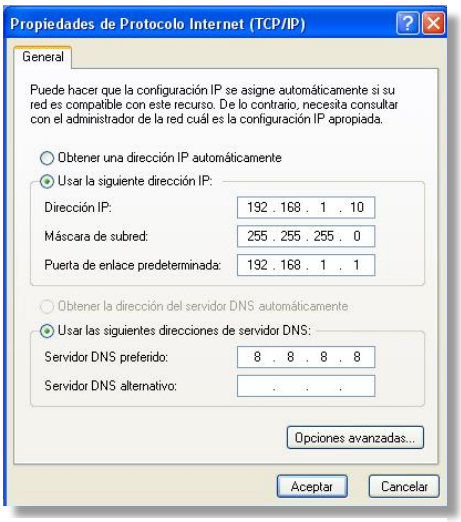



**Figura 4.29. Asignación de IP estática a PC (II)**

Para acceder a la configuración del punto de acceso ingresaremos en la URL la IP asignada por defecto en el firmware de OpenWrt (192.168.1.1).

En el primer acceso, no hay una contraseña definida para el usuario **root**. Para poder definirla, pulsamos sobre el mensaje **Go to password configuration…** de la página principal.

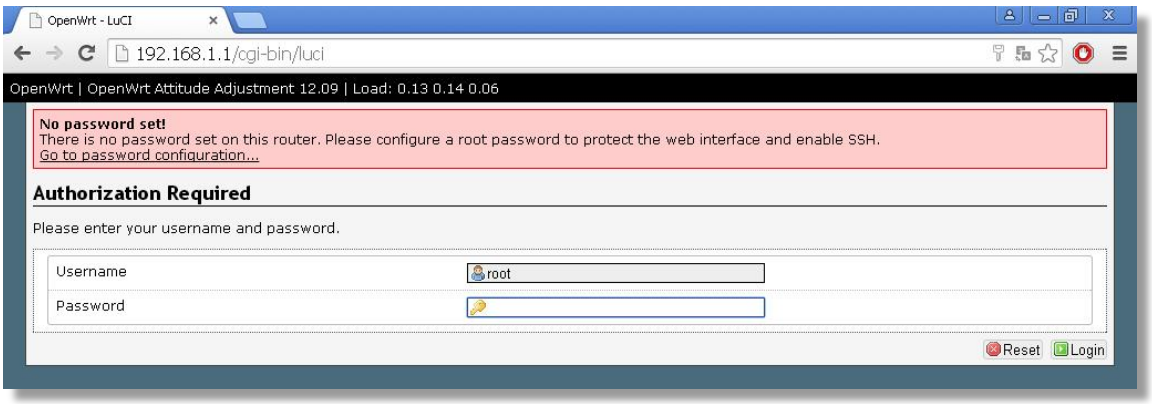



**Figura 4.30. Página principal de OpenWrt**

Una vez asignada la contraseña, accedemos al apartado **Network** y seleccionamos la pestaña **Interfaces**. Para cambiar la configuración de la interfaz, seleccionamos **Edit**.

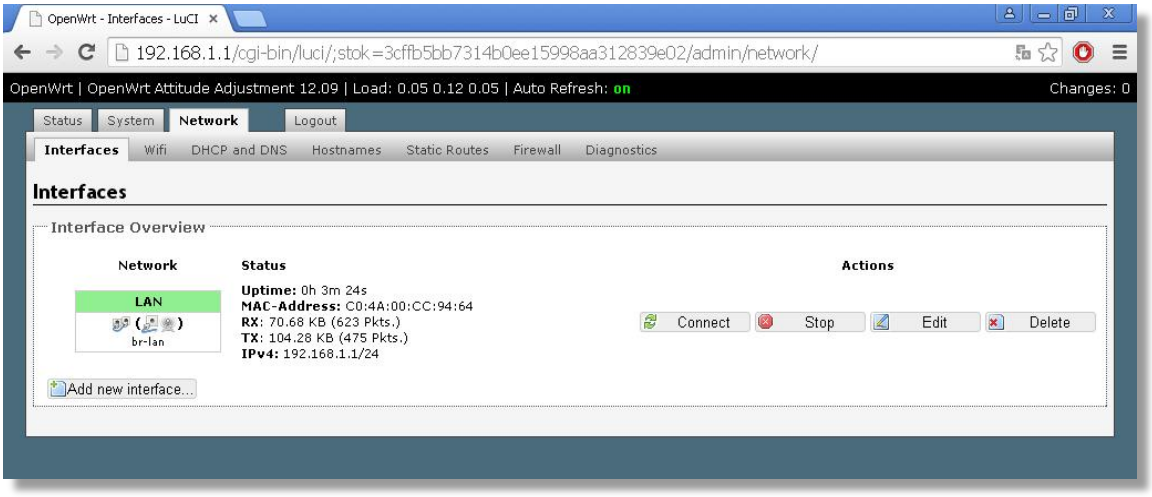



**Figura 4.31. Edición de interfaces del punto de acceso**

Dado que no sabemos la dirección de red a la que se conectará el equipo, la mejor opción es seleccionar una IP mediante protocolo DHCP. Para ello, en la opción **Protocol** seleccionamos **DHCP client**. Para guardar la configuración seleccionamos **Save & Apply**.

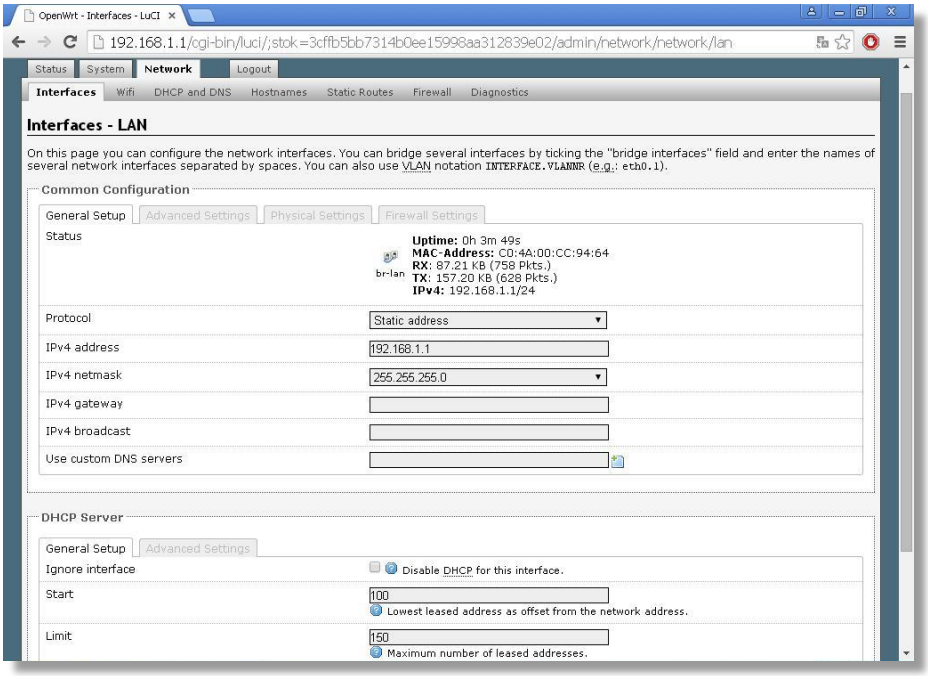



**Figura 4.32. Cambio de Protocolo de la interfaz**

Para compartir el acceso a Internet de forma inalámbrica, debemos crear una nueva interfaz (lan) en la que crearemos la red inalámbrica independiente.

Para crear una nueva interfaz nos dirigimos apartado **Network > Interfaces** seleccionamos **Add new interface.**

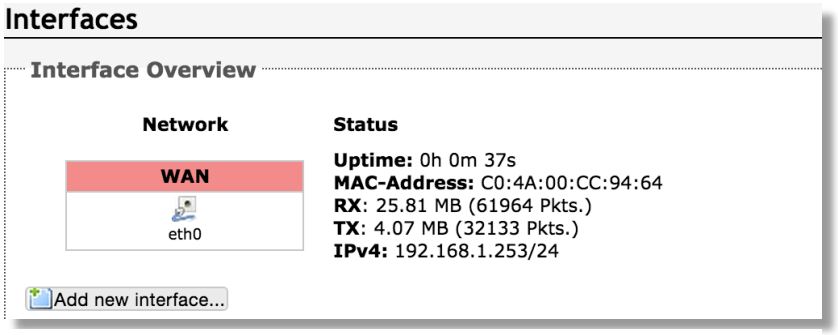



**Figura 4.33. Sección interfaces del punto de acceso**

En la nueva ventana, aparecen los siguientes campos:

**Name of the new interface:** Nombre de la nueva interfaz.

**Protocol of the new interface:** Protocolo asignado a la nueva interfaz.

**Create a bridge over multiple interfaces:** Crear puente entre varias interfaces.

**Cover the following interface:** Cubrir la siguiente interfaz.

En nuestro caso, emplearemos como nombre de interfaz **lan**, protocolo **static address** y en **Cover the following interface** seleccionaremos **Custom interface**, añadiendo el nombre de nuestra interfaz (lan).

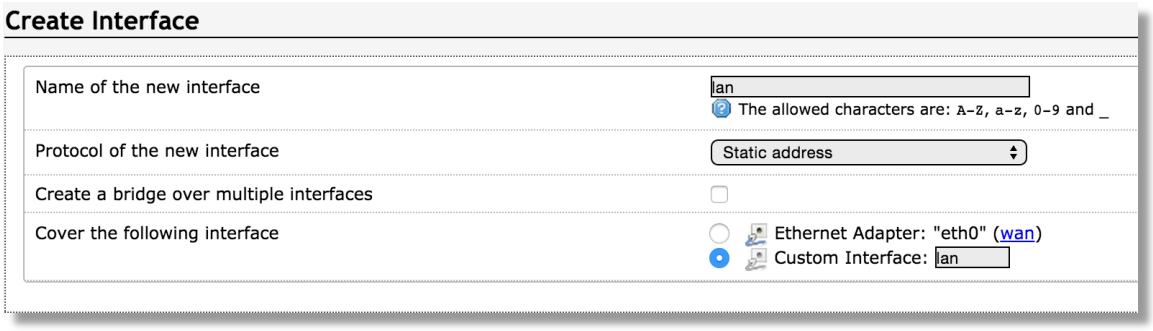



**Figura 4.34. Creación de nueva interfaz**

Para guardar los cambios, hacemos click en **Submit.**

En la siguiente ventana, deberemos configurar los datos de la interfaz. Entre ellos, la dirección IP de la nueva interfaz. En nuestro caso, hemos escogido la dirección IP 192.168.10.1 que hará de puerta de enlace de los usuarios que se conecten a la red inalámbrica.

Este valor debemos asignarlo en el campo **IPv4 address.** Del mismo modo, seleccionamos la máscara de red en el campo **IPv4 netmask** que en nuestro caso será 255.255.255.0. Por último asignamos un servidor DNS por defecto, por ejemplo, 8.8.8.8 en el campo **Use custom DNS servers.**

Para finalizar y guardar la configuración, hacemos click en **Save & Apply**.

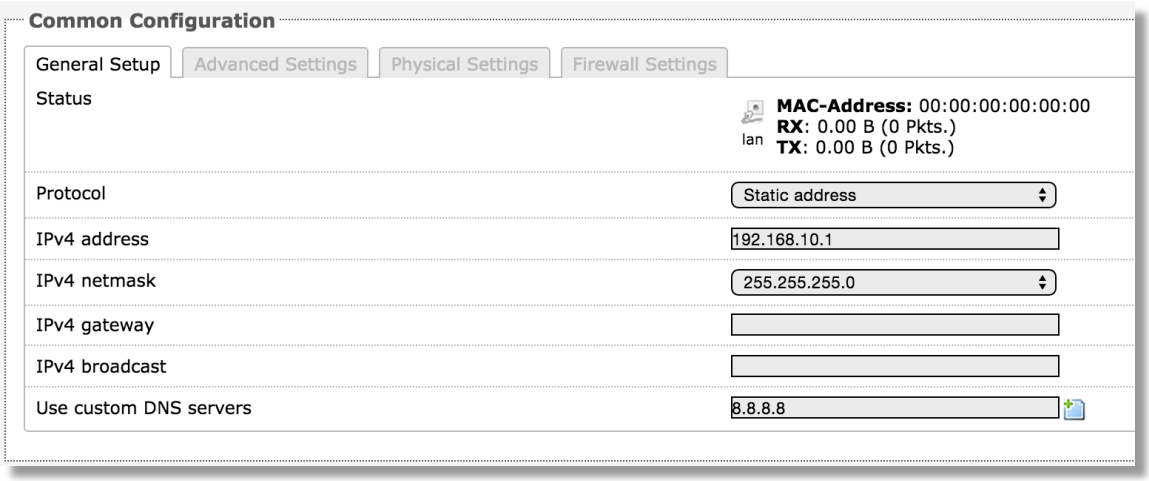



**Figura 4.35. Asignación de dirección de Red a interfaz inalámbrica**

El siguiente paso será crear la red inalámbrica. Para ello, nos dirigimos a la sección **Network >Wifi** y hacemos click en **Add**.

En **Device Configuration** seleccionamos **auto** en el campo **Channel** para que el punto de acceso elija de forma automática el canal en el que creará la red inalámbrica. En el campo **Transmit Power** seleccionamos 20dBm (100mW) que es la potencia máxima de transmisión en redes inalámbricas permitida en España.

En **Interface Configuration**, en la pestaña **General Setup** insertamos en el campo **ESSID** el nombre elegido para nuestra red inalámbrica (en nuestro caso, WeFree), en el campo **Mode** seleccionamos **Access Point**, en el campo **Network** seleccionamos la

interfaz creada anteriormente (lan), la casilla **Hide ESSID** la dejamos desmarcada y dejamos marcada la casilla **WMM**. Para guardar los cambios, hacemos click en **Save & Apply**.

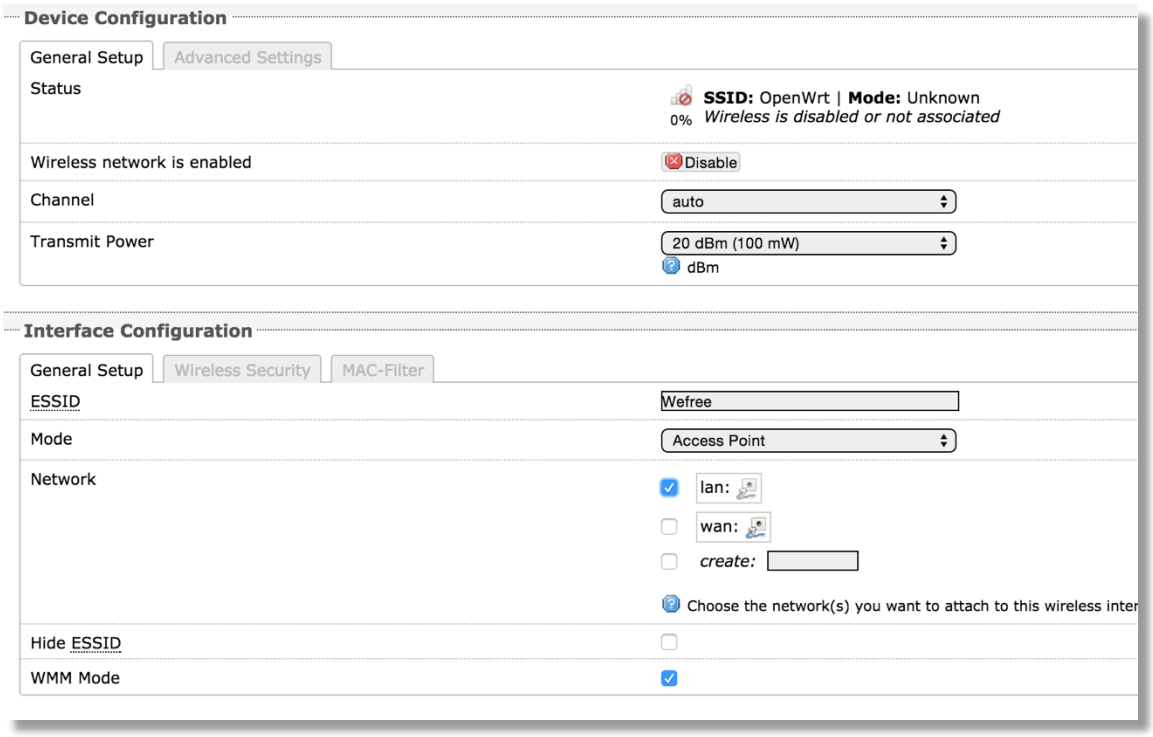



**Figura 4.36. Creación de red inalámbrica**

Ahora podemos comprobar como ha sido creada la red inalámbrica.


**Figura 4.37. Comprobación de la red creada**

### 4.2.3 Ampliación de memoria interna OpenWrt

Debido a la escasa memoria interna del dispositivo TP-Link WR3020 hemos decidido ampliar dicha memoria mediante la conexión USB. Para ello, se ha utilizado una memoria de 2GB de capacidad a la que exportaremos la raíz del dispositivo**.** El proceso de instalación a seguir será:

1.- Creamos 2 particiones (1 de ellas con formato ext4 (1000MB) dedicada a memoria interna y otra swap de 512MB dedicada a memoria RAM) en nuestra memoria USB.

2.- Instalamos en OpenWrt

**block-mount**
**kmod-usb-storage**
**kmod-usb-uhci**
**kmod-fs-ext4**
**mountd**
**mount-utils**
**libext2fs**

Para ello utilizamos los siguientes comandos:

***opkg update***
***opkg install block-mount kmod-usb-storage kmod-usb-uhci kmod-fs-ext4 mountd libext2fs***

3.- Montamos la unidad USB

Mediante la interfaz de OpenWrt podemos montar la unidad USB en el apartado **System > Mount Points.**
Pulsamos en **add** de la sección **Mount Points** y en la siguiente ventana (Mount Entry) marcamos la casilla **Enable this mount** para habilitar el punto de montaje, seleccionamos

la partición /dev/sda6 en **Device** y el punto de montaje mnt/sda6 en **Mount point**. Finalmente, seleccionamos el formato de la partición en **Filesystem** y pulsamos en **Save & Apply.**

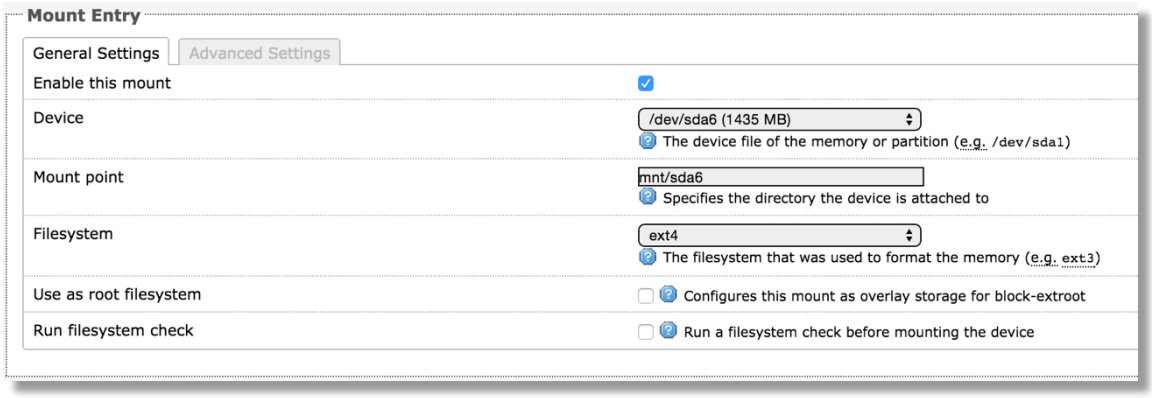



**Figura 4.38. Montaje de partición destinada a memoria interna**

Para montar la memoria SWAP, nos dirigimos a la sección **SWAP** de la ventana **Mount Points** y pulsamos en **Add**. En la siguiente ventana (Mount Points – Swap Entry) marcamos la casilla **Enable this swap** y seleccionamos la partición dev/sda5 en el apartado **Device**. Pulsamos en **Save & Apply** y reiniciamos el punto de acceso.


**Figura 4.39. Montaje de partición destinada a RAM**

4.- Copiamos a nuestra partición la raíz de OpenWrt

***tar -C /overlay -cvf - . | tar -C /mnt/sda6 -xf –***

5.- Editamos **/etc/config/fstab** para que al inicio monte la partición de la memoria USB

***vi /etc/config/fstab***

El archivo fstab quedará de la siguiente forma:

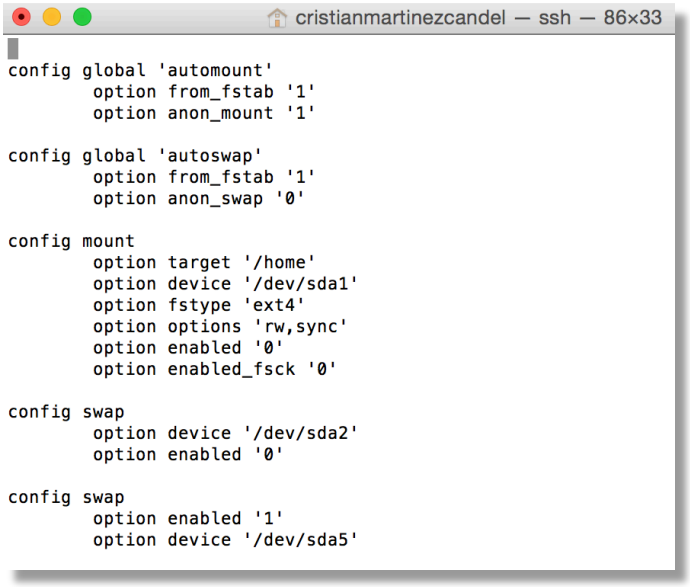



**Figura 4.40. Contenido del fichero fstab**

Al reiniciar el dispositivo, podremos comprobar como la capacidad ha aumentado considerablemente mediante el comando ***df –h***.

```
root@OpenWrt:~# df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                    1.3G    121.3M      1.1G  10% /
/dev/root                 2.0M      2.0M         0 100% /rom
tmpfs                    14.3M    100.0K     14.2M   1% /tmp
tmpfs                   512.0K         0    512.0K   0% /dev
/dev/sda6                 1.3G    121.3M      1.1G  10% /overlay
overlayfs:/overlay        1.3G    121.3M      1.1G  10% /
```

**Figura 4.41. Resultado de la ampliación de memoria interna**

### 4.2.4 Editar banner de inicio ssh de OpenWrt

Para editar la pantalla de inicio (banner) que aparece al acceder mediante ssh, debemos editar el archivo **banner** alojado en el directorio **/etc**. Para ello, ejecutamos el comando ***vi /etc/banner***

Comprobamos que su apariencia inicial es la siguiente:


**Figura 4.42. Apariencia del Banner de OpenWrt por defecto**

Una vez editada a nuestro gusto, tendremos un nuevo banner con el siguiente aspecto:

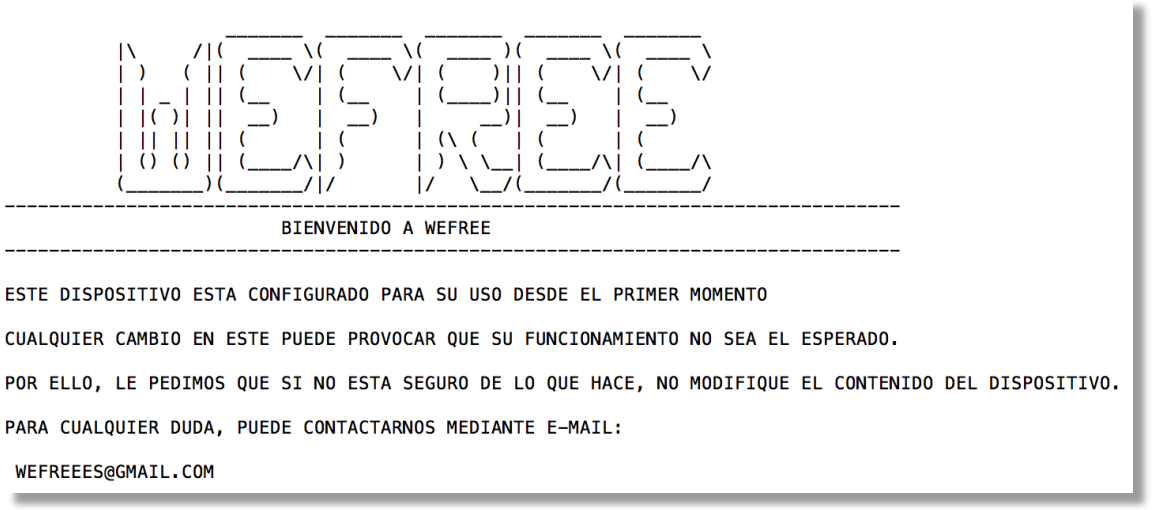



**Figura 4.43. Apariencia del Banner de OpenWrt tras su edición**

### 4.2.5 Implementación de servidor RADIUS en OpenWrt

Primeramente instalamos **openssl** para crear los certificados necesarios para nuestro servidor Radius. Para ello ejecutamos:

***opkg install openssl-utl***

Creamos los directorios necesarios para almacenar los certificados generados por la herramienta openssl:

***mkdir /etc/CertWF && chmod 700 /etc/CertWF && cd /etc/CertWF***
***mkdir certs***
***mkdir newcerts***
***mkdir private***
***mkdir crl***
***touch index.txt***
***echo "01" > serial***
***echo "00" > crlnumber***

Copiamos el archivo de configuración ***openssl.conf*** al nuevo directorio y editamos el archivo, actualizando la ruta donde se generarán nuestros certificados y la ruta de nuestros certificados.
***cp -r /etc/ssl/openssl.cnf /etc/CertWF/openssl.cnf***
***vi /etc/CertWF/openssl.cnf***

**dir= /etc/CertWF**
**certificate= $dir/CertCA.pem**
**private_key= $dir/ClaveCA.pem**

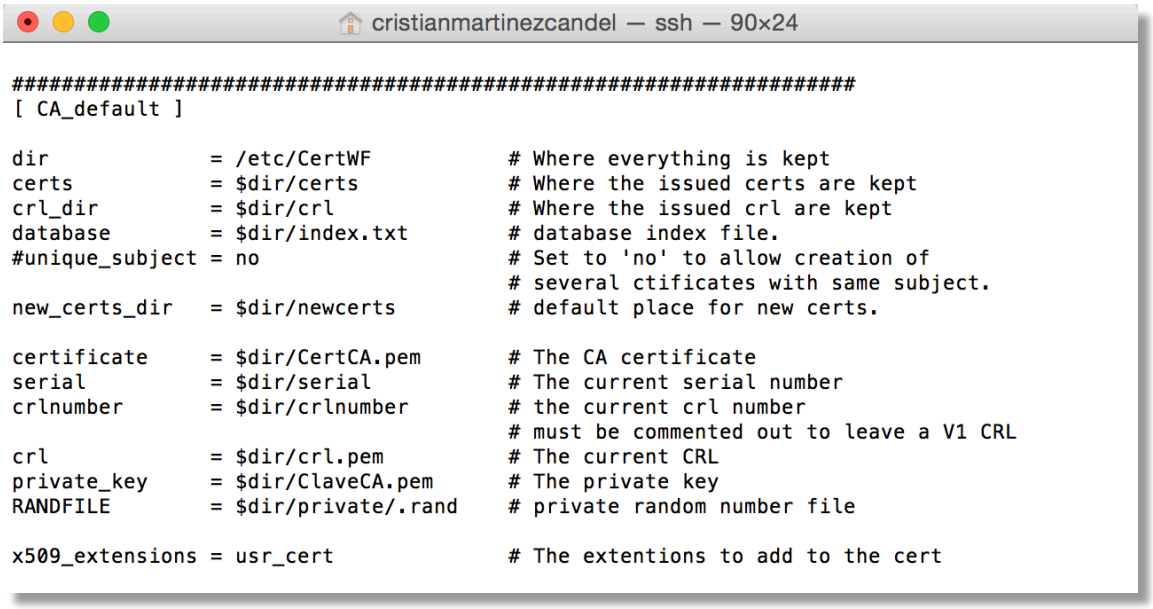



**Figura 4.44. Fichero de configuración openssl.cnf**

Creamos el archivo **xpextensions** con los siguientes **OIDs**:

[xpclient_ext]

extendedKeyUsage = 1.3.6.1.5.5.7.3.2

[xpserver_ext]

extendedKeyUsage = 1.3.6.1.5.5.7.3.1

***touch /etc/CertWF/xpextensions***

***vi xpextensions***

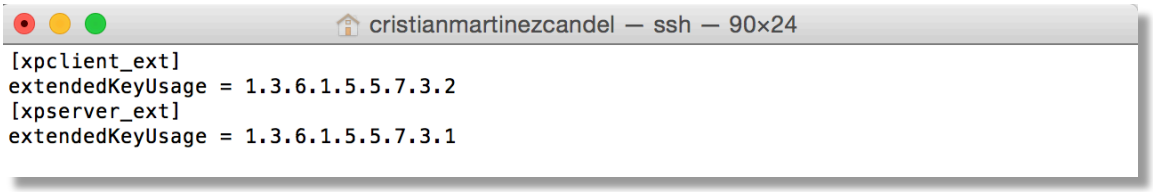



**Figura 4.45. Fichero xpextensions**

Editamos el archivo de configuración de openssl (**/etc/CertWF/openssl.cnf**) para generar certificados con validez de 4 años y actualización de CRL cada 30 días.

*vi / etc/CertWF/openssl.cnf*

```
default_days    = 1460                  # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = default               # use public key default MD
preserve        = no                    # keep passed DN ordering
```

**Figura 4.46. Edición de días por defecto de un certificado y CRL**

### Creamos nuestra propia autoridad certificadora

Para ello, ejecutamos en siguiente comando de la utilidad openssl:

***openssl req -nodes -new -x509 -keyout ClaveCA.pem -out CertCA.pem***

```
root@OpenWrt:/etc/CertWF# openssl req -nodes -new -x509 -keyout ClaveCA.pem -out CertCA.pem
Generating a 1024 bit RSA private key
....++++++
........++++++
writing new private key to 'ClaveCA.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:MU
Locality Name (eg, city) []:CT
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:WeFree Aut.
Email Address []:
```

**Figura 4.47. Creación de autoridad certificadora**

### Creamos certificado para nuestro servidor

Primeramente, debemos crear una petición de certificado del servidor a la autoridad certificadora creada anteriormente. Para ello, ejecutamos:

***openssl req –nodes –new –keyout ClaveServidor.key –out petición.csr***

```
root@OpenWrt:/etc/CertWF# openssl req -nodes -new -keyout ClaveServidor.key -out peticion.csr
Generating a 1024 bit RSA private key
.......++++++
...................++++++
writing new private key to 'ClaveServidor.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:MU
Locality Name (eg, city) []:CT
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:WeFree
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:wefree
An optional company name []:
```

**Figura 4.48. Creación de petición de certificado para Radius**

### Generamos certificado

Una vez creada la petición de certificado, podemos crear el certificado mediante el siguiente comando:

***openssl req -new -x509 -key ClaveServidor.key -out CertificadoServidor.pem***

```
root@OpenWrt:/etc/CertWF# openssl req -new -x509 -key ClaveServidor.key -out CertificadoServidor.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:MU
Locality Name (eg, city) []:CT
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:WeFree
Email Address []:
```

**Figura 4.49. Creación de certificado para Radius**

### Creamos lista de revocación de certificados, Diffie Hellman y Random

En esta lista se almacenarán los certificados revocados, es decir, aquellos certificados que ya no tengan validez. Para crear esta lista, ejecutamos:

***openssl ca -config openssl.cnf -gencrl -out crl.pem***

***openssl dhparam -text -5 1024 -out dh***

***dd if=/dev/random of=random bs=1M count=2***

```
root@OpenWrt:/etc/CertWF# openssl ca -config openssl.cnf -gencrl -out crl.pem
Using configuration from openssl.cnf
root@OpenWrt:/etc/CertWF# openssl dhparam -text -5 1024 -out dh
Generating DH parameters, 1024 bit long safe prime, generator 5
This is going to take a long time
.................................+.....................................................
.....+.............................+.....................+.........................
.............................+.........................++*++*++*
root@OpenWrt:/etc/CertWF# dd if=/dev/random of=random bs=1M count=2
0+2 records in
0+2 records out
```

**Figura 4.50. Creación de lista revocación de certificados**

### 4.2.6 Instalación FreeRadius2 en OpenWrt

Para implementar un servidor Radius en nuestro punto de acceso, debemos instalar la aplicación FreeRadius2 junto a los módulos adicionales almacenados en el repositorio de OpenWrt. Para ello, ejecutamos el siguiente comando:

***opkg install freeradius2 freeradius2-mod-always freeradius2-mod-attr-filter freeradius2-mod-attr-rewrite freeradius2-mod-chap freeradius2-mod-detail freeradius2-mod-eap freeradius2-mod-eap-gtc freeradius2-mod-eap-md5 freeradius2-mod-eap-mschapv2 freeradius2-mod-eap-peap freeradius2-mod-eap-tls freeradius2-mod-eap-ttls freeradius2-mod-exec freeradius2-mod-expiration freeradius2-mod-expr freeradius2-mod-files freeradius2-mod-ldap freeradius2-mod-logintime freeradius2-mod-mschap freeradius2-mod-pap***

Editamos el archivo de ejecución del servidor radius **/etc/init.d/radiusd** cambiando la dirección IP en la que se ejecutará el servidor Radius (en nuestro caso será la dirección IP localhost).

Cambiamos la línea:

**IPADDR=$(ifconfig br-lan | sed -n 's/.*dr:\(.*\)Bc.*/\1/p')**

Por la siguiente:

**IPADDR=127.0.0.1**

```
cristianmartinezcandel — ssh — 122×23
#!/bin/sh /etc/rc.common
# Copyright (C) 2006 OpenWrt.org
START=50

DEFAULT=/etc/default/radiusd
LOG_D=/var/log
RUN_D=/var/run
PID_F=$RUN_D/radiusd.pid
RADACCT_D=/var/db/radacct
#IPADDR=$(ifconfig eth0 | sed -n 's/.*dr:\(.*\)Bc.*/\1/p')
IPADDR=127.0.0.1
```

**Figura 4.51. Configuración de ejecución del servidor Radius**

Más tarde, deberemos editar los archivos de configuración Radius (**radiusd.conf, eap.conf, clients y users).**

## Archivo eap.conf

Para ejecutar el servidor Radius serán necesarios los archivos creados anteriormente (certificado del servidor, certificado de la autoridad certificadora, clave privada, contraseña de clave privada, etc). Para añadir estos archivos, será necesario informar al servidor Radius del directorio en el que se encuentran almacenados. Para ello, editamos el documento eap.conf mediante el siguiente comando:

***vi /etc/freeradius2/eap.conf***

```
tls {
        #
        #  These is used to simplify later configurations.
        #
        certdir = /etc/CertWF
        cadir = /etc/CertWF

        private_key_password = wefree
        private_key_file = ${certdir}/ClaveServidor.key

        #  If Private key & Certificate are located in
        #  the same file, then private_key_file &
        #  certificate_file must contain the same file
        #  name.
        #
        #  If CA_file (below) is not used, then the
        #  certificate_file below MUST include not
        #  only the server certificate, but ALSO all
        #  of the CA certificates used to sign the
        #  server certificate.
        certificate_file = ${certdir}/CertificadoServidor.pem
```

**Figura 4.52. Archivo de configuración eap.conf (I)**

```
#  This parameter is used only for EAP-TLS,
#  when you issue client certificates.  If you do
#  not use client certificates, and you do not want
#  to permit EAP-TLS authentication, then delete
#  this configuration item.
CA_file = ${cadir}/CertCA.pem

#
#  For DH cipher suites to work, you have to
#  run OpenSSL to create the DH file first:
#
#       openssl dhparam -out certs/dh 1024
#
dh_file = ${certdir}/dh
random_file = ${certdir}/random
```

**Figura 4.53. Archivo de configuración eap (II)**

## Archivo radiusd.conf

Editamos la sección "listen" para indicarle el tipo de mensaje, que atenderá, la dirección IP en la que escuchará (empleamos * para filtrar todas las posibles IP) y el puerto (indicando el puerto 0, tomará por defecto el número de puerto 1812).

Por otro lado, editamos la sección "log" para ver los mensajes generados en los registros de usuarios.

***vi /etc/freeradius2/radiusd.conf***

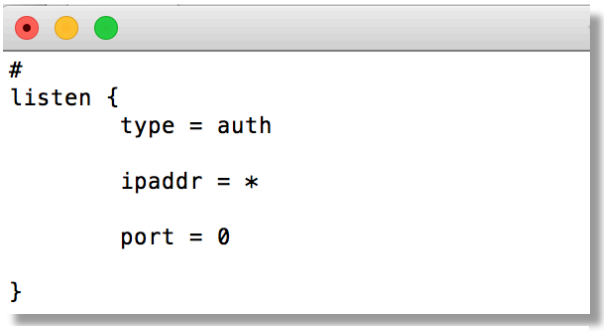



**Figura 4.54. Archivo de configuración radiusd.conf (I)**

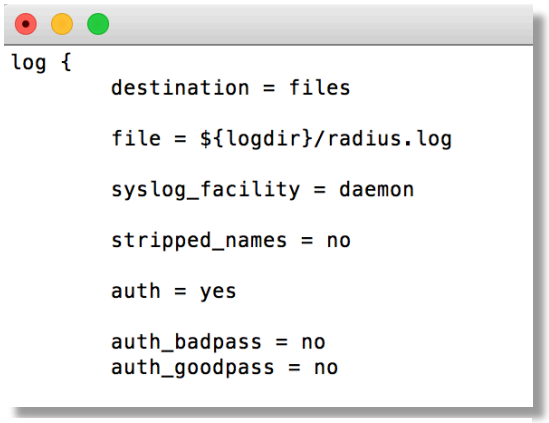



**Figura 4.55. Archivo de configuración radiusd.conf (II)**

## Archivo clients.conf

Editamos el archivo de configuración clients.conf en el que debemos editar el cliente localhost añadiendo la clave del servidor radius mediante la sentencia "secret".

***vi /etc/freeradius2/clients.conf***


**Figura 4.56. Archivo de configuración clients.conf**

## Archivo users

Editamos el archivo de configuración users en el que deberemos agregar los usuarios que tendrán acceso a la red. Dichos usuarios se ingresarán de forma automática mediante un script el cual veremos con posterioridad.

***vi /etc/freeradius2/users***

Añadimos un usuario de prueba para comprobar el correcto funcionamiento del servidor Radius de la siguiente forma:

**upct Cleartext-Password := "upct"**

Mediante la sentencia $INCLUDE añadimos el archivo **users.other** que crearemos más tarde, el cual se actualizará periódicamente con los datos de usuarios almacenados en el servidor.

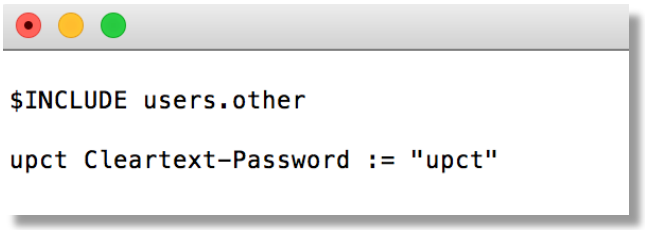



**Figura 4.57. Archivo de configuración users**

Para poder activar en OpenWrt la seguridad WPA2-Enterprise debemos desinstalar el paquete wpad-mini que viene por defecto e instalar el paquete completo denominado wpad.

Para ello ejecutamos:

***opkg remove wpad-mini***
***opkg install wpad***

Una vez instalado dicho paquete, podemos acceder mediante navegador web a nuestro punto de acceso mediante la IP del mismo para comprobar que podemos asignar una seguridad WPA2-Enterprise en la red inalámbrica. Para ello, en la pestaña "Network" > "WiFi" en la sección "Interface Configuration" nos dirigimos a la pestaña "Wireless Security" en la que seleccionaremos el tipo de seguridad que tendrá nuestra red Wifi. En el campo "Encryption" seleccionamos "WPA2-EAP" que hace referencia al protocolo WPA2 Enterprise. Una vez seleccionado, aparecerán otros campos como:

**Cipher:** Tipo de cifrado.

**Radius-Authentication-Server:** Dirección del servidor Radius al que conectará el punto de acceso.

**Radius-Authentication-Port:** Puerto del servidor Radius (Por defecto, 1812).

**Radius-Authetication-Secret:** Clave del servidor Radius.

En nuestro caso, completamos los campos como sigue:

**Cipher:** auto.

**Radius-Authentication-Server:** 127.0.0.1

**Radius-Authentication-Port:** 1812

**Radius-Authetication-Secret:** wfradius

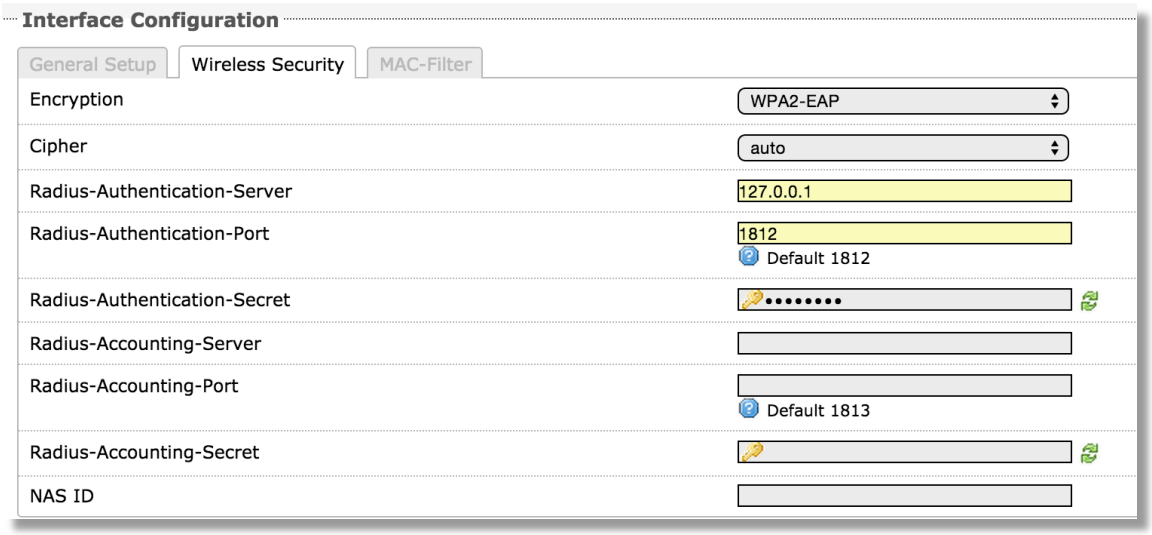



**Figura 4.58. Configuración de seguridad de red inalámbrica**

Para guardar los cambios, hacemos click en **Save & Apply**.

Una vez configurados los archivos, el servidor radius se ejecutará por defecto junto al proceso de inicio de nuestro punto de acceso.

Ahora podremos acceder a la red mediante el usuario de prueba creado anteriormente (upct/upct).

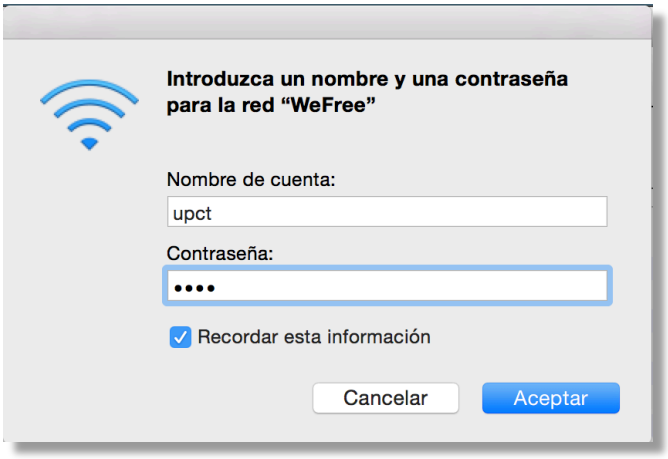



**Figura 4.59. Comprobación de conexión con WeFree**

### 4.2.7 Añadir usuarios activos

Como hemos mencionado anteriormente, utilizaremos un script para añadir de forma autónoma los usuarios obtenidos del servidor central. Dichos usuarios estarán en el servidor almacenados en un archivo encriptado el cual obtendremos mediante la aplicación **wget**. Una vez descargado el archivo al punto de acceso, desciframos el archivo con la herramienta **openSSL** y pasaremos el contenido al archivo de usuarios **users.other**.

Con este script también actualizaremos los datos almacenados en el servidor central sobre el punto de acceso (IP y MAC) pasando dichos parámetros a la página PHP creada (**CompIPMAC.php**) mediante la URL.

Para que los usuarios tengan validez, deberemos recargar el servidor Radius.

Este script será añadido al **cron** de OpenWrt, por ello, el inicio de mismo debe contener la siguiente cabecera:

**#!/bin/sh**

Para crear el script, ejecutamos:

***vi actualiza_ip.sh***

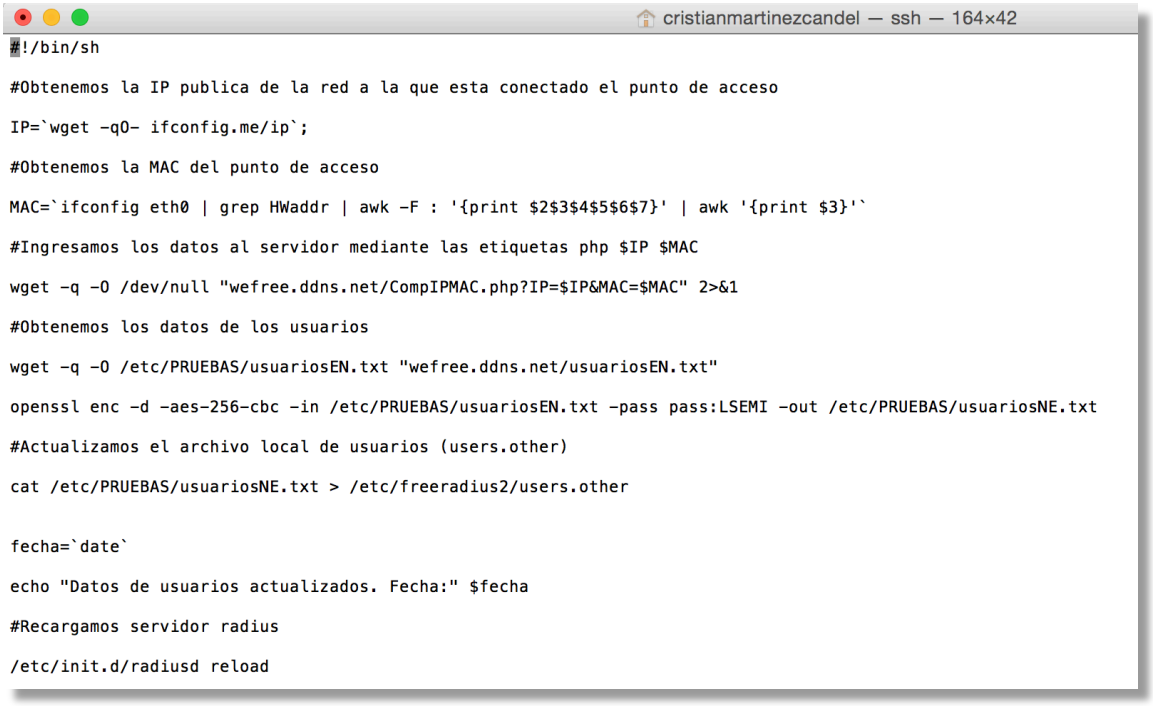



**Figura 4.60. Contenido del script actualiza_ip.sh**

Para actualizar periódicamente los datos de los usuarios habilitados, será necesario ejecutar el script creado de forma periódica (cada 10 minutos). Para ello, creamos un archivo llamado root en el directorio **/etc/crontabs**

***vi /etc/crontabs/root***

En el archivo creado, agregamos la siguiente sentencia:

***/10 * * * * /etc/PRUEBAS/actualiza_ip.sh > /etc/PRUEBAS/act_usuarios.txt**

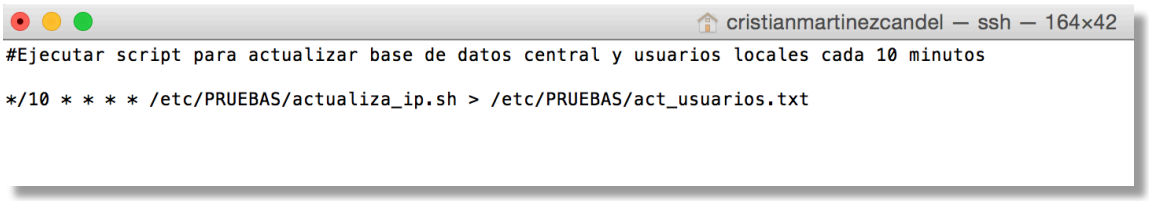



**Figura 4.61. Contenido del archivo /etc/crontabs/root**

Se ha direccionado la salida del script al archivo **/etc/PRUEBAS/act_usuarios.txt** para poder comprobar la fecha de la última actualización de los usuarios.

Una vez editado y guardado, hacemos ejecutable el fichero mediante el comando:
***chmod u+x /etc/crontabs/root***

Si accedemos al archivo **act_usuarios.txt** mencionado anteriormente, podremos observar la fecha de actualización de usuarios:

***vi /etc/PRUEBAS/act_usuarios.txt***

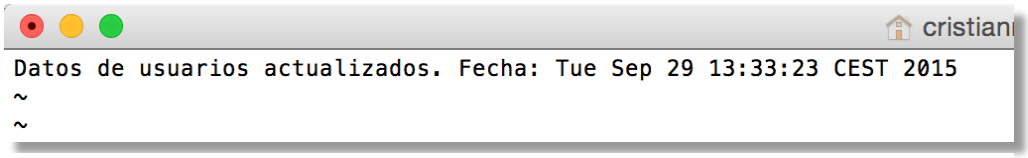



**Figura 4.62. Contenido del archivo act_usuarios.txt**

Si accedemos al archivo de usuarios **users.other** del directorio **/etc/freeradius2** podremos observar los usuarios habilitados para el uso del servicio.
**vi /etc/freeradius2/users.other**

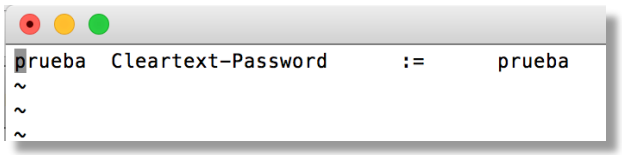



**Figura 4.63. Contenido del archivo users.other**

## 4.2.8 QOS en OpenWrt

Se ha decidido que el ancho de banda a compartir sea de aproximadamente un 30% del total. Esto se consigue aplicando técnicas de calidad de servicio (QOS). En nuestro caso, para implementar dicha calidad de servicio será necesario instalar en los puntos de acceso el paquete **qos-scripts**

Para ello, ejecutaremos:

***opkg update***
***opkg install qos-scripts***

Para configurar la calidad de servicio, debemos editar el archivo **/etc/config/qos**

***vi /etc/config/qos***

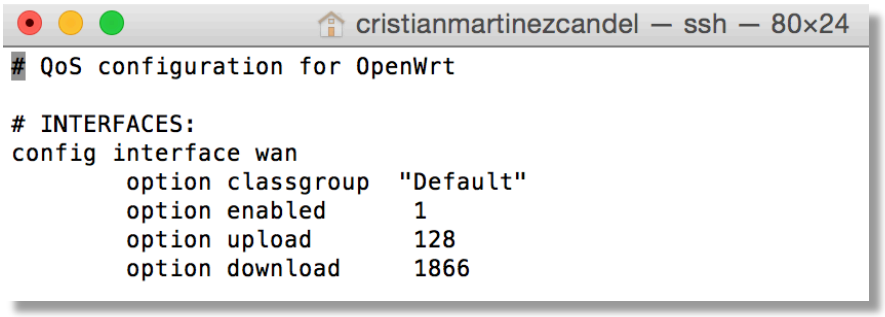



**Figura 4.64. Archivo de configuración qos**

En dicho archivo encontramos los siguientes parámetros:
**Classgroup:** Especifica la clase de grupo usada en la interfaz.
**Enabled:** Habilita o deshabilita la calidad de servicio (1 o 0 respectivamente).
**Upload:** Límite de subida medido en kb/s.
**Download:** Límite de bajada medido en kb/s.

Para activar la calidad de servicio, debemos asegurarnos de que la opción **enabled** tenga valor 1. En nuestro caso la opción **upload** siempre tendrá el mismo valor (128 kb/s) mientras que la opción **download** tendrá un valor dinámico dependiendo del ancho de banda de la red en la que esté conectado el punto de acceso. Para calcular dicho valor utilizaremos un script bash denominado script_qos.sh.
Para crearlo, utilizaremos el editor vi.

***vi script_qos.sh***

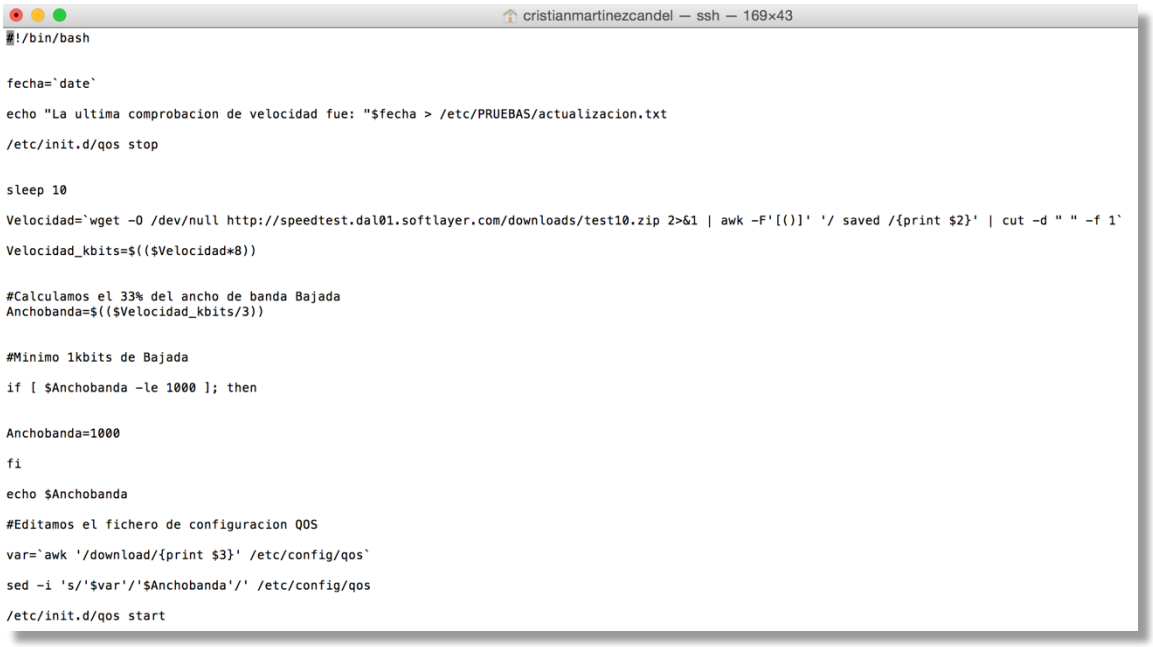



**Figura 4.65. Contenido de script_qos.sh**

El procedimiento de este script es el siguiente:

Mediante los comandos **wget** y **awk** obtenemos la velocidad en KB/s. Dado que las unidades de medida del parámetro **download** son kb/s, debemos convertir el valor obtenido en kb/s. Para evitar que el ancho de banda sea demasiado bajo, forzamos el valor del ancho de banda a 1000kb/s. Más tarde, incluimos el valor obtenido en el archivo de configuración **/etc/config/qos** y reiniciamos el servicio.

Para activar las reglas de calidad de servicio, será necesario ejecutar el comando ***etc/init.d/qos enable*** el cual habilitará el servicio.

Para ejecutar el script al iniciar el punto de acceso, debemos crear un nuevo script con el contenido del script mencionado (**script_qos.sh**) en el directorio **/etc/init.d** con una función de inicio (función start()).

Para ello, creamos el archivo:

***vi /etc/init.d/mi_script***

Y copiamos el contenido dentro de la función **start()**.

Como observamos en la imagen, la cabecera para que sea ejecutable al inicio es la siguiente:

**#!/bin/sh /etc/rc.common**

También podemos observar una variable denominada START que indica la posición de inicio mediante un valor de tiempo.

**START = 50**

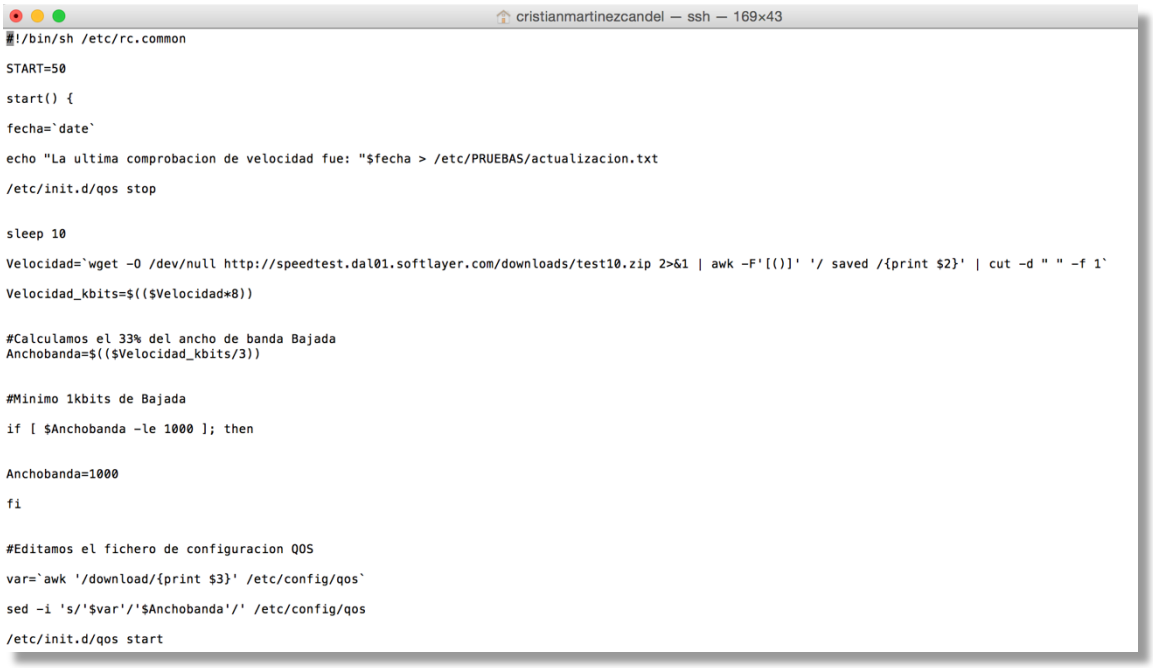



**Figura 4.66. Contenido del archivo mi_script**

Damos permisos de ejecución al script creado y habilitamos el script:

***chmod +x /etc/init.d/mi_script***

***/etc/init.d/mi_script enable***

Podremos comprobar la fecha de la última ejecución del script en el documento **/etc/PRUEBAS/act_qos.txt**

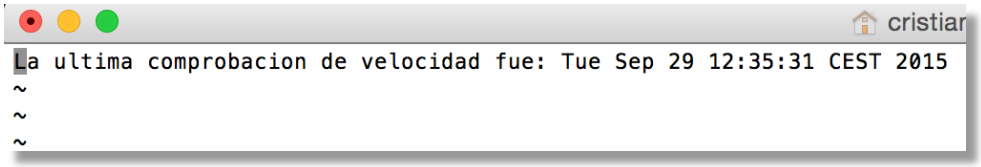
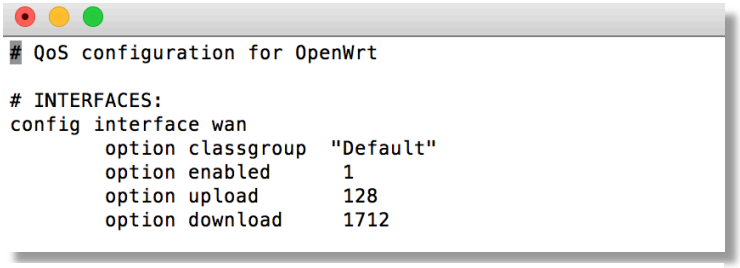



**Figura 4.67. Contenido del archivo act_qos.txt**

Por otro lado, observaremos el valor **download** del archivo **/etc/config/qos:**

*vi /etc/config/qos*



```
# QoS configuration for OpenWrt

# INTERFACES:
config interface wan
        option classgroup  "Default"
        option enabled      1
        option upload       128
        option download     1712
```

**Figura 4.68. Contenido del archivo qos (II)**

Como podemos comprobar, al ejecutar el script que comprueba la calidad de servicio, el valor del campo download ha cambiado.

# 5 Conexión al servicio WeFree

Una vez comprobado que los ejecutables funcionan y los usuarios se actualizan correctamente, pasaremos a comprobar la conexión con el usuario obtenido del servidor central. Para ello, hemos utilizado dos métodos de conexión:

1. **Conexión desde Macintosh**
2. **Conexión desde iOS**

## 5.1.1 Conexión desde Macintosh

En nuestro sistema operativo, nos dirigimos al gestor de redes inalámbricas en la que buscamos la red de nuestro servicio (WeFree). Cuando conectemos, obtendremos el siguiente cuadro de diálogo:

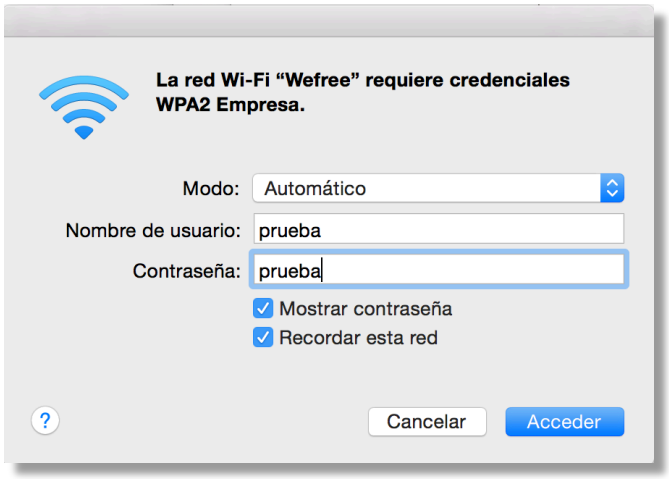



**Figura 5.1. Conexión a Wefree en Macintosh**

Insertamos las credenciales del usuario obtenido:

**Nombre de usuario:** prueba

**Contraseña:** prueba

Ahora, podemos comprobar en nuestro panel de redes inalámbricas como el acceso ha sido satisfactorio:


**Figura 5.2. Comprobación de conexión en Macintosh**

### 5.1.2 Conexión desde iOS

Por otro lado, podemos comprobar que la conexión también tiene efecto desde dispositivos móviles. Para ello, accedemos la red WeFree desde un terminal iPhone insertando las credenciales del usuario obtenido:

**Nombre de usuario:** prueba

**Contraseña:** prueba


**Figura 5.3. Conexión a Wefree desde iOS**

Una vez ingresados los datos, hacemos click en **Conectarse** y obtendremos conexión a la red del servicio WeFree:

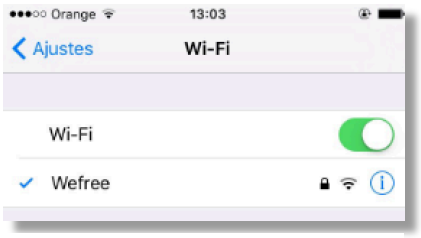



**Figura 5.4. Comprobación de conexión en iOS**

Una vez conectados a la red mediante Wi-Fi, podemos comprobar los valores aplicados mediante un test de velocidad realizado en una aplicación móvil (SPEEDTEST):

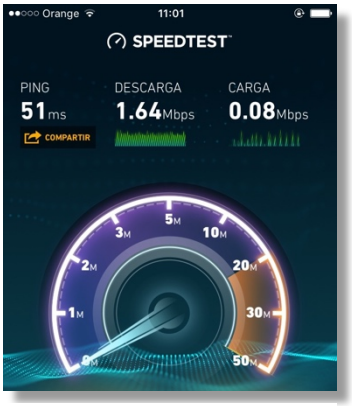



**Figura 5.5. Test de velocidad ejecutado en iOS**

Se puede comprobar como los valores de Descarga y Carga son bastante similares a los configurados en la calidad de servicio (debemos tener en cuenta las pérdidas ocasionadas por la conexión inalámbrica).

# 6 Conclusiones y trabajos futuros

Para el futuro desarrollo del servicio, se llevarán a cabo las siguientes mejoras:

- Implantación de una página web robusta, en la que poder consultar la localización de los puntos de acceso del servicio WeFree y obtener información/ayuda acerca del servicio.
- Mejorar la cobertura del punto de acceso, cambiando el dispositivo usado por otro punto de acceso dotado de mejores características técnicas siempre y cuando sea compatible con el Firmware OpenWrt.
- Crear una base de datos de usuarios en los puntos de acceso para aportar una estructura más clara.
- Controlar el uso y acceso de los usuarios al servicio, limitando el número de accesos por usuario/contraseña.
- Implementar un firewall robusto.
- Analizar y mejorar el procesamiento de colas.

# 7 Bibliografía


**Instituto nacional de estadística:**

http://www.ine.es/ss/Satellite?L=es_ES&c=INESeccion_C&cid=1259925529799&p=1254735110672&pagename=ProductosYServicios%2FPYSLayout&param3=1259924822888

**Raspberyy Pi:**

https://www.raspberrypi.org/downloads/

http://es.engadget.com/2012/08/11/raspberry-pi-model-b-analizado/

**Raspbian:**

https://www.raspbian.org/

**TP-Link WR3020 OpenWrt:**

http://wiki.openwrt.org/toh/tp-link/tl-mr3020

**Openssl:**

https://www.openssl.org/

**Freeradius:**

http://freeradius.org/

http://networkradius.com/doc/FreeRADIUS%20Technical%20Guide.pdf

**Radius:**

http://www.cisco.com/cisco/web/support/LA/102/1024/1024966_32.pdf

**GNU/Linux:**

http://linux.ciberaula.com/articulo/linux_cron/

**MySQL:**

https://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html

**Apache:**

http://wiki.apache.org/httpd/FAQ

**Criptografía:**

Apuntes de la asignatura Seguridad en Redes de Comunicaciones

**Certificados electrónicos:**

http://web.archive.org/web/20150703110000/https://sede.ine.gob.es/ss/Satellite?c=Page&cid=1254734731861&lang=es_ES&p=1254734731861&pagename=SedeElectronica%2FSELayout