ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

Integrando Latch y OpenWRT para el control de acceso

Integrating Latch and OpenWRT for Access Control

Realizado por
Juan Camero Martín
Tutorizado por
Isaac Agudo Ruiz
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO - 2017

Fecha defensa:
El Secretario del Tribunal

Resumen: El proyecto consiste en la creación de un plugin para el firmware libre OpenWRT para routers neutros, que permite gestionar la conexión a internet de dispositivos inalámbricos de forma intuitiva a través de un Smartphone con la aplicación Latch.

Se agregará una capa de seguridad extra para el acceso a internet a través de nuestro router, de forma que podamos evitar el acceso a la red si un atacante consigue sobrepasar las primeras medidas de seguridad como la clave del punto de acceso o un filtro de MAC, usando técnicas de Mac Spoofing, etc.

Dicho objetivo tiene especial importancia en el ámbito de redes de hogar, aunque también es relevante en pequeñas empresas, ya que en la mayoría de los casos los atacantes suelen ser personas, sin conocimientos avanzados, que se dedican a usar software automatizado para obtener acceso a la red.

Dicho perfil de atacante normalmente solo busca el acceso a la red para tener acceso a internet y realizar descargas masivas, pero también puede perjudicar al usuario legítimo si el intruso realiza actividades delictivas desde esa red.

Palabras claves: Seguridad, Informática, WiFi, Redes, inalámbricas, Latch, Eleven Paths, OpenWRT

Abstract: The project involves creating a plugin for OpenWRT that can be installed in neutral routers, which allows you to manage intuitively the wireless Access using a Smartphone with Latch application.

An extra layer of security will be added to access internet through our router, so that we can prevent access to the network if an attacker manages to overcome the first security measures as for instance the access password, filter MAC using Mac Spoofing techniques, and so on

This objective is particularly important in home networks area, but also relevant to small businesses, because in most cases the attackers are usually people without advanced knowledge, using automated software to obtain network access.

This profile of attacker usually only seek access to the network to access the Internet and perform massive downloads, but can also harm the legitimate user if the intruder engaged in criminal activity from that network.

Keywords: Informatic, Security, WiFi, Networks, Wireless, Latch, Eleven Paths, OpenWRT

Tabla de contenido

1. Introducción

Sinopsis:

En este capítulo se hará una introducción al proyecto explicando las motivaciones y la importancia de este desarrollo.

Se explicarán también los objetivos principales de dicho proyecto, así como los posibles escenarios en los que se podría aplicar el proyecto en cuestión.

1.1. Introducción

En el pasado el termino hacker se reservaba únicamente para aquellas personas con talento, conocimiento, inteligencia e ingenuidad que dedicaban su tiempo y recursos en aprender los detalles de los sistemas de computación, así como en intentar extenderlos o sacar provecho de los mismos. Se trataba de un grupo relativamente reducido de personas que destacaban por sus capacidades y logros, ya fueran lícitos o ilícitos.

En la actualidad la cultura hacker ha crecido significativamente y la tecnología está al alcance de prácticamente cualquier interesado, y con el tiempo se han ido creando diversas herramientas que hoy en día se utilizan tanto por los expertos que buscan mejorar la seguridad de los sistemas como por aquellos que buscan romper dicha seguridad. Muchas de estas herramientas están tan automatizadas, y son tan potentes y simples de usar, que las hacen realmente peligrosas en las manos de cualquiera, tenga o no realmente conocimientos sobre el uso de estas. A lo anterior hay que sumar que Internet es una fuente de recursos que facilita adquirir dichas herramientas e incluso guías básicas o avanzadas de uso.

Si entendemos como Hacker al individuo que usa sus habilidades y recursos para invadir sistemas informáticos ajenos, dejamos un hueco en la definición, pues no es tan simple, ya que en la actualidad un Hacker puede ser un niño travieso, un joven delincuente o un gran profesional.

Cada día más y más hogares, así como pequeñas y medianas empresas, son objetivos potenciales de estos hackers que buscan obtener beneficios, ya sean financieros o de otro tipo.

Con este trabajo de investigación y desarrollo se busca intentar añadir una capa extra de seguridad a una de las áreas de seguridad más conocidas, las redes inalámbricas.

Dicha área de la seguridad informática tiene especial importancia ya que es una de las formas “más fáciles” de acceder a una empresa u hogar, existen multitud de aplicaciones comúnmente llamadas de “botón gordo” con las que no necesitas casi conocimientos para su uso, y esta área es normalmente donde empiezan a aprender todos aquellos que sienten cierta atracción por este mundo.

Las grandes empresas tienen, por lo general, robustos sistemas de seguridad en esta área, que han obtenido realizando grandes inversiones económicas en hardware y software propietario.

Por otro lado, los hogares y las PYMES no tienen acceso a esos sistemas de protección dejándolos desprotegidos ante el ataque de cualquiera. Son en estos últimos en los que nos centramos a la hora de crear esta protección, ofreciendo una capa extra de seguridad contra los individuos que quieren atacar y acceder a la red local de los mismos.

1.2. Objetivos

- Crear y establecer una nueva capa de protección para redes inalámbricas que actúe junto a las capas ya existentes añadiendo protección extra.

- Conseguir que esta nueva capa de protección pueda ser instalada en el mayor número de dispositivos posibles, independientemente de su antigüedad, hardware o fabricante, mediante el uso de sistemas operativos potentes y libres creados específicamente para actuar de router, no limitando su uso a estos últimos.

- Diseñar el sistema de forma, que una vez esté instalado y configurado, pueda ser utilizado de forma sencilla e intuitiva por cualquier persona, tenga o no conocimientos de informática, a través de un dispositivo móvil.

- Conseguir que el sistema pueda auto-escalar para que la capa de protección pueda ser utilizado tanto en una casa con un solo dispositivo inalámbrico como en una gran empresa con cientos de conexiones simultaneas.

1.3. Posibles escenarios de aplicación

En esta sección se mostrará algunos ejemplos de posibles usos para la aplicación.

- Administración de la red wifi de una casa, donde el padre o la madre son administradores, y mediante el plugin y el bloqueo de Latch por franja horaria pueden establecer diferentes horarios de conexión para los dispositivos de los hijos.

- Para una escuela o academia. Donde el profesor puede controlar el acceso al wifi de los estudiantes por horas, dispositivos, etc

- Para un bar en el que los camareros controlen el acceso Wi-Fi de los clientes.

- Para un negocio, donde el administrador de la red pueda controlar el acceso, y permitir o no conectar a clientes potenciales en reuniones.

- Para un hotel rural donde cada apartamento y cada router tenga su propia aplicación Latch (O usando una común) y se pueda dar o denegar acceso a dispositivos clientes de habitaciones que hayan pagado previamente

- Controlar el acceso de la red de una casa y detectar si alguien más conoce la contraseña de tu red y está conectando para acceder gratis a internet.

Básicamente se puede usar el sistema en cualquier escenario en el que se necesite el control del acceso a la red a través de Wi-Fi.

2. Tecnologías

Sinopsis:

En este capítulo se hablará de las tecnologías que se van a utilizar en el proyecto, dividiéndola en distintas categorías, y se hará una explicación de las características de cada una de ellas, de su historia, de su funcionamiento, etc.

Se hará especial mención a OpenWRT y a Latch que serán las tecnologías base sobre las que se sustentará el proyecto.

2.1. Tecnologías WEB

2.1.1. HTML

Sigla en inglés de ***HyperText Markup Language*** (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML [1]) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del *World Wide Web Consortium* (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Se considera el lenguaje web más importante siendo su invención crucial en la aparición, desarrollo y expansión de la World Wide Web (WWW). Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

2.1.2. JavaScript

JavaScript [2] (abreviado comúnmente **JS**) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo, en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

2.1.3. CSS

Hojas de estilo en cascada (o CSS [3], siglas en inglés de *Cascading Stylesheets*) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. También permite aplicar estilos no visuales, como las hojas de estilo auditivas.

Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web, y GUIs para muchas aplicaciones móviles

2.2. Lenguajes de programación / Script

2.2.1. Lua

Lua [4] es un lenguaje de programación extensible diseñado para una programación procedimental general con utilidades para la descripción de datos. También ofrece un buen soporte para la programación orientada a objetos, programación funcional y programación orientada a datos.

Lua ha sido usado en muchas aplicaciones comerciales y no comerciales, cuyo número incrementa cada año. Entre las aplicaciones destacadas se encuentra Wireshark que añadió Lua a su proyecto como lenguaje de prototipado y programación de scripts

Gracias a su sencillez, como puede verse en el "Código 1. Sintaxis de Lua", Lua pudo ser portado a múltiples arquitecturas fácilmente. Lo más sorprendente es que existen ports para PSP y Wii, siendo por ahora el primer port el más antiguo y exitoso, mientras que el de Wii deriva de éste.

Debido a que Lua compilado es pequeño (en la mayoría de los casos), veloz y tiene una licencia permisiva ha ganado seguidores entre los desarrolladores de videojuegos. El motor gráfico de Crytek, CryEngine, está programado en Lua, además de en C++. Empresas como VALVe, EA Games y Bethesda decidieron trasladar sus bases de C++ a Lua dado que este es más compatible gráficamente con kernel.

```
function factorial(n)
   if n == 0 then
      return 1
   return n * factorial(n - 1)
end
```

Código 1. Sintaxis de Lua – Cálculo de factorial

2.2.2. Bash

BASH [5] es un shell de Unix (intérprete de comandos de Unix) escrito para el proyecto GNU. Su nombre es un acrónimo de bourne-again shell (otro shell bourne); haciendo un juego de palabras (born-again significa renacimiento) sobre el Bourne shell (sh), que fue uno de los primeros shells importantes de Unix.

BASH es el shell por defecto en la mayoría de sistemas GNU/Linux, además de Mac OS X Tiger, y puede ejecutarse en la mayoría de los sistemas operativos tipo UNIX. También se ha portado a Microsoft Windows por el proyecto Cygwin.

La mayoría de los shell scripts (guiones de órdenes) Bourne pueden ejecutarse por bash sin ningún cambio, con la excepción de aquellos scripts de shell Bourne que hacen referencia a variables especiales de Bourne o que utilizan una orden interna de Bourne.

Cuando se utiliza como un intérprete de órdenes interactivo, bash proporciona autocompletado de nombres de programas, nombres de archivos, nombres de variables, etc., cuando el usuario pulsa la tecla TAB.

Bash permite interactuar directamente con comandos, variables, el sistema de ficheros, etc., del sistema operativo, como se puede ver en el "Código 2. Sintaxis de Bash".

```
#!/bin/bash
#### Comprueba si el paquete wget está instalado
if opkg list | grep wget > /dev/null
then
    echo 'Found wget'
else
    echo 'Updating package list...'
    opkg update > /dev/null
    echo 'Installing wget...'
    opkg install wget > /dev/null
fi
```

Código 2. Sintaxis de Bash – Hola mundo

2.2.3. Python

Python [6] es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Entre otras muchas características que facilitan la legibilidad del código se encuentra el sistema de indentación del mismo, por el que se evita el uso de llaves. La indentación consiste en mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para así separarlo del margen izquierdo y mejor distinguirlo del texto adyacente. Esta característica puede verse en el "Código 3. Sintaxis de Python".

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

```
datos = enum_letras('Oscuridad para asuntos oscuros')
for letras,num in datos.iteritems():
    cont = 'z' if (num < 2) else 'ces'
    print 'La letra %s se repite %s ve%s' % else (letra, num, cont)
```

Código 3. Sintaxis de Python – Cantidad de veces que se repite las letras

2.3. Persistencia

2.3.1. SQLite

SQLite [7] es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, **SQLite** permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

A diferencia de los motores de base de datos completos como MySQL [8], SQLite está orientado a fichero mientras que los motores de base de datos están orientados a servicios, lo cual significa que MySQL es un servidor que atiende peticiones que se le realizan a un puerto e IP determinados mientras que SQLite contiene toda la información en un fichero local.

A diferencia de los archivos de configuración o texto, SQLite permite trabajar con la información de forma rápida y sencilla, permitiendo encontrar la información que se quiere de forma muy rápida, y en general trabajar con toda la información manteniendo la integridad y seguridad de la misma. Además, SQLite permite el acceso concurrente a datos mientras que en los archivos de configuración se produce un bloqueo previo a la modificación de los datos incluidos en los mismos.

2.4. OpenWRT

OpenWRT [9] es un firmware basado en una distribución de Linux empotrada en dispositivos tales como routers personales.

El desarrollo de OpenWRT fue impulsado inicialmente gracias a la licencia GPL, que obligaba a todos aquellos fabricantes que modificaban y mejoraban el código, a liberar éste y contribuir cada vez más al proyecto en general. Poco a poco el software ha ido creciendo y se encuentran características implementadas que no tienen muchos otros fabricantes de dispositivos comerciales para el sector no profesional, tales como QoS, VPN y otras características que dotan a OpenWRT de un dispositivo realmente potente y versátil, apto para utilizar los hardware donde corre OpenWRT no sólo para utilizarlos como routers, sino como servidores de archivo, nodos P2P, servidores de Webcams, firewall o puertas de acceso VPN.

OpenWRT es una distribución GNU / Linux altamente extensible para dispositivos embebidos (típicamente routers inalámbricos). A diferencia de muchas otras distribuciones para estos routers, OpenWRT se construye desde cero para ser un sistema operativo completo y fácilmente modificable para su router. En la práctica, esto significa que usted puede tener todas las características que necesita sin ninguno de los programas no deseables, impulsado por un kernel de Linux que es más reciente que la mayoría de las otras distribuciones.

En lugar de intentar crear un firmware único y estático, OpenWRT proporciona un sistema de archivos totalmente escribible con administración de paquetes opcional. Esto le libera de las restricciones de la selección de aplicaciones y la configuración proporcionada por el proveedor y le permite utilizar paquetes para personalizar un dispositivo incrustado para adaptarse a cualquier aplicación. Para los desarrolladores, OpenWRT proporciona un marco para construir una aplicación sin tener que crear una imagen completa de firmware y distribución alrededor de ella. Para los usuarios, esto significa la libertad de personalización completa, permitiendo el uso de un dispositivo incrustado en formas que el proveedor nunca ha previsto.

OpenWRT se ha establecido por mucho tiempo como la mejor solución de firmware de su clase. Es muy superior a otras soluciones integradas en rendimiento, estabilidad, extensibilidad, robustez y diseño. OpenWRT es libre y Open-Source, es de acceso fácil y libre, y creado por la comunidad.

OpenWRT se configura mediante una interfaz de línea de comandos (shell) o una interfaz web (LuCI). Existen unos 3500 paquetes opcionales de software disponibles para la instalación a través del sistema de gestión de paquetes opkg [11].

OpenWRT puede ejecutarse en varios tipos de dispositivos, incluyendo routers CPE, pasarelas residenciales, teléfonos inteligentes, ordenadores de bolsillo (por ejemplo, Ben NanoNote) y portátiles. También es posible ejecutar OpenWRT en ordenadores personales, que se basan más comúnmente en la arquitectura x86.

Como podemos ver en la Tabla de Hardware de OpenWRT, este es compatible con más de 1100 diferentes hardwares. También existe otra gran cantidad de hardware que ejecuta OpenWRT que no fue oficialmente confirmado y no se muestra en dicha tabla.

Aunque las instrucciones de instalación de OpenWRT pueden diferir en diferentes dispositivos, la mayoría de ellas solo necesitan descargar la imagen correcta e instalarla con la utilidad de actualización oficial de su sistema.

En resumen, OpenWRT es altamente compatible con un gran número de hardware diferente, fácilmente instalable por cualquier persona, y amplía las capacidades del router aumentando el rendimiento del mismo, especialmente en seguridad. Todo esto hace que el desarrollo del plugin LatchOpenWRT, este proyecto, se pueda utilizar en miles de dispositivos diferentes de forma fácil, rápida y eficiente

2.5. Latch

2.5.1. ¿Qué es Latch?

Latch [10] es un servicio en la nube para que los usuarios pueden crear pestillos digitales para poner a sus identidades digitales, y que les permita poner ON/OFF las cuentas con tan solo hacer clic en la identidad que quiere desbloquear o bloquear cuando la vaya a utilizar.

2.5.2. Control de servicios digitales con Latch.

Las contraseñas, el sistema de autenticación más antiguo, son un problema de seguridad con el que tenemos que aprender a convivir. Un segundo factor de autenticación, biometría, gestores de contraseñas... todavía no se ha dado con la fórmula mágica para que el usuario deje de usar contraseñas simples, las reutilice, o las escriba en un papel. Y Latch no es esa solución. Aun en el caso de usuarios avanzados con buenas prácticas en el uso de credenciales, es posible que la contraseña sea robada. El malware que se dedica profesionalmente al robo de credenciales es "habitual" desde hace tiempo. Pero incluso los usuarios más cuidadosos con sus sistemas pueden sufrir problemas si las bases de datos de terceros son vulneradas, y sus credenciales acaban en manos de atacantes. Y Latch, tampoco es la solución para este problema.

Latch no sustituye a las contraseñas, complementa y fortalece cualquier sistema de autenticación.

La aproximación de Latch es diferente. Evitar que los datos de autenticación sean robados es muy complejo. Sin embargo, sí que es posible que el usuario tome el control de sus servicios digitales, y reduzca su tiempo de exposición. "Apagar" el acceso a tu correo, la posibilidad de usar tu tarjeta de crédito, o realizar transacciones online, cuando no esté usando. Bloquearlos incluso cuando se conocen las contraseñas adecuadas. Latch otorga la posibilidad de que sea el propio usuario quien decida cuándo se puede acceder a sus cuentas o usar ciertos servicios. No el proveedor. Ni por supuesto, el atacante.

Latch va a permitir que, incluso si un atacante utiliza un usuario y contraseña robado, una tarjeta de crédito, o cualquier sistema de que necesite una aprobación, al atacante le sea imposible utilizar esta información para hacerse pasar por el usuario fuera de un intervalo definido. En resumen, es posible que el usuario y contraseña que se utilizan para acceder a cualquier servicio, sean (pulsando un botón) solo válidos durante esos pocos segundos en los que es el propio usuario quien los introduce en el sistema.

2.5.3. El esquema Latch

Aunque hemos hablado de contraseñas, Latch se presenta en realidad como un servicio diseñado para proteger los procesos definidos por cualquier proveedor de servicios para interactuar con sus usuarios. La naturaleza y el uso que se pueda dar a estos procesos son independientes de la protección que puede ser proporcionada por Latch.

La idea fundamental sobre la que se estructura esta protección es la limitación del tiempo de exposición del que dispone un atacante para intentar aprovechar alguno de estos procesos en su propio beneficio. El usuario decidirá si sus cuentas están en ON u OFF e incluso las acciones que se pueden realizar desde ellas. Esto supone que se logre la reducción de la ventana de tiempo en la que podría suceder un ataque, asociando a cada operación un control externo. El proveedor de servicio consultará a Latch, véase en la Figura 1, el estado con el que el usuario ha decido configurar la ejecución de esta operación para un momento determinado.

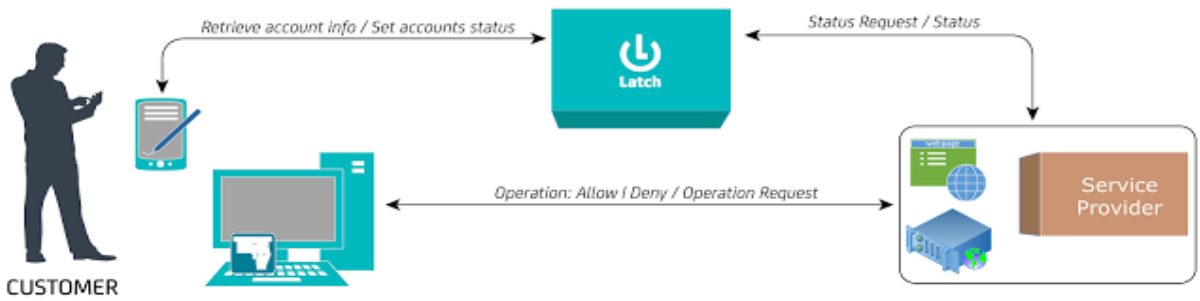


Figura 1. Esquema de funcionamiento de Latch

En la Figura 1, un cliente que solicita la ejecución de una operación a un proveedor de servicios puede obtener la confirmación de que esta operación se ha realizado de forma satisfactoria o que se ha denegado.

La configuración del estado definido por una operación concreta se realiza de a través de un canal alternativo (y considerado más "seguro" que el dispositivo habitual), así que cualquier intento de utilizar la operación que ha sido bloqueada por el propio cliente se puede identificar como una anomalía. Su existencia podría implicar a su vez que el usuario que solicita la ejecución de la operación no sea quien está afirmando ser y se haya detectado así un intento de fraude.

En la práctica, el usuario solo necesitará utilizar su smartphone para "activar" o "desactivar" los servicios pareados con Latch.

Para ello es necesario:

- La creación de una cuenta de usuario en Latch por parte del cliente. Esta cuenta será la utilizada por el usuario para configurar sus estados en las operaciones (establecer a ON u OFF sus cuentas con los proveedores).

- El pareado de su cuenta habitual con el proveedor (la cuenta de correo, o de un blog, por ejemplo) que se desea controlar, con la cuenta de usuario de Latch. Este paso permitirá a Latch sincronizarse con el proveedor de servicio para ofrecerle la respuesta adecuada (configurada por el usuario) dependiendo del proveedor de servicio y la operación concreta dentro de él. Por supuesto, el proveedor de servicio debe ser compatible con Latch. Esto permite a los usuarios que lo decidan, aprovechar esta ventaja ofrecida por el proveedor, sin que este deba imponer la solución a todos sus clientes.

La integración es sencilla, y el proveedor puede mejorar sustancialmente la seguridad que ofrece a sus usuarios, cediéndoles un mayor control de su seguridad y, por tanto, de su identidad en la red.

3. Análisis y Diseño del sistema

Sinopsis:

En este capítulo se busca realizar un análisis de lo que debe realizar el sistema mediante una historia de usuario y una recolección de requisitos. Así mismo se explicará la metodología de desarrollo a utilizar.

También se realizará un diseño inicial basado en el análisis anterior en el que se establecerán los casos de uso y la descripción del esquema de funcionamiento del sistema.

3.1. ¿Por qué desarrollar y usar “Latch OpenWRT”?

Como comentamos anteriormente, los hogares y las pequeñas empresas normalmente no tienen acceso a costosos sistemas de protección de red, y las personas que están empezando a aprender seguridad informática a menudo atacan dichas redes con el propósito de aprender o acceder a Internet de forma gratuita.

Latch OpenWRT debe de poder resolver este problema agregando una nueva capa de seguridad al sistema. Lo importante de este sistema, es que gracias a OpenWRT, debe de ser fácilmente instalable y accesible para la mayoría de los usuarios de forma gratuita. La instalación y la configuración completa se debe de poder realizar en pocos minutos.

Gracias a la tecnología Latch, administrar el acceso de estos dispositivos debería ser tan simple como dar un par de clics y poner un "candado" o no al dispositivo que desea bloquear o desbloquear. Esto también debería permitir al administrador de red administrar los dispositivos inalámbricos que tienen acceso a los mismos independientemente de si el administrador está conectado o no a esa red.

Además, el sistema debe expulsar completamente de la red los dispositivos que están bloqueados en la aplicación Latch, reduciendo también el vector de ataque que tienen los atacantes.

Siendo una nueva capa de seguridad no debe de interferir con otros métodos de seguridad existentes, como el IP estático o el filtro de dirección MAC, dejando a Latch OpenWRT como la última capa de protección en caso de que los otros sistemas fallen o no estén configurados.

Latch OpenWRT también tiene que poder ayudar a detectar casos más complejos de intrusión en la red de forma pasiva. Si un atacante con más conocimientos entiende cómo funciona esta capa de protección, y duplica una dirección MAC que actualmente tiene permisos, sería fácilmente detectado por el administrador, y además el acceso que tendría el usuario sería por tiempo limitado.

Esto se debe a que, independientemente de si un dispositivo conectado tiene permisos de acceso a la red, cuando el dispositivo se desconecta de la red se debe volver a darle esos permisos una vez que se vuelva a conectar, es decir, deben de borrarse automáticamente. (si no es un dispositivo permanente).

Si usted desconecta su dispositivo de la red y después de los segundos transcurridos (el tiempo de recarga) el dispositivo no desaparece de su aplicación de Latch, significa que seguramente alguien está conectado duplicando su MAC, y en este caso sólo tiene que bloquear el dispositivo en la aplicación Latch y cambiar la contraseña al router para dar más trabajo al atacante.

Incluso si no se da cuenta de que el dispositivo sigue conectado a pesar de que lo desconectó, en el momento en que el atacante se desconecta e intenta conectarse posteriormente se tiene que alertar al administrador (si no está conectado), ya que se deben de solicitar los permisos de acceso de nuevo.

Tarde o temprano el atacante terminará aburriéndose a menos que tenga un objetivo más importante que tener acceso libre a Internet.

Gracias al sistema se debería de evitar uno de los perfiles atacantes más comunes y grandes, aquellas personas que usan herramientas automáticas sin ninguna idea de cómo funcionan. Aunque las aplicaciones dan la clave, si el sistema les expulsa no sabrán cómo reaccionar y terminarán buscando otro objetivo.

3.2. Metodología de desarrollo

Para este desarrollo se van a utilizar dos metodologías de desarrollo de software.

Se va a utilizar inicialmente la metodología de prototipado (Figura 2) utilizando el lenguaje Bash para comprobar lo más rápido posible que la idea de este proyecto es viable y compatible con OpenWRT.

Esto permitirá hacer además un primer acercamiento a la idea del proyecto de forma rápida, pero Bash es un lenguaje que genera un código bastante sucio e ilegible, no tenemos una API oficial de Latch para trabajar con Bash y habrá que realizar las peticiones web creándolas manualmente, añadiéndoles las cabeceras, la firma, etc.

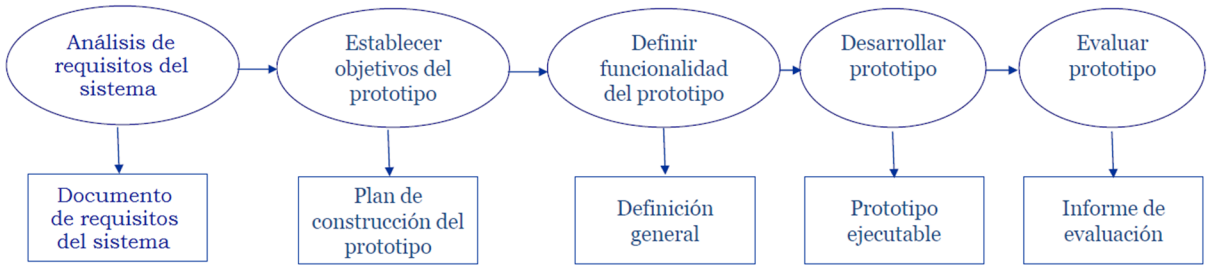


Figura 2. Metodología de prototipado

Tras comprobar mediante el prototipado que el desarrollo es completamente viable y compatible con el firmware OpenWRT se procederá a utilizar la metodología incremental (Figura 3) con la finalidad de desarrollar este software en un lenguaje de alto nivel, Python, que permitirá consumir una API oficial de Latch y realizar todas las gestiones y modificaciones de forma sencilla manteniendo un código limpio y legible.

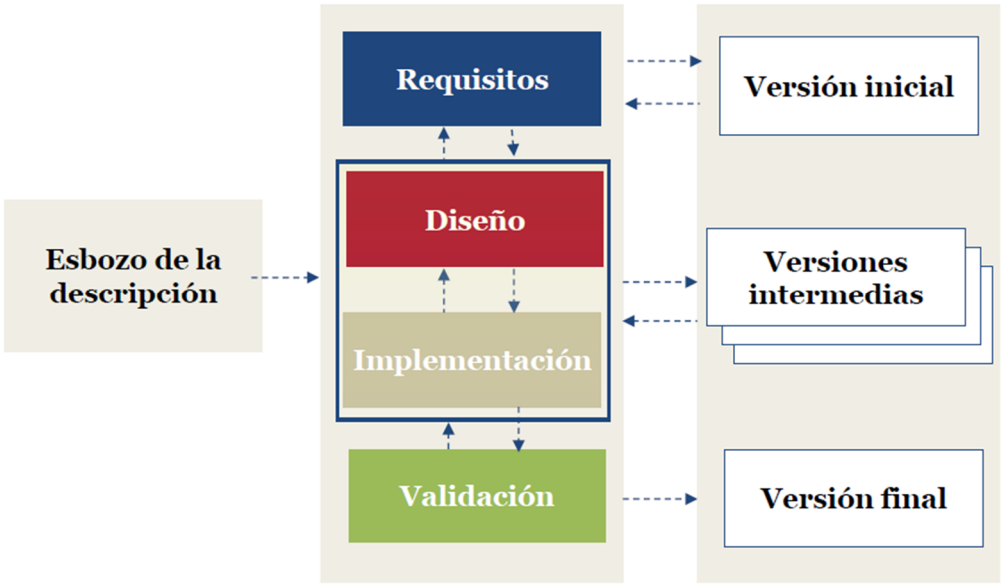


Figura 3. Metodología incremental

3.3. Requisitos

3.3.1. Requisitos funcionales

En este capítulo encontramos los requisitos de funcionamiento del sistema que nos ayudarán a conseguir los objetivos del mismo.

- **RF-01: Configuración del sistema:** El administrador de red debe de ser capaz de cambiar la configuración del sistema en cualquier momento, de forma que solo haga falta un reinicio del servicio para que el sistema funcione con los nuevos datos.
- **RF-02: Gestión de dispositivos de control:** El administrador de red debe de poder establecer o eliminar el dispositivo de control.
- **RF-03: Pareado de dispositivos de control:** El administrador debe de poder parear un dispositivo de control con la aplicación Latch registrada en la configuración (dep. RF-02).
- **RF-04: Despareado de dispositivos de control:** El administrador debe de poder desparear un dispositivo de control, pareado previamente en la aplicación Latch registrada en la configuración (dep. RF-02).
- **RF-05: Administración de dispositivos de usuario desde móvil:** El administrador de red debe de poder administrar la red desde el móvil.
- **RF-06: Desbloqueo de dispositivos usuario:** El administrador de red debe de poder permitir el acceso a la red de un dispositivo conectado (dep. RF-05)
- **RF-07: Bloqueo de dispositivos de usuario:** El administrador de red debe de poder bloquear el acceso a la red de un dispositivo conectado (dep. RF-05)
- **RF-08: Silenciar alertas de dispositivos bloqueados:** El administrador de red debe de poder silenciar las alertas que recibe por el intento de un acceso bloqueado a la red (dep. RF-05)
- **RF-09: Bloquear todos los dispositivos a la vez:** El administrador de red debe de poder bloquear el acceso a todos los dispositivos a la vez (dep. RF-05)
- **RF-10: Establecer autobloqueos por horas:** El administrador de red debe de poder bloquear el acceso a los dispositivos durante una franja horaria (dep. RF-05)
- **RF-11: Administración de dispositivos de usuario desde web:** El administrador de red debe de poder administrar ciertos parámetros de los dispositivos de usuario desde la web.
- **RF-12: Capacidad de ver datos de dispositivos:** El administrador de red debe de poder obtener ciertos datos de los dispositivos de usuario desde la web (dep. RF-11).
- **RF-13: Establecer/quitar dispositivos como permanentes:** El administrador de red debe de poder poner o quitar la propiedad de dispositivo permanente a los dispositivos de usuario desde la web (dep. RF-11).

- **RF-14: Cambiar alias de dispositivos:** El administrador de red debe de poder establecer y/o cambiar el alias de los dispositivos de usuario desde la web (dep. RF-11).
- **RF-15: Administración interna de dispositivos:** El sistema debe de poder permitir o no el acceso a la red por los dispositivos conectados.
- **RF-16: Expulsión de dispositivos y bloqueo:** El sistema debe de poder expulsar a los dispositivos de usuario conectados si su estado es "bloqueado" y debe de poder impedir el acceso de dicho dispositivo a la red durante un tiempo prefijado (dep. RF-15)

3.3.2. Requisitos no funcionales

A continuación, aparecen requisitos no funcionales del sistema.

- **RNF-01:** El sistema debe de estar conectado a internet.
- **RNF-02:** La comunicación con el servidor Latch debe de ser segura.
- **RNF-03:** Solo el usuario de dispositivo de control debe de ser capaz de administrar la red.
- **RNF-04:** La administración de la red debe de estar disponible desde cualquier acceso a internet con el dispositivo de control asignado al sistema.
- **RNF-05:** Los datos de configuración del sistema deben de estar enlazados a una aplicación de los servidores Latch.

3.3.3. Requisitos de información

A continuación, aparece una lista de requisitos de información con la información que debemos de manejar y almacenar.

- **RI-01: Configuración del sistema:** Almacenar configuración del sistema.
- **RI-02: Dispositivo de control:** Almacenar datos del dispositivo de control.
- **RI-03: MAC usuario:** Almacenar dirección MAC de dispositivos usuario.
- **RI-04: Estado usuario:** Almacenar estado de dispositivos usuario.
- **RI-05: Permanencia usuario:** Almacenar estado permanencia de dispositivos usuario.
- **RI-06: Última conexión usuario:** Almacenar última conexión de dispositivos usuario.
- **RI-07: Alias usuario:** Almacenar alias de dispositivos usuario.

3.4. Casos de uso

En esta sección se muestran los casos de uso del sistema que se han tenido en cuenta para el desarrollo del mismo, y que se puede ver su relación de forma más directa en el diagrama (Figura. 4).

Estos casos de uso representan las acciones básicas a las que tienen acceso el administrador y los usuarios que intentan conectar los dispositivos inalámbricos a la red.

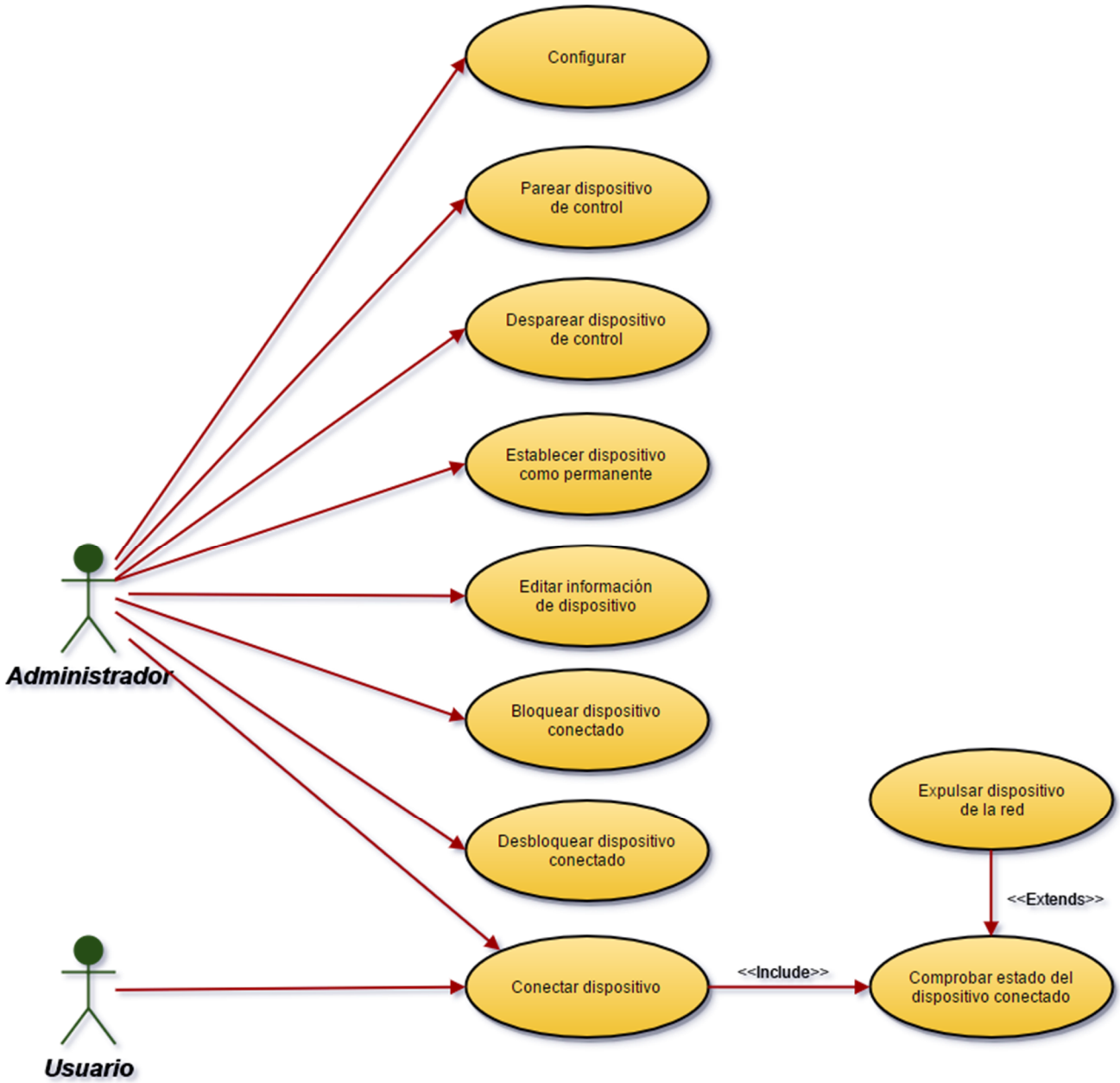


Figura 4. Diagrama de casos de uso

En las siguientes tablas se muestra una descripción más detallada de cada una de las posibles operaciones que se pueden realizar y que se han visto en la Figura 2.

Nombre	Configuración
Actor	Administrador
Descripción	El administrador puede establecer la configuración al plugin a través de la interfaz web, sincronizándolo así con la aplicación en los servidores de Latch que se encargará de mantener el estado de los dispositivos.

Nombre	Parear dispositivo
Actor	Administrador
Descripción	El administrador puede seleccionar un dispositivo compatible con la aplicación “Latch” de Telefónica y parearlo con la aplicación de los servidores de Latch, cuyo enlace se ha realizado previamente. Solo se puede tener enlazado un dispositivo de control a la vez. Esta acción se realiza desde la interfaz web.

Nombre	Desparear dispositivo
Actor	Administrador
Descripción	El administrador puede desparear un dispositivo de control de la aplicación “Latch” previamente pareado con el sistema con la finalidad de desactivar la protección o parear un dispositivo distinto. Esta acción se realiza desde la interfaz web.

Nombre	Establecer dispositivo como permanente
Actor	Administrador
Descripción	El administrador tiene la capacidad de seleccionar un dispositivo conectado a la red y establecerlo como permanente de forma que dicho dispositivo deje de controlarse con la aplicación Latch y tenga acceso sin restricciones a la red. Además, también puede añadir una dirección MAC manualmente sin necesidad de que el dispositivo esté conectado en ese momento.

Nombre	Editar información de dispositivo
Actor	Administrador
Descripción	El administrador puede seleccionar un dispositivo conectado a la red desde la lista de dispositivos de la interfaz web y establecerle un alias con el que será reconocido tanto en la lista de dispositivos de la interfaz web como en la lista de dispositivos de la aplicación Latch.

Nombre	Bloquear dispositivo
Actor	Administrador
Descripción	El administrador puede bloquear un dispositivo que se está intentando conectar a la red, o un dispositivo ya conectado y con permisos de acceso, desde la aplicación “Latch” que había pareado previamente con los servidores de Latch. De esta forma el dispositivo bloqueado será expulsado de la red,

Nombre	Desbloquear dispositivo
Actor	Administrador
Descripción	El administrador puede desbloquear un dispositivo que se está intentando conectar a la red desde la aplicación “Latch” que había pareado previamente con los servidores de Latch. De esta forma el dispositivo desbloqueado podrá navegar por la red hasta que decidamos bloquearlo o se desconecte. Los permisos de acceso son borrados cuando un dispositivo se desconecta, teniendo que dárselos de nuevo cuando vuelva a conectar a la red o poniéndolo como dispositivo permanente.

Nombre	Conectar dispositivo – Bloqueado
Actor	Dispositivo externo
Descripción	Un dispositivo intenta conectar y el sistema le expulsa de la red porque el administrador le ha bloqueado o es la primera vez que conecta y no lo ha desbloqueado. El sistema impide además que el dispositivo expulsado se conecte en los siguientes 5 segundos.

Nombre	Conectar dispositivo – Desbloqueado
Actor	Dispositivo externo
Descripción	Un dispositivo intenta conectar y empieza a usar la red porque el administrador le ha desbloqueado. Los dispositivos permanentes acceden directamente a la red sin pasar por el control de Latch.

3.5. Descripción del funcionamiento del sistema

El esquema general de funcionamiento del sistema será muy sencillo.

El primer paso será configurar el sistema estableciendo la app id y el secret key que se utilizará para conectar con la aplicación creada en los servidores de Latch mediante el uso de una API basada en Python. Además, se tendrá que configurar un tiempo de recarga que se usará internamente por el sistema.

El siguiente paso consistirá en elegir un dispositivo de control y parearlo al sistema (Figura 5) usando una interfaz gráfica web que tendremos disponible para ello. Este dispositivo de control será el que se usará para administrar los dispositivos que se conecten a la red, permitiendo su conexión o denegándola y por lo tanto provocando la expulsión de dicho dispositivo de la red.

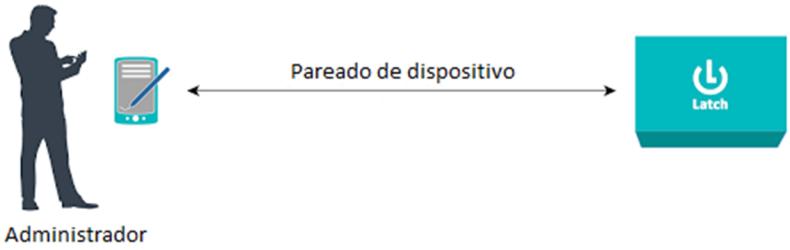


Figura 5. Pareado de dispositivo

Internamente el sistema se dedicará a comprobar cada cierto tiempo, el configurado previamente, que dispositivos son los que están conectados a la red y creará una lista de los mismos.

Con la lista previamente creada, el sistema comprobará uno a uno si existe una instancia de cada dispositivo.

En caso de que no existiese instancia del dispositivo en los servidores de Latch el sistema se encargará de crearla y de bloquearla para que el administrador de la red sea avisado en el dispositivo de control y pueda tomar las medidas de administración que crea oportunas.

En caso de que si existiese la instancia del dispositivo se comprobará el estado, que puede ser bloqueado o desbloqueado.

En el primer caso, el de bloqueado, el sistema avisará al administrador de la red a través de la aplicación Latch y expulsará al dispositivo de la red bloqueándolo durante unos segundos para que no pueda volver a conectarse.

En el segundo caso, el de desbloqueado, el sistema dejará que el dispositivo conectado pueda interactuar correctamente con la red.

El proceso anteriormente descrito será el esquema general de funcionamiento del sistema. Se puede ver un esquema del mismo en la Figura 6.

En cualquiera de los casos, cuando el dispositivo se desconecte de la red, se eliminará de la lista de control y se borrarán todos los permisos, de forma que, aunque previamente tuviesen permisos de conexión, la próxima vez que conecten se bloqueará y avisará al administrador de la red de nuevo.

Por otro lado, tendremos los dispositivos permanentes, que serán dispositivos a los que le daremos un permiso permanente de conexión por la razón que sea y que no aparecerán en la lista de control de Latch ni se bloquearán.

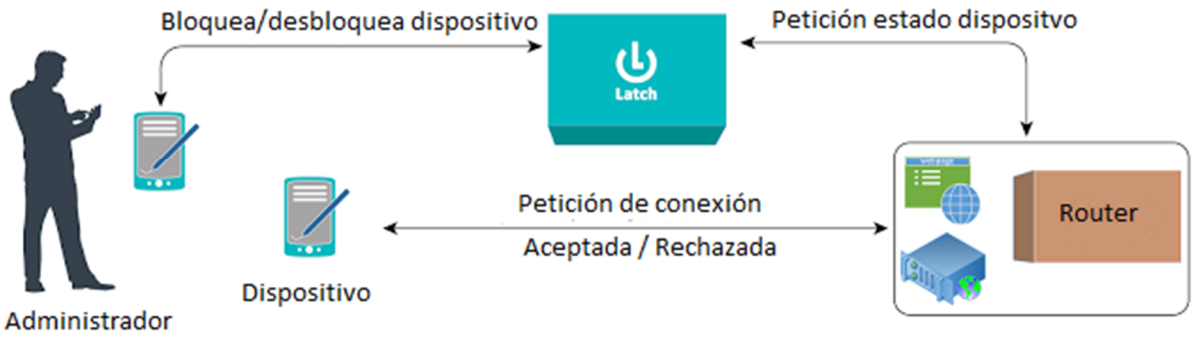


Figura 6. Esquema de funcionamiento del sistema – General

4. Implementación del prototipo

Sinopsis:

En este capítulo se realizará la implementación de un prototipo usando Bash que nos ayudará a comprobar que el proyecto es completamente viable.

Además, haremos un primer esbozo del funcionamiento que buscamos mediante diagramas de flujo y diagramas de interacción de componentes.

4.1. Implementación del prototipo

Como queremos realizar un prototipo que nos permita comprobar si la idea descrita en secciones anteriores es viable y es compatible con OpenWRT, y para no tener que buscar un router compatible con dicho sistema, se montará una máquina virtual sobre la que trabajar.

Está decisión aportará bastantes ventajas como la rapidez de trabajo y compilación al tener más potencia que un router convencional o las Snapshots para tener una copia de seguridad cada vez que se quiera por si se comete algún error.

A su vez, tratándose de una imagen genérica y no para un router especifico, traerá otros inconvenientes, como la dificultad de configuración inicial o los drivers de la interfaz inalámbrica entre otras cosas.

El primer paso consiste en crear una máquina virtual con Oracle Virtualbox. El proceso puede verse en el anexo 3 – Gestión de la máquina virtual.

Tras iniciar la máquina virtual se podrá comprobar que no habrá acceso a internet (Imagen 1), ni a la red local, por lo que no se podrá acceder a la interfaz web desde fuera de la máquina para realizar gestiones sobre OpenWRT.

```
root@OpenWrt:/# ping www.google.es
ping: bad address 'www.google.es'
root@OpenWrt:/# _
```

Imagen 1. Error de conexión a internet

Para solucionarlo, habrá que modificar el archivo de configuración de red adaptándolo a la topología de nuestra red (Imagen 2).

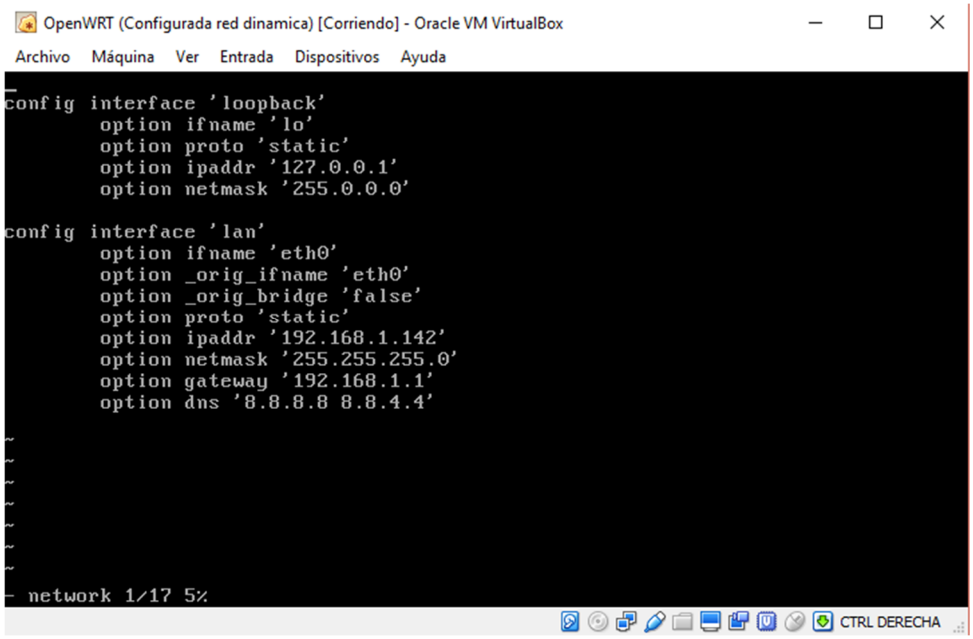


Imagen 2. Configuración de red

Tras guardar la configuración y reiniciar la máquina virtual se podrá comprobar que el problema de acceso a internet ha sido solucionado (Imagen 3).

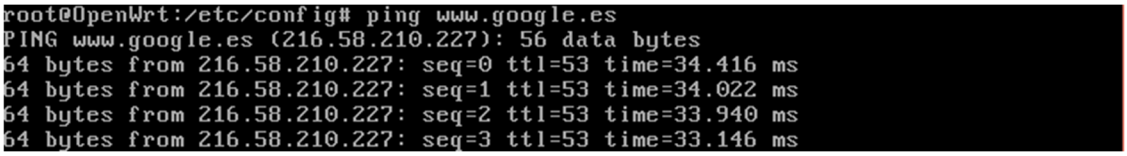


Imagen 3. Acceso a la red correcto

Con esto ya se podrá acceder a la interfaz de administración de OpenWRT (Imagen 4) desde un navegador web disponible en cualquiera de los dispositivos que tengamos conectados a la red.

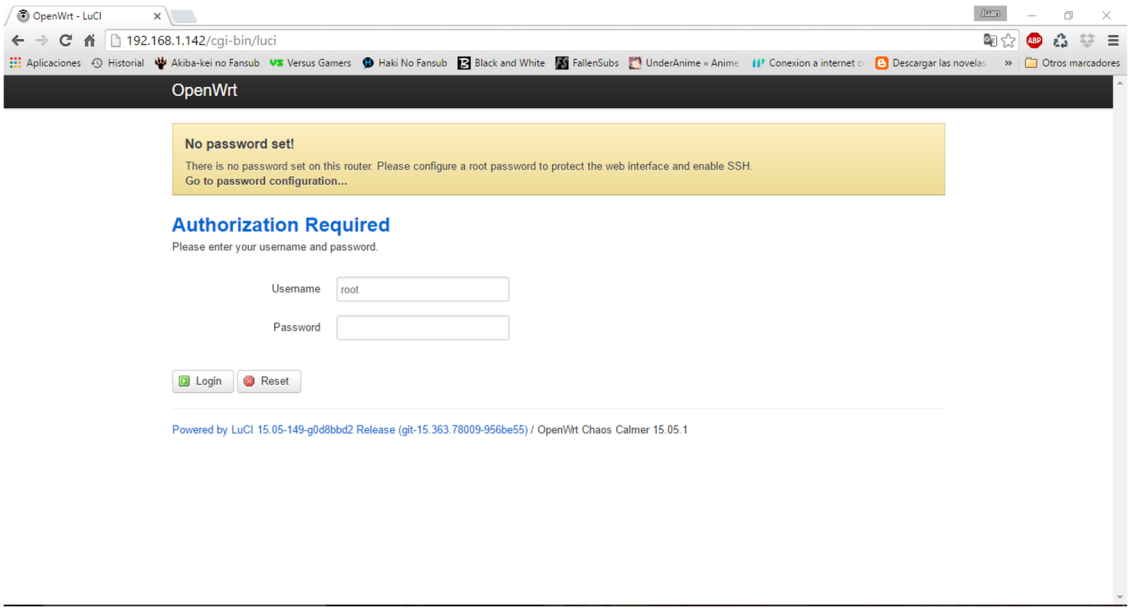


Imagen 4. Interfaz web OpenWRT

Una vez esté OpenWRT funcionando habrá que conectar un adaptador de red por USB para que OpenWRT actúe de router dando conexión.

En caso de que la máquina virtual haya sido creada bajo el software VirtualBox es probable que el adaptador de red conectado no sea detectado, ya que es conocido que VirtualBox da problemas con este tipo adaptadores. En este caso no queda más opción que transformar la imagen de VirtualBox con la que se está trabajando en una compatible con VMWare. Este proceso está descrito en la sección “Anexo – Gestión de la máquina virtual”.

Tras conectar OpenWRT, ahora funcionando en VMWare, debería de detectar correctamente el adaptador de red conectado por USB. No obstante, al haber descargado una imagen genérica es muy probable que la interfaz de administración de OpenWRT no muestre la configuración Wi-Fi (Imagen 5). Esto se debe a un problema de drivers.

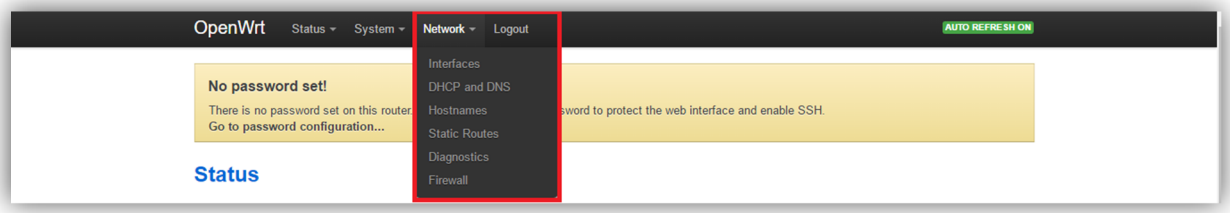


Imagen 5. No hay configuración Wi-Fi

Dado que la instalación de drivers en el sistema depende del dispositivo que se esté intentando conectar, no se muestra el proceso de instalación de los mismos. Una vez instalados los drivers correctamente, y reiniciada la máquina virtual, debería de aparecer la configuración Wi-Fi (Imagen 6) en la interfaz de administración de OpenWRT.

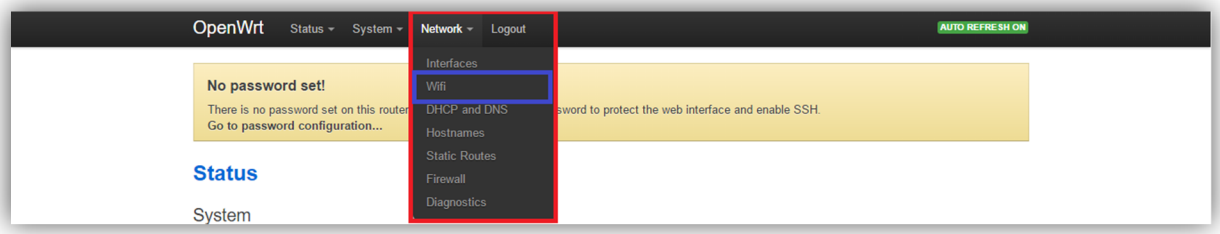


Imagen 6. Configuración Wi-Fi visible

Ya se puede configurar la red inalámbrica desde la interfaz de OpenWRT, y una vez configurada la red se debería de poder conectar correctamente a la misma desde cualquier dispositivo inalámbrico.

Realizado el proceso anterior, se puede empezar a trabajar con Latch. Por lo tanto, habrá que crear una cuenta de desarrollador en los servidores de Latch.

Con la cuenta activa y con acceso al panel de usuario de Latch, se tendrá que crear una aplicación sobre la que trabajar para obtener un APP ID y una Secret Key.

El proceso de creación de cuenta y aplicación Latch es muy sencillo. No obstante, se explica más profundamente con imágenes en la sección “6. Configuración del sistema”.

El siguiente paso será acceder, usando algún programa de conexión SSH, al sistema y empezar a trabajar sobre el mismo para la integración de Latch en OpenWRT.

Se necesitará instalar Bash y Base64 en el sistema dado que son necesarios para realizar el prototipo, y se deberán de crear y dar permisos de ejecución a los ficheros básicos que permitirán parear y desparear un dispositivo con la aplicación Latch que se había creado previamente.

Parte de esos ficheros debería incluir algún archivo de texto plano para almacenar el id de la cuenta que se va a parear y desparear.

Una vez con el sistema de ficheros creado, hay que realizar el desarrollo que permita realizar el pareado de una cuenta de control con la aplicación Latch y que almacene el ID de cuenta del mismo.

Además, en el fichero de registro se debe de crear también una comprobación de prerrequisitos para garantizar que todos los paquetes estén instalados y que por lo tanto todo funcione perfectamente.

El resultado del desarrollo anterior se puede ver en los códigos 4 y 5.

```
#!/bin/bash
####Check procps package, requirement to execute this script
if  opkg list | grep procps > /dev/null
then
    echo 'Found procps'
else
    echo 'Updating package list...'
    opkg update > /dev/null
    echo 'Installing procps package...'
    opkg install procps > /dev/null
fi
####Check coreutils package, requirement to execute this script
if  opkg list | grep coreutils-base64 > /dev/null
then
    echo 'Found coreutils-base64'
else
    echo 'Updating package list...'
    opkg update > /dev/null
    echo 'Installing coreutils-base64 package...'
    opkg install coreutils-base64 > /dev/null
fi
####Check wget package, requirement to execute this script
if  opkg list | grep wget > /dev/null
then
    echo 'Found wget'
else
    echo 'Updating package list...'
    opkg update > /dev/null
    echo 'Installing wget...'
    opkg install wget > /dev/null
fi
####Check openssl-util package, requirement to execute this script
if  opkg list | grep openssl-util > /dev/null
then
    echo 'Found openssl-util'
else
    echo 'Updating package list...'
    opkg update > /dev/null
    echo 'Installing openssl-util package...'
    opkg install openssl-util > /dev/null

fi
####Check bash package, requirement to execute this script (please, check the version is not Shellshocked ;) )
if  opkg list | grep bash > /dev/null
then
    echo 'Found bash'
else
    echo 'Updating package list...'
    opkg update > /dev/null
    echo 'Installing bash package...'
    opkg install bash > /dev/null
fi
####End packages check...
```

Código 4. Comprobación de paquetes correctos.

```
if [ "$3" == "debug" ]; then set -x; fi
applicationId="2kW6yffDwfJj3uZWrRJu"
secretkey="kX6x9gtffsGqWWcQbcbdA7UMmfeu3vB2Y7BbZaqK"
URL="/api/0.6/pair/$2"
Usuario="$1"
LATCH="/root/latch/latch.accounts"

####Delete all errors
grep -Ev 'error' $LATCH > $LATCH


if [ -z "$1" -o -z "$2" ]; then
 echo -e "\nUsage: SITE LATCH-REG <Username> <pair-key>\n"
 exit 0
fi
if [ `grep "^$Usuario:" $LATCH | wc -l` -ne 0 ]; then
 echo -e "\nAlready registered\n"
 exit 0
fi

requestSignature+="GET\n"
date=`date -u '+%Y-%m-%d %H:%M:%S'`
requestSignature+="$date\n\n$URL"
signed=`echo -en "$requestSignature" | openssl dgst -sha1 -hmac "$secretkey" -binary|sed -e 's|.*= \(.*\)|\1|g'`
b64signed=`echo -n "$signed"|base64`
auth_header="Authorization:11PATHS $applicationId $b64signed"
date_header="X-11Paths-Date: $date"
JSON=`wget -q --no-check-certificate -O - --header "$auth_header" --header "$date_header"
"https://latch.elevenpaths.com$URL"`
accountid=`echo "$JSON" | sed -e 's|.*accountId":"\(.*\)"}.*|\1|'| sed -e 's|\(.*\)".*|\1|g'`

if [ -z $accountid ]; then
 echo "Error."
else
 echo "$Usuario:$accountid" >> $LATCH
 echo -e "\nDone.\n"
fi
```

Código 5. Pareado de dispositivos

Si todo va bien el proceso de pareado debe de estar funcionando correctamente. Ahora hay que realizar el desarrollo del fichero que permite desparear un dispositivo (Código 6) de control de la aplicación Latch, de forma que posteriormente se pueda parear otro distinto.

```
#!/bin/bash
if [ "$2" == "debug" ]; then set -x; fi
if [ -z "$1" ]; then
 echo -e "\nUsage: SITE LATCH-UNREG <Username>\n"
 exit 0
fi

applicationId="2kW6yffDwfJj3uZWrRJu"
Usuario="$1"
secretkey="kX6x9gtffsGqWWcQbcbdA7UMmfeu3vB2Y7BbZaqK"
LATCH="/root/latch/latch.accounts"
account=`grep "^$Usuario:" $LATCH |cut -d: -f2`
URL="/api/0.6/unpair/$account"
if [ -z "$account" ]; then echo -e "Error."; exit 0; fi
requestSignature+="GET\n"
date=`date -u '+%Y-%m-%d %H:%M:%S'`
requestSignature+="$date\n\n$URL"
signed=`echo -en "$requestSignature" | openssl dgst -sha1 -hmac "$secretkey" -binary|sed -e 's|.*= \(.*\)|\1|g'`
b64signed=`echo -n "$signed"|base64`
auth_header="Authorization:11PATHS $applicationId $b64signed"
date_header="X-11Paths-Date: $date"
JSON=`wget -q --no-check-certificate -O - --header "$auth_header" --header "$date_header"
"https://latch.elevenpaths.com$URL"`
grep -v "^$Usuario:" $LATCH > /tmp/$$.tmp
mv /tmp/$$.tmp $LATCH
echo -e "\nDone.\n"
```

Código 6. Despareado de dispositivos

Con el proceso de pareado y despareado funcionando correctamente ya solo queda, en el prototipo, crear una versión inicial del que será el protocolo encargado de procesar los dispositivos inalámbricos externos que intenten conectarse a la red, cuyo resultado puede verse en el código 7.

```
#!/bin/bash
applicationId="2kW6yffDwfJj3uZWrRJu"
secretkey="kX6x9gtffsGqWWcQbcbdA7UMmfeu3vB2Y7BbZaqK"
LATCH="/root/latch/latch.accounts"
OPERATIONS="/root/latch/latch.operations"
account=`grep "^$USER:" $LATCH |cut -d: -f2`
if [ -z `echo "$account"|cut -d: -f2`  ]; then exit 0; fi

####Check Internet status####
WAN=`ping -w1 -c1 8.8.8.8 > /dev/null 2>&1 && echo "up" || echo "down" && exit 1`

####If Latch is blocked and Internet status is UP, do these sequences####
if [ `echo "$WAN" | grep "up"` ]; then
    ####List all wireless network interfaces
    for interface in `iw dev | grep Interface | cut -f 2 -s -d" "`
        do
            ####For each interface, get mac addresses of connected stations/clients
            maclist=`iw dev $interface station dump | grep Station | cut -f 2 -s -d" "`
        ####For each mac address in that list...
        for mac in $maclist
        do
            echo "MAC: $mac"
            ####Look for possible operationid
            operationId=`cat latch.operations | grep $mac | cut -d "-" -f 2`

            ####If the MAC has not operation, create one (blocked) and save it
            if [ -z "$operationId" ]; then
                ####Create and save operationid for MAC
                echo "Creating new operation"
                URL="/api/1.0/operation"
                        requestSignature="GET\n"
                        date=`date -u '+%Y-%m-%d %H:%M:%S'`
                        requestSignature+="$date\n\n$URL"
                        signed=`echo -en "$requestSignature" | openssl dgst -sha1 -hmac "$secretkey" -binary|sed -e
's|.*= \(.*\)|\1|g'`
                        b64signed=`echo -n "$signed"|base64`
                        auth_header="Authorization:11PATHS $applicationId $b64signed"
                        date_header="X-11Paths-Date: $date"
                        JSON=`wget -q --no-check-certificate -O - --header "$auth_header" --header "$date_header"
"https://latch.elevenpaths.com$URL"`
                        operationId=`echo -e "$JSON" | sed -e 's|.*operationId":"\(.*\)","name.*|\1|g'`
                echo "$mac-$operationId" >> $OPERATIONS
            fi
            ####Check status
                    URL="/api/1.0/status/$account/op/$operationId"
                    requestSignature="GET\n"
                    date=`date -u '+%Y-%m-%d %H:%M:%S'`
                    requestSignature+="$date\n\n$URL"
                    signed=`echo -en "$requestSignature" | openssl dgst -sha1 -hmac "$secretkey" -binary|sed -e 's|.*=
\(.*\)|\1|g'`
                    b64signed=`echo -n "$signed"|base64`
                    auth_header="Authorization:11PATHS $applicationId $b64signed"
                    date_header="X-11Paths-Date: $date"
                    JSON=`wget -q --no-check-certificate -O - --header "$auth_header" --header "$date_header"
"https://latch.elevenpaths.com$URL"`
                    status=`echo -e "$JSON" | sed -e 's|.*status":"\(.*\)","name.*|\1|g'`

            ####If the MAC is blocked, block the connection for this MAC
            if [ `echo "$status" | grep "off"` ]; then
            echo "Operation Blocked"
            linea=`iptables -nL FORWARD --line-number | grep -i $mac | cut -d " " -f 1`
            iptables -D FORWARD $linea
            iptables -A FORWARD -m mac --mac-source $mac -j DROP
            ####If the MAC is unblocked, unblock the connection for this MAC
             else
            echo "Operation Allowed"
            linea=`iptables -nL FORWARD --line-number | grep -i $mac | cut -d " " -f 1`
                    iptables -D FORWARD $linea
            iptables -A FORWARD -m mac --mac-source $mac -j ACCEPT
            fi
            echo ""
        done
        ####Check if exists any obsolete operationid and delete it
                echo "Deleting obsolete operations"

        ####Reload Firewall
        /etc/init.d/firewall reload
    done
else
    ####If wan connection is down, execute the command####
    echo -e "Error LATCH or not Internet connection" | tr -d '\r'
    exit 0;
fi
```

Código 7. Control de dispositivos

En el prototipo si el dispositivo está bloqueado se realiza un bloqueo de internet por IP Tables, pero en la versión final del desarrollo el bloqueo consistirá en la expulsión del dispositivo de la red.

Llegados a este punto, y con las operaciones descritas anteriormente funcionando correctamente, podemos darnos cuenta de que el desarrollo es completamente viable y de que podemos realizar una implementación en un lenguaje de alto nivel.

4.2. Diagrama de flujo general del prototipo

A continuación, se muestra un diagrama de flujo (Figura 7) que muestra el funcionamiento básico del prototipo cuando quiere comprobar si los dispositivos tienen o no acceso a la red, es decir, el estado que poseen en los servidores de Latch.

Los dispositivos se deben crear a mano como operaciones en la aplicación Latch, y sus ID de operación deben añadirse a mano en un fichero llamado latch.operations en la misma carpeta que los demás ficheros del prototipo.

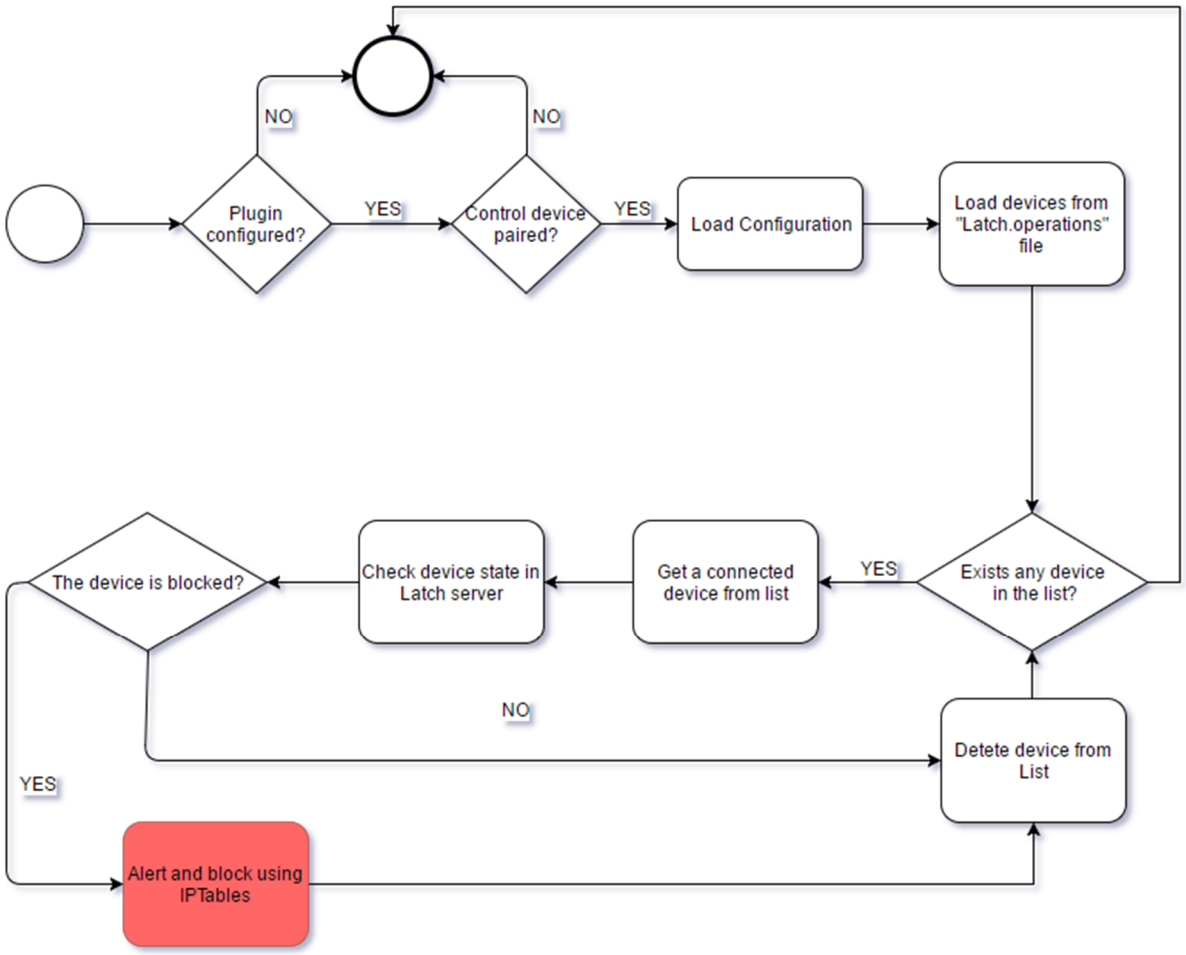


Figura 7. Diagrama de flujo general – Prototipo

4.3. Diagrama de flujo (des)pareado del prototipo

En la Figura 8 se muestra el diagrama de flujo del prototipo cuando quiere parear o desparear un dispositivo. Para parear y desparear se usan 2 ficheros distintos, donde la condición “Register code recived” del diagrama de flujo significa que si se va a usar un código de pareado usaremos el fichero de parear y sino al de desparear.

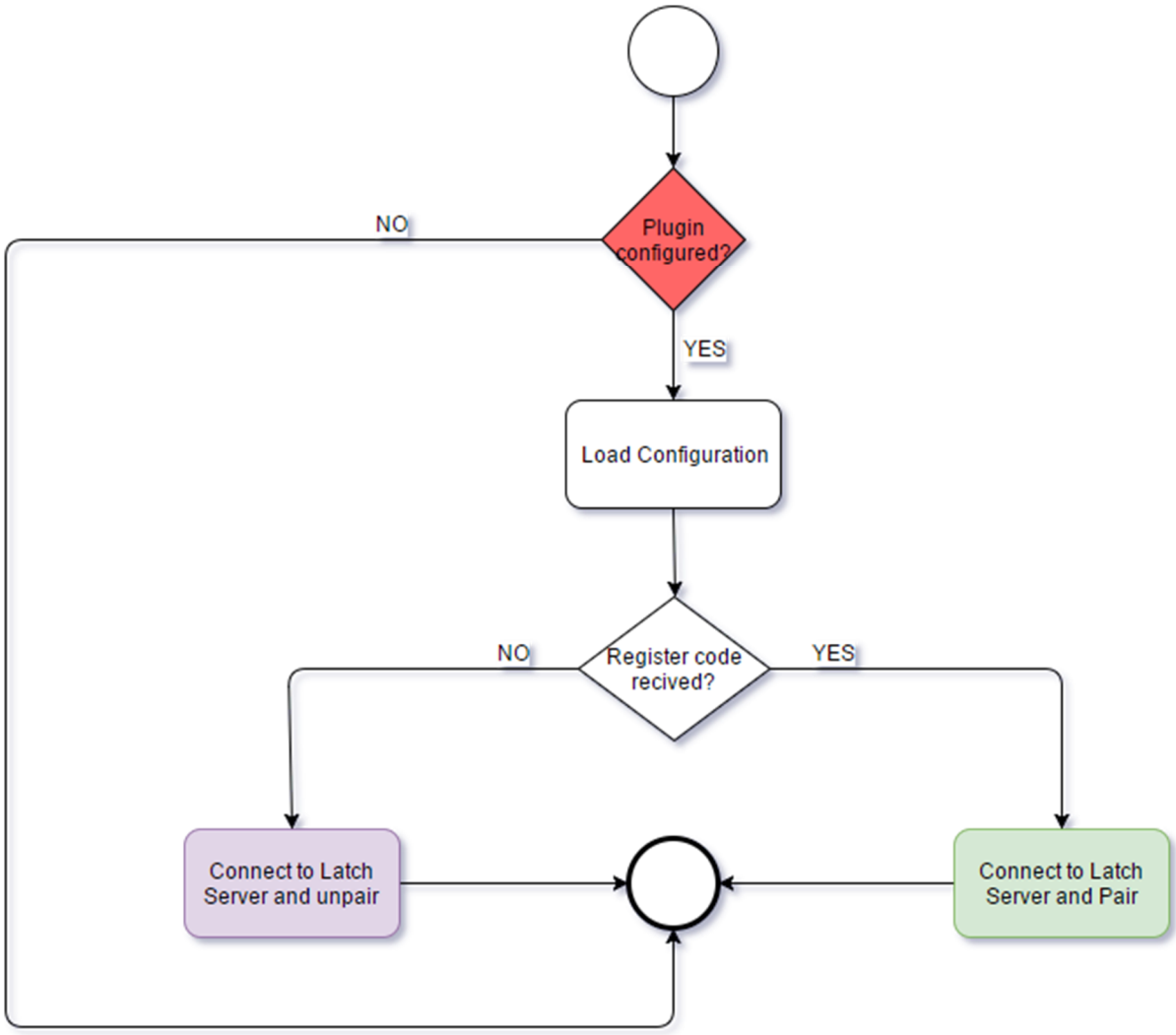


Figura 8. Diagrama de flujo de pareado / despareado – Prototipo

4.4. Diagrama de componentes del prototipo

En esta sección se muestra con un diagrama de componentes (Figura 9) cómo interactúan los componentes en el prototipo. Al ser un prototipo tiene pocas, y muy simples, componentes e interacciones, pero suficientes para comprobar que el desarrollo es viable.

Además, las peticiones al servidor Latch se hacen a mano en los ficheros y sin hacer uso de ninguna API que facilite el trabajo.

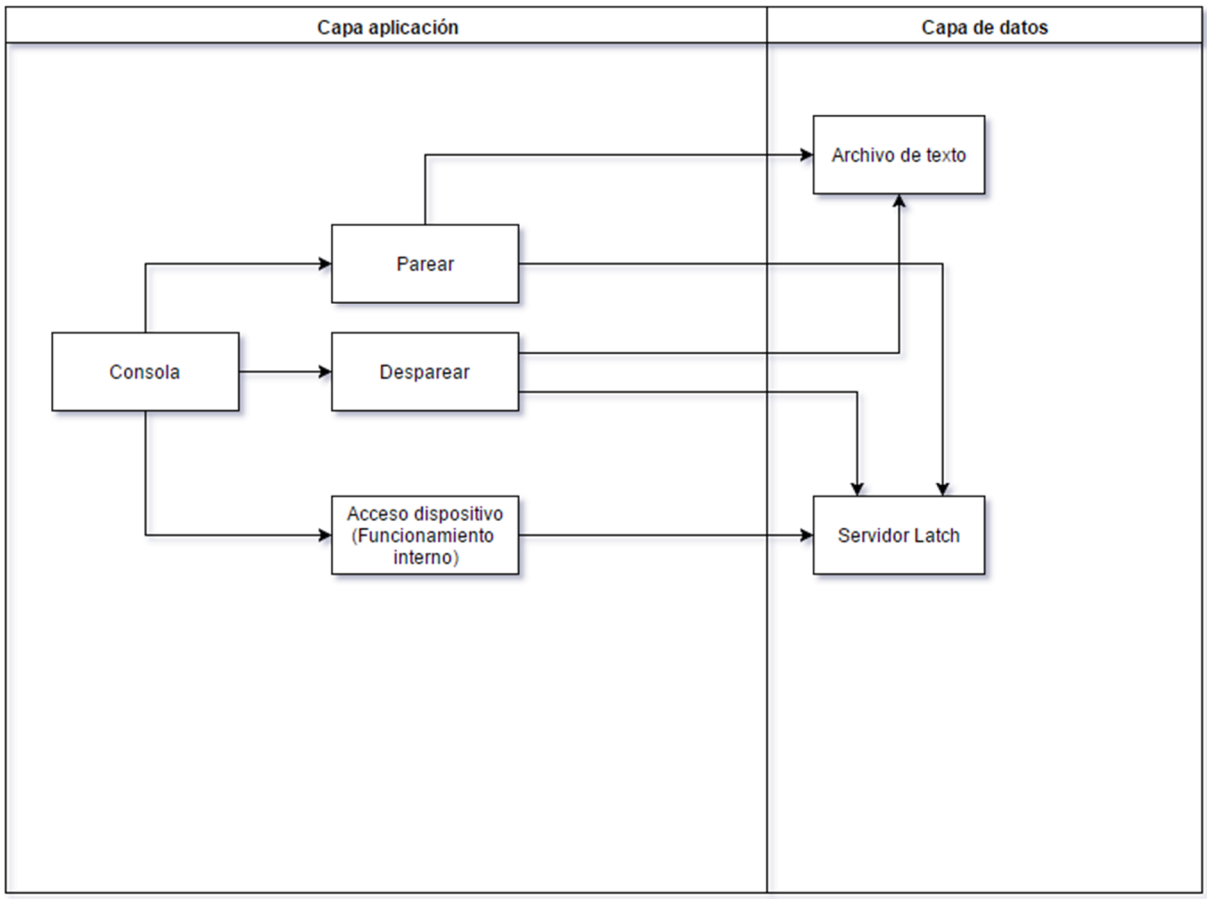


Figura 9. Diagrama de interacción de componentes - Prototipo.

5. Implementación y despliegue del sistema

Sinopsis:

En este capítulo se realizará la implementación del sistema final y se creará el paquete necesario para la instalación del mismo en dispositivos embebidos con OpenWRT.

Se podrá ver cómo, a partir del prototipo implementado en la sección anterior, el sistema crece en funcionalidad.

Se crea un protocolo de análisis de dispositivos conectados que se apoya en una interfaz web que servirá al usuario de herramienta para facilitar la configuración del sistema.

Puede verse como los diagramas de flujo y de componentes han evolucionado a partir del prototipo.

5.1. Implementación del sistema

Con el prototipo ya funcionando con las características principales queda claro que la idea es viable.

Ahora hay que proceder a seleccionar un lenguaje de alto nivel que no ocupe mucho, dada la importancia del espacio en los routers, y a ser posible que sea compatible con alguna de las API de Latch. En mi caso, Python es la solución perfecta.

Tener un prototipo ya funcionando nos da muchas ventajas ya que, a pesar de que el sistema final cambiará bastante, se tiene una primera versión en la que fijarnos y trasladar al lenguaje que hayamos elegido.

Antes de empezar hay que comprobar los requisitos del API de Latch en caso de que se vaya a usar uno, y que componentes harán falta para trabajar con él. En el caso de Python son bastantes, entre los que se incluyen certificados, codecs, logging o SSL entre otros. Además, se deberá de instalar el componente de SQLite para el lenguaje que vayamos a usar, ya que es recomendable guardar toda la configuración, usuarios pareados y alguna que otra información sobre dispositivos en una base de datos de esta tecnología en vez de usar archivos de texto plano.

Lo primero a realizar es el diseño de la base de datos SQLite y crearla para que cumpla las especificaciones que buscamos. El diseño que yo he utilizado se puede ver en la siguiente sección, en “5.2. Implementación del sistema de persistencia”.

Posteriormente debemos de crear los archivos necesarios en el lenguaje que hemos seleccionado para que se pueda realizar el pareado, despareado y control de los dispositivos conectados. En mi caso he creado una clase Python que me permitirá parear, desparear y controlar todo el sistema desde un solo fichero, y una clase auxiliar que será la encargada de interactuar con la base de datos según las peticiones de la clase principal.

Una de las cosas que no se han realizado en el prototipo es la creación automática de instancias de un dispositivo en los servidores de Latch cuando se detecta la conexión de dicho dispositivo. En el prototipo las instancias se deben crear a mano como operaciones en la aplicación Latch y añadirlos a un fichero donde cada id de operación lo asociamos a una dirección MAC concreta.

Para realizar la creación de instancias automáticas por dispositivo hay que realizar una petición a los servidores de Latch a través de la API. Además, para que la aplicación instalada en el dispositivo de control nos avise de que se ha conectado un nuevo dispositivo se necesita que la instancia creada lo haga en estado bloqueado.

El problema es que para realizar una petición que bloquee la instancia que se acaba de crear se necesita una licencia de pago, ya que la versión gratuita no permite bloquear o desbloquear instancias a través de la API. En mi caso, y para la realización de este proyecto, 11Paths, división de seguridad de Telefónica y creadora de Latch, me cedió de forma altruista una licencia.

Una vez conseguido que funcione todo el sistema de pareado, despareado y control de dispositivos, podemos añadir la posibilidad de dispositivos permanentes.

Los dispositivos permanentes son dispositivos que no vamos a controlar desde la aplicación móvil Latch, sino que se les va a permitir hacer uso de la red sin ningún tipo de restricción o aviso.

Llegados a este punto tenemos el sistema interno funcionando. Ahora podemos crear unas interfaces web que se integren en la interfaz de administración de OpenWRT y que permitan gestionar el sistema que acabamos de implementar.

El proceso a realizar para la creación de estas interfaces puede verse en la sección “5.3. Implementación de la interfaz web”. Además, algunos ejemplos del resultado de este proceso pueden verse en “Anexo – Diseño web del proyecto”.

5.2. Implementación del sistema de persistencia

El sistema de persistencia se va a basar en un sistema de base de datos en el que tener toda la información organizada y con la que se pueda trabajar de forma rápida, sencilla y fiable, y de los componentes necesarios para que el lenguaje Python pueda trabajar con dicha base de datos.

La opción más adecuada para implementar esta base de datos es un archivo SQLite. Esta decisión se basa entre otras razones al poco espacio que existe normalmente en los routers, así como la falta de potencia de los mismos. Por lo anterior no es adecuado usar un sistema de base de datos completo como puede ser MySQL.

El archivo SQLite ocupará apenas unos KB y no necesitará de un sistema de gestión de base de datos que lo mantenga, sino que se interactuará directamente con el archivo SQLite usando librerías de Python.

Lo primero que hay que hacer es crear el archivo SQLite con las tablas necesarias, véase la Figura 10, para gestionar la información que queramos. En este sistema hay que gestionar principalmente 3 grupos de información:

- Configuración: En esta tabla se va a guardar el id de aplicación y la clave secreta, parámetros que se usan para la comunicación con los servidores de Latch usando una API, y el tiempo de recarga, parámetro que se usa para realizar una pausa entre cada búsqueda de dispositivos conectados a la red. Además, en el supuesto crecimiento del sistema se guardará en esta tabla cualquier parámetro de configuración adicional.
- Usuarios: En esta tabla se guardan los dispositivos de control que podrán usarse para gestionar la red. En el sistema actual solo se podrá gestionar desde un dispositivo, pero Latch permite la gestión conjunta de dispositivos por lo que en futuras versiones del sistema se podría implementar esta opción y almacenar en esta tabla más de un dispositivo.
- Dispositivos: En esta tabla se almacena información básica de cada dispositivo que está actualmente conectado a la red. Entre otros parámetros se almacena el Alias o la última conexión. En futuras versiones del sistema se podría almacenar más información que luego se puede utilizar para fines de análisis, etc.

Por lo tanto, en esta versión del sistema existirán tres tablas sin relación entre ellas:

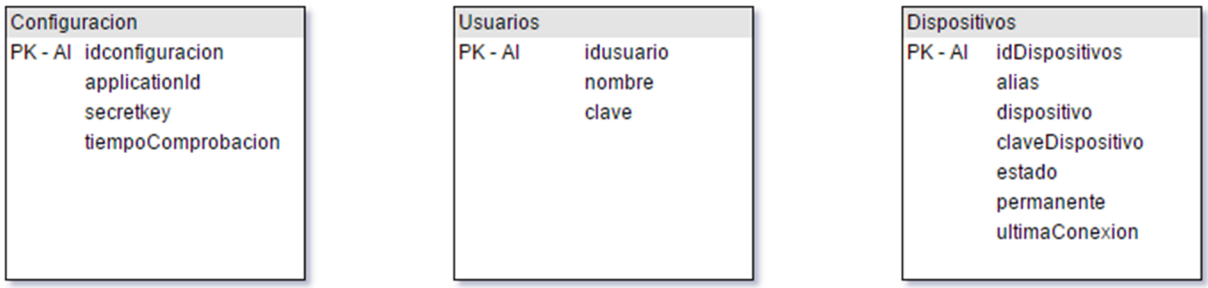


Figura 10. Tablas del archivo SQLite

Por otra parte, una vez creado el archivo SQLite, hay que instalar la librería necesaria para que Python pueda acceder y trabajar con ese fichero. Esta librería se llama python-sqlite3 y se instala fácilmente con el gestor de paquetes OPKG (Imagen 7).

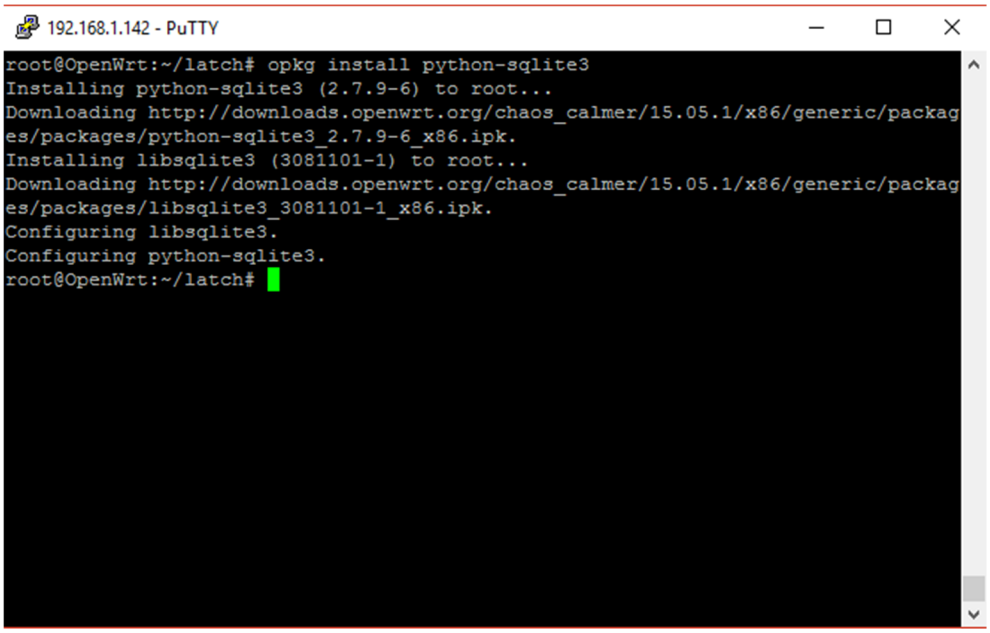


Imagen 7. Instalando python-sqlite3

Con la librería ya instalada se puede interactuar con la base de datos de forma muy sencilla. A continuación, se muestra un ejemplo de obtención de datos (Código 8) y otro de eliminación de datos (Código 9).

```
def cargar_accountid(self):
	con = lite.connect(self.db)
	with con:
		cur = con.cursor()
		cur.execute('SELECT clave FROM Usuarios')
		accountid = cur.fetchone()
		if str(accountid) == "None":
			return ""
		return str(accountid[0])
```

Código 8. Obteniendo datos de SQLite con Python

```
def eliminar_accountid(self):
	con = lite.connect(self.db)
	with con:
		cur = con.cursor()
		cur.execute("DELETE FROM Usuarios”)
		cur.execute("DELETE FROM Dispositivos")
```

Código 9. eliminando datos a SQLite con Python

5.3. Implementación de la interfaz web

Para poder administrar todo el sistema de forma eficiente hay que crear una interfaz web que permita gestionar los principales parámetros y que quede integrada en la interfaz ya existe de administración de OpenWRT.

Para realizar esta interfaz se utiliza el patrón Modelo – Vista – Controlador.

El modelo es el archivo SQLite que se ha creado anteriormente, que guardará toda la información, y que se ha descrito en la sección 5.2.

El controlador en el sistema web de OpenWRT es un archivo programado con Lua que ejecutará las acciones que se pidan. A su vez este controlador Lua utilizará el modelo para gestionar los datos, y realizará llamadas al servicio interno que hemos desarrollado para realizar las acciones de pareado y despareado.

Lo primero a realizar en el controlador es la creación del menú de navegación (Código 10), asociando a cada enlace del menú una acción del controlador.

```
function index()
    entry({"admin", "Latch"}, firstchild(), "Latch", 40).dependent=false
    entry({"admin", "Latch", "Configuracion"}, call("accion_configuracion"),
"Configuration", 1)
    entry({"admin", "Latch", "Pareado"}, call("accion_pareado"), "Pair / Unpair", 2)
    entry({"admin", "Latch", "Dispositivos"}, call("accion_dispositivos"), "Devices", 3)
    entry({"admin", "Latch", "Ayuda"}, call("accion_ayuda"), "Help", 4)
    entry({"admin", "Latch", "editarDispositivos"}, call("accion_editarDispositivos"), nil)
end
```

Código 10. Creación de menú de navegación

Una vez creado el menú de navegación hay que crear una acción para cada enlace, véase el código 11 como ejemplo de una acción.

```
function accion_pareado()
        if luci.http.formvalue("parear") then
               if nixio.fork() == 0 then
                nixio.exec("/root/Latch/Latch-OpenWRT.py", luci.http.formvalue("usuario"),
luci.http.formvalue("parear"))
                io.stderr:write("Can't exec Reg\n")
        end
        sleep(30)
            luci.http.redirect(
                    luci.dispatcher.build_url("admin", "Latch", "Pareado")
                )
        elseif luci.http.formvalue("desparear") then
        if nixio.fork() == 0 then
                        nixio.exec("/root/Latch/Latch-OpenWRT.py",
luci.http.formvalue("desparear"))
        end
        sleep(30)
                luci.http.redirect(
                        luci.dispatcher.build_url("admin", "Latch", "Pareado")

                )
        else
                luci.template.render("Latch/pareado")
        end
end
```

Código 11. Creando acción para el pareado

La vista es la interfaz que verá el usuario y que realizará unos controles de información antes de que se pida al controlador ninguna acción o información.

En la vista se usa HTML para la estructura básica. Con HTML se crea los formularios, listas con información, enlaces entre páginas, y se crean los elementos que luego a través de un archivo de diseño CSS o directamente en dichos elementos con la etiqueta “class” recibirán un diseño visual más llamativo para los usuarios. En el código 12 podemos ver el archivo HTML que se ha creado para guardar la configuración del sistema.

```
<%-
local driver = require "luasql.sqlite3"
env = assert (driver.sqlite3())
con = assert (env:connect("/root/Latch/Latch.db"))
cur = assert (con:execute"SELECT applicationId, secretkey, tiempoComprobacion FROM Configuracion")
row = cur:fetch ({}, "a")
-%>
<%+header%>
<% local eating = luci.model.uci.cursor():get("current", "ice", "flavor") %>
<link rel="stylesheet" href="/estilo.css"/>
<div class="cbi-map" id="cbi-system">
  <h2><a id="content" name="content">Configuration</a></h2>
  <div class="cbi-map-descr">Configuration parameters of the plugin</div>
  <fieldset class="cbi-content">
    <form action="<%=REQUEST_URI%>" method="get" class="cbi-section-node cbi-section-node-tabbed">
      <div class="cbi-value">
        <label class="cbi-value-title" for="applicationId">Application ID</label>
        <div class="cbi-value-field">
          <input class="" type="text" id="applicationId" name="applicationId" value='<%- if row.applicationId== nil then io.write("") else io.write(row.applicationId) end -%>'/>
        </div>
      </div>
      <div class="cbi-value">
        <label class="cbi-value-title" for="secretkey">Secret Key</label>
        <div class="cbi-value-field">
          <input class="" type="text" id="password" name="secretkey" value="<%- if row.secretkey == nil then io.write("") else io.write(row.secretkey) end -%>"/>
        </div>
      </div>
      <div class="cbi-value">
        <label class="cbi-value-title" for="tiempoComprobacion">Reload time</label>
        <div class="cbi-value-field">
          <input class="" type="text" id="tiempoComprobacion" name="tiempoComprobacion" value="<%- if row.tiempoComprobacion == nil then io.write("") else
io.write(row.tiempoComprobacion) end -%>"/>
          <span id="msj">Time in seconds to reload device table</span>
      <span id="error" style="display:none;">The minimun time valor is 15s and we recomend 20s</span>
    </div>
      </div>
      <div class="cbi-page-actions">
          <input type="submit" id="guardar" value="Guardar" class="cbi-button cbi-button-save"/>
      </div>
    </form>
  </fieldset>
</div>
<script type="text/javascript" src="/configuracion.js"></script>
<%-
-- close everything
cur:close() -- already closed because all the result set was consumed
con:close()
env:close()
-%>
<%+footer%>
```

Código 12. HTML para el archivo de configuración

Como se ha comentado anteriormente, se usa CSS para el diseño visual, que en OpenWRT basta con utilizar clases ya creadas de diseño de OpenWRT en las etiquetas en HTML cómo puede verse en el código 12.

No obstante, se puede crear un diseño personalizado si así se quisiese, pero el objetivo en este caso es que las interfaces creadas sean idénticas a las ya existentes en OpenWRT. Algunos ejemplos claros de esto pueden verse en la sección “Anexo – Diseño web del proyecto”.

Finalmente se puede utilizar JavaScript para el control de la información antes de que sea enviada al controlador.

Algunos ejemplos de control de información de este proyecto serian:

- No está vacío el campo
- Tiene un formato valido (Por ejemplo, direcciones MAC)
- Tiene un tamaño exacto (Por ejemplo, los códigos de pareado)
- Avisos de opciones importantes (Por ejemplo, poner dispositivos permanentes)

En el código 13 se puede ver un ejemplo de uso de JavaScript para realizar numerosas comprobaciones en el formulario de configuración antes de que esté pida al controlador que se almacenen los datos.

En este archivo JavaScript se controla que ninguno de los campos del formulario estén vacíos, y además se controla que el campo de tiempo de recarga no tenga un valor menor a 15 segundos.

También se aprovecha JavaScript para avisar por pantalla de estos errores, ya sea poniendo los campos de color rojo con error y verde cuando están bien, o mostrando directamente un mensaje de error al lado del campo correspondiente.

```
var aplicacionID = document.getElementById("applicationId");
var password = document.getElementById("password");
var tiempo = document.getElementById("tiempoComprobacion");
var error = document.getElementById("error");
var msj = document.getElementById("msj");
var boton = document.getElementById("guardar");
var error1 = false;
var error2 = false;
var error3 = false;

function comprobarCampos(e){
    if(aplicacionID.value == "" || /^\s*$/.test(aplicacionID.value)){
        aplicacionID.classList.add("vacio");
        aplicacionID.classList.remove("valido");
        error1 = true;
    }else{
        aplicacionID.classList.remove("vacio");
        aplicacionID.classList.add("valido");
        error1 = false;
    }
    if(password.value == "" || /^\s*$/.test(password.value)){
        password.classList.remove("valido");
        password.classList.add("vacio");
        error2 = true;
    }else{
        password.classList.remove("vacio");
        password.classList.add("valido");
        error2 = false;
    }
    if(tiempo.value == "" || /^\s*$/.test(tiempo.value) || !/^\d*$/.test(tiempo.value) ||
tiempo.value < 15){
    if(tiempo.value < 15){
        error.setAttribute("style", "");
        msj.setAttribute("style", "display:none;");
    }
        tiempo.classList.remove("valido");
        tiempo.classList.add("vacio");
        error3 = true;
    }else{
    tiempo.classList.remove("vacio");
        tiempo.classList.add("valido");
        error3 = false;
    }
    if(error1 || error2 || error3){
        e.preventDefault();
    }
}

boton.addEventListener("click", comprobarCampos);
```

Código 13. JavaScript utilizado para control de formulario de configuración

A lo largo de esta sección se ha visto como ejemplo completo la creación de la sección de configuración. Evidentemente hay que crear el resto de archivos HTML, CSS y JavaScript necesarios para todas las secciones de administración web del sistema.

5.4. Diagrama de flujo del sistema – general

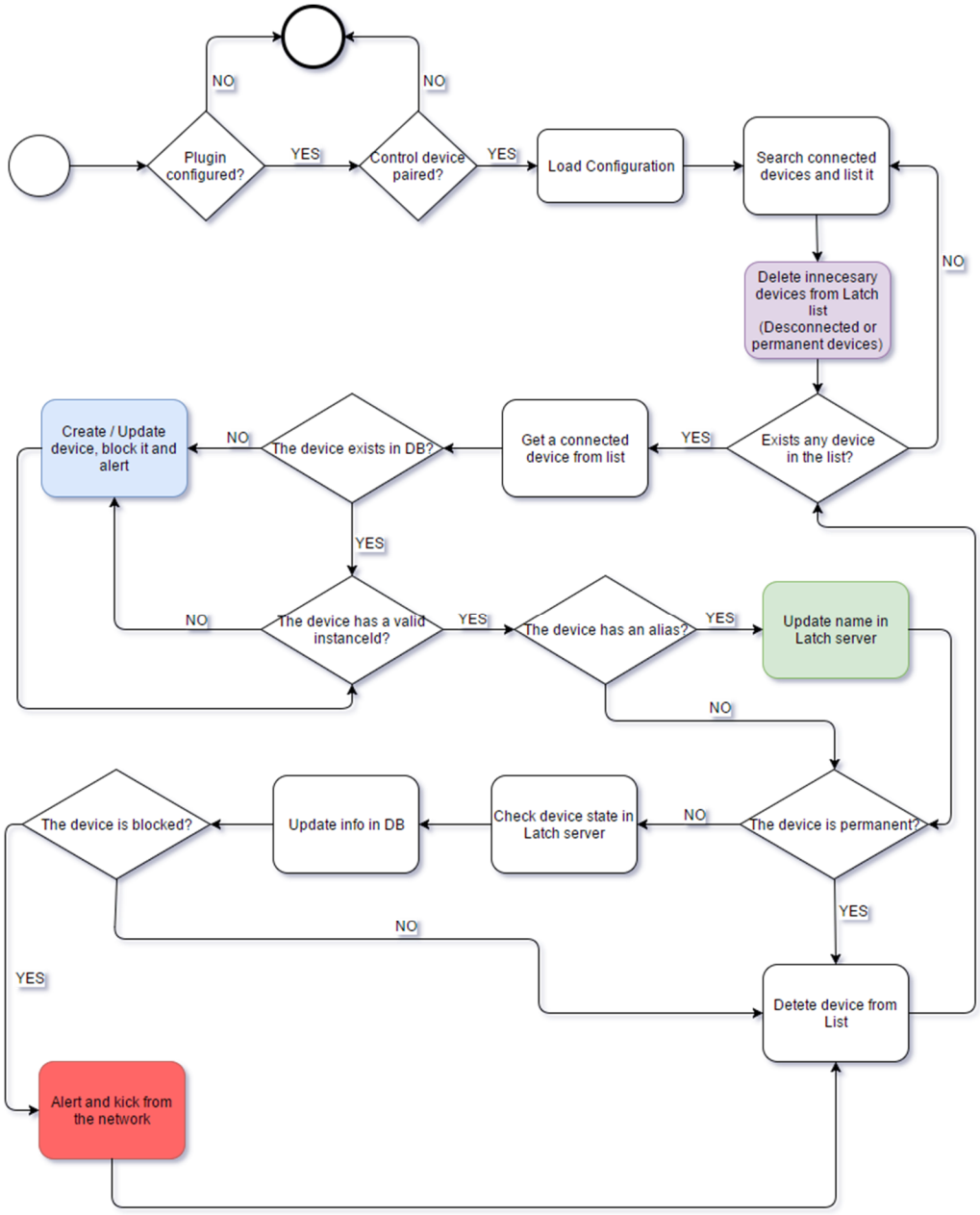


Figura 11. Diagrama de flujo general

Como se puede observar hay distintas operaciones especiales marcadas con colores en el diagrama de flujo (Figura 11). Cada operación especial tiene un estado o alerta asociada en la aplicación Latch que se explicará ahora. En la siguiente página se podrá ver un ejemplo completo de ejecución.

La primera vez que instalas e inicias el servicio del sistema la lista en la aplicación Latch estará vacía, y la operación asociada al color **morado** no se ejecutará.

Cuando el servicio detecte un nuevo dispositivo conectándose (Que no exista en la base de datos), se entrará en la operación asociada al color **azul**, y el servicio creará un nuevo dispositivo (Imagen 8), lo bloqueará (Imagen 9) y avisará al usuario en la aplicación de Latch (Imagen 10).

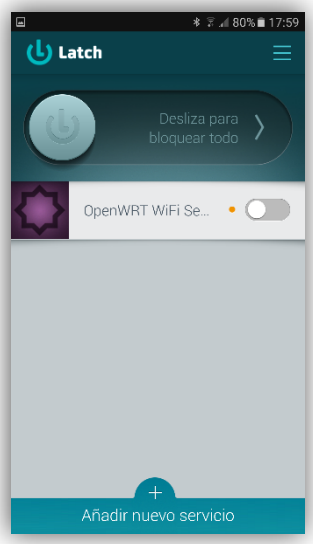


Imagen 8. Crea dispositivo

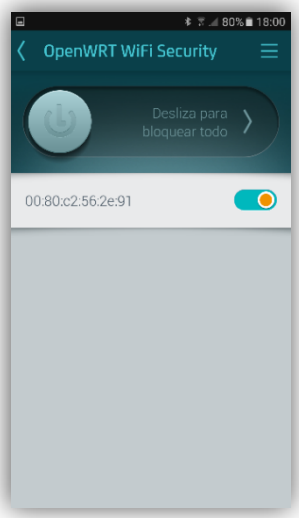


Imagen 9. Bloquea dispositivo

Imagen 10. Aviso bloqueo

Ahora damos permisos y desbloqueamos el dispositivo, y si queremos podemos entrar al menú de administración de dispositivos (Imagen 11) del sistema de Latch y podremos editar el dispositivo para ponerle un alias (Imágenes 12 y 13).

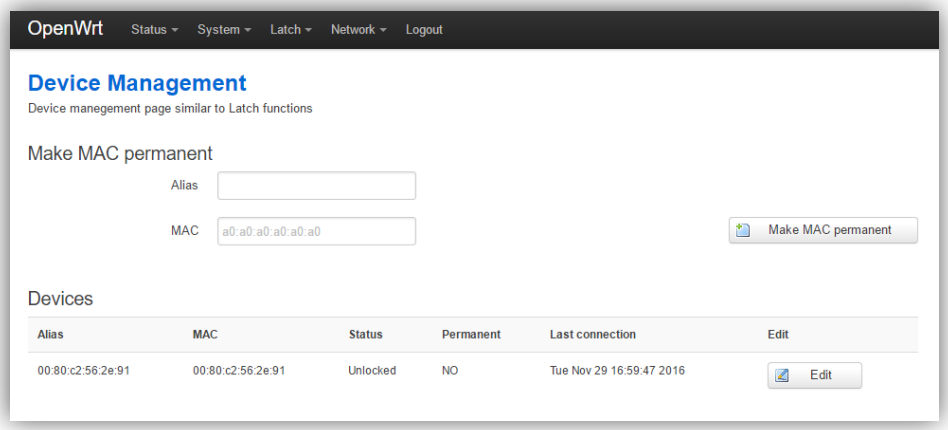


Imagen 11. Lista de dispositivos antes de actualización de alias

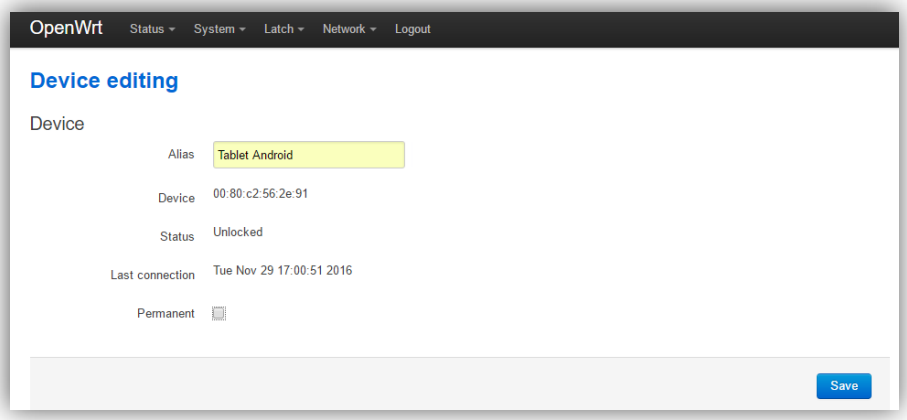


Imagen 12. Editando un dispositivo para poner un alias

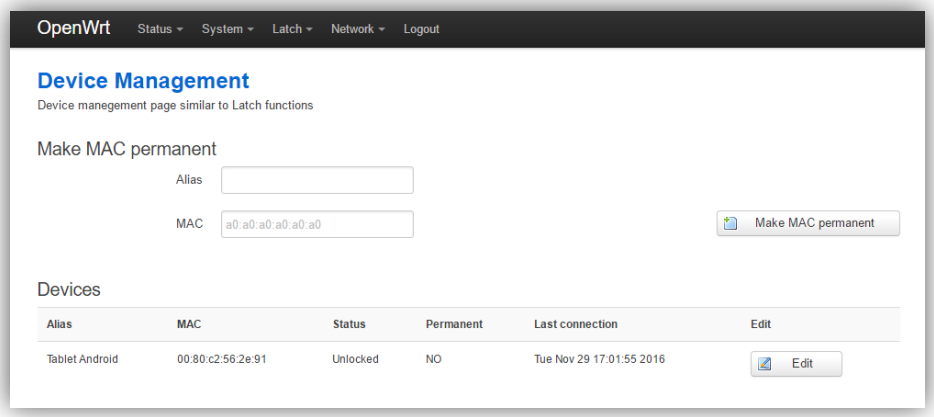


Imagen 13. Lista de dispositivos después de actualización de alias

Ahora debemos esperar el tiempo de recarga. Cuando el servicio del sistema detecte el cambio entraremos en la operación asociada al color **verde**, y se actualizará en Latch (Imagen 14)

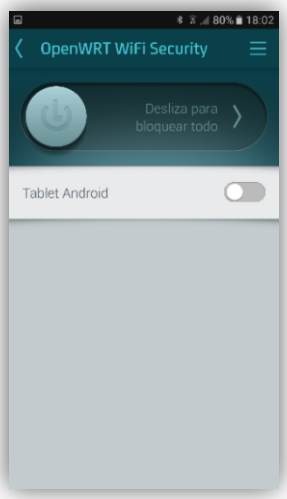


Imagen 14. Alias

Si bloqueamos el dispositivo, después del tiempo de recarga entraremos en la operación asociada al color **rojo** y el sistema expulsará el dispositivo y nos alertará (Imagen 15).

Imagen 15. Intento acceso bloqueado

Finalmente, cuando iniciamos otra vez el flujo del programa y se vuelva a comprobar el dispositivo, tenemos distintas posibilidades:

1. El dispositivo continúa intentando conectar.
    a. Será continuamente expulsado y se recibirán alertas si no se silencian.
2. Ponemos el dispositivo como permanente.
3. El dispositivo que fue expulsado deja de intentar conectarse.

En los casos 2 y 3 entraremos en la operación asociada al color **morado**. Ya sea el dispositivo permanente o haya dejado de intentar conectarse será borrado de la lista de Latch. (Imagen 16)

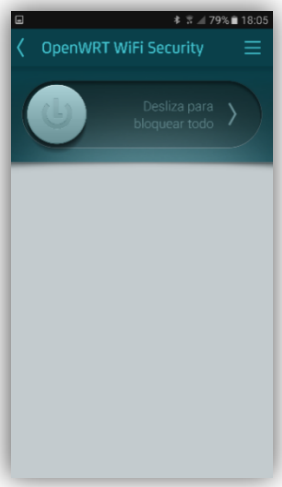


Imagen 16. Lista vacía

5.5. Diagrama de flujo del sistema – (des)pareado

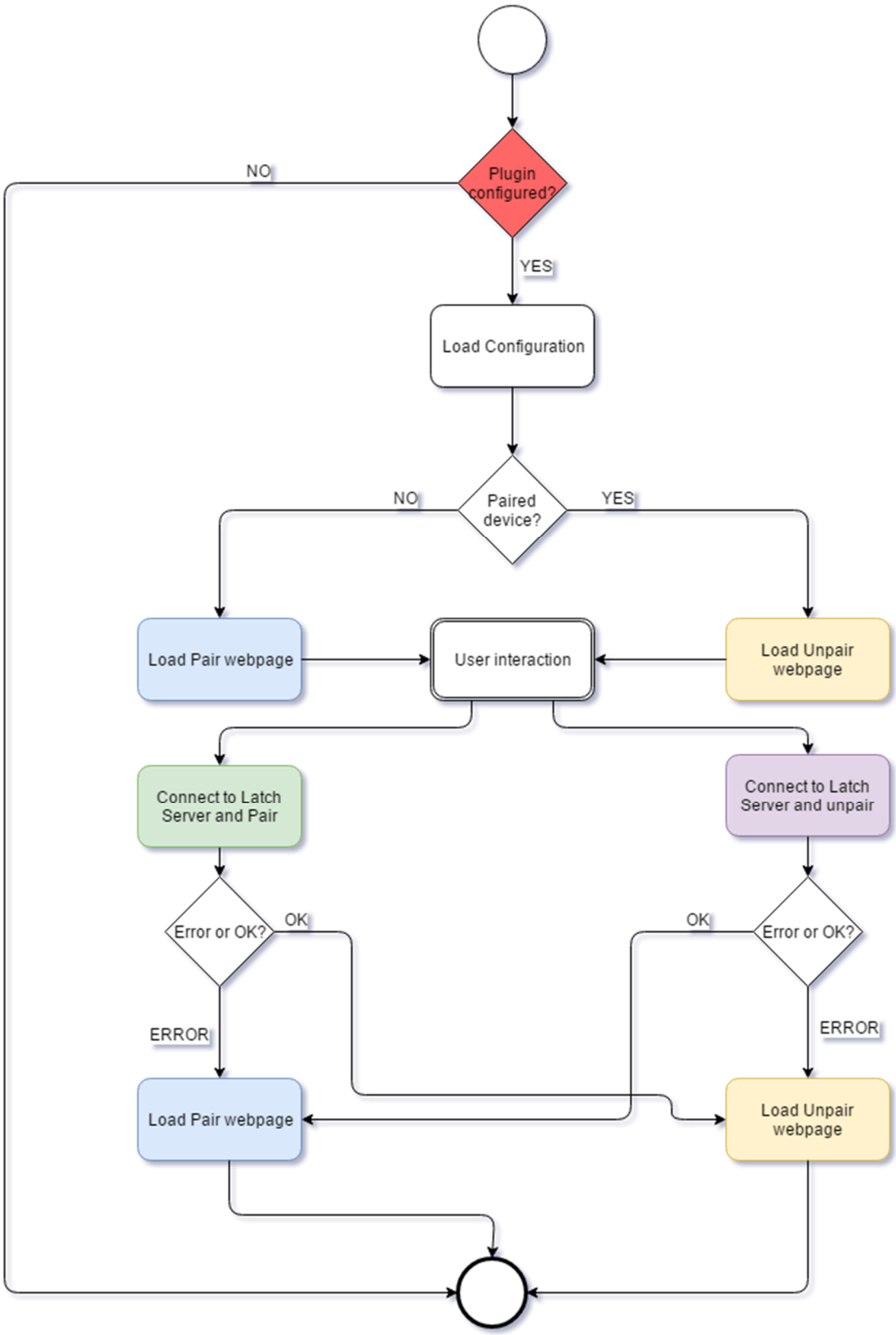


Figura 12. Diagrama de flujo de pareado / despareado

Como en el otro diagrama tenemos operaciones o estados especiales asociados a colores en el diagrama de flujo de pareado y despareado (Figura 12) que se explicarán con un ejemplo de ejecución del programa.

Lo primero es comprobar si el sistema está configurado (Imagen 17), la operación de color **rojo**. Es interesante ver que si no está configurado el sistema no te deja entrar a las páginas de pareado y despareado (Imagen 18).

Esto solo puede suceder cuando has instalado el sistema, ya que una vez que lo configuras no puedes dejar los campos vacíos al menos que lo reinstales.

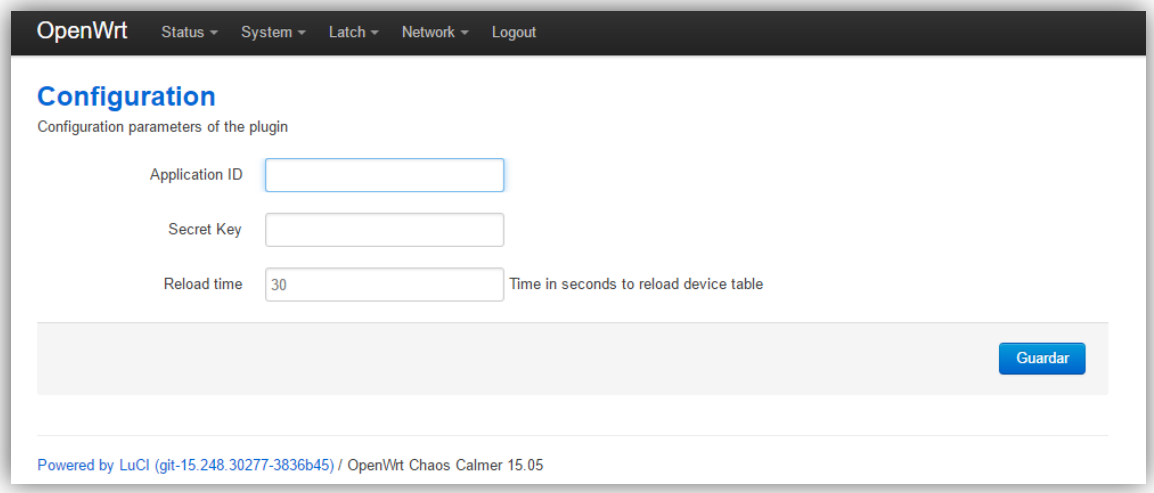


Imagen 17. Sección de configuración de LatchOpenWRT con parámetros vacíos

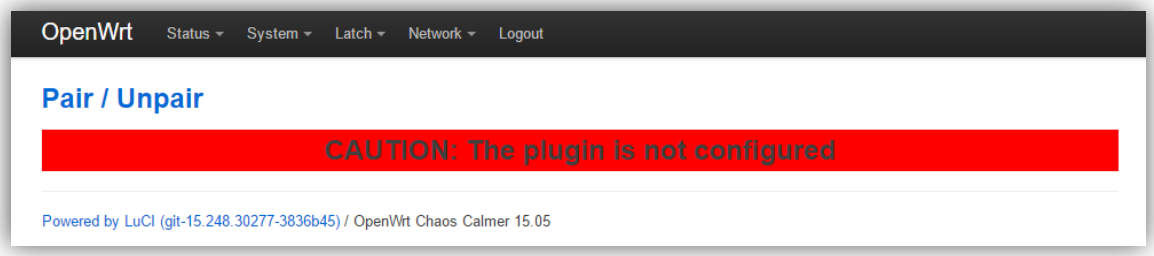


Imagen 18. Página de pareado/despareado de LatchOpenWRT con error

Una vez configurado los parámetros del sistema ya podremos acceder a las páginas de pareado (Imagen 19) y despareado e inicialmente no tendremos ningún dispositivo pareado, por lo que se cargará la página de pareado, el estado de color **azul**. Ahora necesitamos usar la aplicación Latch para obtener el código de pareado (Imágenes 20 y 21).

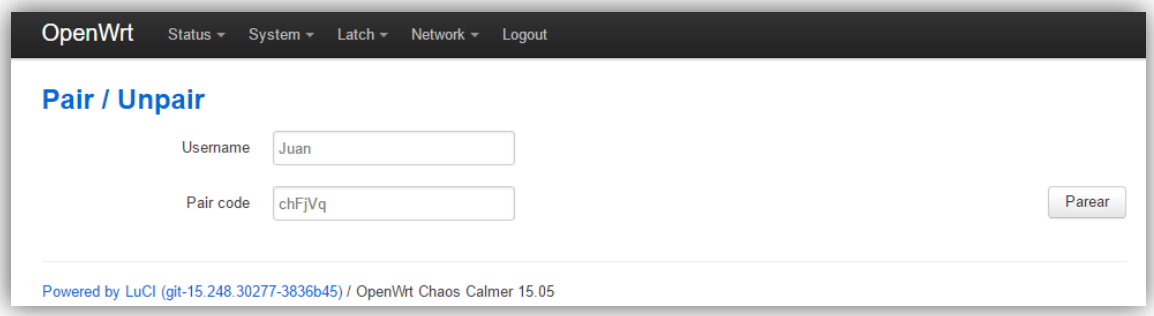


Imagen 19. Página de pareado de LatchOpenWRT

Imagen 20. No hay pareado

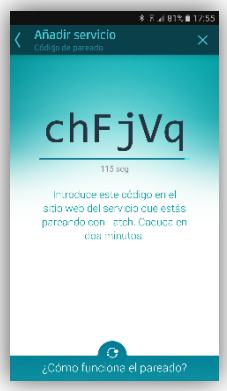


Imagen 21. Cod. de pareado

Necesitaremos introducir el código de pareado y un nombre para reconocer al dispositivo. Si todo es correcto el servicio del plugin se conectará al servidor de Latch y pareará el dispositivo, la operación asociada al color **verde**. Recibirás una notificación en la aplicación Latch (Imagen 22 y 23). En caso de error el plugin cargará de nuevo la página de pareado.

Imagen 22. Aviso de pareado

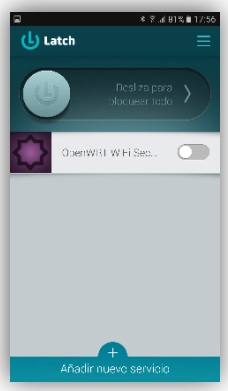


Imagen 23. Dispositivo pareado

Después de parear se cargará la página de despareado (Imagen 24), el estado asociado al color amarillo.

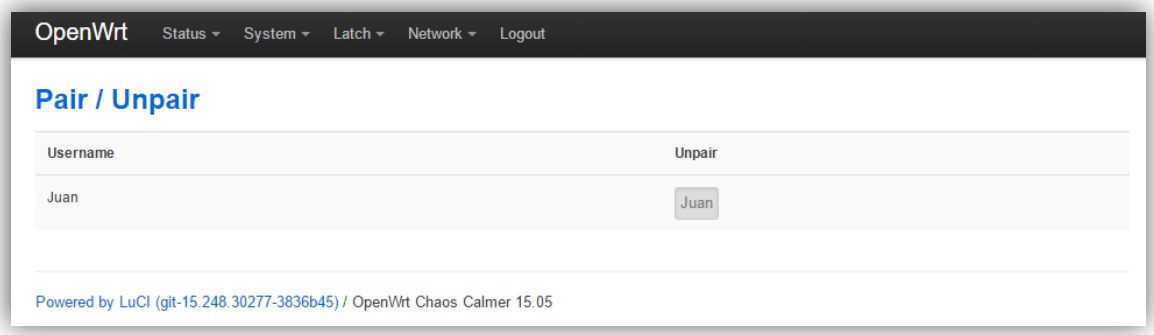


Imagen 24. Pareado guardado en la web

Si quieres desparear el dispositivo solo se necesita darle al botón y el plugin intentará conectar con el servidor de Latch para desparearlo, el estado asociado al color **morado**. Si el despareado fue correcto se cargará la página de pareado (Imagen 19), se avisará (Imagen 25) y se borrará de la aplicación Latch (Imagen 26). En caso de error se cargará de nuevo la página de despareado.

Imagen 25. Despareado correcto

Imagen 26. Lista aplicación vacía

5.6. Diagrama de componentes

En esta sección se muestra un diagrama de componentes (Figura 13) que intenta enseñar de forma generalizada como interacciona la capa de administración web con el resto de la aplicación, así como la interacción del servicio interno con dichos componentes.

Como se puede apreciar la capa web llama a un controlador activando una operación distinta según la tarea a realizar, y este controlador usando la API de Latch se conecta a los servidores realizando la operación correspondiente. Además, estas operaciones de la capa web interactúan con el archivo SQLite que tiene para almacenar, modificar, eliminar o recuperar información según lo necesite.

Por otro lado, el servicio interno, independiente de la capa web, realiza sus operaciones de control por cada dispositivo conectado e interactúa con la API de Latch para pedir información sobre el estado de los dispositivos.

El funcionamiento de las operaciones principales del sistema puede verse con más profundidad en diagramas de flujo en las siguientes secciones.

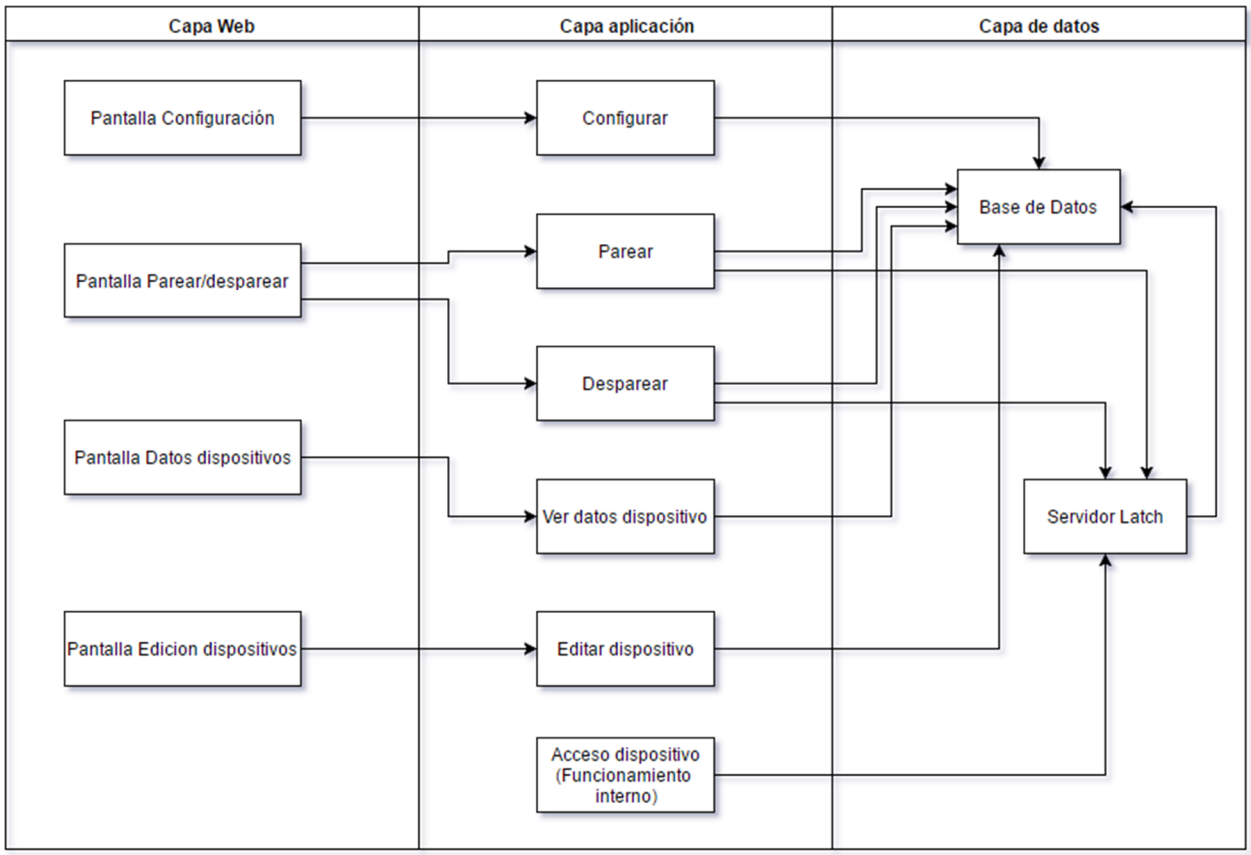


Figura 13. Diagrama de componentes

5.7. Creación del paquete de instalación

Para crear el paquete de instalación del sistema y poder instalarlo en OpenWRT utilizando el sistema de gestión de paquetes OPKG hay que utilizar una distribución Linux como Ubuntu.

La utilidad opkg (una bifurcación del proyecto ipkg) es un gestor de paquetes ligero que se utiliza para descargar e instalar paquetes OpenWRT desde repositorios de paquetes locales o situados en Internet.

Los usuarios de GNU / Linux que ya están familiarizados con apt-get, aptitude, pacman, yum, etc. reconocerán las similitudes. También tiene similitudes con Optware de NSLU2, también hecho para dispositivos embebidos.

OPKG es sin embargo un gestor de paquetes completo para el sistema de archivos raíz, en lugar de sólo una forma de agregar software a un directorio separado (por ejemplo, /opt). Esto también incluye la posibilidad de agregar módulos y controladores del kernel.

A diferencia de otros sistemas de paquetes, al mirar uno de los makefiles de paquetes para OPKG, difícilmente lo reconocería como un makefile. El makefile se ha transformado en una plantilla orientada a objetos que simplifica todo el calvario del formato makefile estándar.

Para crear el paquete OPKG, el primer paso consiste en instalar una serie de prerrequisitos (Imagen 27) para posteriormente poder clonar el SDK de OpenWRT y proceder a la compilación.

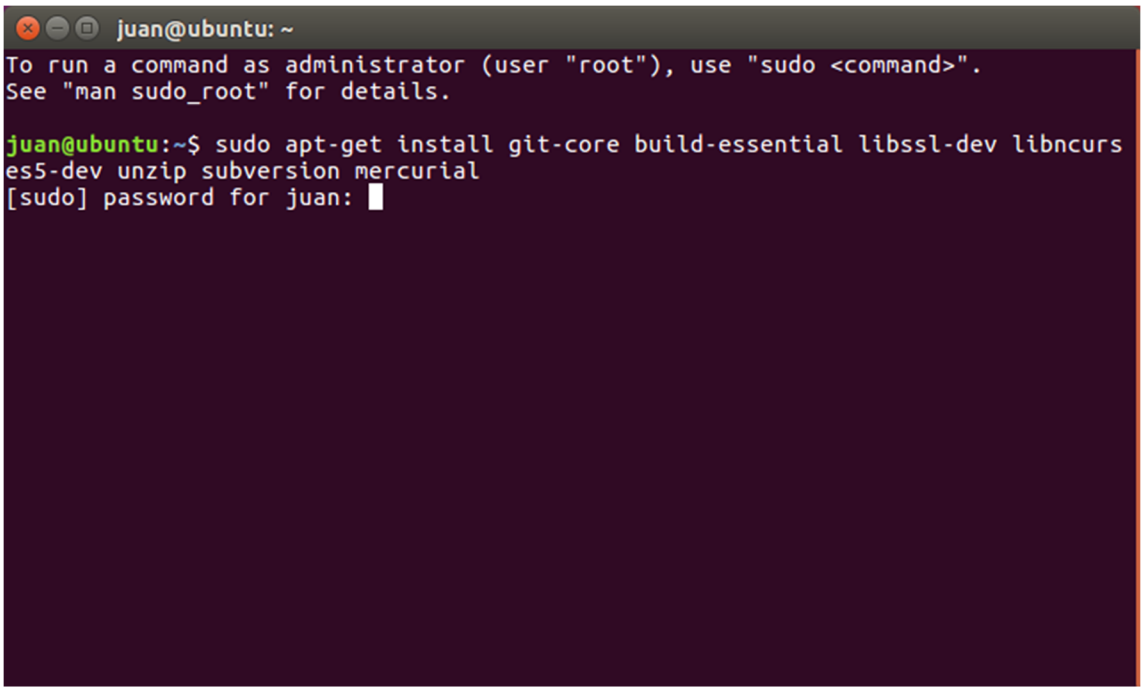


Imagen 27. Instalación de dependencias

Una vez instalado los prerrequisitos podemos clonar el SDK con Git (Imagen 28). Previamente hay que crear una carpeta “Instalación” en la que clonar el repositorio.

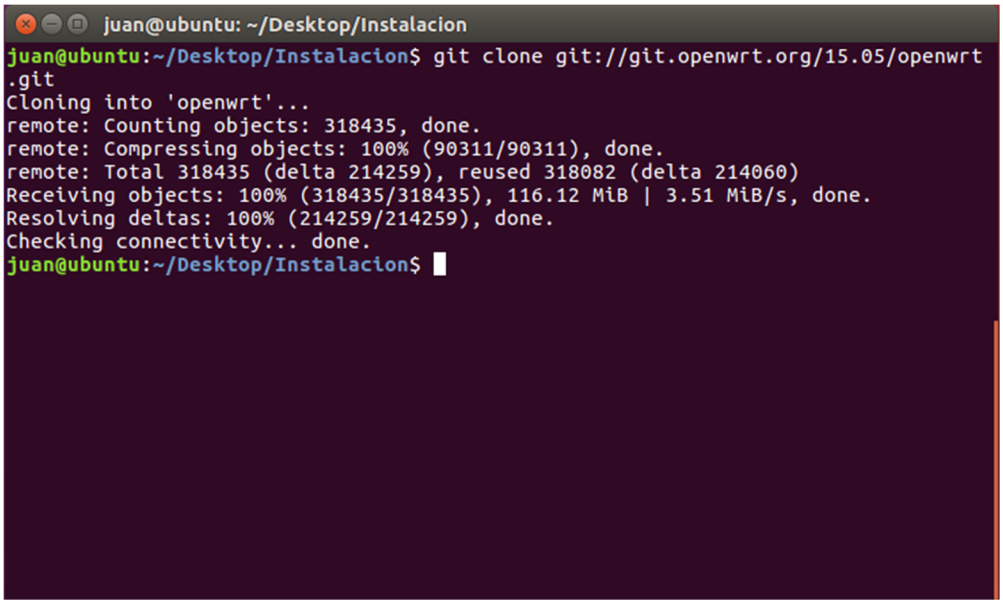


Imagen 28. Clonado de SDK de OpenWRT

Una vez clonado el SDK hay que crear el paquete que luego se usará para realizar la compilación, y crear un MakeFile (Imagen 29) para indicar el procedimiento que se realizará durante la compilación, instalación y desinstalación del paquete.

```
include $(TOPDIR)/rules.mk

PKG_NAME:=LatchOpenWRT
PKG_VERSION:=1.0.0
PKG_RELEASE:=1
PKG_MAINTAINER:=Juan Camero
PKG_LICENSE:=LGPL-2.1
PKG_CONFIG_DEPENDS:=luasql-sqlite3 ca-certificates python-light python-codecs python-sqlite3 python-logging python-openssl
include $(INCLUDE_DIR)/package.mk

PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)-$(PKG_VERSION)

define Package/LatchOpenWRT
  SECTION:=utils
  CATEGORY:=Utilities
  DEPENDS:=+luasql-sqlite3 +ca-certificates +python-light +python-codecs +python-sqlite3 +python-logging +python-openssl
  TITLE:=Latch plugin to control Wi-Fi access
  URL:=https://github.com/JCameroMartin/LatchOpenWRT
  MENU:=1
endef

define Package/LatchOpenWRT/description
 Plugin to integrate Latch technology from Telefonica with OpenWRT for allow or block the access to the network of devices through Wi-Fi
endef

define Build/Prepare
    mkdir -p $(PKG_BUILD_DIR)
    $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

define Build/Compile
endef

define Package/LatchOpenWRT/install
    $(INSTALL_DIR) $(1)/root/Latch
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/Latch/* $(1)/root/Latch/
    $(INSTALL_DIR) $(1)/usr/lib/lua/luci/controller/Latch
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/controller/Latch/* $(1)/usr/lib/lua/luci/controller/Latch/
    $(INSTALL_DIR) $(1)/usr/lib/lua/luci/view/Latch
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/view/Latch/* $(1)/usr/lib/lua/luci/view/Latch/
    $(INSTALL_DIR) $(1)/www
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/www/* $(1)/www/
    $(INSTALL_DIR) $(1)/www/images
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/images/* $(1)/www/images/
    $(INSTALL_DIR) $(1)/etc/init.d
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/initd/* $(1)/etc/init.d/
endef

$(eval $(call BuildPackage,LatchOpenWRT))
```

Imagen 29. Plantilla MakeFile

Teniendo ya el paquete creado se necesita configurar el SDK de OpenWRT con la finalidad de tener el paquete en cuenta y que se compile al realizar los comandos adecuados.

Para realizar esta configuración hay que usar los siguientes comandos.

```
make defconfig

make prereq

make menuconfig
```

El último comando abrirá la ventana de configuración (Imagen 30).

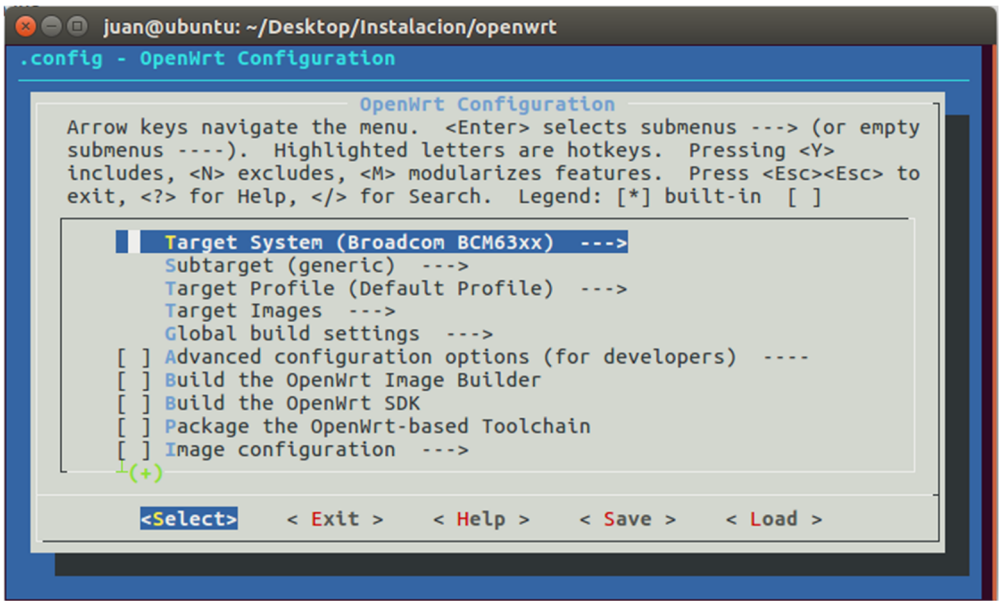


Imagen 30. Selección del sistema para el que compilar

Se deberá seleccionar el sistema para el que queremos compilar en “Target System”, que como se puede ver en la imagen anterior yo tengo seleccionado el “Broadcom BCM63xx” que es el SoC que usa mi router.

Además, debemos entrar al seleccionador de paquetes (Imagen 31), buscar el nuestro y seleccionarlo de forma que la configuración se establezca como “Modular”.

Lo anterior garantiza que al usar los comandos de compilación solo se compile el paquete que queremos y sus requisitos, y no la imagen entera de OpenWRT.

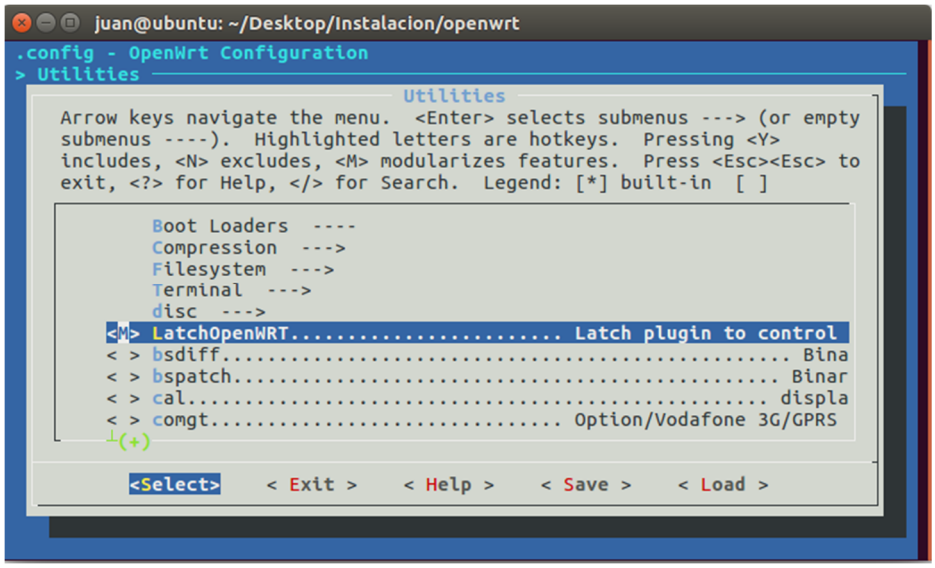


Imagen 31. Selección del paquete a compilar

Podemos ver la información del paquete (Imagen 32) que hemos seleccionado, ofrecida por el makefile que hemos creado previamente, dándole a "?" en el paquete que queramos.

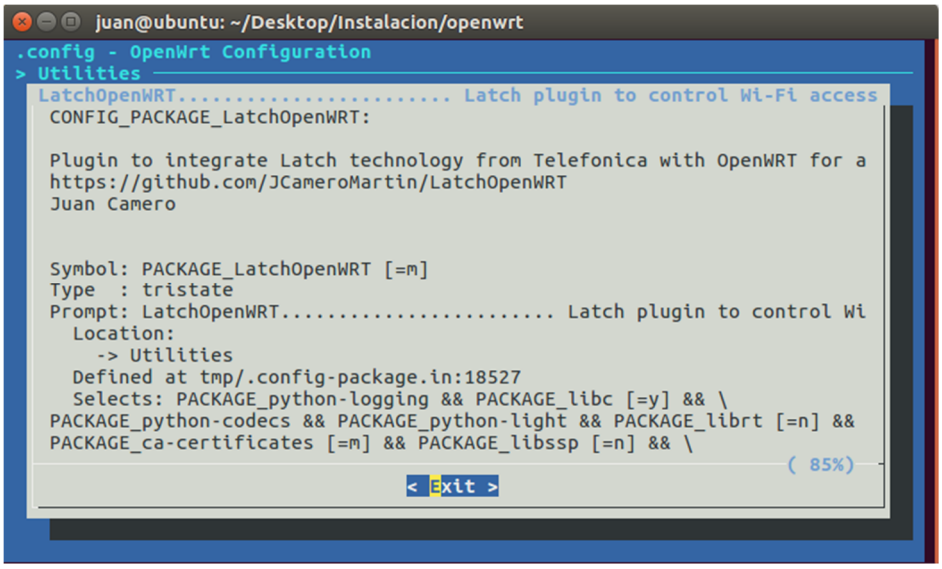


Imagen 32. Información del paquete

Podemos cerrar las ventanas anteriores y proceder a abrir una consola de comandos con la finalidad de ejecutar la compilación y que se cree el paquete. Este proceso es bastante largo y puede tardar más de una hora.

```
make tools/install ; make toolchain/install

make package/LatchOpenWRT/compile
```

Con esto ya tendremos creado el paquete en la carpeta bin.

6. Configuración del sistema

Sinopsis:

En este capítulo se explican los pasos necesarios para poner en funcionamiento el sistema en cuanto a configuración.

Se explica cómo se puede crear la cuenta Latch, y la aplicación Latch necesaria para obtener los parámetros de configuración que luego se introducirán a través de la interfaz de administración del sistema.

6.1. Creación de la cuenta para usar Latch

Crear una cuenta para usar Latch es muy sencillo. Solo tenemos que entrar en https://latch.elevenpaths.com/, darle a Área de Desarrolladores y al botón de registrarse como desarrollador. En la ventana que se abre solo tenemos que darle a la pestaña de "Crear una nueva cuenta de desarrollador" y rellenar los datos (Imagen 33).

Posteriormente sólo tendremos que validar la cuenta (Imagen 34) desde el email para tener una cuenta Latch.

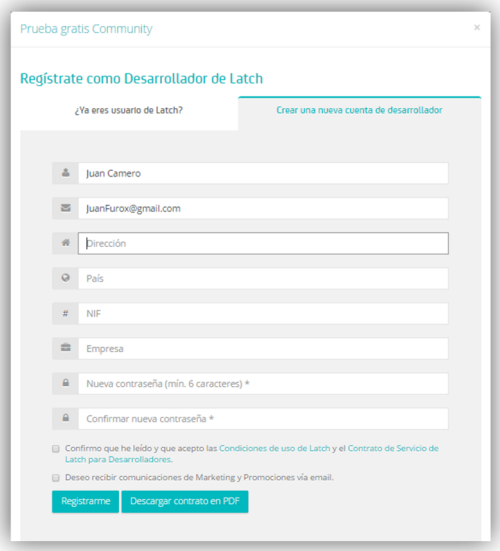


Imagen 33. Creación de cuenta

Imagen 34. Activación de cuenta

6.2. Creación de aplicación Latch

Una vez en el menú de usuario deberemos crear una aplicación. Para ello basta con darle a la opción de “Mis aplicaciones” en el menú lateral izquierdo para entrar en la interfaz de gestión de aplicaciones, véase imagen 35.

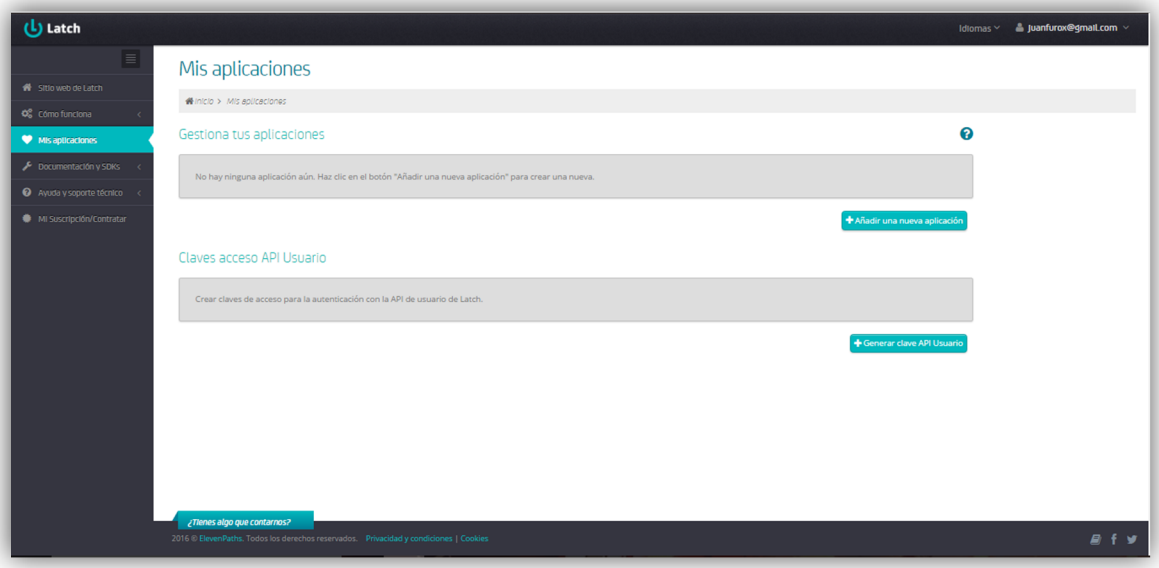


Imagen 35. Gestión de aplicaciones Latch

Ahora hay que darle a la opción “Añadir una nueva aplicación” y poner un nombre a la aplicación que queremos crear (Imagen 36).

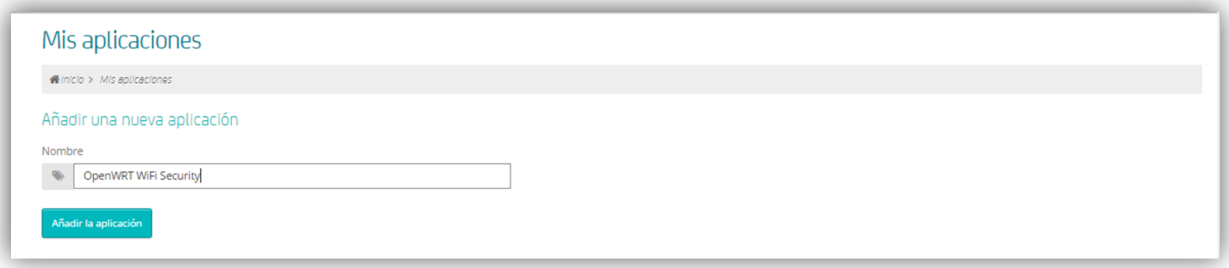


Imagen 36. Creación de aplicación Latch

Con la aplicación creada ya tendremos acceso al APP ID y a la Secret Key de la aplicación (Imagen 37) que hemos creado y que necesitamos para configurar el plugin y empezar a trabajar con él.

El identificador de Aplicación es una cadena que identifica a un proveedor de servicio cuando interactúa con Latch. Se asigna automáticamente a la Aplicación cuando la creas y no se puede cambiar.

Puedes comprobar el ID de Aplicación editando la Aplicación en la sección “Mis aplicaciones” del área privada de Desarrolladores de Latch.

Todos los ID identificadores de Aplicación llevan asociado un Secreto, que se utiliza para firmar todas las solicitudes desde la Aplicación en nuestra API. Se debe proteger la seguridad del Secreto en todo momento, no compartir el Secreto con nadie y no distribuirlo, ni en código ni en configuración. Puede ser modificado en cualquier momento pulsando el botón Renovar en la aplicación.

El secreto básicamente sirve para comprobar que las peticiones realizadas son legítimas, es decir, peticiones realizadas por la aplicación que debe realizarlas, permitiendo así comunicaciones seguras usando la API de Latch que usa el sistema.

El funcionamiento general es muy parecido a OAuth [12] (Open Authorization), que es un protocolo que permite flujos simples de autorización para sitios web o aplicaciones informáticas. Se trata de un protocolo que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y web.

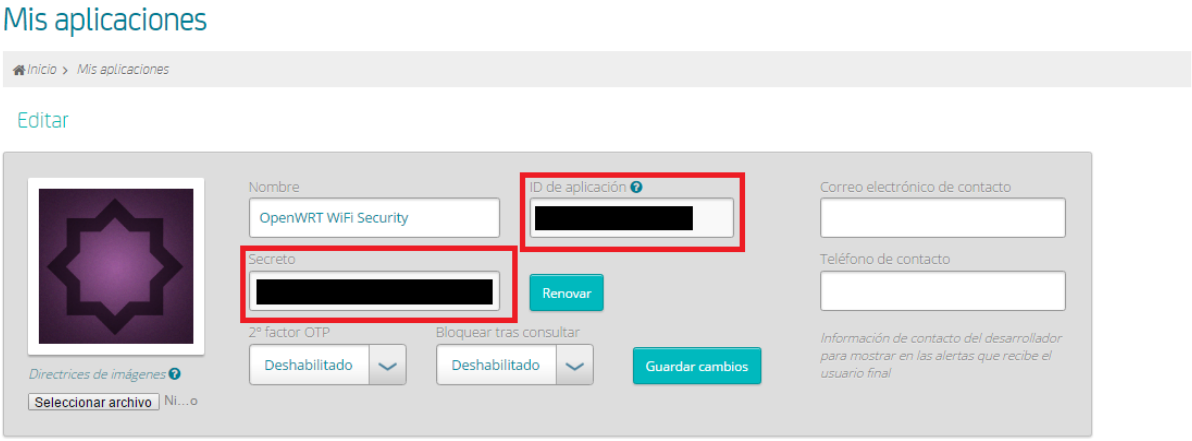


Imagen 37. APP ID y Secret Key de la aplicación Latch

6.3. Configuración del sistema

La configuración del sistema es muy sencilla. Solo hay que ir a “Latch – Configuration” (Imagen 38) para introducir el ID de Aplicación y la clave secreta que se ha obtenido al crear la aplicación.

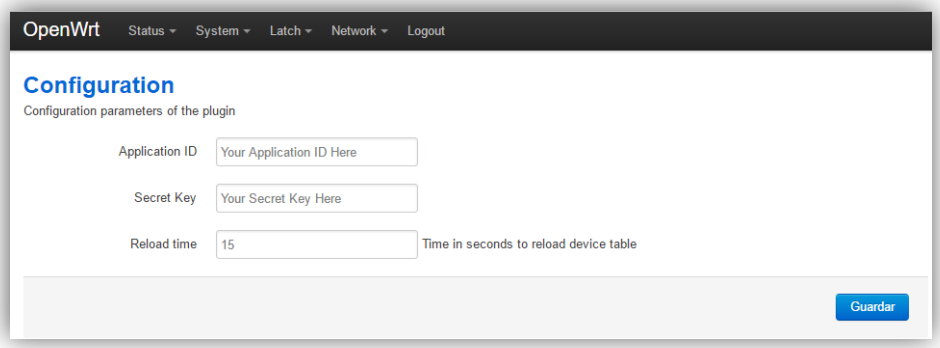


Imagen 38. Ventana de configuración del sistema

7. Pruebas

Sinopsis:

En este capítulo se realizan numerosas pruebas de funcionamiento sobre el prototipo y el sistema final.

Este capítulo se divide en dos secciones, pruebas sin interfaz web y pruebas con interfaz web.

7.1. Pruebas del sistema sin interfaz web

En esta sección se muestran un conjunto de pruebas que se realizan tanto al prototipo como al sistema final en la máquina virtual.

Como la salida por consola del prototipo y del sistema son muy parecidas se muestran solo imágenes del prototipo. No obstante, en la sección 7.1.1 se muestra una tabla resumen con las pruebas realizadas tanto al prototipo como al sistema final.

Todas las pruebas de esta sección se realizan realizando llamadas a los ficheros desde la consola y sin usar ningún tipo de interfaz gráfica.

Como se podrá comprobar en la sección 7.1.1 no todas las pruebas pueden ser realizadas tanto al prototipo como al sistema final. Esto se debe a que el prototipo carece de ciertas funcionalidades que fueron agregadas posteriormente en el sistema final.

7.1.1. Tabla resumen

Aquí se muestra una tabla resumen con las pruebas que han sido realizadas y el resultado en los sistemas.

Tipo de prueba	Prototipo	Sistema final
Pareado	✓	✓
Despareado	✓	✓
Estado global	✓	✓
Estado dispositivo	✓	✓
Alias de dispositivo	✗	✓
Dispositivo permanente	✗	✓

7.1.2. Prueba de pareado

La prueba de pareado consiste en intentar parear un dispositivo con los servidores de Latch para que actúe como dispositivo de control del sistema.

Lo que se hace en esta prueba es realizar una petición a los servidores de Latch (Imagen 39) con el código de registro obtenido de la aplicación Latch instalada en el Smartphone, y esperar una alerta (Imagen 40) en dicho Smartphone para comprobar que se ha pareado correctamente.

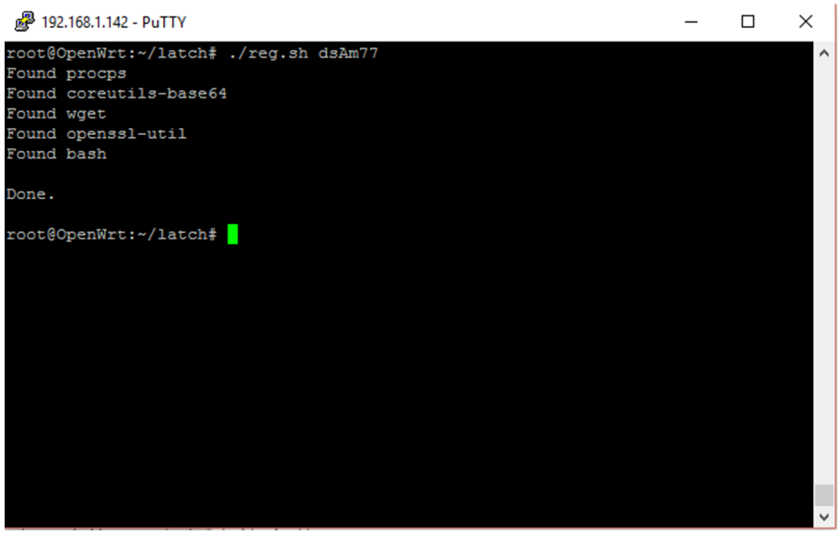


Imagen 39. Petición de pareado

Imagen 40. Alerta de pareado

Una vez recibida la alerta se puede ver como en la lista de servicios pareados de la aplicación Latch (Imagen 41) aparece la que acabamos de parear, y si comprobamos el archivo (Imagen 42) o base de datos donde se debe de guardar el id de la cuenta pareada se podrá comprobar que se ha almacenado correctamente.

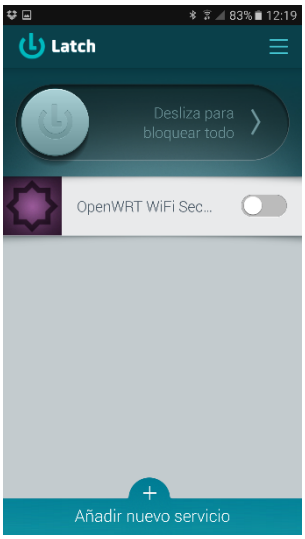


Imagen 41. Listado

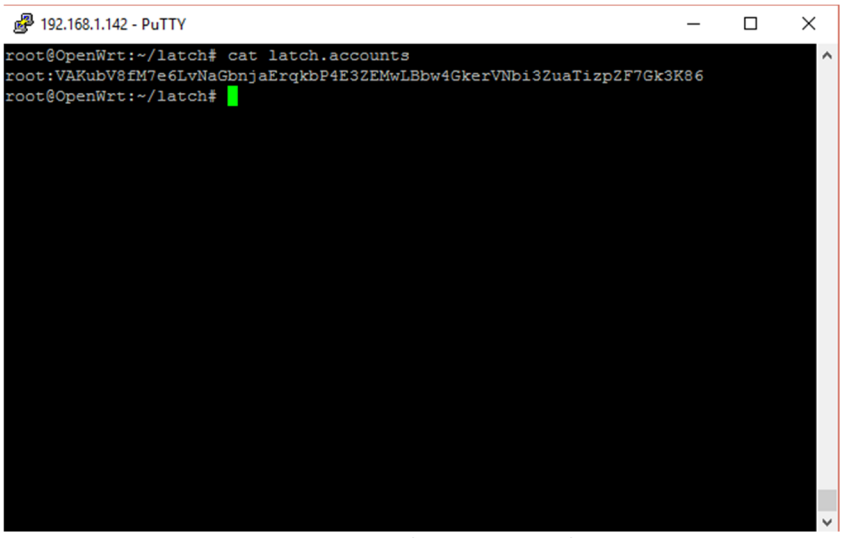


Imagen 42. ID de cuenta pareada

7.1.3. Prueba de despareado

La prueba de despareado consiste en desvincular un dispositivo de control que hayamos pareado previamente con la finalidad de poder parear uno distinto o de liberar el Smartphone del servicio que controla la red.

Para ello realizamos una petición de despareado (Imagen 43) que hará que el sistema envíe una petición a los servidores de Latch para que se desvincule el Smartphone. En caso de que todo haya sido correcto el Smartphone recibirá una alerta (Imagen 44) avisando de que un servicio ha sido despareado.

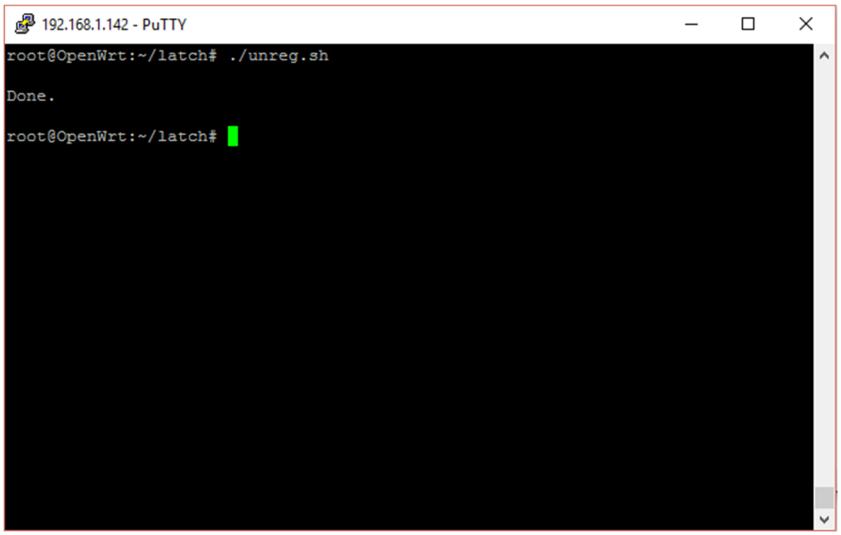


Imagen 43. Petición de despareado

Imagen 44. Alerta de despareado

Cuando hayamos recibido la alerta de despareado se puede comprobar que la lista de servicios (Imagen 45) de la aplicación Latch está vacía, y que el archivo o tabla de la base de datos que almacena el id de la cuenta pareada también ha sido eliminado (Imagen 46).

Imagen 45. Listado vacío

Imagen 46. ID de cuenta eliminado

7.1.4. Prueba de estado de dispositivo

En esta prueba se comprueba se comprueba que el sistema pueda comprobar los estados de los dispositivos para trabajar con ellos.

Esta prueba tiene diversas variantes como son los dispositivos permanentes y la gestión de alias. No obstante, estas pruebas se verán en la sección 7.2 cuando se realicen con interfaz gráfica.

Por lo tanto, esta prueba lo que hace es realizar una petición a los servidores Latch para comprobar el estado.

En un primer caso tenemos la red sin bloquear (Imagen 47) y al realizar la petición de estado podemos comprobar cómo nos dice que el acceso está permitido (Imagen 48).

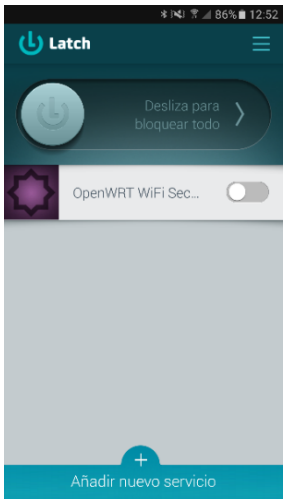


Imagen 47. Desbloqueado

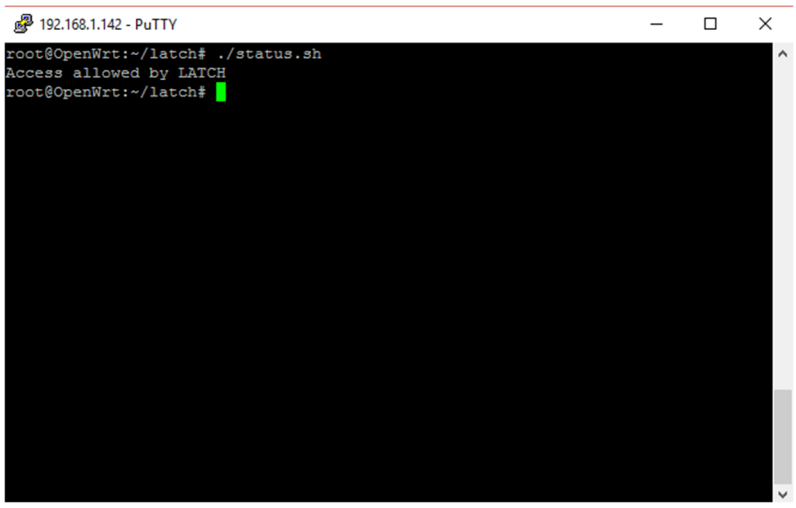


Imagen 48. Petición de estado - Devuelve permitido

En un segundo caso tenemos la red bloqueada (Imagen 49) y al realizar la petición de estado podemos comprobar cómo nos dice que el acceso está bloqueado (Imagen 50).

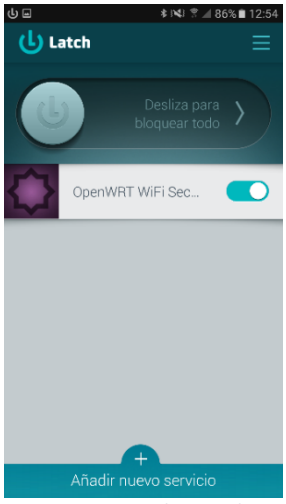


Imagen 49. Bloqueado

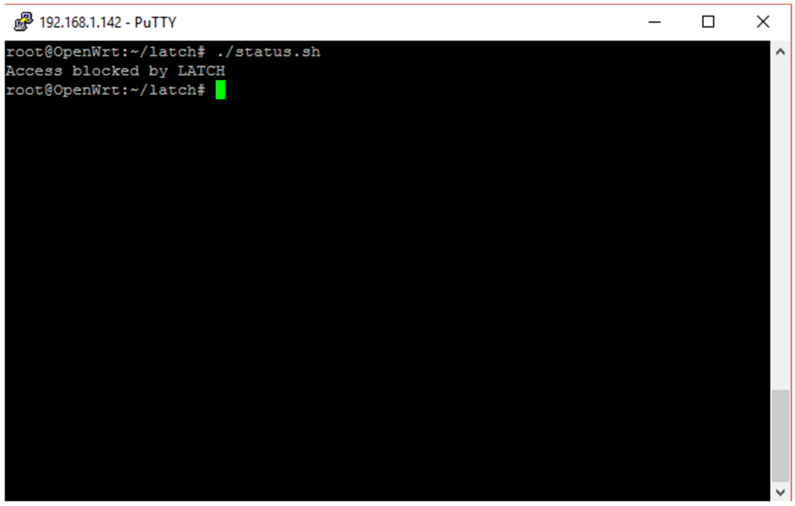


Imagen 50. Petición de estado - Devuelve bloqueado

Además, en este caso la aplicación Latch nos avisa (Imagen 51) de que se está intentando realizar un acceso al servicio cuando este está bloqueado.

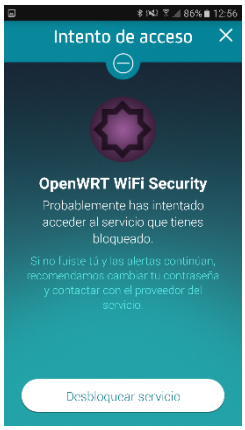


Imagen 51. Aviso bloqueado

7.2. Pruebas del sistema con interfaz web

En esta sección se realizan las mismas pruebas que se realizaron en la sección anterior, pero haciendo uso de la interfaz gráfica y en un router real. Además, se realizan un conjunto de pruebas propias de la interfaz.

7.2.1. Tabla resumen

Aquí se puede ver la tabla resumen de las pruebas realizadas

Tipo de prueba	Sistema final
Pareado	✓
Despareado	✓
Estado global	✓
Estado dispositivo	✓
Alias dispositivo	✓
Dispositivo permanente	✓
Navegación por interfaces	✓
Control de formato	✓
Control de campos vacíos	✓
Control de interacción con el servicio	✓
Diseño homogéneo y autoadaptable	✓

7.2.1. Prueba de navegación por interfaces

La prueba de navegación por interfaces es muy simple. Esta prueba consiste, tal y como su nombre indica, en navegar por las interfaces comprobando que no existe ningún enlace roto y que todas las secciones navegan a donde se supone que deben de hacerlo.

7.2.2. Prueba de control de formato

La prueba de control de formato consiste en comprobar que los datos estén bien formados antes de enviarlos al controlador.

Un ejemplo de esta prueba reside en la sección de pareado donde el código debe de ser de longitud 6, o en la gestión de dispositivos a la hora de añadir una dirección MAC como permanente donde se comprueba que la dirección introducida tenga el formato de una dirección MAC.

7.2.3. Prueba de campos vacíos

La prueba de campos vacíos consiste en comprobar los campos de los formularios de forma que no se pueda enviar vacíos al controlador, evitando de esta forma evitamos muchos errores y tiempo

Está prueba afecta prácticamente a todos los campos de formulario, pero más concretamente se puede ver un ejemplo de esto en la configuración o en la sección de pareado.

7.2.4. Prueba de interacción con el servicio

La prueba de interacción con el servicio consiste en realizar pruebas a los controladores de la aplicación para comprobar que cada interfaz hace su función correctamente.

En el caso de la configuración consiste simplemente en interactuar con el archivo de base de datos SQLite.

En otros casos como el del pareado y despareado consiste en interactuar con el servicio que hemos programado para lanzar peticiones a los servidores de Latch mediante la API.

7.2.5. Prueba de diseño homogéneo y autoadaptable

La prueba de diseño homogéneo consiste en comprobar que el diseño de las interfaces es idéntico al del resto del sistema de administración de OpenWRT de forma que parezca que el sistema venía instalado por defecto en dicho sistema.

La prueba de diseño autoadaptable consiste en instalar el sistema en versiones anteriores de OpenWRT para comprobar que el diseño sigue siendo el mismo que el del resto del sistema de administración de OpenWRT a pesar de que este último haya cambiado completamente. Es decir, comprobar que el diseño del sistema se adapta al diseño de OpenWRT independientemente de la versión de OpenWRT sobre la que se trabaje.

7.2.6. Prueba de gestión de dispositivos – Alias y permanencia

En esta prueba se comprueban que el sistema en general funcione correctamente detectando dispositivos que se conecten, avisando de dicha conexión (Imagen 52) al administrador, creándolos en la aplicación Latch en el smartphone (Imagen 53), bloqueando el dispositivo (Imagen 54) y mostrándolos en la sección de administración de dispositivos del sistema (Imagen 55).

Imagen 52. Aviso conexión

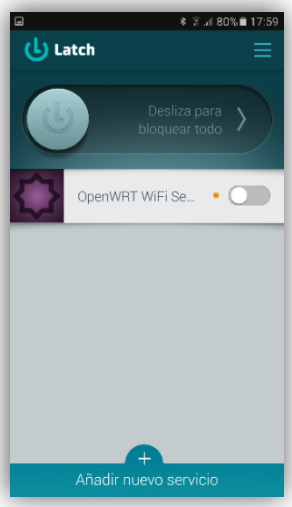


Imagen 53. Creación dispositivo

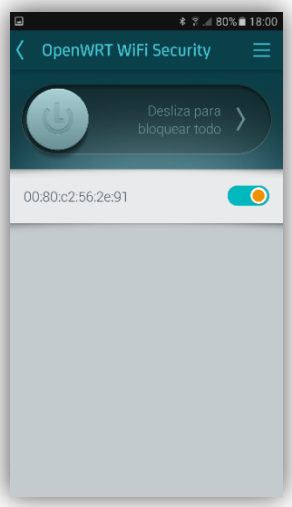


Imagen 54. Bloqueado

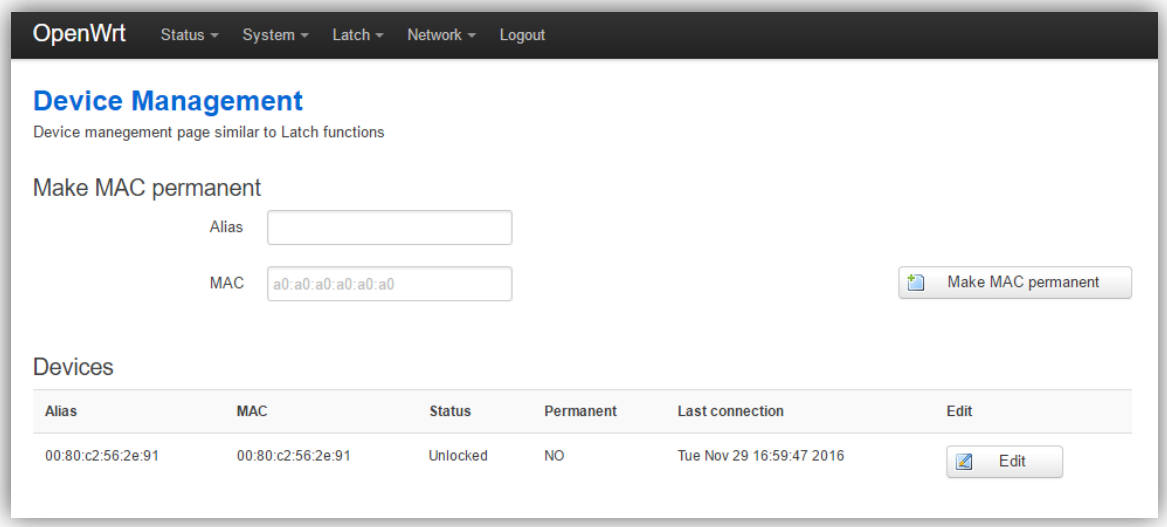


Imagen 55. Gestión de dispositivos - Dispositivo conectado

En esta prueba editaremos el alias del dispositivo (Imagen 56) conectado para ponerle "Tablet Android" y lo guardaremos. Se puede comprobar que el cambio aparece correctamente en la gestión de dispositivos (Imagen 57) de la interfaz web

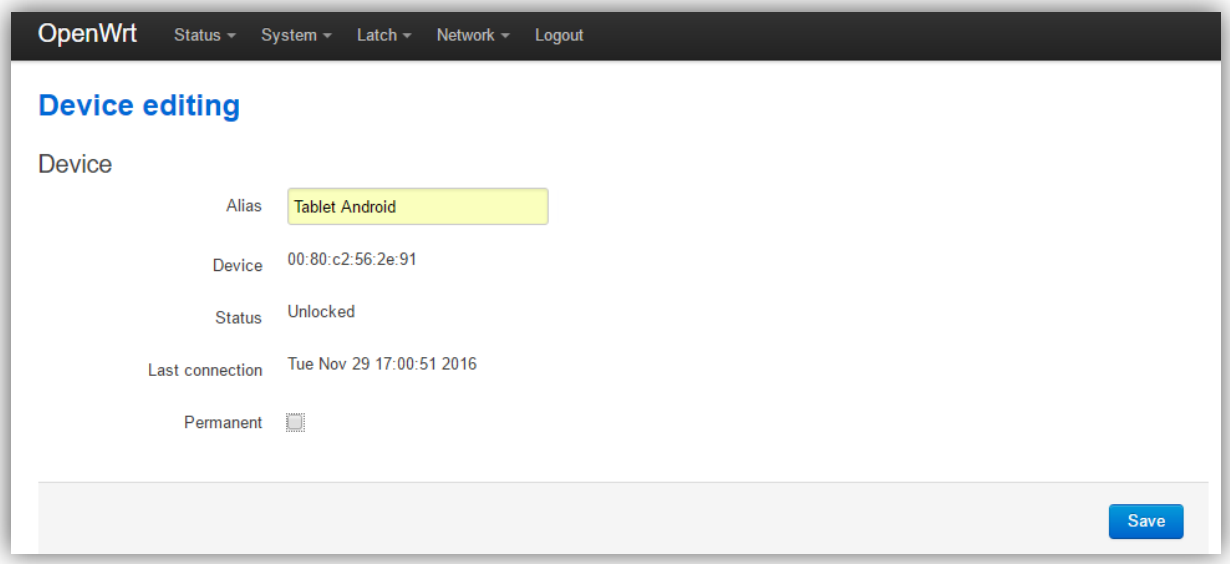


Imagen 56. Edición de dispositivo

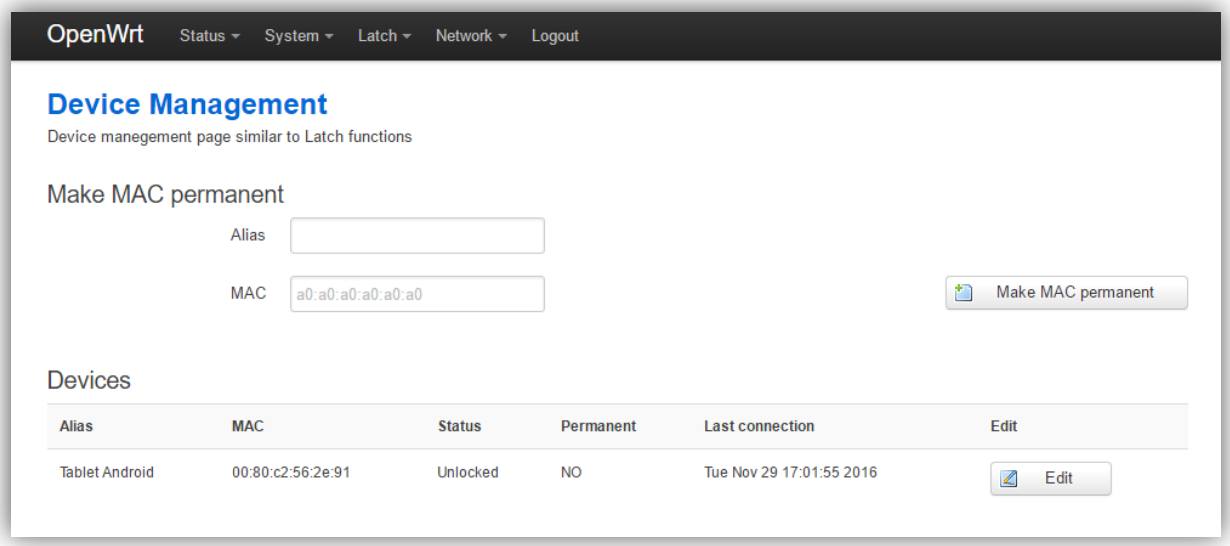


Imagen 57. Gestión de dispositivos

En la siguiente iteración del servicio del sistema se actualizará este alias (Imagen 58) en la aplicación Latch del Smartphone. Además, podemos editar estos dispositivos para ponerlos como permanentes si así lo quisiésemos, de forma que ese dispositivo desaparecerá de la lista (Imagen 60) y se administrará automáticamente.

En caso de que no pongamos el dispositivo como permanente, y lo bloqueemos, el sistema avisará (Imagen 59) al administrador de que un dispositivo bloqueado está intentando conectarse y lo expulsara de la red y si en la siguiente iteración del sistema no se ha vuelto a conectar se borrara de la lista de administración de dispositivos en la aplicación Latch (Imagen 60)

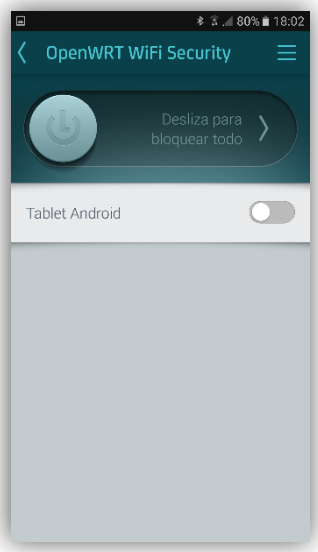


Imagen 58. Alias

Imagen 59. Aviso

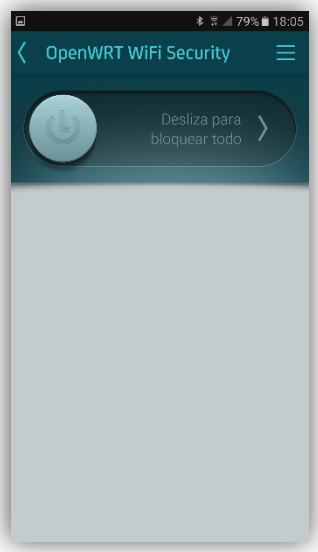


Imagen 60. Lista vacía

8. Conclusiones

Sinopsis:

En este capítulo se detalla el resultado y las conclusiones, así como el posible trabajo futuro para mejorar el sistema que ha sido desarrollado a lo largo de este documento.

También se incluyen las fuentes bibliográficas de las que se ha obtenido información para la realización del mismo

8.1. Resultado y conclusiones

El trabajo realizado consiste en la creación de un módulo o plugin para el firmware para routers OpenWRT utilizando la tecnología Latch desarrollada por Telefónica. El resultado es un sistema muy fácil de administrar que permite gestionar el acceso a la red a nivel de dispositivo, dando permiso o denegándolo a los dispositivos de forma que puedan acceder y utilizar la red o ser expulsados de la misma.

Latch OpenWRT es más que un plugin. Se trata de un gran sistema completamente integrado en el firmware OpenWRT que hace uso de diferentes tecnologías para facilitar la configuración del plugin y la administración de los dispositivos inalámbricos desde Latch.

Algunas de las tecnologías utilizadas para la creación de este plugin son Python, Lua, HTML, CSS, JavaScript, Bash, SQLite, etc.

Latch OpenWRT está tan integrado en OpenWRT que no notarás diferencia en el diseño web al entrar al plugin (Imágenes 61 y 62). Parecerá que está instalado por defecto en el sistema.

Imagen 61. Sección de administración de OpenWRT

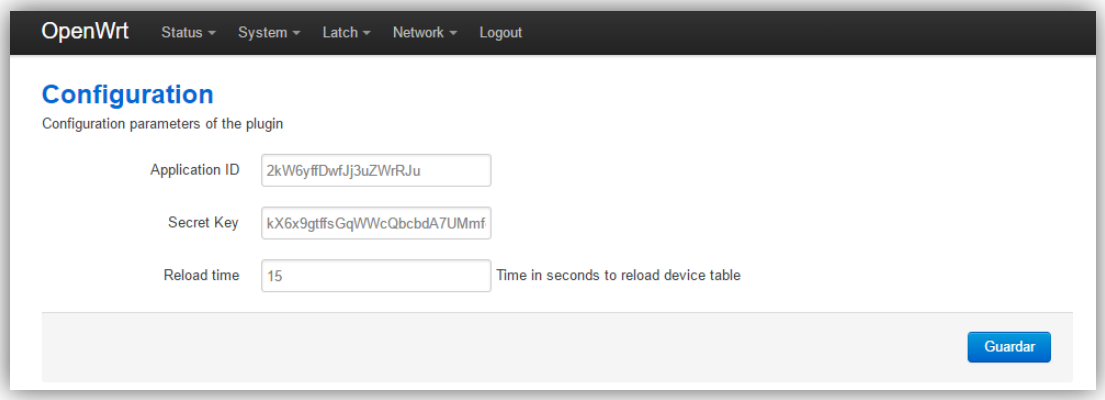


Imagen 62. Sección de Configuración de LatchOpenWRT

La integración visual ha sido realizada de forma que se adapta automáticamente a distintas versiones de OpenWRT sin necesidad de hacer ninguna configuración adicional o cambios en los ficheros del plugin.

Además, podemos comprobar y cambiar el estado de la protección directamente desde la sección de OpenWRT (Imagen 63) destinado a ello, como si fuese otro servicio nativo.

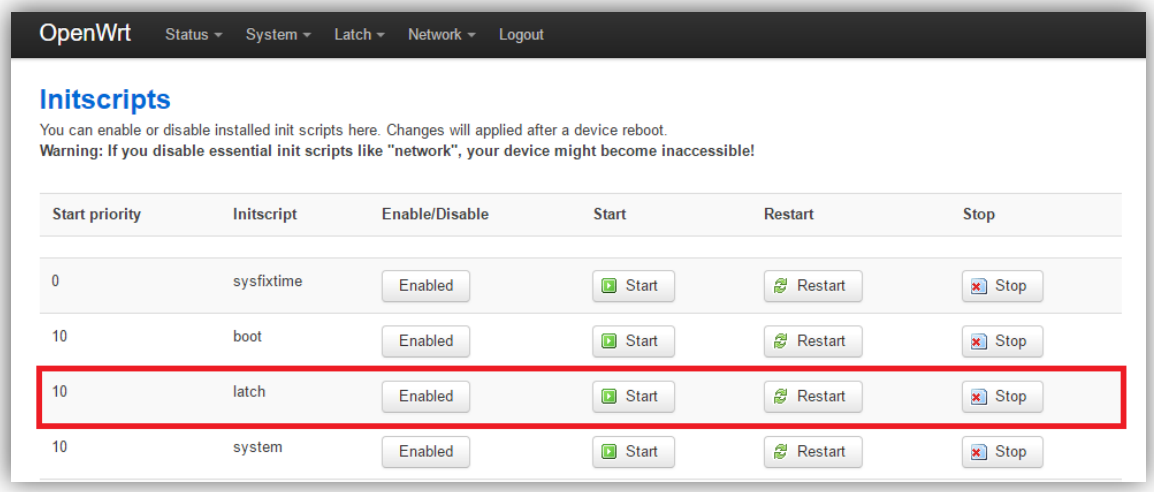


Imagen 63. Sección "Startup" de OpenWRT

La desinstalación del plugin también está totalmente integrado en OpenWRT, basta con entrar a "System – Software" (Imagen 64) y buscar el paquete para darle a "Remove" y que desaparezca de OpenWRT.

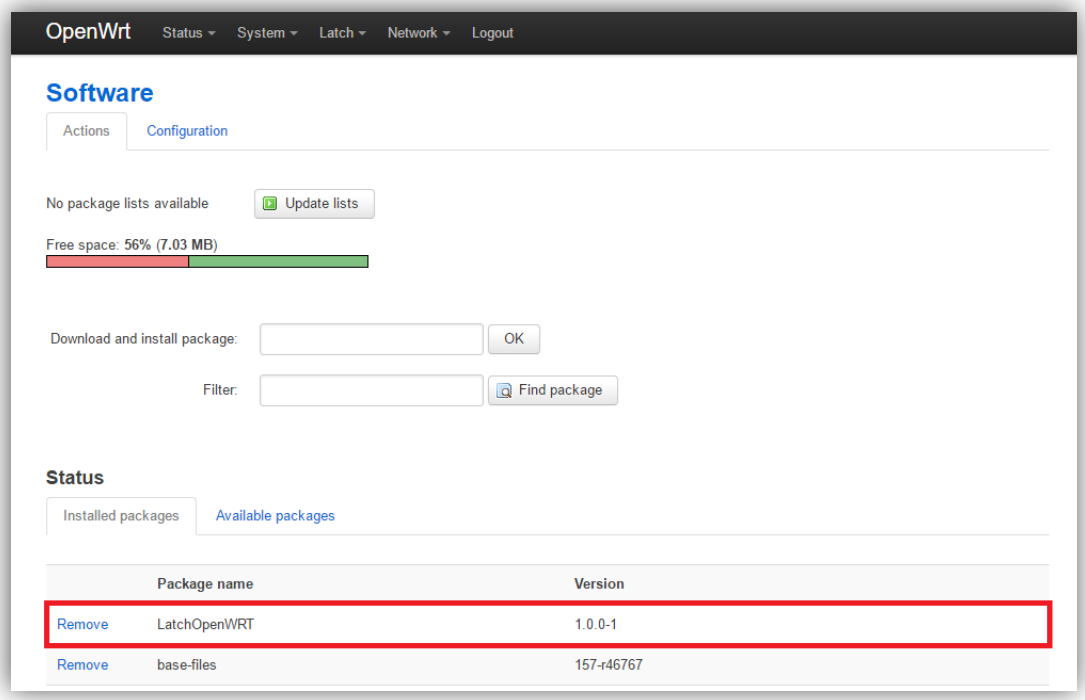


Imagen 64. Sección de gestión de paquetes de OpenWRT

Debido a los diferentes tipos de licencias de Latch, este proyecto tiene una gran escalabilidad. Podemos usar el plugin tanto para una casa o una pequeña compañía, usando las licencias más baratas o la gratuita, como para medianas o grandes compañías, usando licencias más potentes y manejando así más dispositivos.

A diferencia de otros proyectos que hacen uso de Latch y que pueden ser más o menos útiles en el mundo real y en el uso diario, Latch OpenWRT usa una tecnología de seguridad, como es Latch, para la seguridad informática y tiene un impacto directo y diario en la misma.

La instalación y configuración del plugin es tan sencilla que puedes tenerlo funcionando en pocos minutos. Además, la administración inteligente de dispositivos del plugin te ayudará a enfocarte en administrar los dispositivos de forma simple y rápida desde la aplicación de Latch.

Un video completo sobre la instalación, configuración y uso del sistema sobre un router real puede ser encontrado en YouTube buscando “Latch OpenWRT”, o entrando directamente al siguiente enlace: https://www.youtube.com/watch?v=RyBP2_bliDo&

Todo lo anterior ha llevado a este proyecto a ganar uno de los premios del Concurso de tecnología Latch realizado por Telefónica en su edición 2016.

8.2. Trabajo futuro

A pesar de que el trabajo realizado hasta ahora nos provee de un sistema completo y funcional cumpliendo todos los objetivos y especificaciones que habíamos establecido todavía nos encontramos ante distintos retos y posibles mejores que se pueden realizar con la finalidad de crear un sistema más avanzado.

- Mejora en el reconocimiento de dispositivos.
    - Actualmente el reconocimiento de dispositivos se basa en la asignación y uso de un Alias. Este alias es muy útil para reconocer a los dispositivos permanentes de un vistazo, pero no lo es a la hora de reconocer dispositivos temporales.
    - Esto se debe a que en el momento que se desconecta un dispositivo temporal se borra toda la información asociada al mismo y por lo tanto la próxima vez que se conecta no tiene alias asociado, aunque se lo hayamos puesto en otra ocasión.
    - La forma de solucionar esto sería creando una tabla que almacene diferentes datos de los dispositivos, asociados por dirección MAC, entre ellos el alias y que recupere esta información en la próxima conexión.

- Detección inteligente de MAC duplicadas.
    - Actualmente solo se almacena un alias y la fecha de la última conexión del dispositivo a la red.
    - Sería interesante recopilar más datos para poder analizarlos y crear perfiles de conexión, etc.

- MAC Duplicadas.
    - Actualmente el control de duplicidad de MAC se controla pasivamente por el router y por lógicas de conexión, como ver que tu MAC se ha conectado, aunque tú no estás conectado a dicha red.
    - Sería interesante crear un control activo y más exhaustivo de las posibles MAC duplicadas. Por ejemplo, se podría usar los perfiles de conexión de los que hablábamos en el apartado anterior para detectar irregularidades en la conexión, como por ejemplo que una MAC se haya conectado a las 01am cuando durante semanas o meses nunca se había conectado a dicha hora, pudiendo realizar así un bloqueo preventivo.

- Personalización del tiempo de bloqueo de conexión.
    - Actualmente el tiempo de bloqueo de conexión para un dispositivo que ha sido expulsado de la red es 5 segundos sin posibilidad de ser modificado.
    - Sería interesante que este tiempo de bloqueo pueda ser modificado desde el panel de configuración para que cada usuario lo personalice.

8.3. Referencias bibliográficas

[1] HTML (www.w3schools.com/html/html_intro.asp)

[2] JavaScript (www.w3schools.com/js/js_intro.asp)

[3] CSS (www.w3schools.com/css/css_intro.asp)

[4] Lua (www.lua.org)

[5] Bash (www.gnu.org/software/bash/)

[6] Python (www.python.org)

[7] SQLite (www.sqlite.org)

[8] MySQL (www.mysql.com)

[9] OpenWRT (www.openwrt.org)

[10] Latch (latch.elevenpaths.com)

[11] OPKG (wiki.openwrt.org/doc/techref/opkg)

[12] OAuth (www.oauth.net)

9. Anexo – Figuras e Imágenes

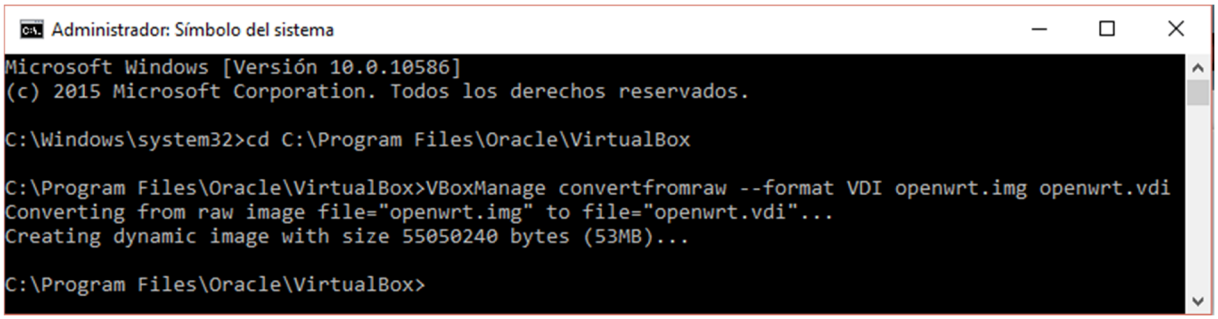
Tabla de figuras

Tabla de imágenes

10. Anexo – Gestión de la máquina virtual

Preparando máquina virtual

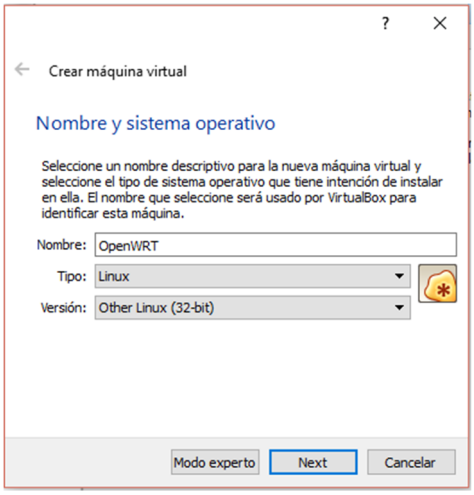
Primero hay que descargar una imagen de OpenWRT genérica desde la sección de descargas de su página oficial. Una vez se descargue la imagen de OpenWRT hay que transformarla a formato vdi como se puede ver en la imagen XXX, qué es el formato de las máquinas virtuales de Oracle VirtualBox.



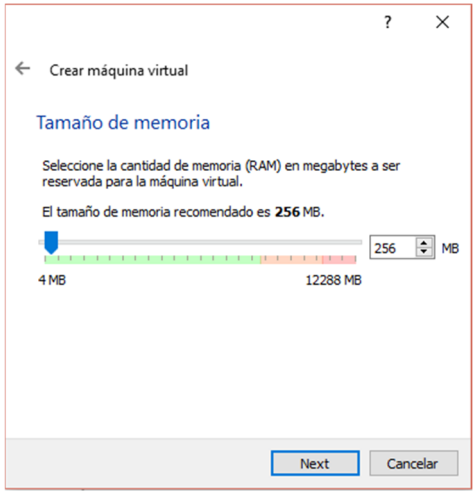
Transformación de imagen genérica a vdi

Ahora hay que crear una nueva máquina virtual y configurar todos sus parámetros. Lo primero que hay que hacer es establecer un nombre a la máquina virtual y el tipo de sistema operativo que se va a instalar en ella como se muestra en la imagen XXX.

Después debemos de establecer una cantidad de memoria RAM a la máquina, véase la imagen XXX, que estamos creando, con 256MB de memoria RAM es suficiente



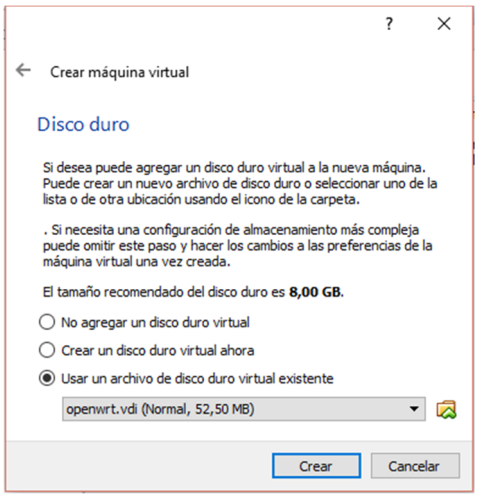
Nombre máquina



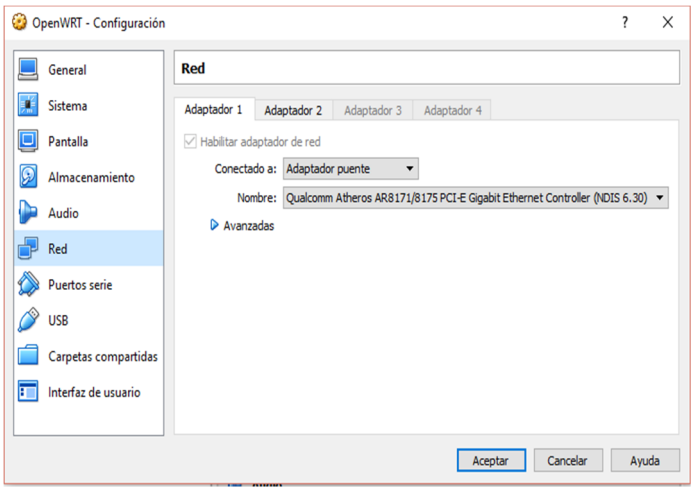
Memoria máquina

Continuando con el proceso de creación de la máquina virtual deberemos de seleccionar el disco duro que queremos asociar a dicha máquina. En el caso que nos ocupa hay que seleccionar la imagen de OpenWRT que se ha transformado anteriormente, como puede verse en la imagen XXX. Una vez finalizado este paso ya tiene que estar creada la máquina.

Para finalizar con el proceso de preparación de la máquina con la que se va a trabajar se debe de preparar el adaptador de red. Para ello hay que entrar a las propiedades de la máquina que hemos creado, ir a Red, y configurar el adaptador como “Adaptador puente” seleccionando el adaptador de red del ordenador, tal y como se muestra en la imagen XXX.



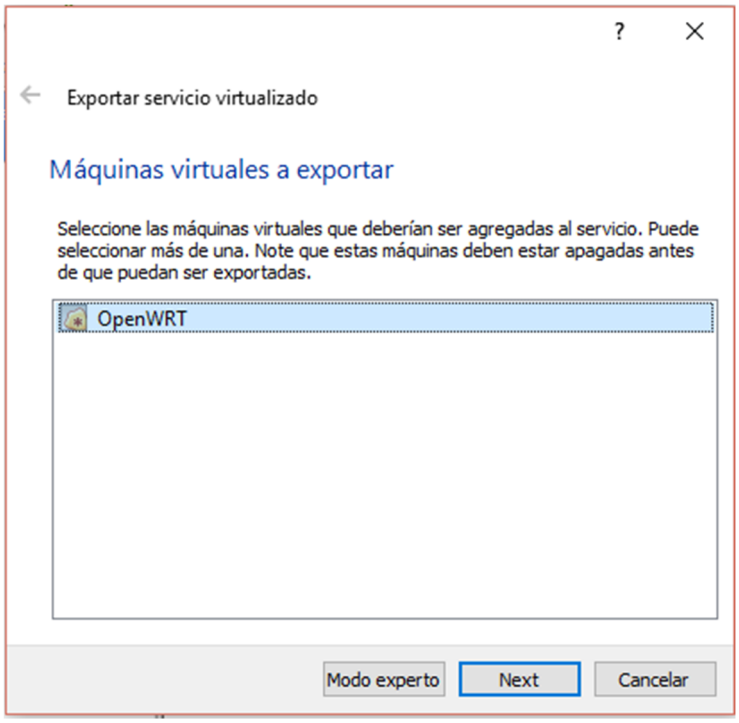
Disco duro máquina



Adaptador de red máquina

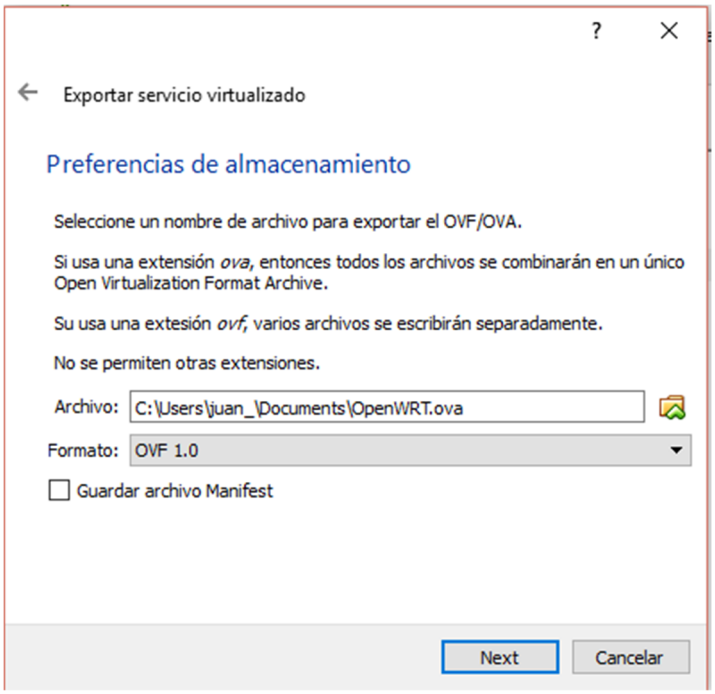
Exportando desde VirtualBox

Para exportar una máquina virtual desde Virtual Box se debe de entrar a la opción de exportación y seleccionar la máquina virtual que se quiera exportar.



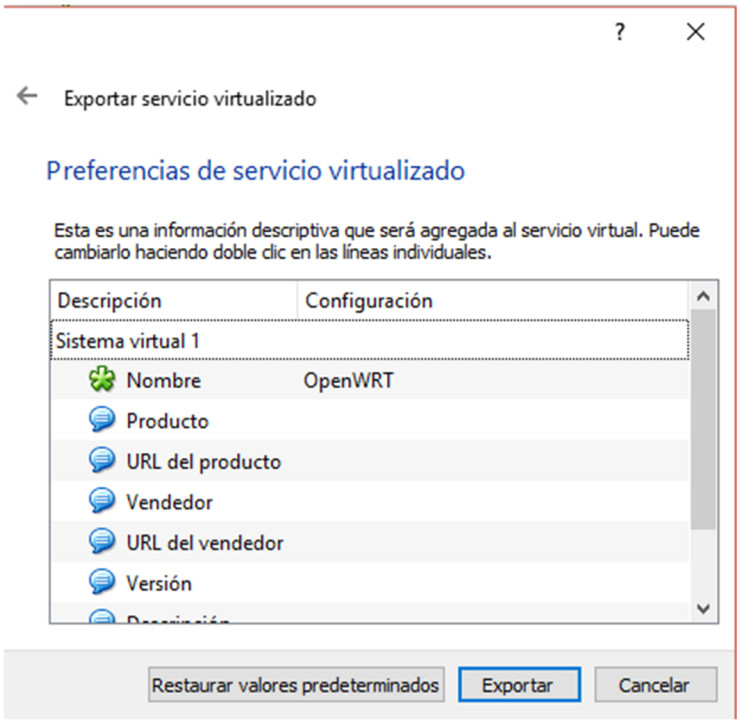
Selección de máquina virtual

Una vez seleccionada la máquina que se quiere exportar se deberá de seleccionar un formato de exportación y la ruta donde se quiere guardar la máquina exportada.



Selección de formato y ruta

En la siguiente ventana se puede establecer distintas propiedades para la máquina virtual que estamos exportando. Se pueden dejar por defecto si así se desea.



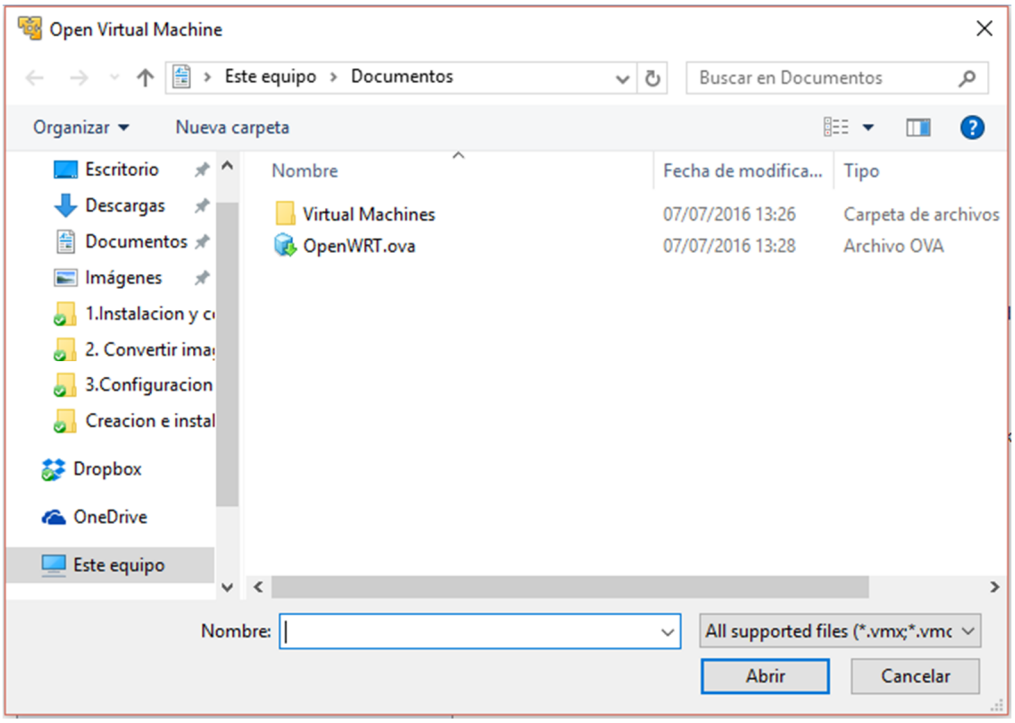
Preferencias de exportación

Para finalizar la exportación de la máquina virtual solo hay que darle al botón de exportar y empezará a realizarse la exportación.

Cuando finalice dicha exportación el sistema mandará un aviso y la máquina estará exportada en la ruta que se ha configurado previamente.

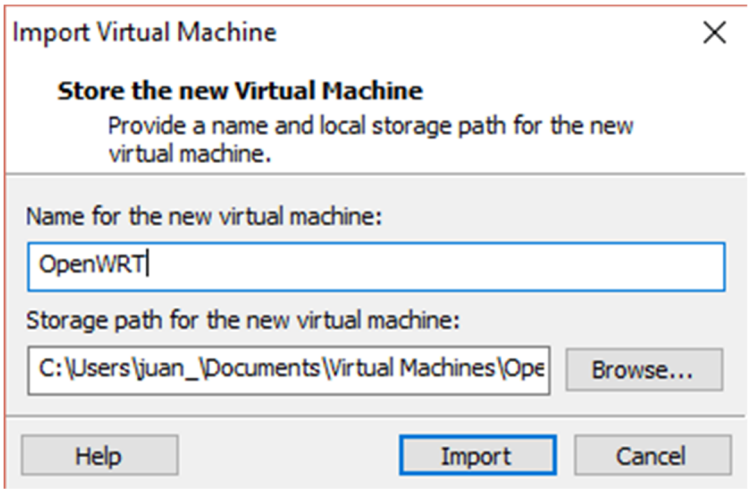
Importación en VMWare

Realizar una importación de máquina virtual en VMWare es muy sencillo. Solo hay que entrar en la opción de importar máquina virtual y seleccionar la máquina que se quiere importar.



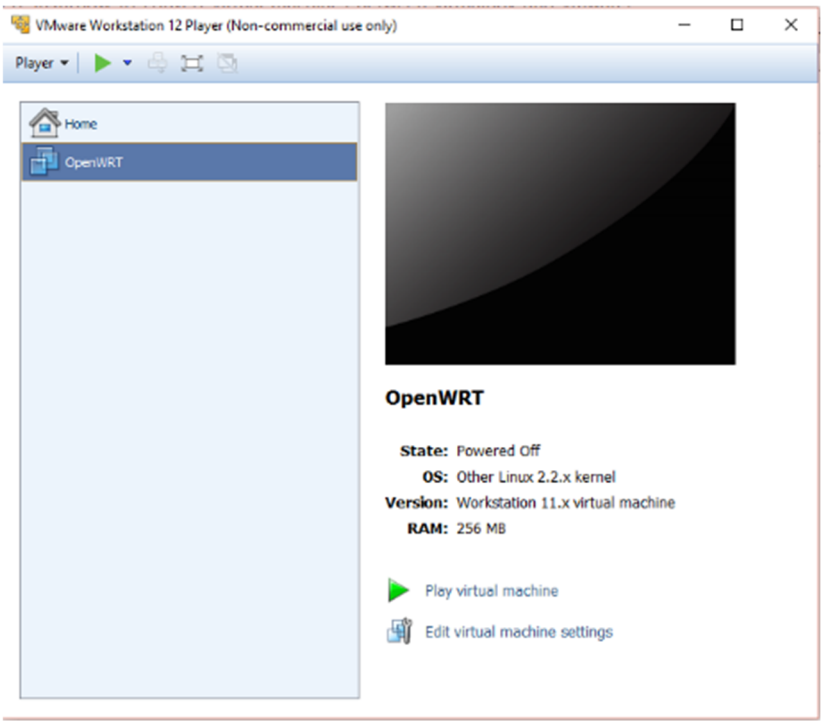
Selección de máquina para importar

Una vez seleccionada la máquina a importar hay que establecer un nombre y una ruta en la que almacenarla.



Propiedades importación

Finalmente basta con darle al botón “Import” para realizar la importación de la máquina y que aparezca en la lista de máquinas virtuales en el menú principal del programa.

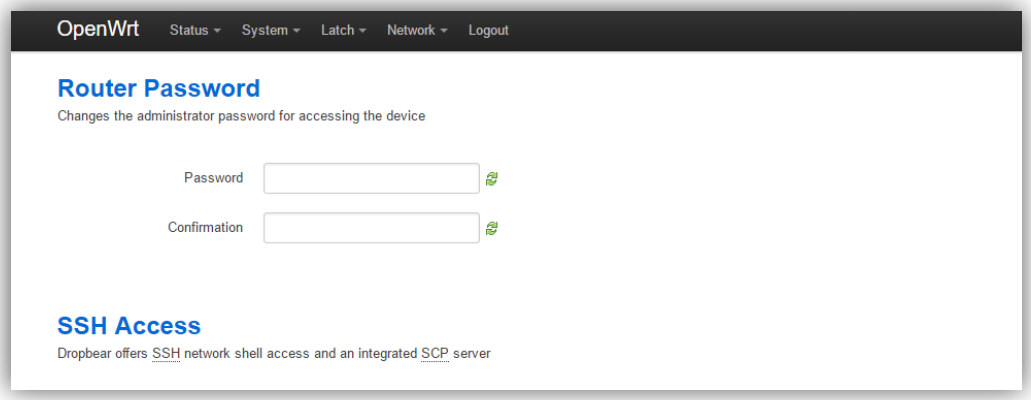


Listado de máquinas virtuales

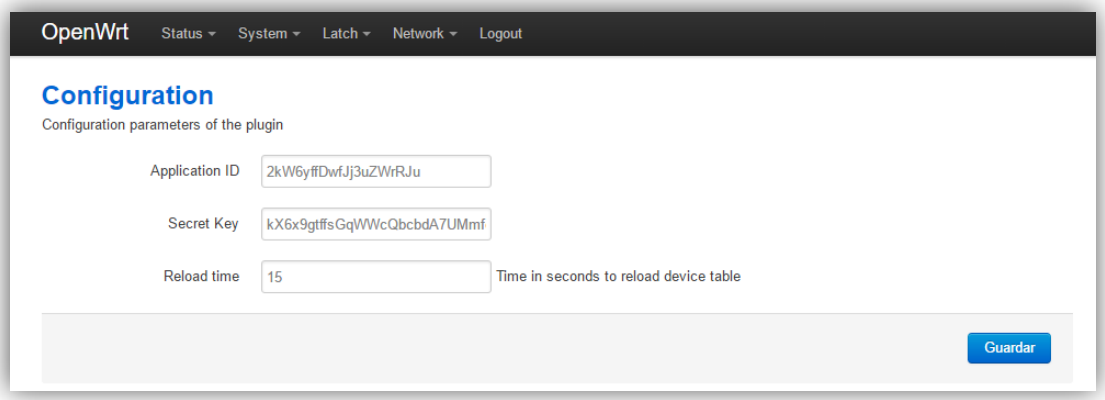
11. Anexo – Diseño web del proyecto

Comparación con diseño OpenWRT

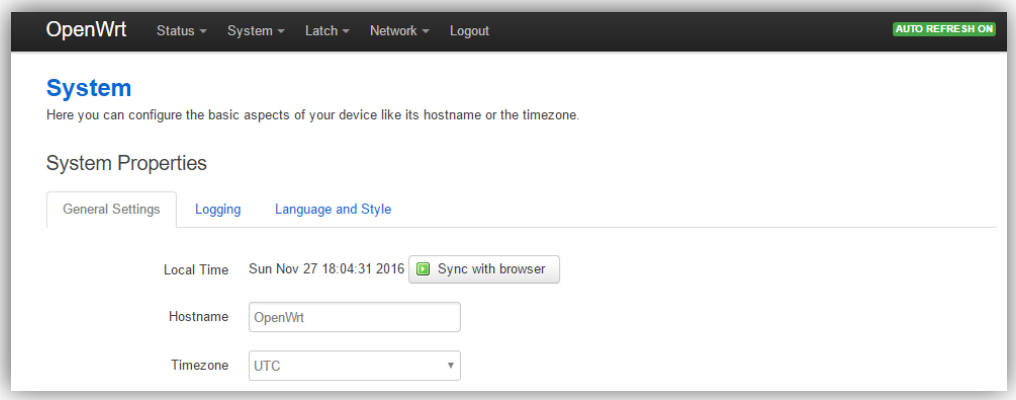
Tal y como se ha comentado en el documento, el diseño del plugin creado en el proyecto es idéntico al del sistema OpenWRT de forma que parece que el sistema viene instalado por defecto en el mismo.



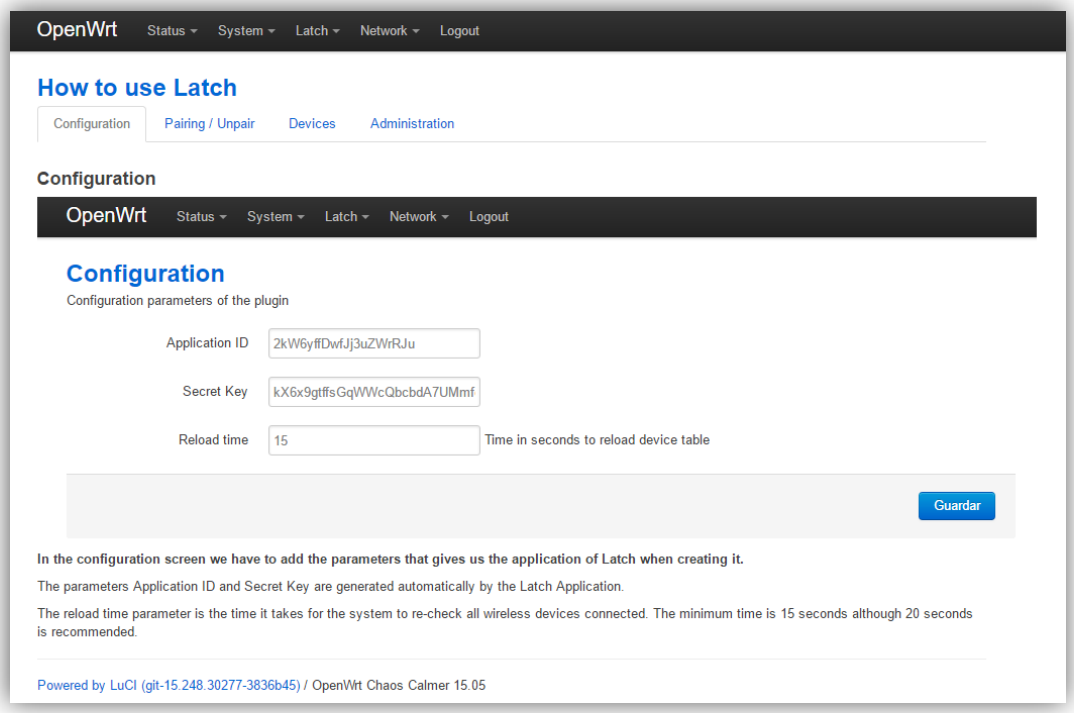
Sección de administración de OpenWRT



Sección de Configuración de LatchOpenWRT

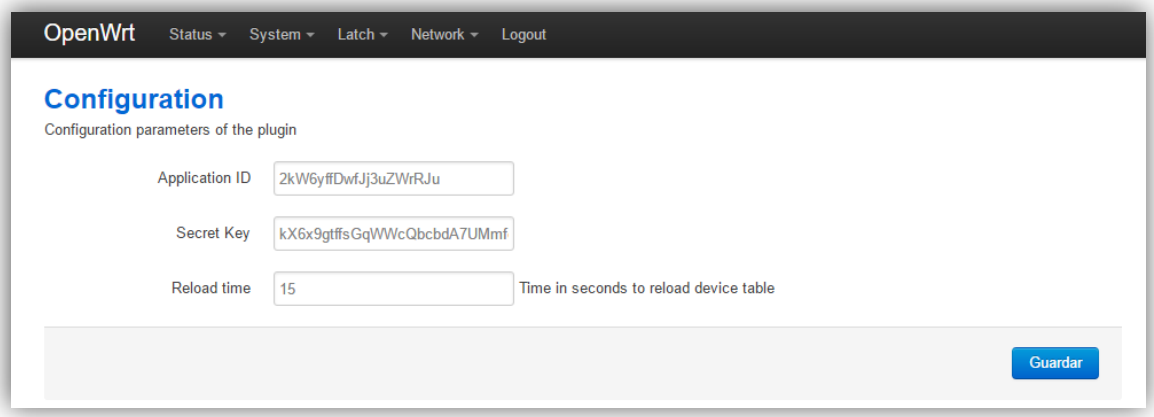


Sección de sistema de OpenWRT

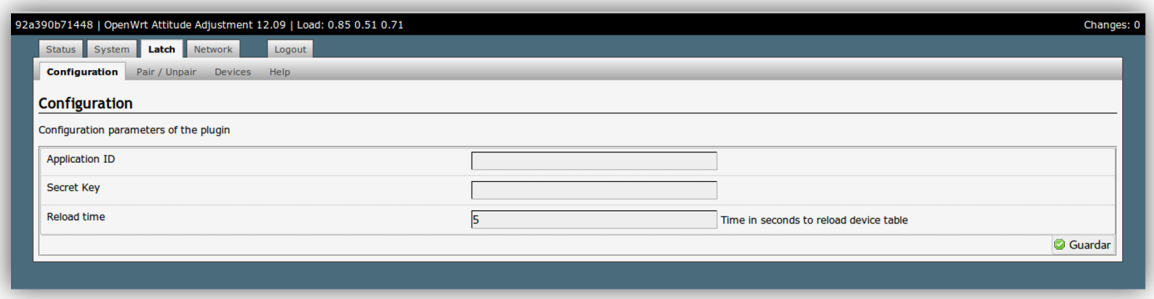


Sección de ayuda de LatchOpenWRT

Además, el diseño realizado se adapta automáticamente a la versión de OpenWRT. Aquí se puede ver un ejemplo de esto.



OpenWRT Chaos Calmer



OpenWRT Attitude Adjustment

12. Anexo – Instalación y uso del sistema

Latch

Manual de Latch OpenWRT

Prerrequisitos

- Instalación de OpenWRT previamente configurada con acceso a internet y contraseña de administrador.
    - Aunque el plugin ha sido creado para trabajar con OpenWRT, este manual no va a explicar la instalación o configuración del mismo dado que dicha instalación y configuración depende de cada router.
    - Si se explicará brevemente como instalarlo en el router "Homestation ADB PDG A4001N".
    - La configuración de red de OpenWRT para tener acceso a internet desde el mismo depende de la topología de cada red.

- Una cuenta creada en "latch.elevenpaths.com" y una aplicación, creada en esa cuenta, con el nombre que se desee, por ejemplo: "OpenWRT WiFi Security".
    - No se necesita configurar nada de la aplicación que se ha creado. Se pueden dejar todos los parámetros por defecto y solo necesitará apuntar el id de la aplicación y la clave secreta de dicha aplicación para configurar el plugin.

- El paquete de instalación para el router que tengamos.
    - El paquete de instalación utilizado en este manual ha sido compilado para routers que usen la familia de SoC Broadcom BCM63xx, y será instalado en el router "Home Station ADB PDG A4001N" de Movistar.
    - Si necesitas el paquete de instalación para otro router puedes descargar el código de compilación desde GitHub y compilarlo manualmente para la familia SoC que necesite.

- Un programa de conexión por SSH para instalar el plugin.
    - En este manual usaremos Putty.

Preparando OpenWRT

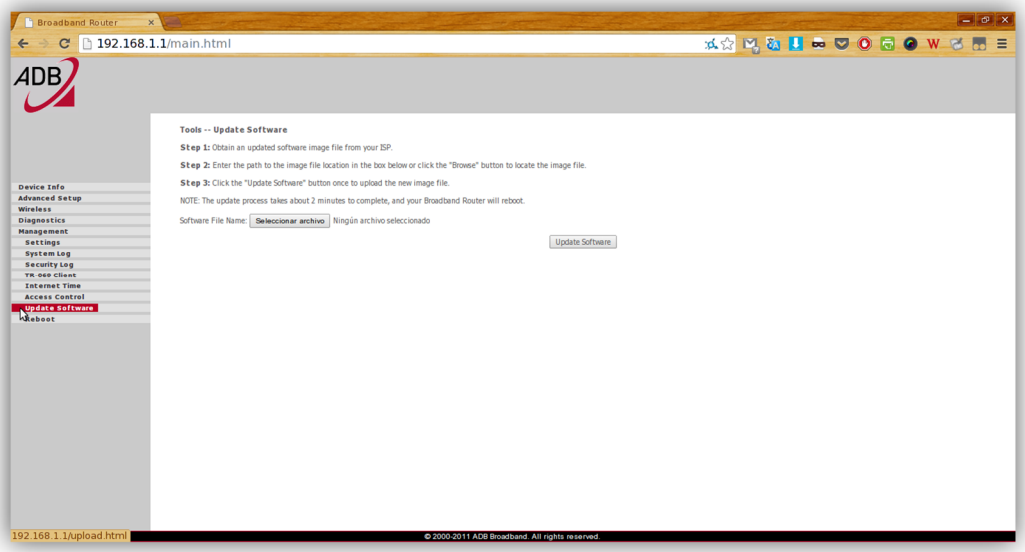
Lo primero que necesitas es un router compatible con OpenWRT y el firmware correcto para el router en el que queremos instalarlo. También necesitarás saber los pasos de instalación para ese router en concreto ya que dependiendo del router los pasos pueden ser distintos.

En este caso usaremos el router “Home Station ADB PDG A4001N” de Movistar.

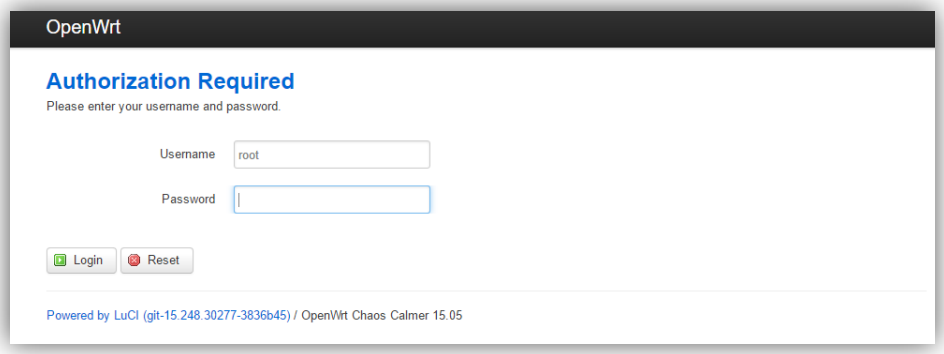
Este router usa el SoC Broadcom BCM6328, así que necesitaremos una versión de OpenWRT que sea compatible con ese SoC, en este caso la versión “brcm63xx”. Puedes obtener la versión correcta para este router en mi GitHub.

La instalación de OpenWRT en este router es realmente simple. Solo necesitamos introducir el firmware como si de una actualización oficial se tratase.

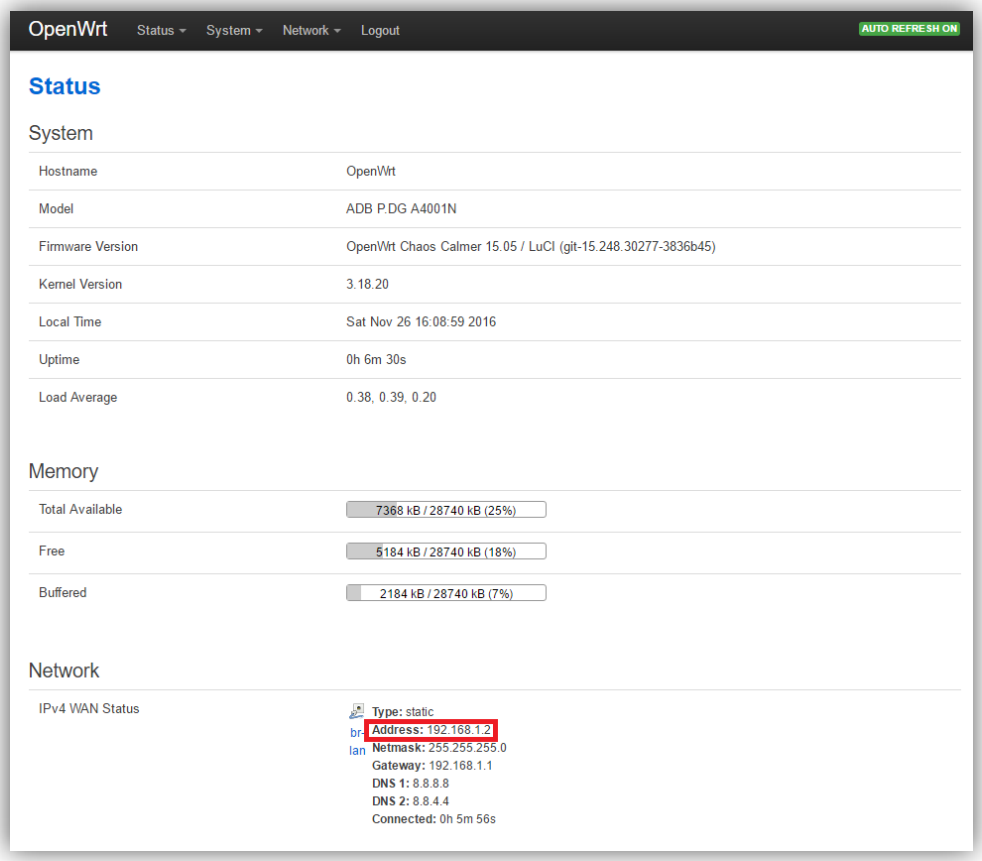
1. Acceder a la página web de administración: http://192.168.1.1/main.html
2. El sistema te preguntará por un usuario y una contraseña. Si no la has cambiado deberías de poder entrar poniendo “1234” en ambos campos.
3. En el menú de la izquierda selecciona “Management” y dale clic a “Update Software”. Dale al botón de examinar para seleccionar el firmware de OpenWRT que descargaste y dale a actualizar software. En unos minutos el router reiniciará y deberías ver la interfaz de OpenWRT.



Instalación del Plugin



Pantalla de Login de OpenWRT



Una vez que tengamos OpenWRT instalado y configurado podemos acceder al router con cualquier programa de conexión SSH. En nuestro caso usaremos Putty.

Para acceder al router deberás usar la IP y la contraseña que configuraste para acceder a la página de administración de OpenWRT.

La instalación del plugin es realmente sencilla. Una vez conectado al mismo por SSH solo necesitaremos enviar el paquete de instalación del plugin al router y ejecutar la instalación. Lo anterior se puede realizar usando los siguientes comandos.

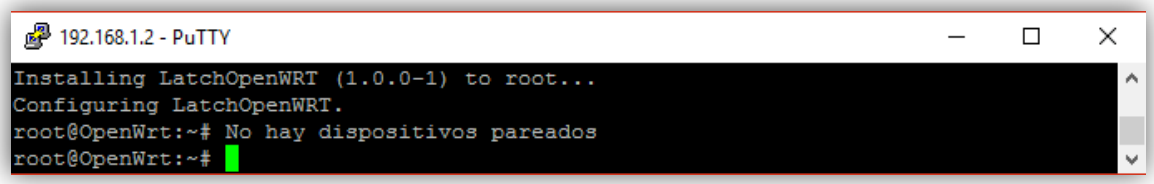
```
scp <package name> root@RouterIP:/

opkg update ; opkg install <name package>
```

Donde el "package name" es el nombre del paquete de instalación para tu router y "RouterIP" es la IP de acceso a la administración del router.

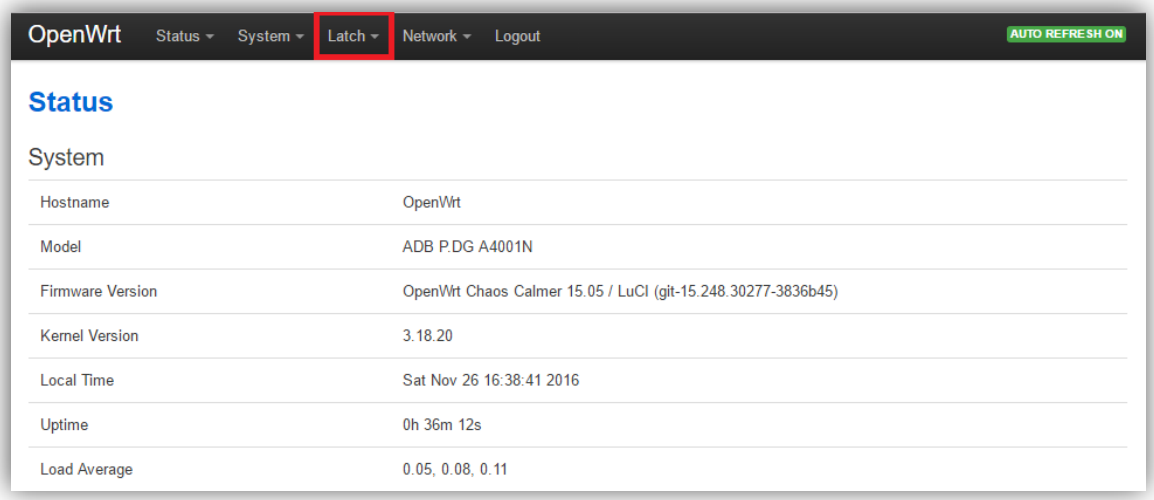
En mi caso, tengo el paquete de instalación para mi router compilado y guardado en mi GitHub, así que puedo instalar el plugin directamente, sin descargarlo a mi ordenador y enviarlo al router. Para eso usaré el siguiente comando:

```
opkg update ; opkg install openssl-util ; opkg install
http://github.com/JCameroMartin/LatchOpenWRT/raw/master/LatchOpenWRT_1.0.0-
1_brcm63xx.ipk
```

Este proceso tardará un rato. Sabrás que la instalación ha terminado cuando en la pantalla veas algo similar a la siguiente imagen.

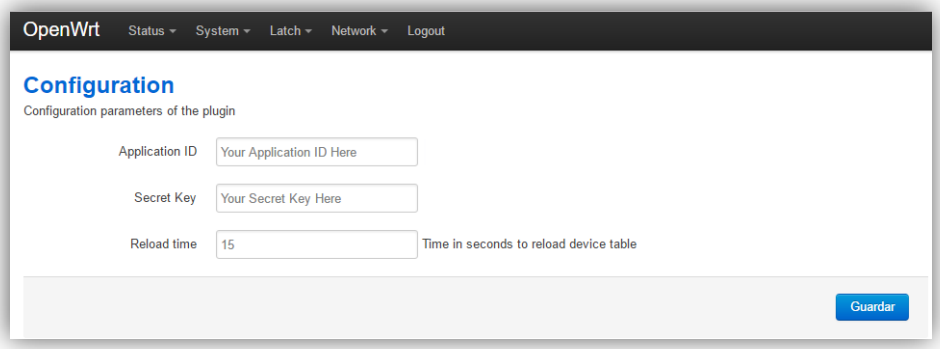


Ya tienes instalado el plugin. Ahora puedes ir a la página web de administración de OpenWRT y comprobar como hay una nueva sección indicando que la instalación ha sido correcta.



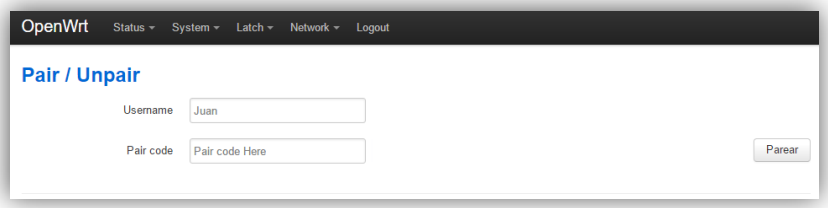
Configuración

La configuración del plugin es incluso más sencilla que su instalación. Solo necesitas ir a “Latch – Configuration” para introducir el ID de Aplicación y la clave secreta.

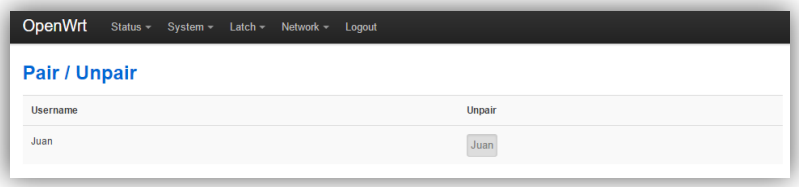


Después necesitaras parear un dispositivo con el que controlar el plugin. Para ello ve a “Latch – Pair/Unpair”.

Para parear un dispositivo solo necesitamos poner un nombre con el que reconocerlo y el código de pareado que se obtiene de la aplicación Latch instalada en el dispositivo que quieras parear.



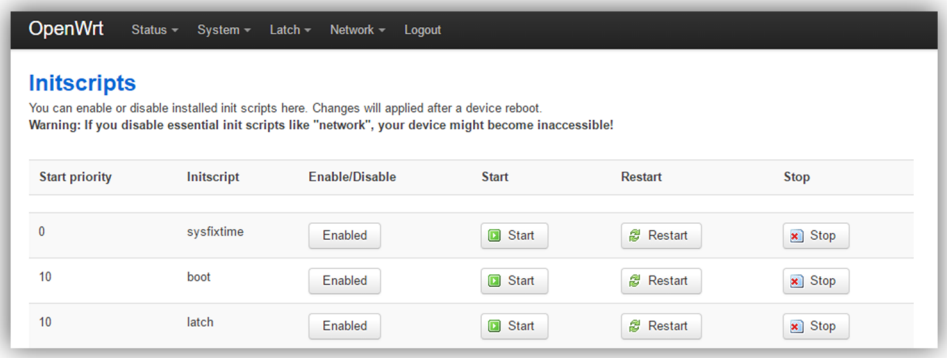
Después de 30 segundos la página se recargará y si todo ha ido bien veréis una pantalla como la siguiente:



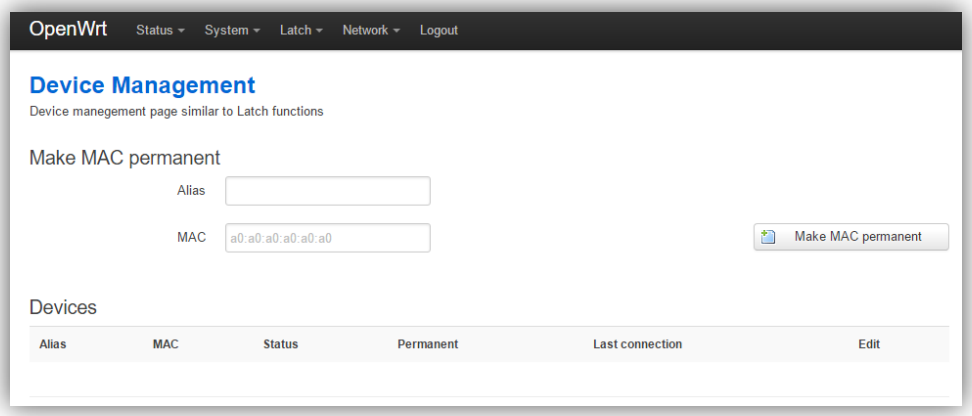
Para desparear un dispositivo y volver a parear otro solo necesitas darle al botón.

En este paso ya has terminado de instalar y configurar el plugin. Para iniciar el servicio del plugin solo tienes que ir a “System – Startup”, encontrar el servicio “latch” y darle al botón de “Start”.

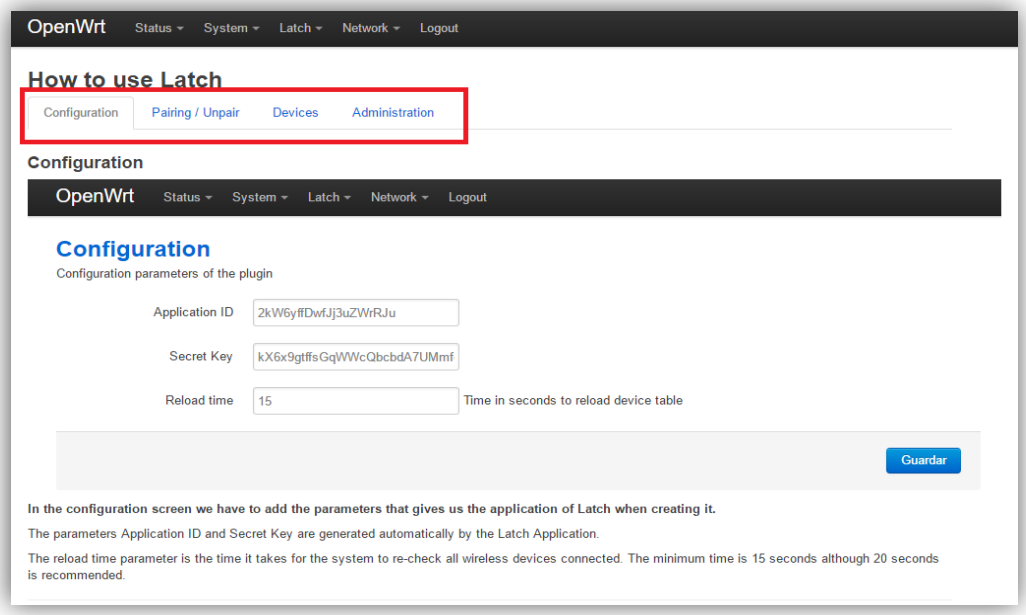
NOTA: Tendrás que reiniciar el servicio si despareas el dispositivo para parear otro o si cambias cualquier parámetro de la configuración del plugin.



Aunque ya puedes cerrar la página de administración y dejar que el plugin haga todo el trabajo por ti, puedes añadir dispositivos permanentes y editar información de los mismos en “Latch – Devices”



Información avanzada de uso puede ser encontrada en la ayuda, en “Latch – Help”

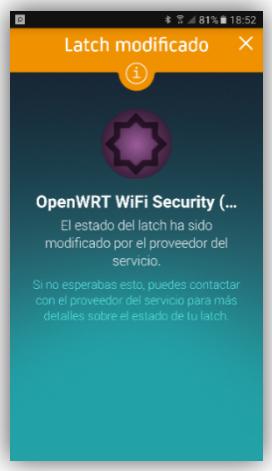


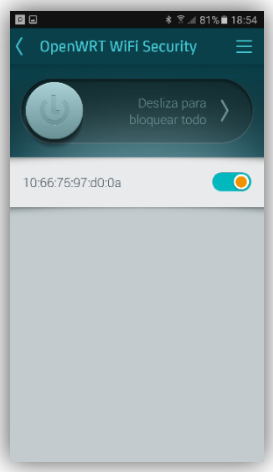
Usando el plugin

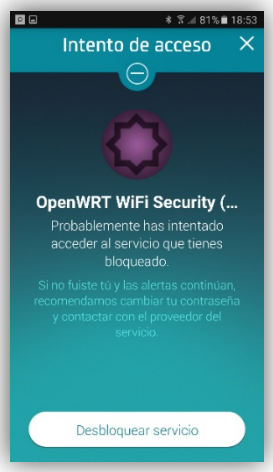
Una vez hayamos iniciado el servicio del plugin, este empezará a controlar los dispositivos que intenten conectarse a la red Wi-Fi.

Si un dispositivo inalámbrico intenta conectarse a nuestro router, el dispositivo que hayamos pareado nos mostrará una alerta y lo bloqueará. Tendremos algo de tiempo, el configurado previamente en el "Reload Time", para dar permisos o no de acceso a la red a este dispositivo.

Después de este tiempo, el servicio volverá a analizar la lista de dispositivos conectados y si el dispositivo anterior sigue conectado y no tiene permisos seguirá bloqueado y será expulsado de la red, mostrándonos otra alerta.







El dispositivo bloqueado seguirá siendo expulsado continuamente de la red hasta que se le de permisos o deje de intentar conectarse.

Si el dispositivo deja de intentar conectarse, cuando se vuelva a comprobar la lista de dispositivos conectados el servicio del plugin lo borrará de la lista en nuestro dispositivo pareado.

Como puede verse en las imágenes hay dos alertas diferentes. La primera se utiliza para avisar que un dispositivo que no estaba conectado se está intentando conectar. La segunda alerta te avisa de que un dispositivo que ha intentado conectarse y no le has dado permisos se sigue intentando conectar a la red.

Obviamente las alertas se pueden silenciar para cada dispositivo si resultan molestas porque un dispositivo no deje de intentar conectarse.

Todo esto te ofrece la garantía de que tendrás una forma inteligente, rápida y fácil de administrar todos los dispositivos que intentan conectarse a tu red desde tu dispositivo pareado.

Renuncia

- Los nombres de marcas, logotipos y marcas comerciales utilizados en este documento son propiedad de sus respectivos propietarios.
- Esta lista de cualquier firma o sus logotipos no pretende implicar ningún endoso o afiliación directa con Movistar o OpenWRT

13. Anexo – Installing and using the system

Latch

Latch OpenWRT Manual

OpenWrt
Wireless Freedom

Prerequisites

- Installation of OpenWRT previously configured with internet access and administrator password
    - Although this plugin was created to work in OpenWRT, this manual wont explain the installation or configuration of OpenWRT because it depends of each router.
    - We will only describe how to install on “Homestation ADB PDG A4001N” router
    - The configuration of OpenWRT to get internet depends of the topology of each network

- One account created at “latch.elevenpaths.com” and one application with the name you want. For instance: “OpenWRT WiFi Security”.
    - You don´t need configure anything in the application, only save the Application id and Secret key for configuring the plugin

- Installation Package of plugin for your router
    - The installation package used in this manual was compiled for routers with Broadcom BCM63xx SoC family, and will be installed in the router “Home Station ADB PDG A4001N” from Movistar.
    - If you need the installation package for other router you can download the source code from github and compile it for other SoC family.

- Any SSH Connection software to install the plugin.
    - In this manual we will use Putty.

Preparing OpenWRT

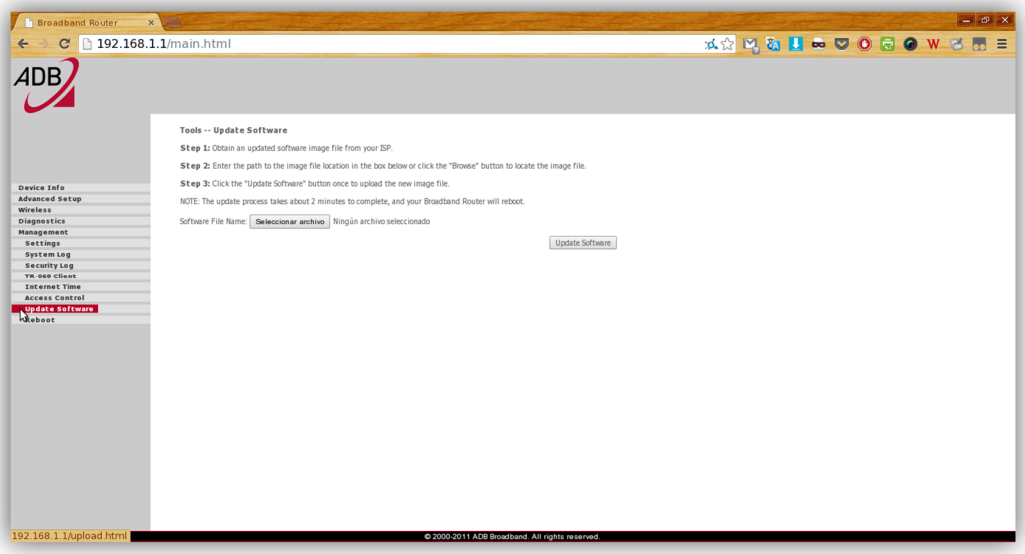
First of all you need a router compatible with OpenWRT and the correct firmware version to install it. Also you need know how to install it because the installation of OpenWRT depends of the router.

In this case we will use a “Home Station ADB PDG A4001N” router from Movistar.

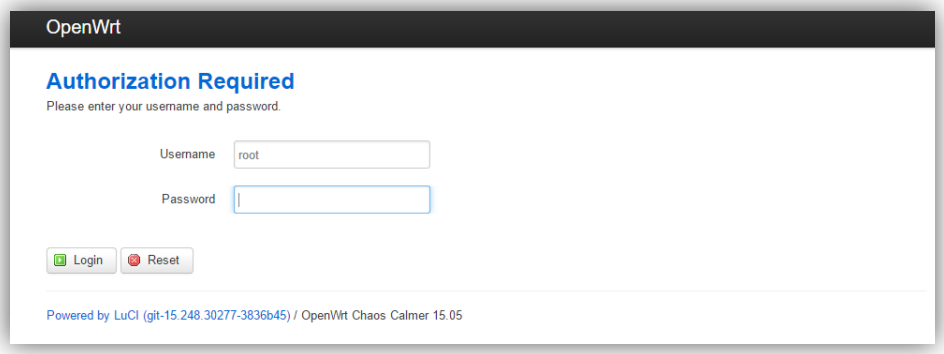
It uses a Broadcom BCM6328 SoC, so we will need a firmware of OpenWRT compatible - “brcm63xx” in this case. You can get the correct image for this router from my GitHub.

The installation of OpenWRT is really simple in this model. You only need update the firmware like if you do with the oficial firmware.

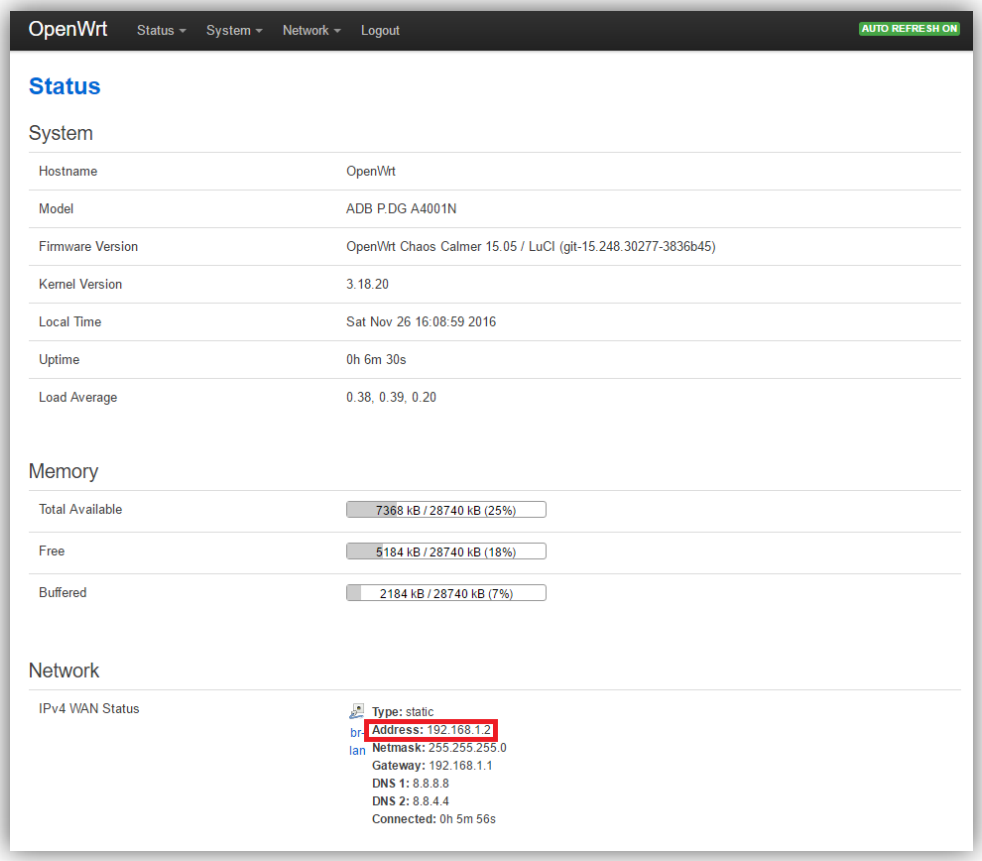
4. Access to the administration web page at: http://192.168.1.1/main.html
5. The system will ask you for a user and password. You should enter 1234 in both input if you didnt change it before.
6. On the left menu select “Management” and click on “Update Software”
7. Click on search button and select the OpenWRT firmware you downloaded before, click to “Update Software” and wait around 1-3 minutes. After the restart you should see the interface of OpenWRT.



Plugin Installation



OpenWRT Login Interface



Once we have the OpenWRT installed and configured we can connect with it using any SSH software we selected before. In our case we will use Putty.

Remember the IP address that you configured to Access to the router administration interface and the password to adminístrate it because we will need the both to connect using SSH.

The plugin installation is really easy, you only need connect to router using Putty or other SSH connection software, send the installation package to the router using SCP commands and execute it.

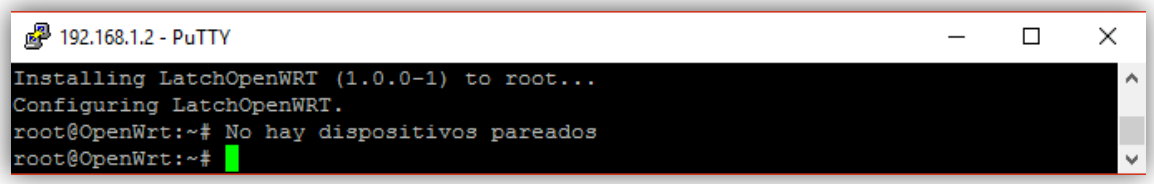
```
scp <package name> root@RouterIP:/

opkg update ; opkg install <name package>
```

Where the “package name” is the name of your installation package and RouterIP is the ip used to administer the router.

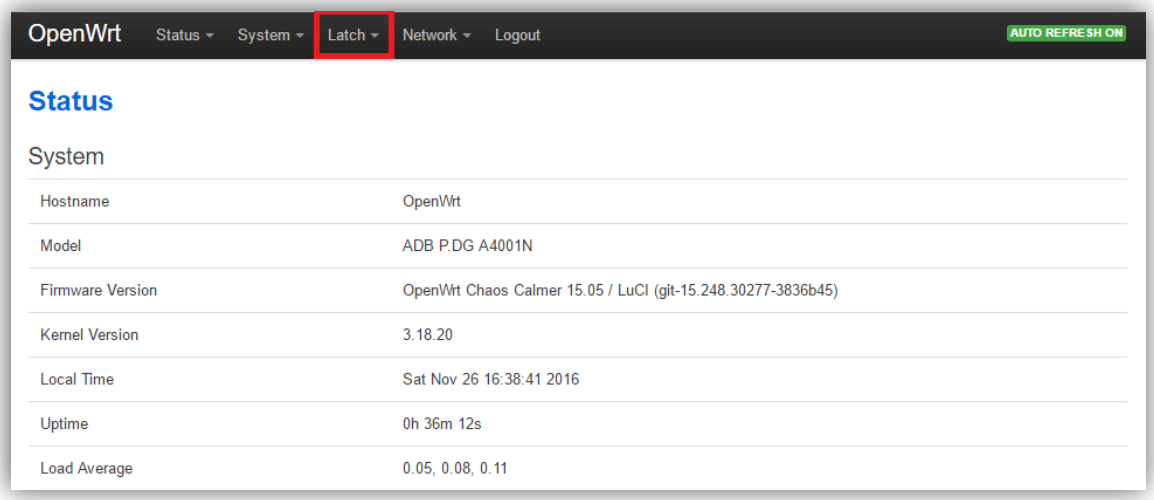
In my case i have the installation package for this router on my Github, so i can install the plugin without download the package previously and do all the job using the next command:

```
opkg update ; opkg install openssl-util ; opkg install
http://github.com/JCameroMartin/LatchOpenWRT/raw/master/LatchOpenWRT_1.0.0-
1_brcm63xx.ipk
```

This process will take a while. You will know that the installation has finished when you see something like the next window.

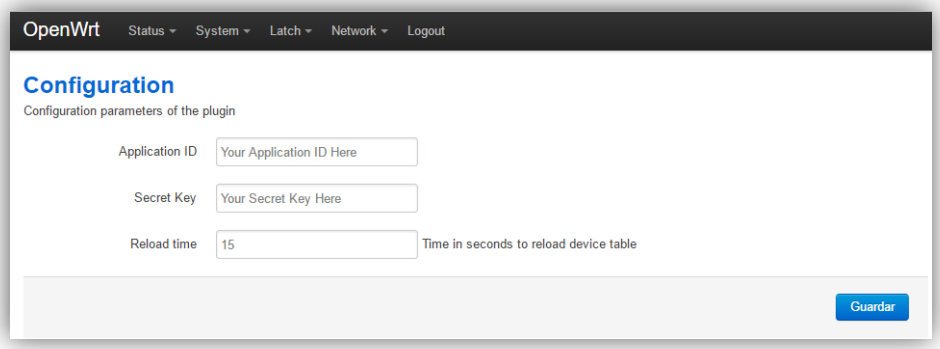


Now you can go to the OpenWRT Administration Web and see that the plugin was succesfully installed.



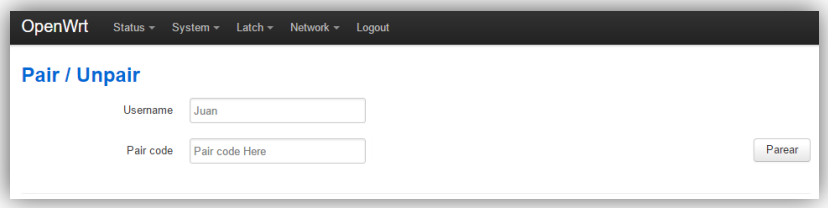
Configuration

The configuration of the plugin is really easy. You need go to “Latch – Configuration” to insert your application id and secret key.

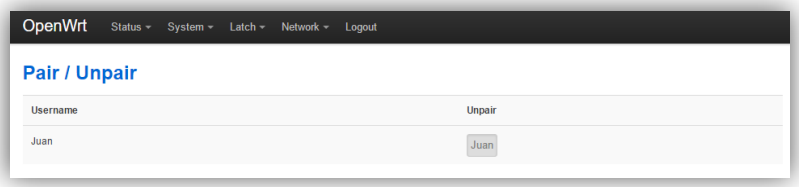


After save the configuration you need to pair a control device to administrate the plugin. You need go to “Latch – Pair/Unpair” to do it.

It is really simple, only need put a name to recognise the control device and use the pair code provided for the Latch Application previously installed in the control device.



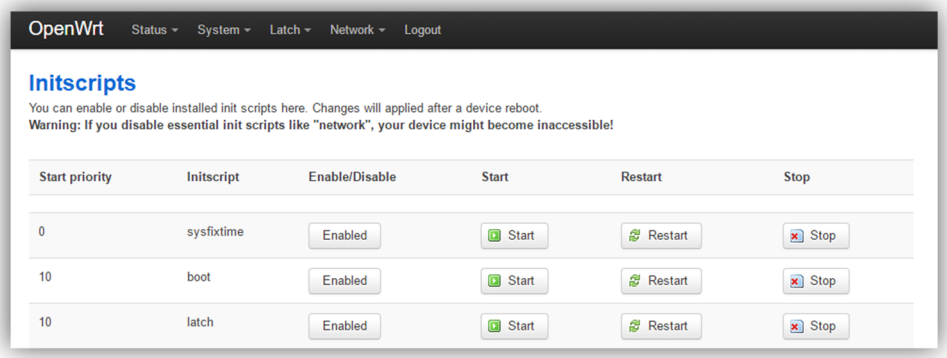
After 30 seconds this window will be reloaded and if the control device was succesfully paired you will see a new window like this:



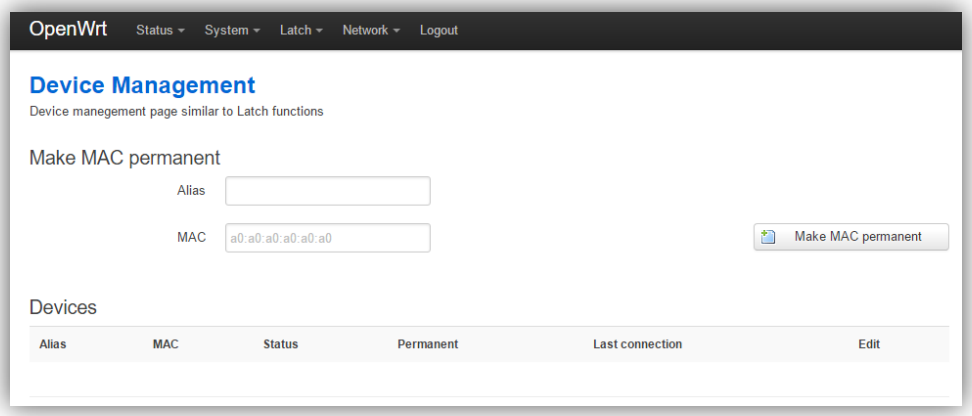
You can unpair the control device clicking in the button with it name to pair a new control device if you want.

At this step you have finished to install and configure the plugin. To start the service you only need go to “System – Startup”, find the “latch” service and click in Start or Restart button.

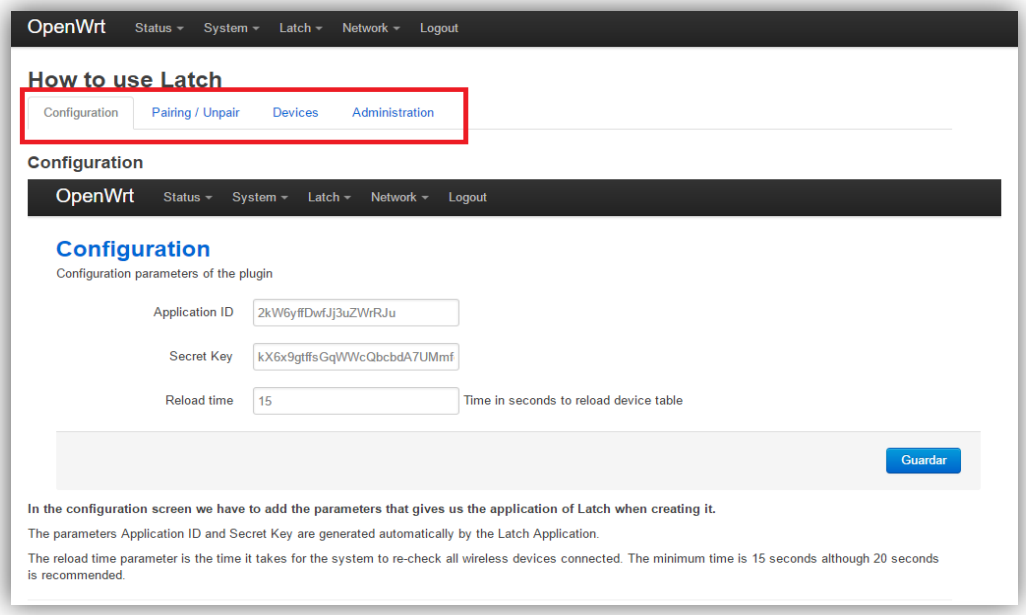
NOTE: You should restart the service if you unpair the control device and pair a new one, or if you change any configuration parameter.



Although you can close the administration web page and the plugin will do all the work for you, you can add a Permanent MAC or edit some information of this permanent devices in "Latch – Devices"



Advanced information can be found in the plugin help. See it at "Latch – Help"

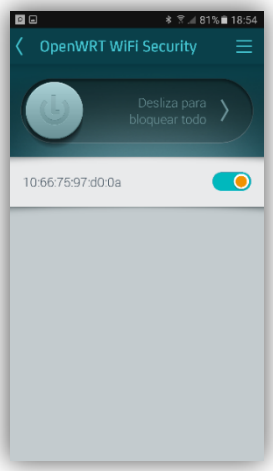


Using the plugin

Once we have started the plugin service it will start the control of Wireless devices trying to connect to our router.

If one wireless device try to connect to our router, the control device will show us a alert and will block it. We have some time, the "Reload time" (The time configured in the administration web page), to give, or not, permissions to access at our network.

After this time the service will reload the devices table and if it continue connected and blocked it will alert us and expulse it from our network.



The device will continue blocked for the “Reload time” and if the device continue trying to connect before the reload time it will be continuesly expulsed.

If the device dont try to connect and the reload time finalize, the device is deleted from the control device.

As you can see you have 2 different alerts. The first one is used to alert that a device not connected now are trying to connect. The second one is used to alert that a device previously blocked continue trying to connect.

Obviously you can silence a alert for 1 device if it not stop try connect and ignore it.

This give you a warranty than you have a smart, fast and easy administration. Managing all wireless connections from your control device

Disclaimer

- Brand names, logos and trademarks used herein remain the property of their respective owners.
- This listing of any firm or their logos is not intended to imply any endorsement or direct affiliation with Movistar or OpenWRT