

UNIVERSIDAD  
REY JUAN CARLOS

DOBLE LICENCIATURA EN INGENIERÍA  
SUPERIOR DE TELECOMUNICACIONES Y  
ADMINISTRACIÓN Y DIRECCIÓN DE  
EMPRESAS

Curso Académico 2015/2016

Proyecto Fin de Carrera

*CloudTrust: Aplicación Web de Configuración de red  
en un entorno virtualizado*

Autor : David Gutiérrez Melendro

Tutor : Dr. Gregorio Robles

Co-Tutor: Ing. Beatriz Sánchez Castañeda

*Dedicado a  
mi familia*



# Agradecimientos

Este trabajo como estudiante de la doble licenciatura de Ingeniería de Telecomunicación y Administración y dirección de empresas, es sólo una pequeña parte, pero sí que es una en las que más esfuerzo, corazón y sacrificio he puesto. Lo mejor de pertenecer a una comunidad es que se comparten alegrías y sufrimientos, y en mi trayectoria en la universidad, tengo que agradecer mucho a muchas personas.

Me siento un verdadero afortunado al haber contado con maravillosos docentes, que no solo han sido profesores sino también compañeros de los que he aprendido y he recibido además una formación humanística.

Me gustaría mencionar especialmente el trabajo de uno de esos docentes, que transforma su trabajo en pasión, gracias Gregorio, por ser cercano, comprensivo y un excelente formador.

No puedo sino también agradecer enormemente la oportunidad que me ha brindado Alcatel-Lucent (ahora Nokia), de pertenecer y aportar en uno de sus proyectos de desarrollo e investigación, así cómo de aprender tareas de gestión y de Project Management.

Así cómo agradecer la ayuda inestimable de los otros dos organismos colaboradores dentro del proyecto, la Universidad Politécnica de Madrid, y el grupo Innovati. Beatriz, gracias a ti he aprendido a ser un profesional y a ser crítico, así cómo, he aprendido a reconocer el valor de mi trabajo. Gracias por enseñarme cosas tan importantes.

En el apartado más personal, me llevo muchos buenos momentos de mi paso por la vida universitaria, pero me llevo mucho más, un hermano, gracias Fernando por formar parte de esta etapa de mi vida, y las que quedan. Y como él, muchos más, MariLuz, Patricia... demasiados para mencionarlos a todos.

Por último y quizá por ello más importante, quiero agradecer a mis padres su apoyo incondicional. A mi madre primero por madre, que eso ya de por sí tiene infinito valor, por luchadora, mamá eres la persona más fuerte que conozco y por tu amor incondicional y desinteresado. A

mi padre, gracias por ser amigo y padre a la vez, formador si la situación lo pedía, gracias por sacrificar todas esas cosas por mí, gracias a ti conozco el significado del trabajo duro, el sacrificio y lo importante de asumir responsabilidades, cualquier cosa que diga se quedará corta, en general por todo, gracias papá y mamá.

# Resumen

En la actualidad el navegador Web se ha convertido en un elemento imprescindible para el uso de Internet, la evolución de los lenguajes y tecnologías Web han dado como fruto múltiples herramientas asociadas directamente al navegador.

La integración de programas y protocolos han llevado a los navegadores a crecer rápidamente soportando nuevas funcionalidades, opciones, etc. que suponen una mayor complejidad de los mismos y por tanto un incremento de los factores de riesgo.

Actualmente los entornos web se mueven hacia la virtualización y la nube. Dentro de esta tendencia se observan distintas vulnerabilidades en la navegación web y en el hogar, así como sus posibles soluciones, para poder ofrecer servicios seguros tanto desde el punto de vista del ofertante, del poseedor de la plataforma que da el servicio, como del usuario final.

Dentro del hogar de los usuarios, las vulnerabilidades son muy variadas. Desde accesos no deseados a través de la red de nuestro hogar y no fácilmente controlables, hasta webs de las que nos debemos proteger.

Este proyecto de fin de carrera consiste en el diseño, implementación e integración de una aplicación web de configuración y gestión de una red virtual implementada según el proyecto de seguridad en la nube denominado *CloudTrust*.

El proyecto *CloudTrust*, ha sido un proyecto financiado por el Ministerio de la Competencia durante 3 años y desarrollado conjuntamente por tres empresas colaboradoras, Alcatel-Lucent (actualmente Nokia), La Universidad Politécnica de Madrid, y el grupo Innovati.

El objetivo principal del proyecto *CloudTrust* es construir un entorno de navegación segura integrado con los actuales sistemas de hosting/housing y que permita atajar de manera radical las actuales vulnerabilidades en la navegación de los usuarios. La solución de *CloudTrust* es una solución que aporta confianza bidireccional, ambos extremos están seguros en la navegación, la empresa puede ofrecer una imagen en Internet sabiendo que el usuario final verá lo que

realmente se oferta y no es víctima de ningún ataque.

*CloudTrust* alcanza una solución software que ofrece una alternativa virtual a los actuales CPE (Customer Premise Equipment). La transformación se realiza mediante la construcción de un entorno que simula y virtualiza un entorno doméstico estándar. Su objetivo principal, es conseguir un paquete completo de software que sea comercial y que resulte competitivo, sin por ello comprometer la usabilidad y aún mucho menos la seguridad del usuario.

Así es tanto que, la seguridad del cliente se convierte en uno de los objetivos principales. Por ello, a parte de asegurar las condiciones básicas de conectividad de un CPE, se agregaron y desarrollaron sistemas como la detección de dispositivos, controles de consumo, identificación de usuarios o el registro de eventos.

Se quiso poner especial énfasis en la importancia de crear un entorno amigable para el usuario, de tal forma que el cliente menos experimentado con entornos en red fuese capaz de explotar todas las bondades de *CloudTrust*, y lo hiciera siempre de forma segura. Por ello se implementó, una aplicación web de configuración basada en Django y escrita en Python, que se integrase completamente con el entorno de red desarrollado y con todas sus funcionalidades.

Así pues, el objetivo de este proyecto de Fin de Carrera, es realizar un estudio previo de los requerimientos necesarios para implementar correctamente esta aplicación web, desarrollarla y codificarla, y finalmente crear un producto cerrado integrable dentro de la solución software *CloudTrust*.

# Summary

Currently web browsers have become an essential element for the Internet navigation, the continuous developments on web technologies has delivered numerous tools strongly related to the web browsers.

New programs or protocols has lead to the web browsers to grow exponentially supporting new features, options, etc. that leads to them to grow in complexity and increasing the risk factors.

Today, web is moving towards virtualization and cloud. Within this trend, new vulnerabilities can be observed on the web browsing and in particular at home networks. So, new solutions are appearing to develop new tools to fight back those new risks.

Inside the internet residential network, the vulnerabilities are diverse. Starting from not desired accesses, that are not easily manageable, to websites that we need to protect from.

This project consists on the design, implementation and integration of a web application that allows to configure and manage a virtualized network implemented inside the software solution of safe browsing on the cloud named *CloudTrust*.

The *CloudTrust* project, has been financed by the Ministerio de la Competencia for a period of three years, and developed together by three collaborating companies, Alcatel-Lucent (currently Nokia), the Universidad Politécnica de Madrid, and the Innovati group.

The main goal of the *CloudTrust* project is to build a safety environment for a safer browsing, making use of the current hosting/housing services that allows to perfectly manage the various vulnerability issues previously mentioned.

The *CloudTrust* solution, is a solution that provides two-way confidence, both sides of the connection are safe on the navigation of the net, the company that offers the service can offer this package as a true solution for a potential client for the safety of his home network.

*CloudTrust* reaches a software solution that offers a virtual alternative to the current CPE



(Customer Premise Equipment). The transformation happens through the construction of an environment that simulates and virtualizes a standard home network. The main goal is to achieve a complete commercial software package, that results competitive, but without compromising usability and even more important, safety.

That is so, the safety of the client is the first commitment for *CloudTrust*. To achieve this, various functionalities have been implemented, as device detection system, users identification or log events record.

Special emphasis wanted to be placed on the creation of an user-friendly environment for the users, specially for those users which aren't as experienced on browsing their own network as others, so they could do it as safe as the rest. That is why the web application was implemented based on Django, coded on Python and fully integrated on the environment developed and with all his functionalities.

To summarize, the main goal of this final project, is to make a previous study on the needed requirements to correctly implement this web application, develop it, coded it, and finally create a closed solution package integrated inside the *CloudTrust* project.

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Contexto . . . . .	1
1.2. Motivación . . . . .	2
1.3. Estructura . . . . .	3
<b>2. Objetivos</b>	<b>5</b>
2.1. Objetivos generales . . . . .	5
2.2. Objetivos específicos . . . . .	6
<b>3. Requisitos y Especificaciones</b>	<b>7</b>
3.1. Requisitos funcionales generales . . . . .	8
3.2. Requisitos funcionales de la aplicación de gestión del hogar . . . . .	8
3.3. Requisitos de interfaces de usuario . . . . .	9
3.4. Requisitos de configuración de la aplicación web. . . . .	10
3.5. Requisitos del entorno de desarrollo de la aplicación web. . . . .	11
3.6. Requisitos de rendimiento . . . . .	12
<b>4. Estado del Arte</b>	<b>13</b>
4.1. Django . . . . .	13
4.2. Bootstrap . . . . .	17
4.3. Python 3 . . . . .	17
4.4. PostgreSQL . . . . .	18
4.5. Servidor web . . . . .	19
4.6. VNX . . . . .	19
4.7. HTML . . . . .	20

4.8. CSS . . . . .	21
4.9. Open vSwitch . . . . .	21
4.10. Frenetic . . . . .	21
<b>5. Diseño e Implementación</b>	<b>23</b>
5.1. Diagrama Funcional . . . . .	25
5.2. Diagrama Físico . . . . .	26
5.3. Diagrama de Red . . . . .	28
5.4. Estructura y definición del entorno de desarrollo . . . . .	29
5.5. Estructura y definición de la aplicación web de configuración . . . . .	30
5.6. Estructura de la Base de Datos . . . . .	33
5.7. Estructura de las carpetas de la aplicación . . . . .	35
<b>6. Resultados</b>	<b>39</b>
6.1. Caso de estudio 1: Primer acceso . . . . .	39
6.2. Caso de estudio 2: Acceso desde un dispositivo . . . . .	49
6.3. Caso de estudio 3: Algunas funciones más . . . . .	52
6.4. Caso de estudio 4: Acceso de usuario no administrador . . . . .	57
<b>7. Conclusiones</b>	<b>59</b>
7.1. Lecciones aprendidas . . . . .	59
7.2. Conocimientos aplicados . . . . .	61
7.3. Trabajos futuros . . . . .	62
<b>A. Acrónimos Requisitos y Especificaciones</b>	<b>63</b>
<b>Bibliografía</b>	<b>65</b>

# Índice de figuras

4.1. Modelo Vista Controlador . . . . .	14
4.2. Extracto de código: Class <i>Register</i> . . . . .	15
4.3. Extracto de código: Class <i>Command</i> . . . . .	16
4.4. Arquitectura Frenetic . . . . .	22
5.1. Diagrama Funcional . . . . .	25
5.2. Diagrama Físico . . . . .	26
5.3. Diagrama de Red . . . . .	28
5.4. Arquitectura de Red . . . . .	29
5.5. Index WebAPP . . . . .	31
5.6. Página principal . . . . .	32
5.7. Diagrama de la Base de Datos. . . . .	33
5.8. Estructura de carpetas. . . . .	35
6.1. Index: Acceso root . . . . .	40
6.2. Primer acceso: Guía de navegación, bienvenida . . . . .	41
6.3. Primer acceso: Guía de navegación, vista de dispositivos conocidos . . . . .	41
6.4. Primer acceso: Guía de navegación, vista de dispositivos desconocidos . . . . .	42
6.5. Primer acceso: Guía de navegación, vista de usuarios . . . . .	43
6.6. Primer acceso: Guía de navegación, registro de usuarios . . . . .	43
6.7. Primer acceso: Vista de usuario David . . . . .	44
6.8. Primer acceso: Cambio de perfil . . . . .	44
6.9. Primer acceso: Nuevo dispositivo en la red . . . . .	45
6.10. Primer acceso: Vista dispositivo desconocido . . . . .	45

6.11. Primer acceso: Navegación dispositivo h1 . . . . .	46
6.12. Primer acceso: Dispositivo h1 asignado a usuario David . . . . .	46
6.13. Primer acceso: Cambio de perfil a usuario David . . . . .	47
6.14. Primer acceso: Dispositivo h1 con acceso a red . . . . .	47
6.15. Primer acceso: Configurando h1 . . . . .	48
6.16. Acceso desde un dispositivo: Primer acceso . . . . .	49
6.17. Acceso desde un dispositivo: Segundo acceso . . . . .	50
6.18. Acceso desde un dispositivo: Acceso a Google . . . . .	50
6.19. Acceso desde un dispositivo: h2 asignado a David . . . . .	51
6.20. Algunas funciones más: h3 trata de navegar . . . . .	52
6.21. Algunas funciones más: Nuevo usuario Guest María . . . . .	52
6.22. Algunas funciones más: vinculamos h3 a María . . . . .	53
6.23. Algunas funciones más: comienza la conexión . . . . .	53
6.24. Algunas funciones más: termina la conexión . . . . .	53
6.25. Algunas funciones más: Vista dispositivos David . . . . .	54
6.26. Algunas funciones más: Función del control parental . . . . .	55
6.27. Algunas funciones más: Log . . . . .	55
6.28. Algunas funciones más: Función Show only . . . . .	56
6.29. Acceso de usuario no administrador: Vista de dispositivos de David . . . . .	57
6.30. Acceso de usuario no administrador: Histórico de consumo de David . . . . .	57

# Capítulo 1

## Introducción

### 1.1. Contexto

En el año 2015 en España, el 78,7 % de la población disponía de acceso a internet en su hogar y el 77,8 % con conexión de banda ancha según datos del Instituto Nacional de Estadística [11].

Para el acceso a este servicio universal, es necesaria una línea telefónica en casa del usuario, que ésta se encuentre dada de alta en el servicio de internet y por último disponer de un home Customer Premises Equipment (*hCPE*) instalado en el hogar. El *hCPE* es el terminal encargado de conectarse con el operador en el Punto de Terminación de Red y por el pasará todas las comunicaciones que entren o salgan del hogar del usuario a internet.

En la actualidad los *hCPE*, son equipos con funciones limitadas que:

- Requieren una configuración inicial por parte del operador que únicamente puede ser realizada de manera presencial. Además, la gestión del dispositivo es en general costosa para el operador.
- La adición de servicios novedosos requiere frecuentemente de la sustitución de los equipamientos.
- Las modificaciones de configuración en tiempo real del equipamiento para adaptarse a las condiciones de la red son limitadas.
- Las actualizaciones para resolver vulnerabilidades del ámbito del software requieren de conocimientos técnicos del usuario o del desplazamiento de personal cualificado al hogar

del usuario.

La rigidez de un equipamiento indispensable para proporcionar el servicio de acceso a internet, como es el *hCPE*, sumado a la velocidad con la que aparecen y evolucionan los servicios, los convierte en un punto vulnerable de la arquitectura.

Los *hCPE* suponen un gran gasto de recursos para los operadores, que han de gestionarlos y mantenerlos de manera presencial y un gran impedimento para la innovación y la introducción de nuevos servicios a los usuarios en un tiempo razonable.

Todos estos motivos son razón suficiente para elaborar un nuevo entorno de red virtualizado que permita desarrollar e innovar junto a conceptos antes mencionados, como son la nube o la virtualización, siento éste lo más seguro posible, incluyendo todos los beneficios de las arquitecturas actuales, y añadiendo nuevas funcionalidades.

Y éste es precisamente el objetivo de éste proyecto, crear una aplicación de configuración web que de sentido a todo el conjunto y permita al usuario (ó incluso cliente) final entender de qué herramientas nuevas dispone en su nueva red de hogar.

## 1.2. Motivación

A lo largo de mi trayectoria estudiantil, y en las etapas finales durante mi trayectoria profesional, siempre me he sentido atraído por las nuevas tecnologías, por lo más *puntero*. Actualmente nos encontramos en un punto de la historia de Internet en el que, lo más novedoso es precisamente el uso y la explotación de las tecnologías en la nube. Su uso, promete interesantes avances en los campos de la seguridad en la red, de la versatilidad de Internet o también en el ámbito empresarial con su fuerte impacto en la reducción de costes y en nuevas estrategias corporativas.

Tuve la enorme oportunidad de entrar a trabajar en el proyecto *CloudTrust* en una gran empresa del sector de las telecomunicaciones, Alcatel-Lucent (actualmente Nokia) que compartían mi visión de investigador, y me ofrecieron las herramientas necesarias para poder llevar a cabo este proyecto.

Mi participación en él se vio doblemente justificada, pues no sólo realicé funciones de desarrollo sino también de gestión y de explotación del proyecto.

Las grandes compañías del sector se encuentran en un momento de expansión y redescubrimiento de la red, muchas de las limitaciones anteriores han desaparecido gracias al concepto de la nube, y la reestructuración de la arquitectura es inminente. Las empresas que consigan ser líderes marcarán en gran medida el futuro de Internet y de cómo lo usamos.

El problema reside en que no solo las empresas ven en esta herramienta una nueva fuente de ingresos, sino que también terceros malintencionados pueden aprovecharse de este proceso de descubrimiento y encontrar vulnerabilidades cuando aún se están desarrollando nuevas aplicaciones y/o nuevas utilidades. Es por ello, que una de las principales preocupaciones del sector es, como es lógico, desarrollar e innovar, pero siempre poniendo desde el primer momento gran atención a la seguridad tratando de anteponerse a desarrolladores malintencionados.

Estas y otras preocupaciones de carácter personal, me han llevado a realizar mi proyecto *Aplicación Web de Configuración de red en un entorno virtualizado*.

## 1.3. Estructura

Se presenta a continuación una estructura resumida del proyecto:

- Para empezar, se presenta un resumen tanto en español como en inglés, del proyecto.
- El primer capítulo, denominado *Introducción*, contiene el contexto del proyecto, dónde se explica la situación actual en la que se ha desarrollado el proyecto así como su justificación. En segundo lugar encontramos un apartado de Motivación donde se recogen las razones que me han llevado a realizar el proyecto, y por último encontramos este resumen del esqueleto del proyecto.
- El segundo capítulo, *Objetivos*, recoge dos apartados, un primer apartado dónde se presentan los objetivos más genéricos del proyecto, y un segundo, donde se presentan objetivos más concretos.
- En el tercer capítulo, *Requisitos y especificaciones*, encontramos una descripción detallada de los requerimientos del proyecto definidos al comienzo del mismo.
- El cuarto capítulo, *Estado del arte*, explica cada una de las tecnologías usadas en este proyecto, así como su impacto en él, o su contribución.



- En el quinto capítulo, *Diseño e implementación*, se detalla la estructura y se define cómo se ha construido el proyecto. Se presenta una primera visión dividida en tres partes, diagrama funcional, donde se ven las relaciones funcionales del proyecto, el diagrama físico, en el que se presenta la estructura de los principales elementos hardware necesarios, y por último, un diagrama de red, donde se detalla las relaciones a nivel de red. A continuación, dentro de este mismo capítulo, se pretende dar una visión global de dónde y cómo se ha elaborado el proyecto, también se responde al porqué se ha hecho siguiendo esa estructura.
- El sexto capítulo, corresponde al capítulo de *Resultados*, donde se presentan una serie de casos de estudio habiendo finalizado el proyecto. Que tratan de representar las bondades de la solución, en situaciones potencialmente reales.
- El último capítulo, *Conclusiones*, se echa la vista hacia atrás y se presenta un resumen de los principales lecciones aprendidas, cuales han sido los conocimientos aplicados y qué trabajos futuros se pueden seguir desarrollando.

# Capítulo 2

## Objetivos

### 2.1. Objetivos generales

El objetivo principal consiste en elaborar una aplicación web que permita interactuar al potencial cliente de un proveedor de Internet o ISP, con la solución de *CloudTrust*. Este proyecto pretende dar respuesta, a su vez, a otra serie de cuestiones, que se enumeran a continuación:

- Presentar la situación actual y futura de las tecnologías NFV (Network Function Virtualization), y las oportunidades que ofrece el mercado, cómo explotarlo y obtener beneficios.
- Introducir las tecnologías SDN (Software Defined Network), así como, hacer uso de ellas para ofrecer entornos seguros de navegación
- Explicar cuáles son los beneficios ó vulnerabilidades que tiene trabajar con funciones de red virtualizadas. Y sobretodo, también explicar, cómo y cuánto de seguro es.
- Delatar cuáles son los servicios que va a ofrecer *CloudTrust* al usuario final.
- Virtualizar un dispositivo y estudiar cómo afecta esto al usuario, así como, qué beneficios podemos obtener al hacerlo
- Describir qué personas, o agentes del sector han contribuido en el proyecto
- Justificar la continuidad del proyecto, ¿Es viable económicamente, o dimensionamente posible?

## 2.2. Objetivos específicos

Para poder dar respuesta a los objetivos anteriores, podemos enumerar otra serie de objetivos más concretos que hemos ido completando hasta finalizar el proyecto:

- Crear un entorno de desarrollo virtual con *VirtualBox* que simule un escenario real de red doméstica, que no comprometa ni pierda funcionalidades básicas de conectividad de los servicios tradicionales.
- Desarrollar e implementar una solución definitiva que resuelva el compromiso entre la simplicidad de uso y la seguridad que se pueda alcanzar en una red doméstica de hogar.
- Hacer uso de varias herramientas de código libre, como pueden ser *Django*, *Python* ó *HTML* durante la fase de desarrollo, implementación y explotación del proyecto.
- Alcanzar un mayor grado de conocimiento y aplicar lo aprendido durante mis estudios universitarios en una herramienta con potencial real de convertirse en una solución software ofertable por proveedores ISP.
- Realizar un estudio preliminar de cuáles serían los componentes añadidos necesarios para poder virtualizar una red.
- Identificar cuales son los requisitos y especificaciones que requiere un proyecto de esta envergadura. Así cómo, elaborar los puntos de acción y responsabilidades de los agentes que van a contribuir en el proyecto.
- Alcanzar mayores conocimientos en competencias y administración sobre base de datos.
- En el terreno personal; Alcanzar competencias en administración de procesos y a aumentar la agilidad de gestión. Así cómo estudiar y aplicar estrategias de gestión de negocio.
- Elaborar una página web del proyecto (<http://CloudTrust.grupoinnovati.com/>) así como una cuenta de Twitter que dé visibilidad al proyecto (<https://twitter.com/mihogarenlanube>).
- etc.

## Capítulo 3

# Requisitos y Especificaciones

Previo elaboración del proyecto, se identificaron cuáles iban a ser los objetivos específicos que se querían alcanzar al finalizar el mismo. A continuación se presentan esos requisitos de funcionamiento.

Dentro de nuestra red privada o doméstica, el usuario final será capaz de acceder a una serie de funciones y/o servicios que le permitirán una navegación segura, tanto controlando el estado del hogar, como el acceso a servicios en Internet. A continuación, se expondrán cuales son los requisitos de las funciones que el usuario podrá realizar. Creemos importante definir lo que consideramos como red y como administrador dentro del contexto de nuestra aplicación web para no llevar a malinterpretaciones:

- Red: Entorno que realiza las funciones de envío/recepción de información entre dispositivos y entre el resto de unidades instaladas.
- Administrador: Entenderemos por administrador a aquel usuario con el rango de *super-user* que tendrá determinados privilegios (definidos en la especificación).

(Se han utilizado acrónimos para clasificar los requisitos. Especificados en el Apéndice A al final de este proyecto).

### 3.1. Requisitos funcionales generales

- DEF.HOG.FUN.001 - El entorno en el que desarrollaremos nuestra aplicación se trata de un domicilio u hogar familiar. Y será extensible a redes corporativas y públicas.
- ASU.HOG.FUN.002 - Por cada hogar tendremos un único vCPE (Router virtual).
- REQ.HOG.FUN.003 - Por cada hogar, al menos el administrador podrá configurar su red de acceso a Internet.
- ASU.DISP.FUN.004 - Todos los dispositivos en la red doméstica tendrán acceso tanto inalámbrico como por cable mediante el OpenvSwitch a redes externas.
- ASU.OVSW.FUN.005 - Todos los dispositivos podrán realizar reenvíos de paquetes a nivel 2.
- ASU.SIS.FUN.006 - El entorno cubrirá las funciones y servicios de conectividad básicas de un entorno doméstico normal (encaminamiento, DHCP o DNS).
- ASU.SIS.FUN.007 - El entorno permitirá realizar conexiones a nivel de aplicación, juegos en red, hogar virtual (aplicaciones de domótica), invitados WiFi, accesos a VPN corporativas.
- REQ.CON.FUN.008 - Toda persona dentro del hogar tendrá asignado un par *username/password*.
- REQ.SIS.FUN.009 - El administrador podrá acceder de forma remota a la configuración del router desde una red externa a la del hogar.

### 3.2. Requisitos funcionales de la aplicación de gestión del hogar

- REQ.CON.FUN.010 - Existirá una aplicación web disponible para la gestión de la red en el hogar.

- REQ.CON.FUN.011 - Para acceder a la aplicación los usuarios deberán introducir su *username/password*.
- REQ.USU.FUN.012 - Desde la aplicación un usuario podrá consultar sus dispositivos que están registrados en el hogar.
- REQ.USU.FUN.013 - Cada usuario mediante uno de sus dispositivos conectados podrá consultar el estado de actividad de los dispositivos de la red (activos/inactivos).
- REQ.USU.FUN.014 - Un administrador podrá activar o desactivar cualquier dispositivo de la red del hogar.
- REQ.USU.FUN.015 - Desde la aplicación cada usuario podrá acceder a su propio histórico de consumo.
- REQ.USU.FUN.016 - El usuario administrador podrá acceder al histórico de consumo del resto de dispositivos en la red.
- ASU.CON.FUN.017 - El controlador actualizará un fichero de log que mantenga un registro de los sucesos que ocurren en la red.
- REQ.USU.FUN.018 - Solo el administrador podrá acceder al fichero de sucesos log del controlador, para consultar los eventos que ocurran en la red.
- REQ.CON.FUN.019 - La aplicación ofrecerá la posibilidad de establecer controles parentales al administrador según márgenes horarios y por dispositivo.

### 3.3. Requisitos de interfaces de usuario

- REQ. APPWEB.INTF. 001 - La configuración del entorno seguro del hogar se realizará mediante una aplicación web.
- REQ. APPWEB.INTF.002 - La aplicación web deberá funcionar en *smartphones, tablets* y *PCs*.
- REQ. APPWEB.INTF.003 - Concretamente los navegadores soportados serán *Safari* y *Mozilla*.

- REQ.APPWEB.INTF.004 - La aplicación deberá ser accesible de forma simple e intuitiva.

### 3.4. Requisitos de configuración de la aplicación web.

- ESP.APPWEB.CONF.001 - Cuando se realice la instalación de la red *CloudTrust* en el hogar por primera vez, se creará un usuario administrador por defecto (*username*), una contraseña de usuario administrador, así como una contraseña de acceso a la red.
- ESP.APPWEB.CONF.002 - La aplicación web, una vez acceda este administrador por defecto podrá cambiar el *username* del administrador, su contraseña y la contraseña de acceso a la red.
- REQ.APPWEB.CONF.003 - La aplicación web ha de permitir configurar los nombres de usuarios.
- ASU.APPWEB.CONF.004 - Un usuario pertenece a un perfil
- ESP.APPWEB.CONF.005 - Por defecto, al iniciar por primera vez la aplicación web, se crearán 4 perfiles de usuarios, *admin*, *user*, *not validated* y *guest*.
- REQ.APPWEB.CONF.006 - Sólo el usuario administrador podrá cambiar el perfil de cualquier usuario.
- ESP.APPWEB.CONF.007 - El perfil de *admin* podrá, además, configurar los siguientes servicios.
- REQ.APPWEB.CONF.008 - Podrá activar/desactivar dispositivos en la red mediante un botón.
- REQ.APPWEB.CONF.009 - Podrá establecer controles parentales de acceso según dispositivo y según horario, mediante una franja horaria.
- ESP.APPWEB.CONF.010 - El perfil de *User* no podrá configurar ningún servicio dentro de la aplicación web, pero sí podrá tener acceso a los servicios de la aplicación web.

- ESP.APPWEB.CONF.011 - El perfil de *Guest* no tendrá acceso a la Aplicación web, únicamente podrá acceder a los recursos de red.
- REQ.SIS.CONF.012 - La aplicación web ha de permitir al administrador añadir/eliminar dispositivos de cada usuario, valdrá con añadir la dirección MAC.
- REQ.SIS.CONF.013 - A nivel global se podrá definir un número máximo de dispositivos por hogar.
- REQ.SIS.CONF.014 - A nivel global se podrá definir un número máximo de usuarios por hogar.
- REQ.APPWEB.CONF.015 - La aplicación web permitirá configurar repartos de ancho de banda según un determinado porcentaje para cada dispositivo.
- REQ.APPWEB.CONF.016 - La aplicación web permitirá configurar prioridades de acceso al ancho de banda en caso de congestión para cada dispositivo.
- REQ.APPWEB.CONF.017 - La aplicación web permitirá configurar controles parentales según márgenes horarios para cada dispositivo

### **3.5. Requisitos del entorno de desarrollo de la aplicación web.**

- ESP.SWEB.DES.001 - El servidor web funcionará bajo el framework *Django* con lenguaje *Python*.
- ESP.SWEB.DES.002 - El servidor web trabajará con bases de datos *PostGres*.
- ESP.SWEB.DES.003 - El servidor web trabajará con *HTML 5*, *Javascript* y con *CSS 3*.
- ESP.APPWEB.DES.004 - La aplicación web en el lado del cliente trabajará con *HTML 5*, *Javascript* y *CSS 2*.
- ASU.OVSW.DES.005 - El home CPE será un *OpenvSwitch*.
- ASU.OVSW.DES.006 - Todos los dispositivos se conectarán a la red de acceso mediante un *Openvswitch* que utilice los protocolos *OpenFlow*, *NetFlow* y *SFlow*.



### 3.6. Requisitos de rendimiento

- ASU.CON.REND.001 -El tráfico generado por el controlador deberá tener una influencia mínima en los recursos de red, y no generar congestión de tráfico en ningún momento
- ASU.CON.REND.002 - El controlador deberá realizar un reparto de ancho de banda equitativo
- ASU.CON.REND.003 - Por defecto en casos de congestión de red se asegurará igual ancho de banda a todos los dispositivos conectados
- ASU.SIS.REND.004 - El protocolo de comunicaciones seguro deberá asegurar un consumo de procesador de los terminales móviles mínimo.
- ASU.SIS.REND.005 - El tiempo de respuesta de la aplicación web no debe superar en ningún momento los 5 segundos.

# Capítulo 4

## Estado del Arte

### 4.1. Django

Durante la fase de desarrollo de la aplicación, se llegaron a varios consensos que delimitaron el ámbito de trabajo, así como la definición de las herramientas de las que haríamos uso a lo largo del proceso de desarrollo de todas las funcionalidades implicadas dentro de la solución *CloudTrust*.

Así, una de las primeras decisiones que se tomaron fue el uso del framework de *Django* [1]. Django es un framework de alto nivel basado en Python que promueve la creación de aplicaciones web de forma rápida, limpia y con diseños pragmáticos.

*Django es un entorno de desarrollo web escrito en Python que fomenta el desarrollo rápido y el diseño limpio y pragmático.*

Está basado en un diseño denominado MVC o *Modelo Vista Controlador* que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario. Es altamente recomendable su uso, pues permite reutilizar código, permite realizar cambios o actualizar aplicaciones sin alterar el resto de las partes. La arquitectura, como antes mencionada, sigue el siguiente patrón:

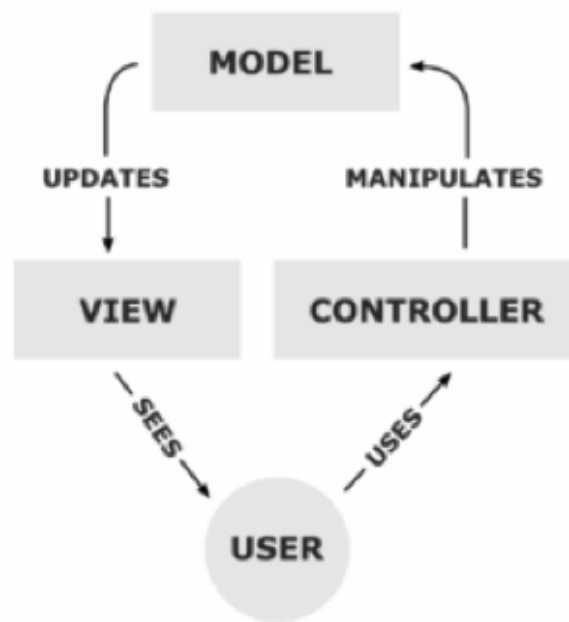


Figura 4.1: Modelo Vista Controlador

Son especialmente interesantes de mencionar algunos de los módulos u opciones incluidos dentro del framework de *Django*, de los cuáles, hemos hecho uso los siguientes:

- **Módulo admin:** Una de las partes más poderosas de *Django* es la interfaz automática de *admin*. Puede leer *metadata* del modelo y proveer al usuario inmediatamente de una interfaz lista para comenzar a añadir contenido al sitio. Ha sido una herramienta tremendamente útil en las labores desarrollo y testeo.
- **Mapeador de URLs:** *Django* hace uso de un esquema elegante de URL's basados en expresiones regulares, que permite elaborar arquitecturas de navegación dentro de la página web de alta calidad. Se basa en un método muy simple que responde ante peticiones que contienen esas expresiones
- **Models:** Otra de las características principales de *Django* reside en la simplicidad con la que trabaja con bases de datos. El modelo de nuestra base de datos, nace de una única y definitiva fuente de información, el archivo *models.py*. Contiene los campos esenciales y la estructura de la información que estamos almacenando. Este es un ejemplo de una de las tablas:

```
class Register(models.Model):
    user = models.ForeignKey(User)
    network_element = models.ManyToManyField(ElementoDeRed)
    registered_date = models.DateTimeField(null=True, blank=True)
    min_limit = models.IntegerField(max_length=2, blank=True, null=True)
    max_limit = models.IntegerField(max_length=2, blank=True, null=True)
    def __unicode__(self):
        return smart_unicode(self.user)
```

Figura 4.2: Extracto de código: Class *Register*

- API base de datos: *Django* provee además de una API con multitud de métodos y funciones muy útiles para establecer relaciones y una comunicación bi-canal entre el usuario y la base de datos.
- Plantillas con herencia (*Inheritance Templates*): Se ha hecho uso de otra herramienta más de *Django*, que nos permite establecer una jerarquía de plantillas *HTML*, para construir el esquema de navegación mucho más simple. Contamos por tanto, con simplicidad, rapidez de carga (que a la hora de la navegación web es muy importante), y claridad de código. Mediante la herencia, podemos crear plantillas *padre* y múltiples plantillas *hijas* que heredan de la plantilla *padre*, métodos, funciones y/o grandes extractos de código.
- *Templatetags*: Durante el desarrollo de este proyecto, se he hecho uso en múltiples ocasiones de esta funcionalidad de *Django*. La limitación que presenta *HTML* al no poder codificar lógica sino es mediante otras codificaciones externas, es fácilmente salvable mediante el uso de *Templatetags*. Cuando se produce una petición de un usuario solicitando un determinado contenido, que depende exclusivamente de algún elemento de entrada, las *Templatetags* nos permiten establecer funciones que ante este parámetro devuelvan las salidas deseadas y las presenta, si se desea, en formato *HTML*. En mi opinión personal, a la hora de trabajar con *Django*, creo que esta es, en efecto, una de sus herramientas más poderosas.
- Host de archivos *static*: En la fase de desarrollo, una vez más, *Django* facilita la vida al programador habilitando un servidor provisional de ficheros estáticos que sirve con el propósito de servir todas las plantillas, archivos *CSS*, *js* y/o imágenes.

- *Admin custom commands*: Otra herramienta interesante de mencionar de la que se ha hecho uso es la de crear nuestros propios comandos ejecutables por línea de comandos. Para la realización de este proyecto entre las funcionalidades planteadas se encuentra la de permitir al usuario administrador establecer un control de acceso a la red según horario (control parental). Se necesitó elaborar un demonio que verificase periódicamente si un usuario tenía permiso de acceso según la hora de acceso y otorgarlo o denegarlo según la situación. A continuación se muestra el extracto de código que lo habilita:

```

"""Para lanzar el comando ejecutamos en terminal python manage.py comandos"""
"""en este caso python manage.py parentalControl"""
class Command(BaseCommand):
    help = "USAGE: python manage.py parentalControl"

    option_list = BaseCommand.option_list + (
        make_option('--myoption', action='store',
                    dest='myoption',
                    default='default',
                    help='option help message'),
    )

```

Figura 4.3: Extracto de código: Class *Command*

A modo de resumen sus principales ventajas residen principalmente en lo comentado, se trata de un entorno de trabajo que facilita la implementación de los distintos módulos típicos dentro de una aplicación web. Por otro lado, ofrece un alto nivel de seguridad, y esto es así, porque entre sus características más básicas encontramos las de servicio seguro de autenticación de usuarios y un código fuertemente integrado con mínimos o inexistentes *leaks* de código.

*Django* a su vez permite alcanzar altos niveles de escalabilidad de la aplicación web gracias a su estructura piramidal, según la cual módulos como la base de datos (limitantes a priori en términos de escalabilidad) son o pueden ser independientes del esquema de navegación.

Por otro lado, *Django* es un *framework* usado a nivel mundial, con una comunidad de desarrollo amplísima y muy activa. Y es por esta misma razón, múltiples desarrolladores crean aplicaciones y librerías de código fácilmente integrables con *Django*.

## 4.2. Bootstrap

Otra de las herramientas principales de las que se ha hecho uso dentro del desarrollo de la aplicación web es *Bootstrap* [4]. *Bootstrap* es un *framework* de código libre de *Twitter* que se usa para el desarrollo de páginas web. Es limpio, intuitivo y rápido.

El *framework* trae varios elementos con estilos predefinidos fáciles de configurar: Botones, Menús desplegables, formularios incluyendo todos sus elementos e integración *jQuery* para ofrecer ventanas y *tooltips* dinámicos.

Es especialmente útil su diseño adaptativo pues se auto configura y se recoloca según el dispositivo en el que se esté visualizando el contenido, ya sea una pantalla grande o una de un *smartphone* por ejemplo. Lo cual acelera mucho el proceso de crear una versión final.

En su repositorio *GitHub* oficial podemos encontrar páginas con toda la información necesaria para utilizar Bootstrap en nuestro proyecto web, además también podemos encontrar ejemplos de utilización de este *framework*.

Su instalación e integración con *Django* es tan sencilla como simplemente descargar los ficheros que contienen las implementaciones de las funciones *CSS* y *JavaScript* de sus paquetes e incluirlos y referenciarlos dentro del árbol de carpetas de nuestra aplicación web basada en *Django*.

## 4.3. Python 3

El lenguaje de programación en el que está basado *Django* es *Python* [3]. Más concretamente la versión 1.6.5 de *Django* hace uso de *Python 3*, la nueva versión de *Python*, lanzada en Diciembre de 2008.

Su instalación es muy simple pues se trata de software libre, y si el sistema operativo no lo trae de serie como algunas distribuciones de Linux, basta con descargarse el paquete y ejecutar un instalador sencillo.

- Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional.
- Interpretado, no necesita compilación, el ordenador es capaz de ejecutar la sucesión de

instrucciones sin traducción. Esto hace que cualquier código es independiente de la máquina o del sistema operativo y por tanto se hace portable.

- Usa tipado dinámico, una misma variable puede tomar valores de distinto tipo en distintos momentos.

Cuenta con una licencia de código abierto, “Python Software Foundation License”.

Podemos encontrar amplísima documentación en la página oficial <https://www.python.org/>, tanto sobre el lenguaje, como sus funciones, librerías, uso, etc.

## 4.4. PostgreSQL

La base de datos utilizada es *PostgreSQL 9.3* [2], se trata de una base de datos de código abierto relación-objeto. Corre en la mayoría de sistemas operativos y es de fácil integración con varios lenguajes. La elección de esta base de datos se basa en su potencial de escalabilidad para proyectos futuros en entornos reales con un número de usuarios y dispositivos mucho mayor, así como la velocidad de respuesta de sus búsquedas recursivas.

Además la integración con el entorno de *Django* es tan simple como la instalación de una API extra de comunicación entre la base de datos y el demonio de Django, esta API se llama *psycopg2* de fácil instalación basada en *Python*.

Administrar la base de datos es muy sencillo, aparte de lo anterior, contamos con simples e intuitivas instrucciones propias para manejar algunas opciones básicas de la base de datos. A continuación se presentan algunas de las instrucciones de arranque y preparación de la base de datos al iniciar el escenario de pruebas.

```
sudo -u postgres psql -c ``ALTER USER postgres WITH PASSWORD '1234';``;
sudo -u postgres psql -c ``CREATE DATABASE CloudTrust WITH OWNER postgres;``;
/home/CloudTrust/webdjango/mysite/manage.py syncdb --noinput;
echo ``from django.contrib.auth.models import User;
User.objects.create_superuser('root', 'root@textit{CloudTrust}.org',
'root')`` | python /home/CloudTrust/webdjango/mysite/manage.py shell;
python /home/CloudTrust/webdjango/mysite/InitialScript.py & &
```

- **ALTER USER:** Para que *Django* pueda acceder a la base de datos de PostGres, en el fichero de configuración *setting.py*, se define el *username* y el *password* de acceso. Mediante

esta instrucción asignamos al usuario postgres una nueva *password* que corresponde con lo definido en este archivo.

- CREATE DATABASE: Una vez definido que el usuario postgres es el usuario con acceso a la base de datos, se crea una tabla dentro de esta con el nombre de *CloudTrust*.

## 4.5. Servidor web

Como mencionamos anteriormente, *Django* incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda *Apache 2* [5] con *mod\_python*. Aunque *Django* soporta la especificación *WSGI*, por lo que puede correr sobre una gran variedad de servidores como *FastCGI* o *SCGI* en *Apache* u otros servidores (particularmente *Lighttpd*). En la fase de desarrollo nos hemos limitado a hacer uso del servidor por defecto de *Django* pero como vemos es escalable a varios y distintos servicios.

## 4.6. VNX

Para la creación de los escenarios de pruebas utilizados para el desarrollo, pruebas y demostraciones del proyecto se ha utilizado la herramienta de software libre Virtual Networks over Linux (VNX [8]), desarrollada y mantenida por la Universidad Politécnica de Madrid y que ha sido mejorada en el contexto del proyecto *CloudTrust* para incluir capacidades relacionadas con las SDN.

La herramienta VNX permite crear escenarios de red virtuales formados por un conjunto de máquinas virtuales interconectadas según una topología de red definida por el usuario. El escenario se especifica mediante un documento escrito en un lenguaje basado en XML, y la herramienta se ocupa de manera transparente de crear, configurar, interconectar y gestionar las máquinas virtuales, ocultando al usuario toda la complejidad y los múltiples comandos necesarios para poner en marcha el escenario. Además, VNX permite especificar conjuntos de comandos a ejecutar en las máquinas virtuales, lo que permite no solo arrancar las mismas sino llevarlas a un determinado estado. Esta última característica es especialmente útil para la realización de pruebas o demostraciones como las creadas en el proyecto *CloudTrust*.



VNX soporta diversas tecnologías de virtualización como KVM/QEMU, LXC, Dynamips (Cisco), Olive (Juniper) y UML, así como diversos sistemas operativos en las máquinas virtuales (Ubuntu, Debian, Fedora, CentOS, FreeBSD, Windows 7 y XP e incluso Android), permitiendo la creación de escenarios de red altamente heterogéneos. En el caso del proyecto *CloudTrust* se ha utilizado casi exclusivamente la virtualización ligera basada en contenedores LXC. Las ventajas de LXC en este contexto se resumen en:

- Ligereza, que permite arrancar decenas de máquinas virtuales incluso en ordenadores de baja potencia.
- Permite una virtualización anidada, esto es, el arranque de máquinas virtuales LXC dentro de otras máquinas virtuales Linux arrancadas con VirtualBox o VMware. Este aspecto es esencial para la realización de demostraciones, ya que permite crear máquinas virtuales en formato estándar OVA con todo el software necesario ya instalado.

## 4.7. HTML

*HTML* [6], siglas de *HyperText Markup Language*, es un lenguaje de marcado para la elaboración de páginas web. Es un estándar mundialmente utilizado, y de enorme uso que pretende homogeneizar el código en la web. El *W3C (World Wide Web Consortium)* está a cargo del estándar. Se trata del estándar que se ha impuesto a la hora de visualizar páginas web y el que todos los navegadores han adoptado.

Es un lenguaje sencillo, basado en etiquetas (rodeados por corchetes angulares “<>”), que definen la forma en la que se despliega o se muestra el contenido al usuario. Se estructura típicamente en dos partes, una primera parte que es una cabecera (<head></head>), que suele contener información del documento que se presenta, y un cuerpo (<body></body>), que define el contenido del documento.

A su vez dentro de estas etiquetas o elementos del documento, encontramos atributos, que suelen ser pares nombre-valor, usado en múltiples ocasiones durante la elaboración del proyecto, como por ejemplo el atributo “class” que es fundamental para la integración con *Bootstrap*, de tal forma que se puedan identificar cuáles son los estilos a aplicar a cada elemento de la página web.

## 4.8. CSS

*CSS* [6] o *Cascading Style Sheets*, se trata de un lenguaje que define la presentación de un documento escrito en *HTML*. Al igual que la codificación *HTML*, el *W3C* es el encargado de elaborar el estándar que finalmente han optado por seguir todos los navegadores. La sintaxis de la que hace uso *CSS* es muy sencilla, se basa en el uso de palabras clave, según propiedad: valor.

La idea fundamental de *CSS* reside en separar la estructura del documento de la vista del mismo. La información de presentación se puede encontrar en el mismo fichero *HTML* o en un documento separado. En este PFC se ha hecho uso de muchos de los estilos de *Bootstrap*, que contienen archivos *CSS* fácilmente configurables, y además de la etiqueta `<style>`, para colocar algunos elementos de la página web, y estructurar la vista de forma más ordenada.

## 4.9. Open vSwitch

*Open vSwitch* [9]: Se trata de un *Switch* implementado por software que dispone de compatibilidad para integrarse con diversos protocolos de control y monitorización como *Openflow*, *Netflow*, *Sflow*, etc.

## 4.10. Frenetic

La arquitectura de control SDN propuesta en el proyecto Frenetic [10] se compone de tres niveles.

- **POX**: Este es el nivel de abstracción más bajo de la arquitectura. Se encarga de actuar como interfaz para el protocolo *Openflow* en ambas direcciones. Ofrece una API para el empleo del protocolo *Openflow* en lenguaje Python. Al disponer de un nivel de abstracción bajo, las aplicaciones que pueden desarrollarse no disponen de un alto grado de complejidad, siendo necesario el empleo de un nivel superior.
- **Pyretic**: Este nivel emplea la API expuesta por POX, y ofrece niveles de abstracción mucho mayores, como la composición de reglas o la monitorización. Este nivel contendrá las aplicaciones desarrolladas que no requieran de interacción con el servidor web, como

el switch, el histórico de consumo, el descubridor de dispositivos, log de la red o medidor de la actividad de un dispositivo.

- Pyresonance: Este es un nivel de abstracción que se superpone con Pyretic y que permite la interacción del controlador con elementos externos. Habilitando la recepción de mensajes que modifiquen el comportamiento del controlador, provenientes del servidor web. Este nivel contendrá la aplicación de denegación de servicio.



Figura 4.4: Arquitectura Frenetic

## Capítulo 5

# Diseño e Implementación

La solución *CloudTrust* ofrecerá una estructura de red en varios niveles sustancialmente distinta de la convencional, los papeles que realizarán suministradores de operadoras, las mismas operadoras o los clientes serán distintos. Pero es precisamente esta diversidad lo que permite explotar y explorar nuevas fronteras de las redes de comunicación.

Plantaremos varios escenarios posibles en los que *CloudTrust* tiene cabida, por ejemplo, podremos implantar *CloudTrust* en los hogares, podremos implantar *CloudTrust* en grandes o pequeñas empresas, o también podremos implantarla en recintos públicos como Ayuntamientos o Aeropuertos.

Por ello es de sumo interés realizar una descripción detallada, y realizar un análisis completo de la red para el usuario final.

Para que el usuario final, ya sea un cliente en su hogar, o un cliente en su lugar de trabajo, le sea transparente el cambio a *CloudTrust*, y únicamente sea capaz de ver las ventajas de acogerse a esta solución de una forma simple e intuitiva, se llevará a cabo la creación de un portal web de configuración de la red final (entendida por red final, la red personal de hogar, o del establecimiento público o privado), al cual tendrá acceso, para poder llevar a cabo las acciones posibles (descritas en apartados anteriores) que considere oportunas.

Así por un lado, el usuario que utilice *CloudTrust*, se dará cuenta rápidamente de que tiene acceso a un portal web en el que podrá configurar fácilmente sus dispositivos, y acceder a información de una forma mucho más nítida y simple sobre cómo interactúan estos dispositivos con la Red.

Entendemos que es totalmente necesario que bajo todo el cambio estructural de red que

se plantea bajo la solución *CloudTrust*, además, se dote al cliente final de esta aplicación web mediante la cual, se traduzcan las ventajas de usar *CloudTrust* en algo visual, fácilmente configurable y comprensible.

A continuación presentaremos una descripción de la arquitectura de la solución *CloudTrust*, y además lo haremos, dividido en tres planos distintos pues la escala del proyecto lo exige. Estos tres planos son un diagrama de red, un diagrama físico y un diagrama funcional.

## 5.1. Diagrama Funcional

En el siguiente diagrama vemos la organización funcional de los distintos bloques de la solución.

Desde el punto de vista de la red final encontramos el bloque identificado como Configurador, que hace referencia al servidor que trabaja bajo *Django* y que hostea la aplicación web de configuración de la red del usuario final.

Dentro de la aplicación web encontramos algunas de sus principales funcionalidades como el Portal de inicio, que actuará de *captive-portal*, el interfaz vCPE pues la aplicación hará de intermediario entre la red y el dispositivo CPE, la función de histórico de consumo, la gestión de usuarios, controles parentales, descubrimiento de dispositivos o filtrado MAC entre otras.

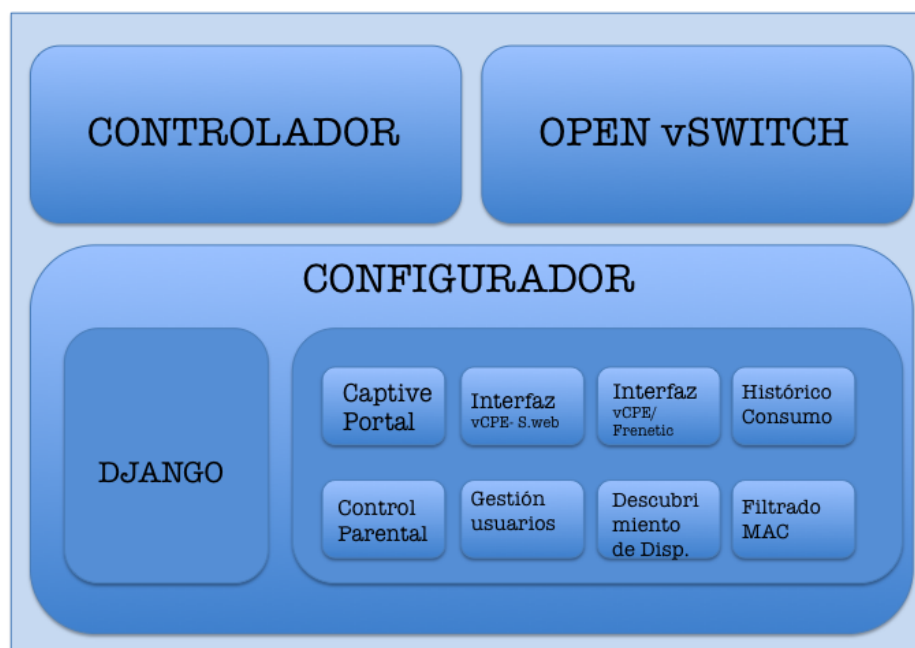


Figura 5.1: Diagrama Funcional

## 5.2. Diagrama Físico

En el siguiente diagrama vemos representados los elementos hardware básicos mínimos de la solución *CloudTrust*. Con la información del anterior diagrama, ya podíamos intuir muchos de los componentes que describiremos a continuación.

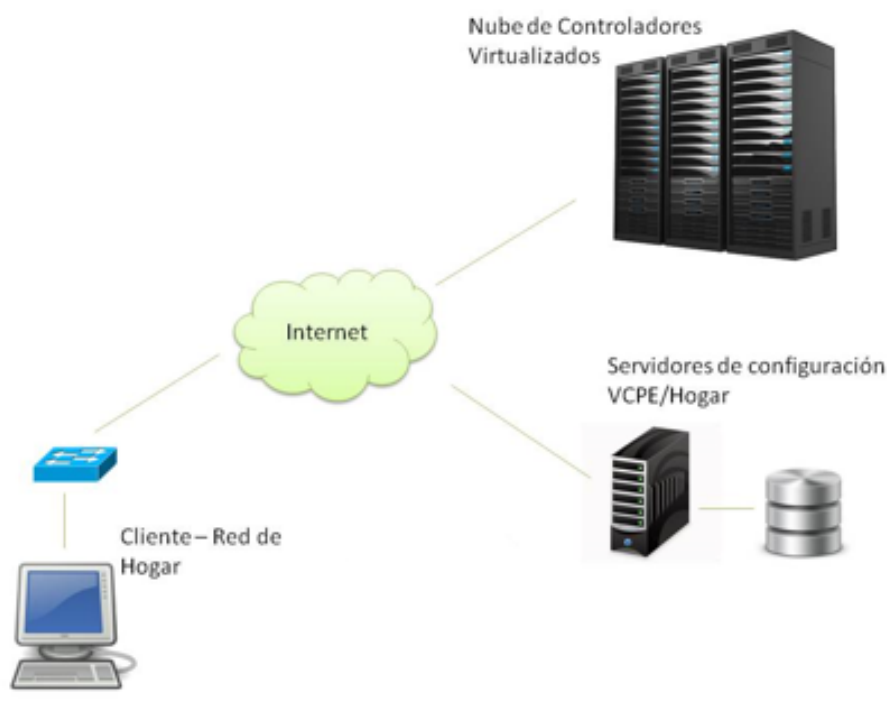


Figura 5.2: Diagrama Físico

En este apartado describiremos los requisitos necesarios para llevar a cabo la implementación del Servidor de configuración web. Necesitaremos realizar una previsión de recursos dependiente de dos parámetros. En nuestro servicio implementaremos un sistema seguro de sesión que verifique la identidad de los usuarios finales, añadiendo así una capa más de seguridad. Por tanto, la primera variable que deberemos manejar en términos de escalabilidad del servidor será el número de sesiones esperado que deberemos soportar. Es decir, dependerá enteramente del número de usuarios que manejemos.

Por otro lado, haremos uso del número de peticiones como segundo indicador de escalabilidad, es decir, el número de clicks que un usuario realice durante una sesión.

Las características del servidor básico a utilizar en un despliegue aproximado de 10.000 usuarios activos serían las siguientes:

- Servidor: DL380p Gen8 - 2 \* Xeon E5-2620 6-core 2.0GHz, 32 GB RAM (2\*16GB LV RDIMMs), 2\*300GB-10K disks, 750W redundant AC power, 4 \* 1GbE embedded LAN ports
- Licencia iLo Advanced: Insight Lights-Out Advanced Pack - Electronic licence software and licence-to-use for 1 rackmount or tower server - includes 1 year 24\*7 SW support and updates
- Additional memory module - 16 GB (1 \* 16GB dual-rank) PC3 12800 ECC Registered DIMM, factory integrated

En aras de aportar mayor seguridad al sistema, se le aplicaría una política de n+1 equipos de servidores para asegurar redundancia en caso de pérdida de energía o similares.

Como segundo elemento hardware importante requeriremos de una nube de almacenamiento para la base de datos de los nombres de usuario y sus sesiones, así como de la información de sus dispositivos.



### 5.3. Diagrama de Red

En el siguiente diagrama encontramos resumidas las diferentes conexiones de red que encontramos dentro de la solución. Fijémonos en el cliente 1 por ejemplo, una red de hogar.

El cambio principal reside en el cambio del router ubicado en el hogar (o empresa o localización elegida) por un OpenVSwitch, es decir, se trata de un cambio hardware mínimamente invasivo.

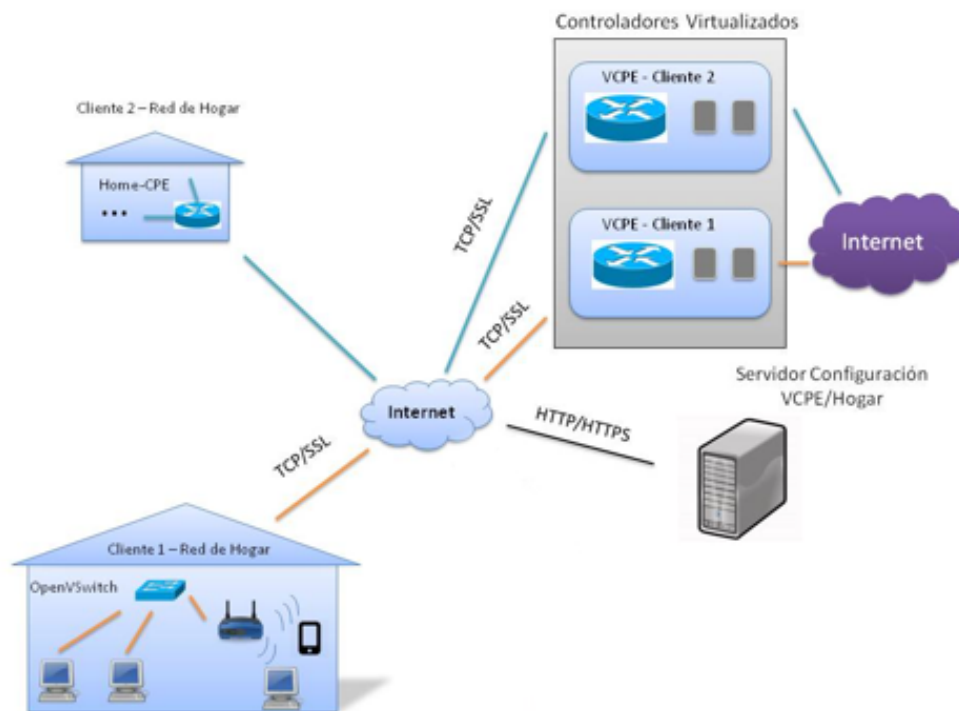


Figura 5.3: Diagrama de Red

Además de la red del hogar, en la que se cambia un router por un OpenVSwitch, el primero de los componentes de *CloudTrust* que encontramos son los Controladores Virtualizados o vCPE's. Al extraer el router de la fase final de la red, necesitamos introducir la figura del controlador que haga las veces de controlador de red, pero que lo haga de forma virtualizada y externa a la red final bajo los protocolos *TCP/SSL*.

Este controlador tendrá las funciones avanzadas de un router, de tal manera que se minimizan las capacidades funcionales que tiene el hardware en casa del cliente, y por tanto nos lleva a procesos de integración de nuevos items y mantenimiento mucho más baratos.

El segundo elemento que encontramos es precisamente el Servidor del portal de configura-

ción que hemos comentado anteriormente en este mismo apartado. El acceso a este portal estará hospedado en este servidor aquí representado y se realizará por acceso *HTTP/HTTPS*.

## 5.4. Estructura y definición del entorno de desarrollo

El entorno que se presenta se trata de una máquina Linux modificada según las necesidades de la solución. Esta máquina virtual contiene todo lo necesario para crear un escenario potencial real de una red de hogar con todos los dispositivos anteriormente definidos.

En la figura que se presenta a continuación, se pueden identificar fácilmente las máquinas virtualizadas que se crean una vez arrancado el escenario. De esta forma se crean máquinas que simulan dispositivos de red como ordenadores personales, así como, el Controlador, el servidor DHCP, el servidor web y el router hacen las veces de las máquinas necesarias para llevar a cabo las funciones de conectividad de la red, según lo especificado en anteriores apartados.

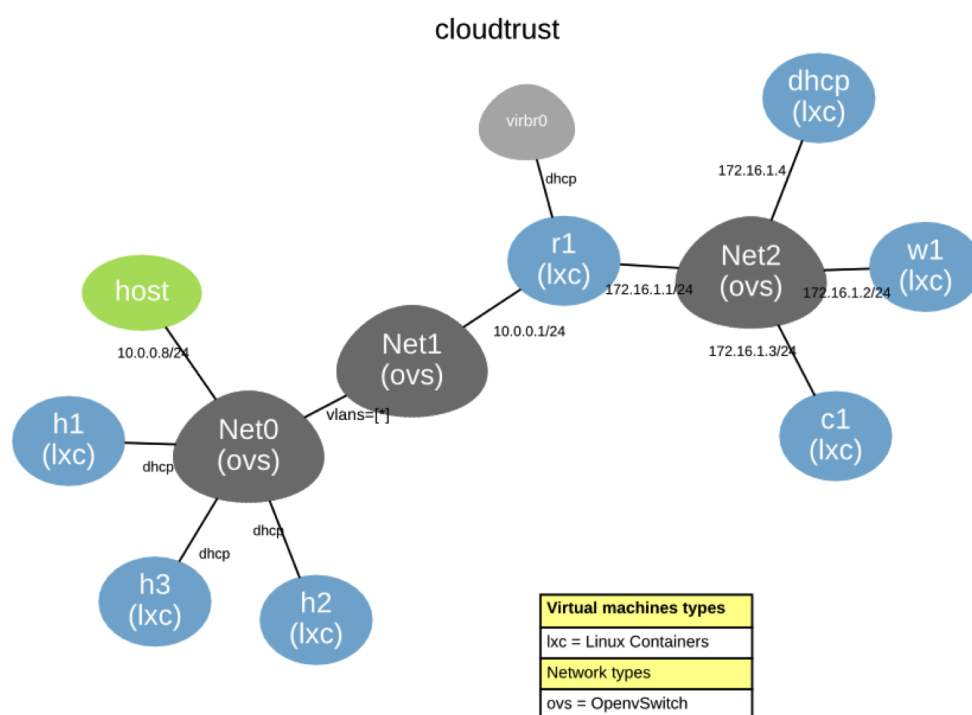


Figura 5.4: Arquitectura de Red

Para arrancar este entorno necesitamos el fichero en formato .ova y ejecutarlo en un programa que soporte este tipo de formato, en nuestro caso el programa que hemos utilizado en fase de desarrollo es *Oracle VM VirtualBox*.

Una vez arrancamos la máquina virtual se nos presenta un escritorio *Linux*. Para arrancar el escenario debemos iniciar una ventana de terminal, y tras ello acceder como súper usuario con el comando “sudo su”, y ejecutar los siguientes comandos: (con la opción “-t” al final, se arranca, con la opción “-P” se cierra el escenario):

```
vnx -f CloudTrust-v03.xml -v -t
vnx -f CloudTrust-v03.xml -v -x start-srv
vnx -f CloudTrust-v03.xml -x start-parental
vnx -f CloudTrust-v03.xml -v -start -M h1
```

Con estos comandos lo que conseguimos es arrancar efectivamente el escenario de pruebas, así, aparecerán sucesivamente una serie de terminales, directamente en los dispositivos del entorno. Cada una de estas máquinas son accesibles con las credenciales “root” y contraseña “xxxx”.

Con el escenario bajo estas condiciones las máquinas que simulan las de usuarios de la red carecen de conectividad, pues queremos dársela únicamente bajo los criterios que consideremos oportunos.

La segunda línea de código arranca el servidor de *Django*, la tercera línea, arranca un demonio de *Linux* que ejecuta un código periódicamente (cada 30 segundos), de tal forma que se verifica según la hora actual si un dispositivo tiene acceso a la red o no, la aplicación más directa de esta opción, es ofrecer controles parentales. La última línea de código arranca un dispositivo cualquiera “h1” en el entorno, y por tanto simula que un nuevo elemento aparece en la red.

## 5.5. Estructura y definición de la aplicación web de configuración

En este apartado se hará una descripción detallada de las características, formato y funcionalidades de las que dispone la aplicación web de configuración.

Una vez contamos con el escenario arrancado, tenemos enfrente una serie de terminales que simulan a los distintos dispositivos que nos podemos encontrar en una red de por ejemplo de hogar, con los que podemos interactuar a través de la aplicación web.

A partir de los requisitos y especificaciones definidos en el capítulo 3, y a partir de las consideraciones que introduciremos a continuación conformamos una aplicación web intuitiva

## 5.5. ESTRUCTURA Y DEFINICIÓN DE LA APLICACIÓN WEB DE CONFIGURACIÓN<sup>31</sup>

y sencilla para el usuario.

Para poder configurar nuestra red, contamos con el acceso al portal de configuración, hospedado en la máquina virtual ServidorWeb bajo la IP 172.16.1.2 y puerto 8000. Por tanto, si abrimos el navegador de nuestra máquina Linux y accedemos desde el campo de URL's a *http://172.16.1.2:8000/* (como vemos en la Figura 5.4 esta es la IP que corresponde al servidor web), se desplegará la página de inicio de la aplicación web de configuración de *CloudTrust*.

Una de las primeras consideraciones que tratamos de cubrir en el diseño de la aplicación es, la de la implantación de una página de índice desde la cual el usuario pueda acceder según sus credenciales a los servicios del sistema. En esta primera página le daremos la bienvenida al usuario y encontraremos el logo de la solución *CloudTrust* en el centro. En la parte superior se situará el menú de acceso a la aplicación, y desde la parte inferior de la página se podrán acceder a las páginas de soporte o redes sociales del proyecto.

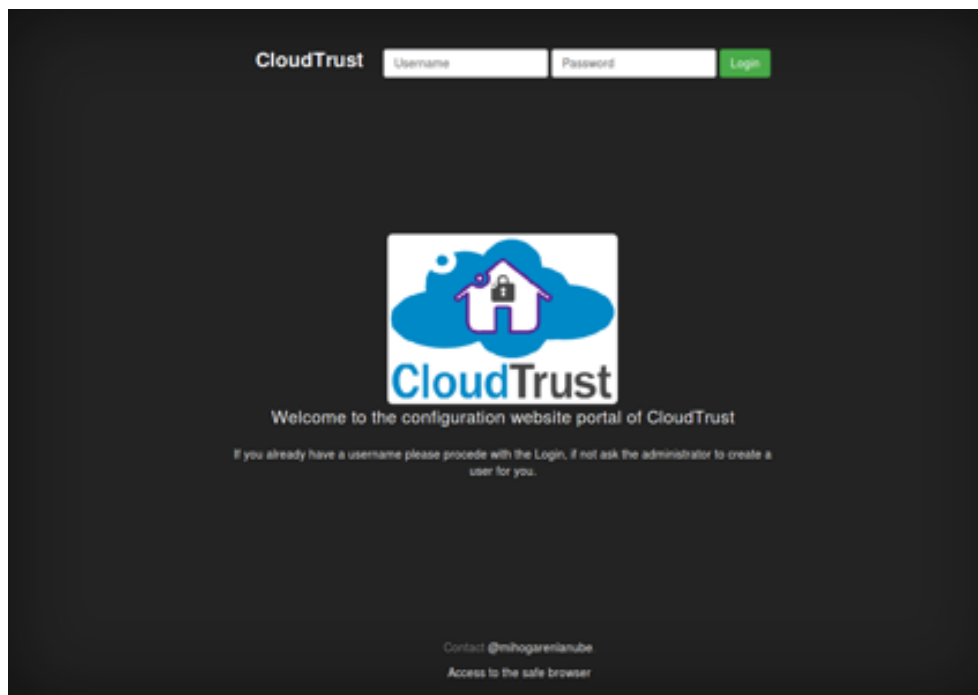


Figura 5.5: Index WebAPP

Una vez accedemos con las credenciales de administrador, en nuestro caso, el usuario administrador se llama “root/root”. Lo primero que recibimos es la bienvenida al portal de configuración y algunas indicaciones de los primeros pasos que debemos llevar a cabo.

Se dividirá en tres bloques principales, el primero de ellos será un banner en el que se

colocará a la izquierda un acceso directo a la página principal, y a la derecha información sobre las credenciales del usuario y un botón de logout.

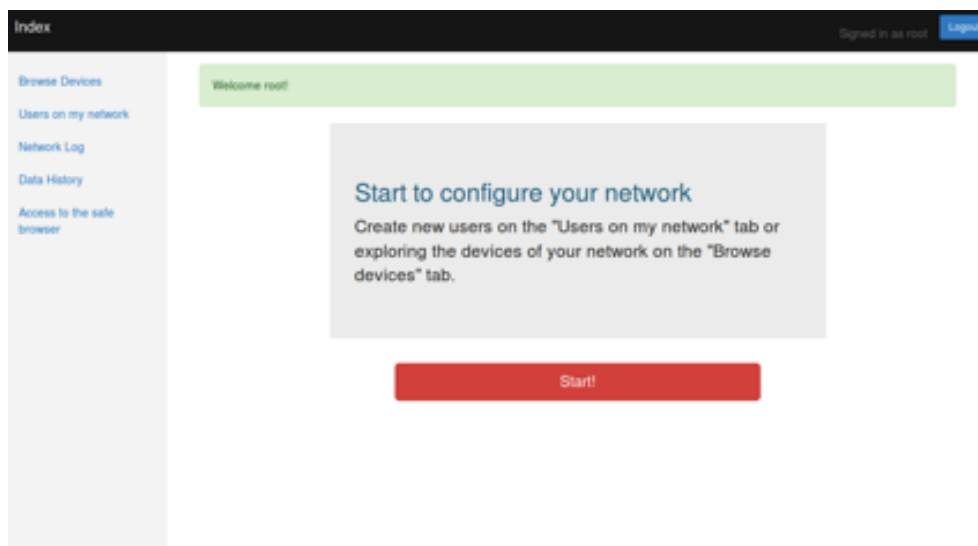


Figura 5.6: Página principal

El segundo bloque principal ocupará el lateral izquierdo de la pantalla y será un menú de acceso a las distintas funciones de la aplicación, este menú será estático y será permanentemente accesible por el usuario. Estas serán, “Browse Devices” (Acceso a la configuración de dispositivos), “Users on my network” (Acceso a la gestión de los usuarios de la red), “Network Log” (Acceso a la información de Log de sistema) y por último, “Data history” (Acceso al histórico de consumo).

El tercer bloque principal ocupará el resto de la página y es dónde se colocará la información de cada uno de los menús a los que se accede desde el segundo bloque. Así si por ejemplo el usuario accediera a “Browse devives”, en este bloque encontraríamos el listado de los dispositivos conocidos y desconocidos de la red del usuario. De la misma forma si clickase en la opción “configurar usuario” se desplegará un listado de todos los usuarios de la Red, con su consecuente información, y opciones de configuración.

Por último, desde el punto de vista de los componentes de la aplicación, se establece un modelo vista-control-datos.

## 5.6. Estructura de la Base de Datos

A continuación se presenta la definición de la estructura de la base de datos del proyecto. En la siguiente imagen podemos ver que existe una relación directa y en cascada entre varios elementos, y alternativamente se hace uso de otra pareja de tablas anexas.

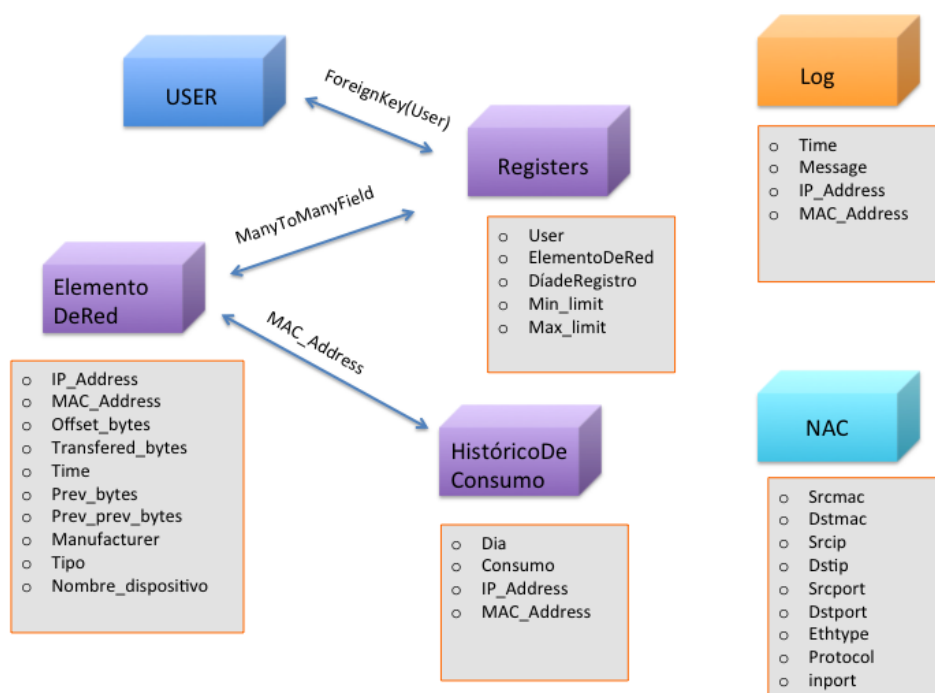


Figura 5.7: Diagrama de la Base de Datos.

Pasemos a describir cada uno de estos elementos. El primero de estos corresponde a la tabla “USER”. Nuevamente hacemos uso de una de las funciones nativas incorporadas en Django. La gestión de usuarios es muy sencilla con el paquete `UserAuthentication`[13] de Django, este genera una tabla de usuarios con su pareja de *username/password* e información sobre su sesión, como el día de registro o el último *login*.

Linkado con la tabla “USER”, encontramos la tabla “Registers”, en ella vamos a contener la relación que existe con los usuarios y sus elementos de red, o dispositivos de red, así como su información. Es decir, para cada usuario dentro de la red, se creará un elemento nuevo dentro de la tabla Registers, que recoge la información de usuario, sus dispositivos y en este caso también se guardan los campos límites inferior y superior del control parental. Como vemos la conexión entre ambas tablas se establece bajo una herramienta típica en el uso de bases de

datos, que se denomina *ForeignKey*. Lo que implica que depende de un elemento externo como clave ("llave"), en nuestro caso User.

Si seguimos la estructura definida en la figura, la siguiente tabla que encontramos es "ElementoDeRed". Se trata de una tabla que contendrá por cada dispositivo de red un registro distinto. Cuando el controlador detecte un dispositivo nuevo, creará este registro y almacenará la información relativa al dispositivo a nivel de red como su dirección IP o su dirección MAC. Por otro lado cada vez que se haga uso de la red, se mantendrá un registro de los *bytes* transferidos en el último acceso en *Transfereed bytes*. Por último, se almacenará también datos como el fabricante (*manufacturer*), o el tipo de dispositivo o el nombre del dispositivo, que son variables configurables por el usuario.

La relación que existe entre las dos tablas anteriores se define según una llave denominada *ManyToManyField*, de esta forma, un "ElementoDeRed" puede pertenecer a un "Register", y un "Register" puede tener múltiples registros de "ElementoDeRed".

La última tabla enlazada que encontramos se trata de la tabla "HistóricoDeConsumo". En ella se mantendrá un registro del gasto de un dispositivo identificado por su dirección MAC (linkado con la tabla "ElementoDeRed), a través de la suma de *bytes* consumidos en un día.

Se hace además uso de dos tablas más, la tabla de "Log", y la tabla "NAC". La primera se usa para llevar un registro de los eventos que han ocurrido en la red. Se almacena el momento temporal del evento, un mensaje descriptivo del suceso, y las direcciones IP y MAC del dispositivo involucrado.

La segunda tabla, "NAC" (*Network Control Access*) , almacena la información referente al control de acceso a la red.

## 5.7. Estructura de las carpetas de la aplicación

Como se ha comentado anteriormente, *Django* requiere de una estructura de contenidos peculiar y precisa. Por lo que con esta imagen se quiere mostrar y explicar el porqué del árbol de carpetas que se presenta en la solución *CloudTrust*.

El primer comentario importante que se debe hacer al respecto, sería la distinción entre lo referente a los archivos estáticos y los archivos referentes a la aplicación web o pertenecientes a *Django*, así como las API's que se han desarrollado para crear el entorno. Así, todo lo contenido bajo la carpeta *static*, será contenido estático, es decir imágenes, plantillas, iconos, etc. de los que hacen uso la aplicación.

Por otro lado tenemos el contenido referente a las funcionalidades propias de la aplicación. Podemos identificarlo como aquel contenido bajo la carpeta *mysite*. Dentro de esta misma carpeta podemos encontrar varias subcarpetas cada una de ellas con su propio sentido de uso.

Bajo la carpeta “*mysite*” encontramos algunos ficheros que no se pueden englobar en las categorías anteriormente comentadas. Debemos mencionar tres ficheros que son “*InitialScript.py*”, “*floodlightInitialScript.py*” y “*manage.py*”.

1. El primero de estos ficheros, *InitialScript.py*, contiene una serie de instrucciones que ejecutan y arrancan las funcionalidades principales del escenario como son el Controlador y el ServidorWeb de Django.
2. El segundo de los ficheros, *floodlightInitialScript.py*, corresponde al arranque de los servicios requeridos por el Controlador *Floodlight*.
3. El tercero de los archivos, *manage.py*, es un fichero básico dentro de la estructura de Django. Gracias a este fichero pode-

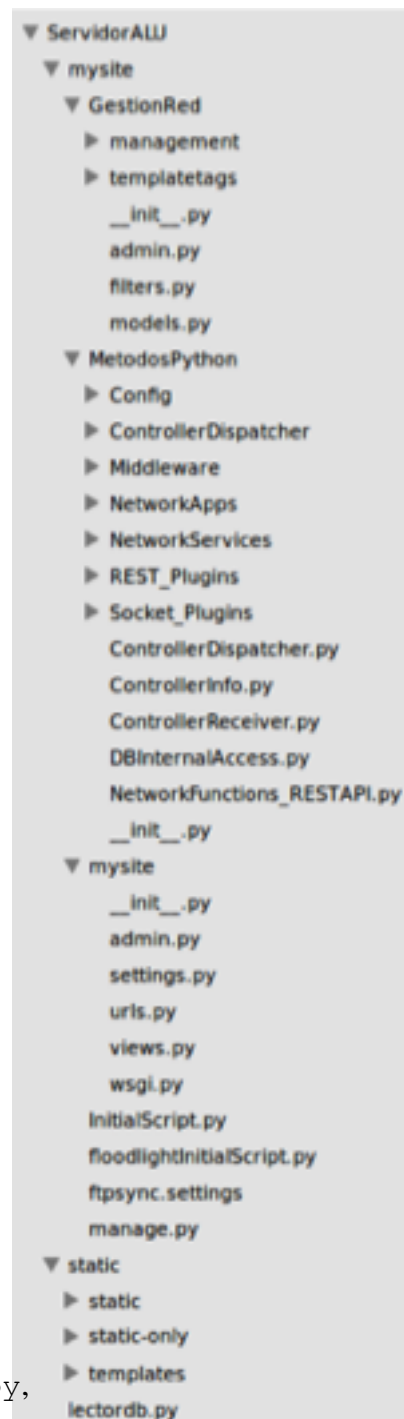


Figura 5.8: Estructura de carpetas.



mos ejecutar múltiples comandos preestablecidos que el *framework* de *Django* establece, como *syncdb*, para sincronizar la base de datos, o *collectstatic* que sirve para recolectar todos aquellos archivos estáticos en un único punto, y facilitar el servir estos ficheros en producción final.

Por orden, la carpeta *GestionRed* contiene las utilidades que hemos creado o que *Django* provee para que nuestro “site” funcione. De esta forma, por ejemplo, el fichero `admin.py` contiene el código Python referente a una herramienta que se puede integrar dentro de proyectos de *Django*, a la que se puede acceder a través del navegador invocando tras la petición la palabra clave `admin`, `HTTP://host:port/admin`. Esto nos permitirá acceder a los contenidos de la base de datos, muy útil a la hora del desarrollo de la aplicación. Se trata de un servicio no extensible a la solución *CloudTrust*, y por tanto se limita a la fase de *developing*.

Bajo la carpeta *GestionRed* encontramos otro fichero muy importante, y este es `models.py`, en él encontraremos en código Python, bajo la nomenclatura específica de *Django*, la estructura deseada de la base de datos, sus tablas y el contenido de cada una de ellas. *Django* provee de un método de sincronización con la base de datos, de tal forma que, las consultas que se realizan a la base de datos desde la aplicación web, son transparentes, pues únicamente recurrimos a los métodos específicos que nos provee el framework de *Django* para trabajar. Dentro de esta misma carpeta, *GestionRed*, encontramos dos subcarpetas más que identificamos como “management” y como “templatetags”. La primera de estas carpetas, contiene a su vez otra carpeta “commands”.

Gracias a *Django*, podemos situar dentro de esta carpeta módulos Python ejecutables por línea de comandos, para crear nuestros propios comandos. En el desarrollo de nuestra aplicación entre las múltiples funciones de configuración que ofrecemos al usuario, se encuentra el control parental, es decir, requeriremos de un script que se ejecute periódicamente y que compruebe la hora actual, y otorgue o deniegue la conectividad a los dispositivos de un usuario dependiendo de la franja horaria en la que se situé. Y es precisamente bajo esta carpeta donde situamos este script “parentalControl.py”. La segunda carpeta, “templatetags”, contiene un archivo llamado “GestionRedtags.py”. Según *Django* los archivos que situemos bajo esta carpeta, contendrán funciones, que pueden llamar las plantillas HTML, de tal forma que podemos devolverle al renderizador del navegador la información requerida, tratada en Python, eliminando complejidad de procesamiento del navegador, y llevando la inteligencia software fuera de las plantillas HTML.

Se trata de una herramienta muy poderosa de Django, que facilita mucho las labores de codificación.

La segunda carpeta que encontramos, se llama `MetodosPython`, aquí se encuentran desarrolladas todas las API's o secciones de configuración, de las que hace uso el controlador, que interactúan en este caso con las funciones de red. A efectos de este proyecto, muchas de estas interfaces de programación realizan acciones de consulta y escritura en la base de datos. Por ello se decidió crear una API específica de tratamiento de la base de datos, denominada `DBInternalAccess.py`. Aquí se desarrollan todos los métodos python que cualquier elemento de la red pueden invocar para realizar acciones con la base de datos.

La siguiente carpeta a comentar es “mysite” bajo la carpeta “mysite”, esta carpeta, contiene lo que podríamos denominar como la aplicación web en sí. Aquí definiremos el comportamiento de la aplicación web. Encontramos bajo esta carpeta tres ficheros de gran importancia, “settings.py”, “urls.py” y “views.py”.

1. El primero de estos archivos, “settings.py”, como los dos restantes son ficheros específicos que Django crea según se arranca un nuevo proyecto. La función principal de “settings.py” es la definición de variables globales del proyecto, como pueden ser la IP y el puerto de nuestro servidor, la información de configuración de la base de datos, la zona horaria, el código de lenguaje, la localización de dónde se están sirviendo los archivos estáticos o la inclusión de APP externas de las que queramos hacer uso.
2. El segundo de los archivos previamente mencionados, “urls.py”, se trata de un fichero bajo el cual Django establece el patrón de navegación de la aplicación de configuración web. Es decir, realiza la conexión entre cada solicitud HTTP a una URL específica y el método Python que lo ejecuta. Por ejemplo, podemos encontrar dentro de fichero esta línea de código:

```
url(r'^Index/$',          'mysite.views.Index'),
```

Mediante el uso de expresiones generales, podemos definir bajo que formato de peticiones queremos responder y con qué método queremos hacerlo. En este caso, esta línea de código específica que para peticiones a nuestro servidor y puerto, seguidas de “/Index” respondaremos con la función Python definida en `mysite/views.py` bajo el nombre de `Index`. Así y bajo

esta misma filosofía construiremos la estructura de navegación completa de la aplicación de configuración.

El tercero de estos archivos, muy relacionado con el segundo, es “views.py”. En él colocaremos todas las funciones que responderán a las peticiones tipo GET y POST que realicen los navegadores de los usuarios, para cada URL especificada y para cada usuario.

La siguiente imagen que vamos a comentar es la estructura de archivos estáticos del entorno, así como la estructura de plantillas o templates en las que se basa la navegación de la página web.

Bajo la carpeta static, todos los archivos con formato “.png” son imágenes. Bajo las carpeta js, encontramos todas las funciones JavaScript y bajo la carpeta CSS los archivos que definen los estilos de los elementos HTML.

Siguiendo la estructura típica de las páginas HTML, la página de inicio la denominamos Index.html, en esta se despliega el formulario de autenticación, y según seamos administrador de la red o usuario, nos redirige a Logged.html para el primero y Logged\_user.html para el segundo. Desde esta página principal podemos acceder a todas las funcionalidades desde el menú de la izquierda como hemos comentado antes, podemos acceder a la vista de dispositivos según seamos administrador (devices\_superuser.html) o usuario (Devices\_user.html).

Si somos administradores desde el Index seremos capaces de acceder a gestionar los usuarios de la red. Para ello el template a renderizar es el de Manage\_users.html Sin embargo si queremos visualizar el histórico de consumo, si somos administradores podremos ver el de todos los dispositivos de la red, desde Data\_superuser.html, o si somos usuarios el de nuestros dispositivos desde Data\_user.html. Por último podemos consultar el fichero de eventos desde Log.html.

# Capítulo 6

## Resultados

En este apartado pasamos a describir algunos escenarios reales con los que nos podemos encontrar en un entorno de red doméstico, y cómo nuestra aplicación interactúa con ellos. Se llevarán a cabo pruebas empíricas que confirmen y corroboren el funcionamiento de los distintos Requisitos y Especificaciones planteados en el apartado 3.

### 6.1. Caso de estudio 1: Primer acceso

Una vez hemos arrancado el escenario virtualizado con VNX, y tengamos funcionando todas las máquinas virtuales que simulan una red (router, servidor DHCP, Controlador y Servidor web), el propio entorno de pruebas nos abre un navegador predeterminado (Firefox) y nos carga la URL donde se sirve la aplicación web de Django, previamente configurado y arrancado.

La siguiente imagen, es una captura de pantalla de la página de Index que un usuario de *CloudTrust* vería en cualquier dispositivo con acceso a la Red si tratara de acceder por primera vez. Este portal hace las veces, de página de bienvenida y de captive portal, pues si el dispositivo del usuario no está reconocido dentro de la red, se le pedirán unas credenciales de acceso.

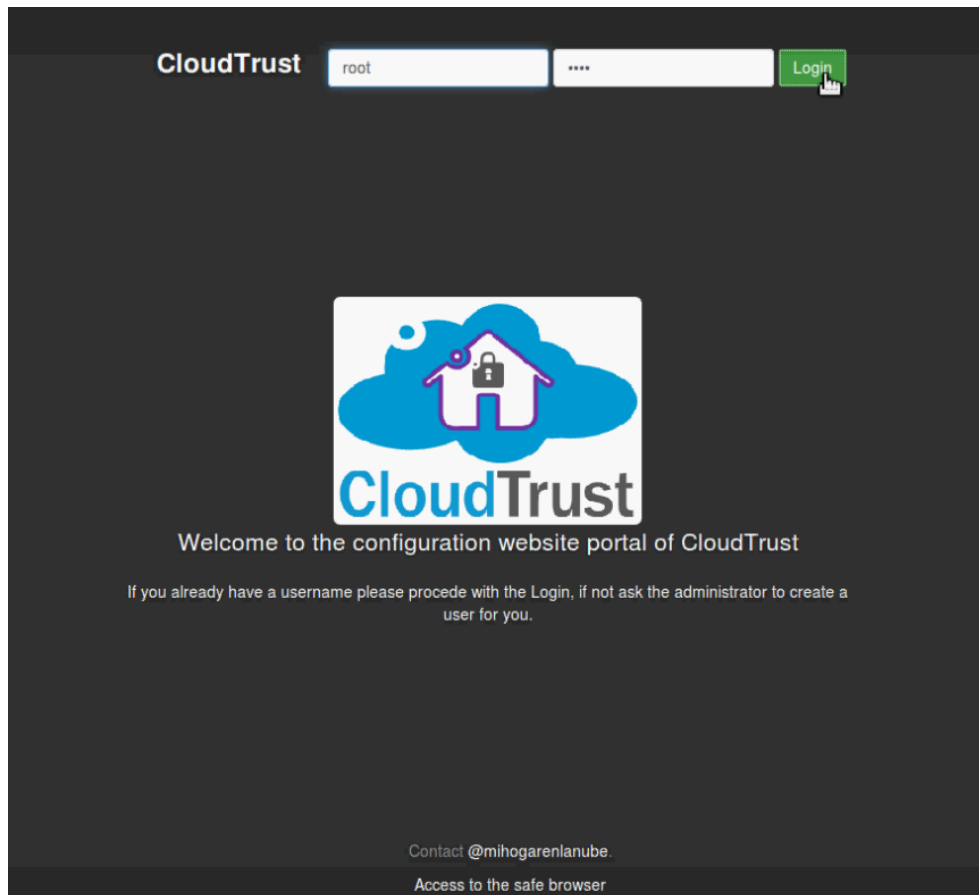


Figura 6.1: Index: Acceso root

La solución *CloudTrust*, depende de la existencia de un administrador único de red, que tenga acceso a la gestión de su red personal, según lo definido en el apartado 3. Estas credenciales pueden ser fácilmente definibles gracias a la estructura de Django y a su funcionalidad “admin”. Es importante no confundir al administrador de la red con los distintos tipos de perfiles que se han definido (User, Guest, Not Validated).

Así pues, una vez hayamos accedido al portal de configuración por primera vez, la propia aplicación web, nos guiará entre las principales funcionalidades del entorno:

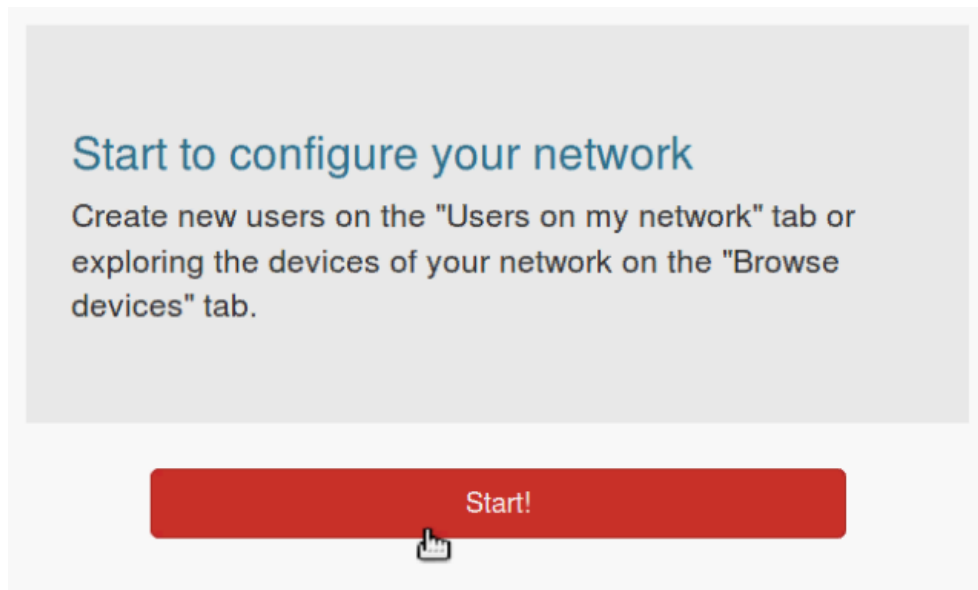


Figura 6.2: Primer acceso: Guía de navegación, bienvenida

Como vemos la aplicación web nos invita a comenzar a configurar los usuarios de nuestra red, o bien a verificar hemos configurado en nuestra red. Clickamos en el botón de "Start!", y nos redirige al panel de vista de dispositivos:

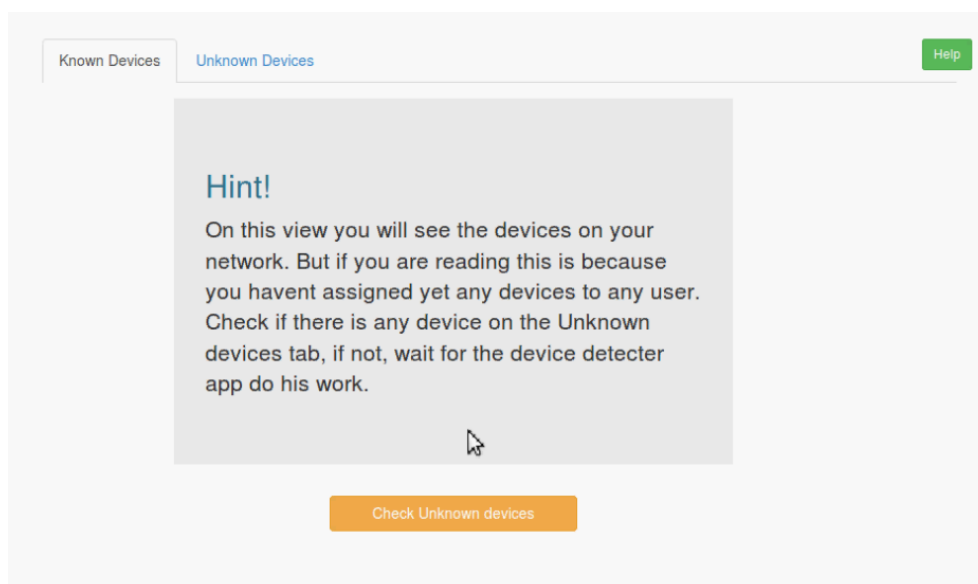


Figura 6.3: Primer acceso: Guía de navegación, vista de dispositivos conocidos

La aplicación web, nos guía entorno al uso de la esta vista. Como se describe lo primero que debemos notar es que existen dos vistas ("tabs") distintas, una para los dispositivos conocidos

("Known devices") y otra para los dispositivos desconocidos ("Unknown devices"). Como nos indica, al ser la primera vez que accedemos al portal no tenemos ningún dispositivo vinculado a ningún usuario.

Continuamos haciendo click en el botón de "Check Unknown devices":

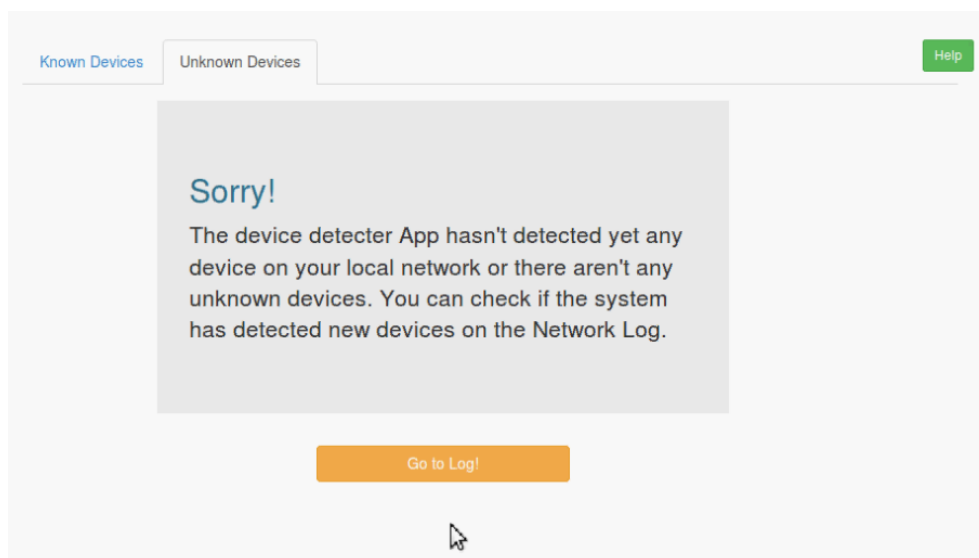


Figura 6.4: Primer acceso: Guía de navegación, vista de dispositivos desconocidos

En esta vista se nos muestra un extracto de texto, que nos comunica que en la red no ha aparecido ningún dispositivo de red todavía (esto es así, porque todavía no hemos arrancado ningún dispositivo dentro del escenario, dado que estamos en un entorno que hace uso de DHCP con el primer mensaje de discover IP ya se detectará el dispositivo). A pesar de que el escenario nos plantea continuar la navegación explorando la pestaña de Log, pasamos a la pestaña de "Users on my network", y obtenemos la siguiente presentación:

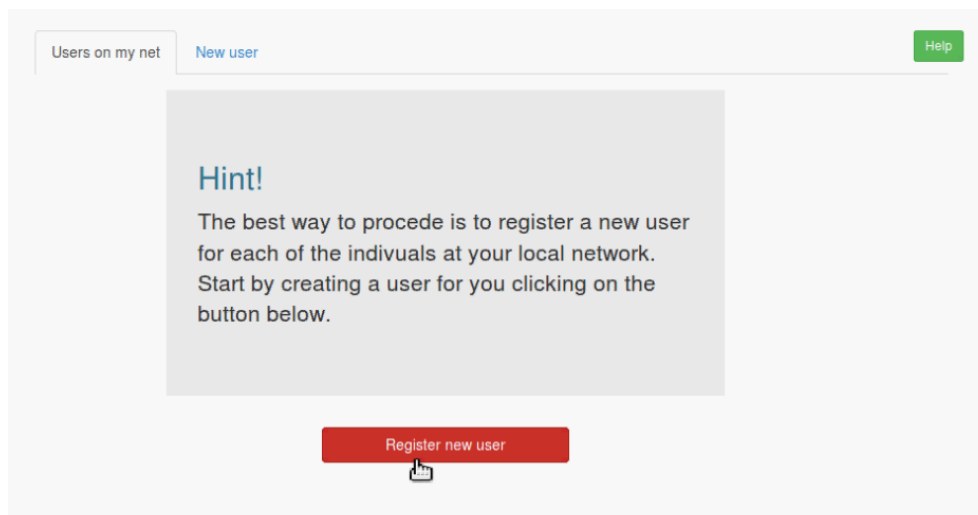


Figura 6.5: Primer acceso: Guía de navegación, vista de usuarios

Como vemos, nuevamente la aplicación nos guía y nos hace saber que la mejor forma de continuar, es proceder a crear un usuario en nuestra red, así pues, tendremos que configurar una única vez perfiles de usuario para cada uno de, por ejemplo, los integrantes de nuestra unidad familiar.

Procedemos a crear un primer usuario, en este caso, el usuario “David” con su pareja de “username/password”:

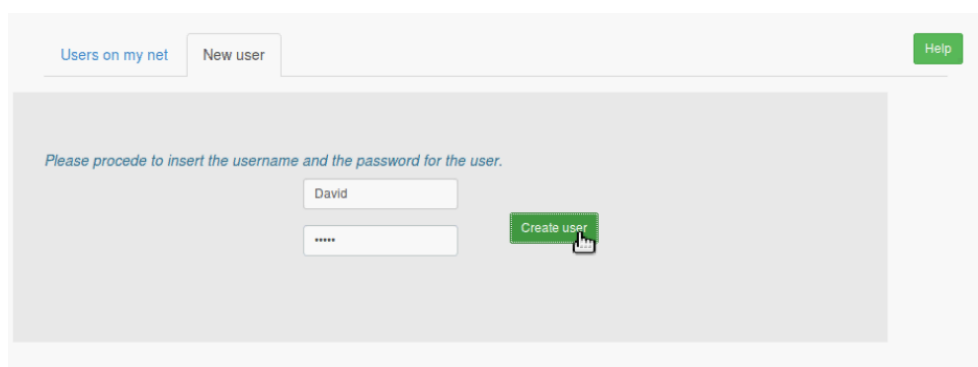


Figura 6.6: Primer acceso: Guía de navegación, registro de usuarios

Tras una página de confirmación de que nuestro usuario se ha creado correctamente, nos redirige automáticamente la vista de usuarios:



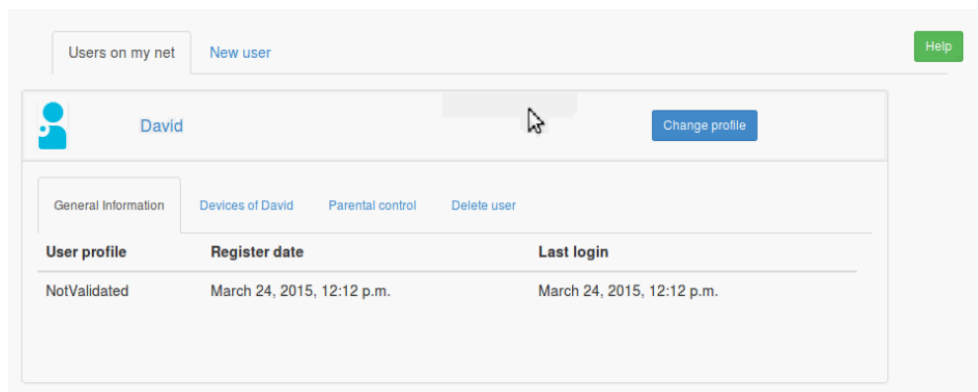


Figura 6.7: Primer acceso: Vista de usuario David

La vista que se nos presenta es un desplegable del usuario que acabamos de crear, David, con su información más relevante desplegada. Así como aquellas pestañas que permitirán configurar ciertos parámetros por usuario. Podemos crear más usuarios, tantos como deseamos, creamos otro usuario “Beatriz” para poder ver la vista del desplegable de usuarios.

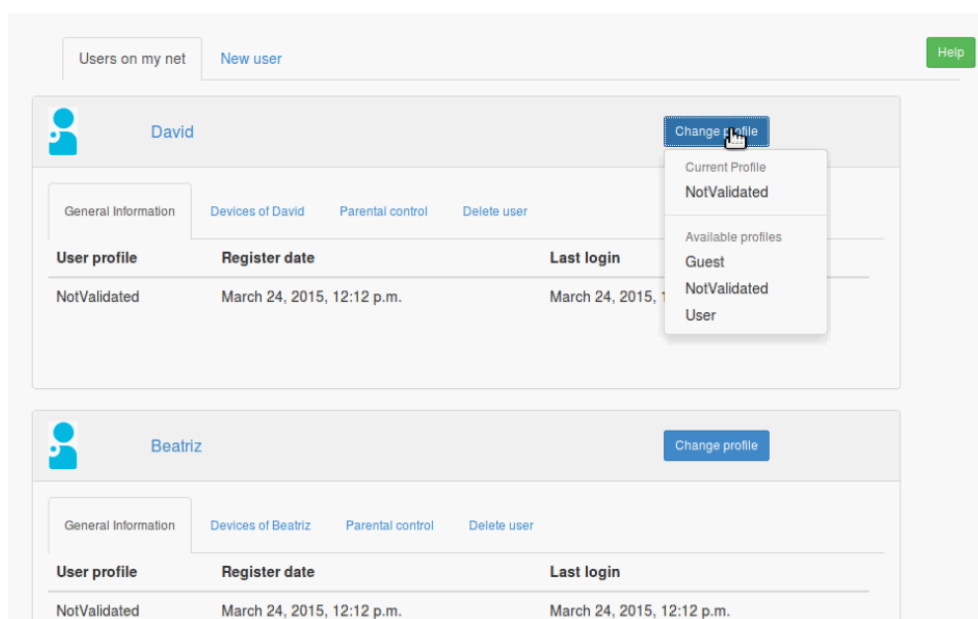


Figura 6.8: Primer acceso: Cambio de perfil

A continuación, y con el escenario planteado, pasamos a simular que aparece un dispositivo en la red, arrancamos el dispositivo h1:

```

root@ctrust:/home/cloudtrust/ctrust-demo#
root@ctrust:/home/cloudtrust/ctrust-demo#
root@ctrust:/home/cloudtrust/ctrust-demo# vnx -f cloudtrust-v03.xml -v -start -M
h1

```

Figura 6.9: Primer acceso: Nuevo dispositivo en la red

Con esta instrucción, arrancamos en vnx una nueva máquina virtual cargada con la imagen de un dispositivo de red, previamente configurada que tratará de acceder a la red.

Nuestra red, va a reconocer el dispositivo y si navegamos a la página de dispositivos desconocidos (simplemente recargando) podremos ver cómo efectivamente el entorno ha recibido la petición de acceso y se nos muestra el desplegable de un dispositivo de red desconocido en la red:



Figura 6.10: Primer acceso: Vista dispositivo desconocido

Esta vista es muy importante, pues el usuario de *CloudTrust* no tiene porque contar con conocimientos sobre redes o dispositivos de red, y a de ser un entorno easy, y user-friendly. Así pues, la información que se despliega sobre el dispositivo es únicamente la más relevante, como pueden ser la dirección MAC o la dirección IP que el servidor DHCP le ha asignado. Además, en función de la dirección MAC, y los 24 primeros bits podemos averiguar quién es el fabricante, con una simple consulta a una base de datos. Como vemos todavía no le hemos asignado ningún usuario a este dispositivo.

El usuario dueño del dispositivo h1, tratará de navegar pues como vemos tiene dirección IP asignada 10.0.0.10, y digamos que por ejemplo va a su navegador e introduce la URL de [www.google.es](http://www.google.es):

```
* Documentation: https://help.ubuntu.com/
root@h1:~# ping www.google.es
PING www.google.es (216.58.211.227) 56(84) bytes of data.
```

Figura 6.11: Primer acceso: Navegación dispositivo h1

Tras varios segundos verá que no tiene acceso a Internet pues el encaminador no lo reconoce como un dispositivo con permisos.

Así que, el administrador de la red a través de la aplicación web, pasa a asignar el dispositivo h1 (10.0.0.10) al usuario David, que ya hemos creado previamente, y si vamos a la vista de dispositivos conocidos vemos el resultado del cambio:

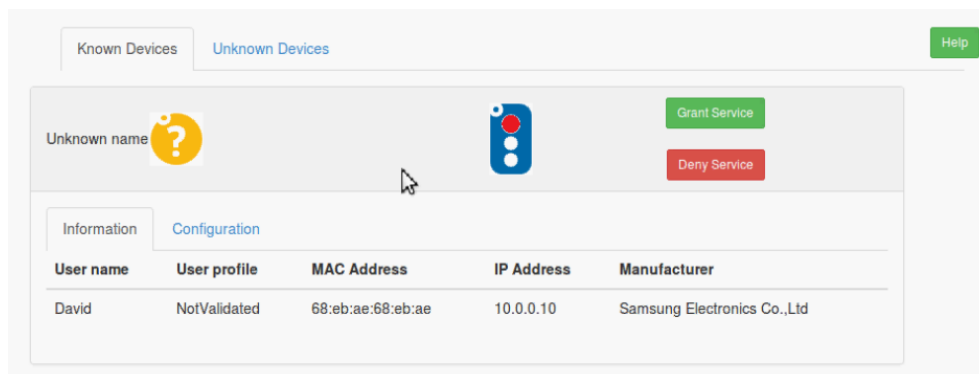


Figura 6.12: Primer acceso: Dispositivo h1 asignado a usuario David

El usuario David, es un usuario que no ha sido validado, es decir, no tiene acceso a los recursos de red, y por ello, el icono del semáforo nos indica que este dispositivo no tiene acceso a la red.

Procedemos a configurar el usuario David y le asignamos el perfil de “User” (usuario con acceso total a los recursos de red), clickando en change profile. Ahora cualquier dispositivo asociado a David, tendrá acceso a la red.

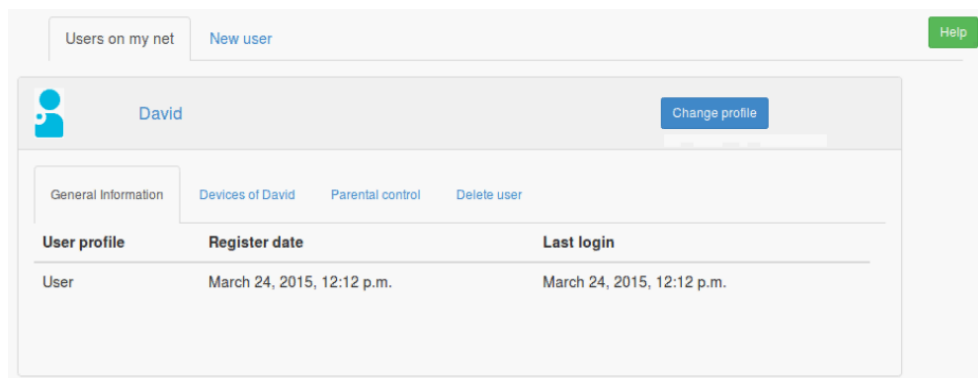


Figura 6.13: Primer acceso: Cambio de perfil a usuario David

Ahora, en vista de dispositivos veremos el resultado del cambio, y como inmediatamente el dispositivo pasa a navegar libremente pues ya es un dispositivo conocido asignado a un usuario con un perfil con acceso.

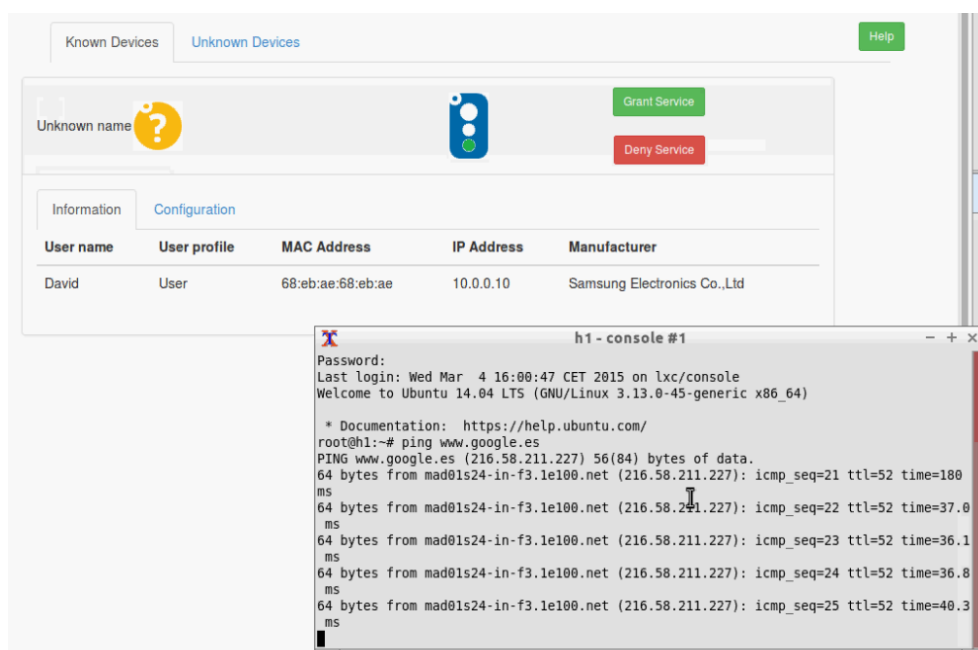


Figura 6.14: Primer acceso: Dispositivo h1 con acceso a red

Así pues, el dispositivo h1 una vez se le haya concedido el acceso, y reconocido y asignado a un usuario, se podrá configurar dentro de la aplicación web, indicando un nombre de dispositivo o un icono de dispositivo para que sea más fácilmente reconocible:



Figura 6.15: Primer acceso: Configurando h1

## 6.2. Caso de estudio 2: Acceso desde un dispositivo

A continuación se trata de representar un caso real, en el que un usuario no administrador de la red, trate de acceder desde su dispositivo a la red.

Así pues, igual que en el caso de estudio anterior, arrancamos un segundo dispositivo, en este caso el dispositivo "h2", que aparece como nuevo en la red y lo identifica el entorno igual que el con el dispositivo h1.

Digamos que este usuario trata de acceder a cualquier web en su navegador, será entonces redirigido al captive portal pues su dispositivo todavía no ha sido identificado ni verificado: (Para esta parte del proceso hemos hecho uso de un navegador basado solo en texto, elinks).



Figura 6.16: Acceso desde un dispositivo: Primer acceso

Como vemos el usuario se ha encontrado con el captive portal del entorno y trata de acceder con las credenciales del usuario Beatriz, que recordemos se trataba de un usuario no validado (sin acceso a los recurso de red). Como este usuario no tiene los permisos habilitados, se le redirige de nuevo al captive portal, y ahora trata de acceder con las credenciales del usuario David.



Figura 6.17: Acceso desde un dispositivo: Segundo acceso

Tras introducir las credenciales de David, como usuario habilitado, y clicar en el login, el dispositivo se asigna automáticamente al usuario en la base de datos, y si ahora trata de acceder a cualquier URL, podrá hacerlo:

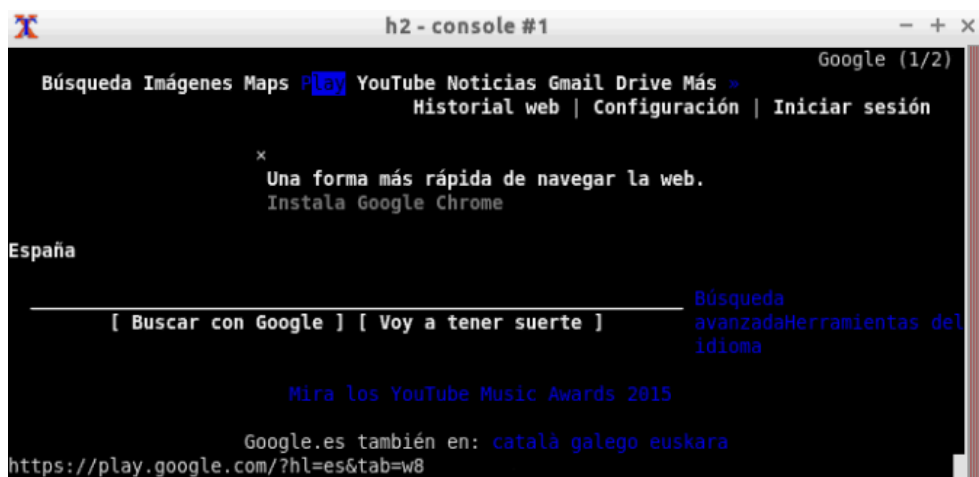


Figura 6.18: Acceso desde un dispositivo: Acceso a Google

En la siguiente imagen, se presenta cómo el nuevo dispositivo h2 (10.0.0.11), ha sido automáticamente asignado y guardado como un dispositivo de David.



Figura 6.19: Acceso desde un dispositivo: h2 asignado a David

A partir de ahora, cada vez que el usuario dueño del dispositivo h2 esté dentro de la red, no tendrá que introducir sus credenciales pues el dispositivo ya está reconocido y habilitado para navegar, a no ser que el administrador decida denegarle el servicio al usuario o bien se lo deniegue por dispositivo.



### 6.3. Caso de estudio 3: Algunas funciones más

Con la intención de mostrar algunas de las funcionalidades añadidas en la aplicación web, se va a mostrar un supuesto práctico en el que podremos ver cómo se comporta ante otra posible situación real:

Supongamos que aparece un tercer dispositivo en la red, para ello, arrancamos una tercera máquina virtual, en este caso h3. Imaginemos que este dispositivo es de un invitado en nuestro hogar. Como era de esperar si trata de navegar obviamente no podrá, pues no se le ha concedido acceso:

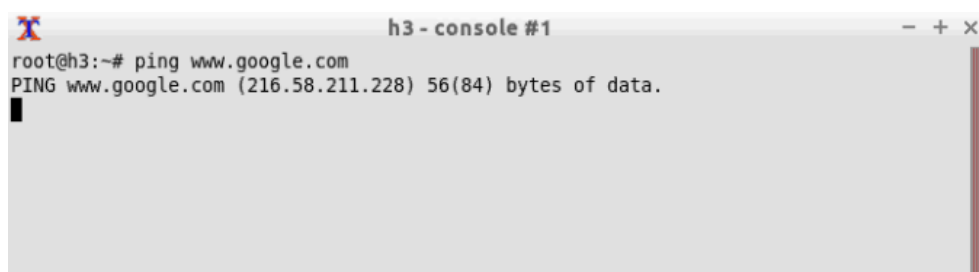


Figura 6.20: Algunas funciones más: h3 trata de navegar

Podemos crear un usuario con el perfil de invitado para esta persona:

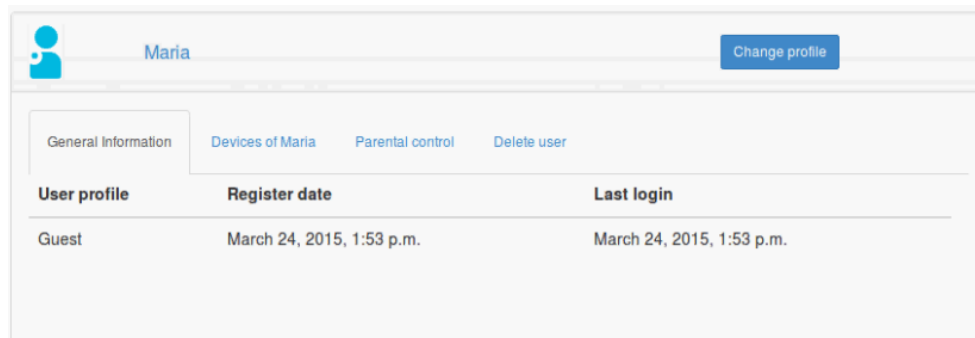


Figura 6.21: Algunas funciones más: Nuevo usuario Guest María

Y le asignamos el dispositivo a María:



Figura 6.22: Algunas funciones más: vinculamos h3 a María

El dispositivo h3 con IP 10.0.0.12 y con dirección MAC 04:1e:64:04:1e:64, que ha sido reconocido como un dispositivo de Apple está asignado a un usuario con el perfil de Guest. Se ha configurado de tal forma que un usuario Guest, tenga acceso a la red únicamente durante un período de tiempo limitado, de tal forma que desde que al usuario se le asigna el perfil, comienza a correr el tiempo, de cara la demostración que presentamos a continuación, se ha fijado un tiempo de conexión de 25 segundos:

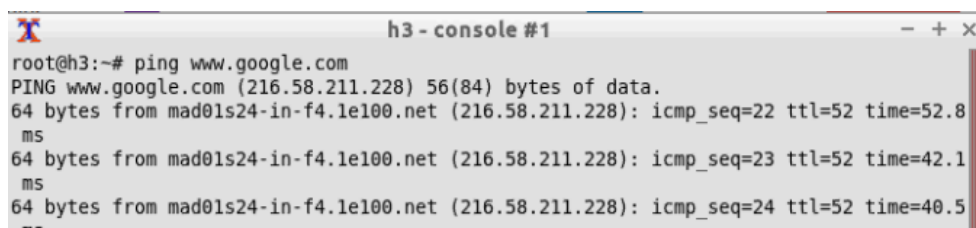


Figura 6.23: Algunas funciones más: comienza la conexión

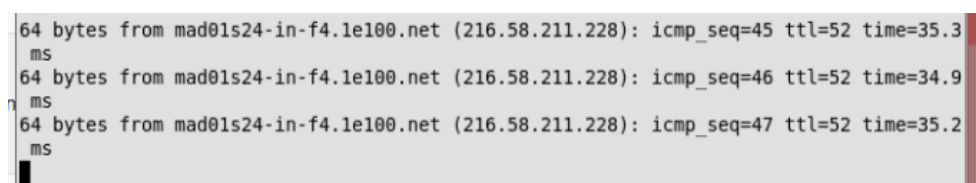
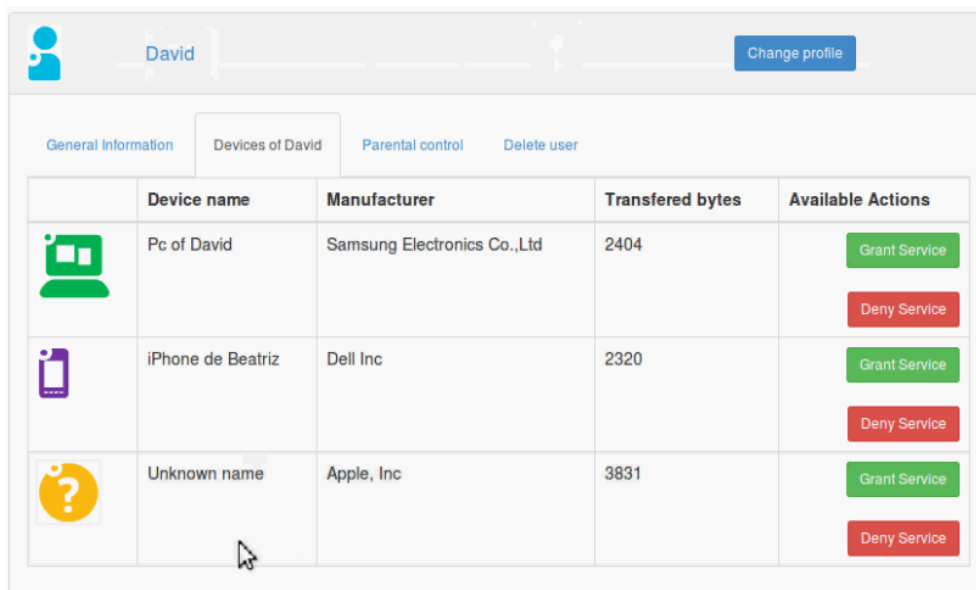


Figura 6.24: Algunas funciones más: termina la conexión

Como podemos ver, se lanzó una petición a [www.google.es](http://www.google.es), y tras varios intentos fallidos ( $\text{icmp\_seq} < 22$ ) ya que el dispositivo no estaba habilitado para navegar, comienza la transmisión a partir del instante en el que se le conceden los permisos (a partir de  $\text{icmp\_seq} = 22$ ) hasta que transcurre el tiempo límite (hasta  $\text{icmp\_seq} = 47$ ).

Probemos ahora a explorar algunas de las otras funciones que se han configurado, por ejemplo, podemos asignar el dispositivo h3 al usuario David para que tenga acceso total y ver un despliegue de todos los dispositivos que tiene David:






	Device name	Manufacturer	Transferred bytes	Available Actions
	Pc of David	Samsung Electronics Co.,Ltd	2404	<a href="#">Grant Service</a> <a href="#">Deny Service</a>
	iPhone de Beatriz	Dell Inc	2320	<a href="#">Grant Service</a> <a href="#">Deny Service</a>
	Unknown name	Apple, Inc	3831	<a href="#">Grant Service</a> <a href="#">Deny Service</a>

Figura 6.25: Algunas funciones más: Vista dispositivos David

Se incluye en esta vista, toda la información de carácter más relevante para el usuario, de forma que los dispositivos sean fácilmente identificables, estos son el nombre o el icono de dispositivo que se le haya asignado, o el fabricante del dispositivo. Además se incluye el contador de consumo de los dispositivos bajo la columna de “Transferred Bytes”. También la opción de denegar el servicio por dispositivo.

Se ha implementado también la posibilidad de establecer controles parentales por usuario así pues, imaginemos que el usuario David sólo tenga acceso desde las 0 a.m. hasta las 10 a.m., para ello nos vamos a la pestaña de “Parental Control” y asignamos la franja horaria que deseemos:

Figura 6.26: Algunas funciones más: Función del control parental

Como se ha explicado anteriormente en el capítulo 5.6, hay un demonio corriendo que comprueba periódicamente la hora actual y la contrasta con los límites establecidos, si está fuera, deniega el servicio.

Por otro lado, se presenta a continuación, la funcionalidad de log de la que dispone la aplicación web:

Show All	Sort by date	Sort by event	Show only ▾
Network Log			
Date	Event	Message	
2015-03-24 13:52:41	New device appeared	A new device with IP No descubierta and mac 82:d7:6f:b0:6d:47 is new to the network	
2015-03-24 13:52:48	User log in	root log in	
2015-03-24 13:52:56	New user	The user David has been created.	
2015-03-24 13:53:04	New user	The user Beatriz has been created.	
2015-03-24 13:53:07	Traffic used	82:d7:6f:b0:6d:47consumed Bytes:118860	
2015-03-24 13:53:11	New user	The user Maria has been created.	
2015-03-24 13:53:17	User changed profile	The user David now belongs to the profile User	

Figura 6.27: Algunas funciones más: Log

El Network log nos permite ver cuáles son las principales acciones que se han llevado a cabo, las que se presentan en la imagen superior son algunos de los eventos que han ocurrido tras realizar los casos de estudio que se plantean en este documento.

Por último, y a modo de ejemplo se muestra el desplegable de los eventos más relevantes

sobre los que se puede filtrar:

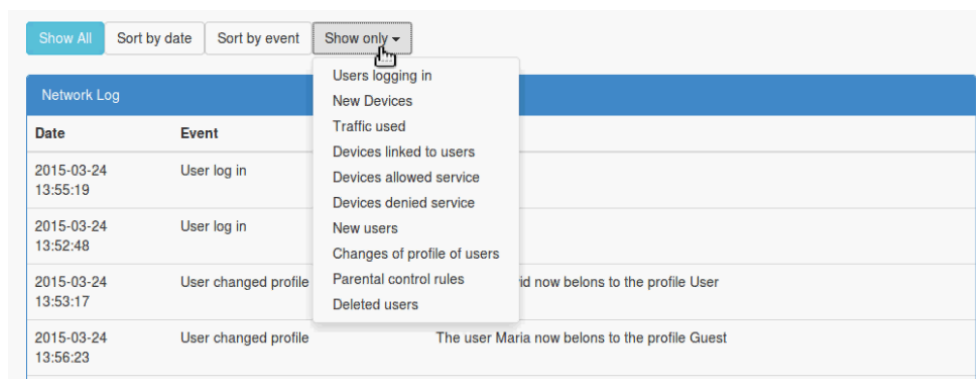


Figura 6.28: Algunas funciones más: Función Show only

## 6.4. Caso de estudio 4: Acceso de usuario no administrador

Finalmente, se ha construido y se permite el acceso a la aplicación web de configuración para aquellos usuarios que no son usuarios administrador. Estos usuarios tendrán acceso únicamente a la información referente a sus dispositivos o su historial de consumo, y no al del resto de usuarios, privilegio reservado para el usuario administrador.

Las dos siguientes capturas de pantalla reflejan un par de vistas de lo que el usuario David vería si hiciera una consulta sobre sus dispositivos (véase que únicamente puede leer la información y no configurarla) o una consulta sobre su consumo, resultado de las pruebas de los casos de estudio anteriores.

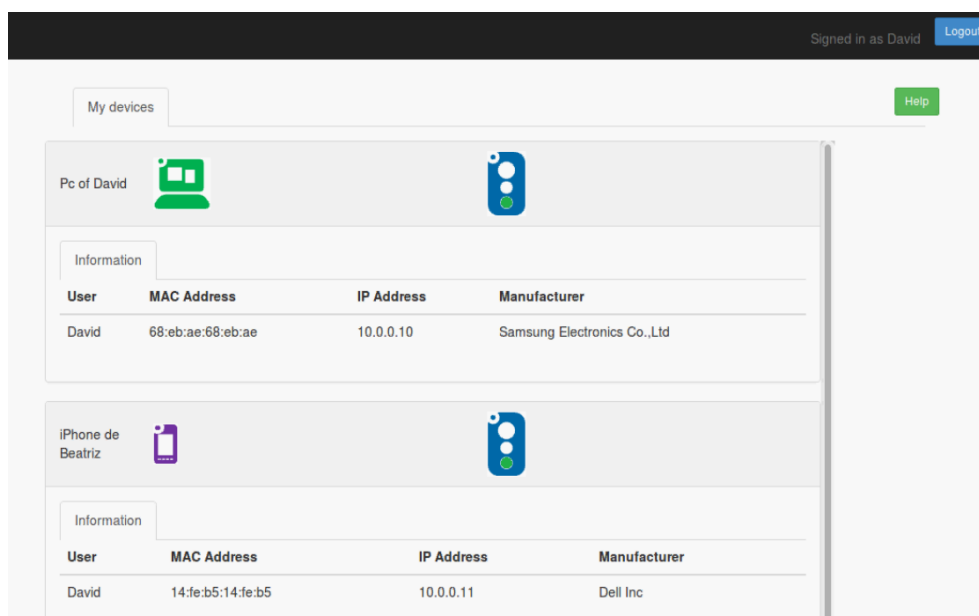


Figura 6.29: Acceso de usuario no administrador: Vista de dispositivos de David

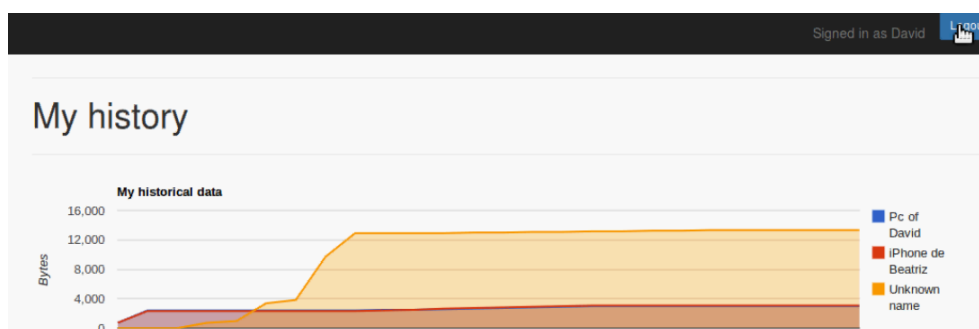


Figura 6.30: Acceso de usuario no administrador: Histórico de consumo de David



# Capítulo 7

## Conclusiones

El capítulo que cierra este proyecto evalúa los compromisos y objetivos alcanzados, y refleja cómo lo aprendido y desarrollado nace para complementar y completar la formación recibida en la Universidad en la titulación de Ingeniero Superior de Telecomunicación.

Tras un año de trabajo, la solución *CloudTrust* se encuentra empaquetada, completamente funcional, y lista para presentar a potenciales clientes interesados.

### 7.1. Lecciones aprendidas

El proyecto *CloudTrust* es un proyecto de un largo recorrido, y tuve la suerte de participar en la segunda mitad de su proceso de desarrollo. Cuando se comenzó a trabajar en la solución de *CloudTrust* tenía mucho trabajo por delante, y se requirió la inclusión de una persona que fuera capaz de juntar el trabajo de las tres empresas participantes, y actuase bajo la figura de director de proyecto. Por ello cuando me incorporé y durante la duración del proyecto, no sólo tuve que realizar tareas de desarrollo de código, sino que también llevé a cabo tareas de gestión, así como tareas de gestión de proyecto.

En un apartado más técnico, las capacidades adquiridas son muchas, por ejemplo, y por enumerar algunas de las más importantes, he aprendido y extendido mis conocimientos sobre bases de datos, sobre su estructura, su correcta implementación, o la forma más eficiente de implementarlas. Con Django, esto es extremadamente sencillo. También he tenido la oportunidad de extender mis conocimientos sobre programación en lenguaje Python, particularmente con el trato de bases de datos, de usuarios o como construir páginas web en HTML apartando la lógica



con Python.

Otra de las características nuevas aprendidas digna de mención, es el uso de los template-tags de Django, estos permiten hacer un preprocesado de la petición del usuario en Python, y devolver la información de tal forma que el navegador no tiene que ejecutar ningún script y la velocidad de carga de la página es mayor. Esto permite además manejar la información en local y no depender de codificaciones externas susceptibles de cambios o actualizaciones.

Siguiendo en la línea de lo aprendido entorno a la creación de páginas webs, en el proyecto se tomó la decisión conjunta de utilizar el framework de Bootstrap para mayor sencillez en la construcción de las vistas. Bootstrap plantea algunas plantillas de las que se pueden hacer uso, y que son fácilmente configurables, le ahorran al programador, tareas engorrosas como la colocación de los elementos, o el diseño de los mismos. Incluye además algunas partes de código ejecutables, a las que se llama en función de un formato de clases siguiendo la estructura de HTML. Es decir, el proceso de aprendizaje no se limita exclusivamente al uso de las herramientas más utilizadas, sino que también he aprendido a ser mejor profesional y aprender la importancia y el valor de mi tiempo.

Referente a la elección del entorno de trabajo, he de mencionar que una de las grandes lecciones aprendidas, ha sido la de la importancia de crear un preproceso de definición de requisitos, pues en proyectos tan grandes como este, un mal estudio previo puede desembocar en que las tareas no puedan ser completadas. Un ejemplo de esto es la elección de la estructura Frenetic, hay muchas estructuras de control de datos para SDN pero esta es la que mejor se adecuaba a nuestros requerimientos, pues su sencillez que permite abstraerse al desarrollador e implementar aplicaciones complejas con menor esfuerzo. Además permite usar el lenguaje Python desplegable en cualquier sistema operativo, que dispone de un amplio abanico de desarrolladores activos. Sin embargo, depende de que todas las reglas de monitorización pasen por el controlador y esto limita su escalabilidad, o para aplicaciones que requieran de anchos de banda muy grandes.

Como se ha comentado anteriormente en puntos anteriores del proyecto, el valor añadido de participar en un proyecto en el que participan varias personas y empresas, me ha hecho comprender lo importante que son cosas como la viabilidad empresarial, el cumplimiento de tareas, así cómo, la correcta definición de tiempos y procesos.

En términos generales considero que se han visto cumplidos mis objetivos personales y

los objetivos técnicos del proyecto, y no sólo eso, sino que he conseguido tender un puente que vincule el mundo de la formación con el mercado laboral, sentando las bases para futuros proyectos.

## 7.2. Conocimientos aplicados

En mi opinión personal, gran parte del motivo de estudiar una carrera era la adquisición de conocimientos, pero no sólo eso, la carrera te prepara en muchos aspectos. Los primeros años, una carrera técnica como la de Ingeniería de Telecomunicación, se centra mucho en aprender los conceptos más básicos, y sentar la base para conocimientos más profundos y complejos. Y en cambio, en las fases finales, el proceso de aprendizaje cambia y es redirigido hacia la solución de problemas, es decir, nos prepara para afrontarlos y darles solución. Este aspecto es el más importante, el conocimiento siempre está disponible, la capacidad para entenderlo no.

Transformar todo este entendimiento junto con mi crecimiento personal y la formación que he recibido durante mi periodo como estudiante ha derivado en mucho trabajo duro puesto en este proyecto. Algunas asignaturas de la carrera han tenido mayor impacto, y las enumero a continuación:

- Fundamentos de la programación y Métodos de programación: Ambas asignaturas en gran medida como se comentaba anteriormente, sentaron las bases sobre la lógica, la arquitectura y el bien hacer a la hora de programar.
- Fundamentos de los computadores I y II: Igual que las dos asignaturas anteriores, el conocimiento que recogían sobre software y hardware es de básico conocimiento a la hora de realizar cualquier proyecto técnico que se precie.
- Sistemas Telemáticas I y II, Información audiovisual en Redes de Ordenadores Planificación y Gestión de redes de ordenadores: Todas estas asignaturas combinadas dan una vista general y completa del funcionamiento y administración de redes, así tanto cómo su arquitectura, los protocolos de comunicación y su correcto diseño.
- Servicios y aplicaciones telemáticas: Ciertamente esta es una de las asignaturas que tiene una relación más directa con lo desarrollado en este proyecto. La primera toma de contacto con herramientas como Django, o lenguajes de programación como Python o HTML,

fueron gracias a esta asignatura. Los conocimientos adquiridos en este PFC no han hecho más que extender, junto con otras asignaturas, lo aprendido en ellas.

- **Proyectos:** Esta asignatura, es el primer acercamiento real hacia lo que se entiende como gestión de proyectos. Conceptos como métricas, KPI, orden de relación entre actividades, gestión de riesgos o toma de decisiones, las aprendimos en Proyectos. Dado que este proyecto se ha desarrollado en un entorno empresarial y desde la figura de gestor de proyecto, los conocimientos adquiridos y finalmente aplicados son directos.

### 7.3. Trabajos futuros

Como resultado de las pruebas se han observado distintas líneas de trabajo en las que continuar para un despliegue real de la solución:

- Mejorar el sistema de acceso, autorizando a los usuarios dependiendo del distinto nivel de acceso a información que deban tener. Establecer el concepto de administrador general del sistema en contraposición con el de hogar/cliente/contrato.
- Finalmente, no se consiguió implementar el sistema de control del ancho de banda de la solución debido a la simplicidad de la estructura de control. Se espera poder desarrollar avances en esta materia.
- Investigar los últimos avances en la tecnología Frenetic, y ampliar el controlador virtual para que soporte nuevas funciones que puedan ser interesantes.
- La construcción de gráficos sobre datos de consumo depende de un servicio externo, se desearía realizar una codificación propia.
- Implementar la solución, y recibir feedback de usuarios activos del servicio.

# Apéndice A

## Acrónimos Requisitos y Especificaciones

**ASU** Asunción

**APPWEB** Aplicación web

**CON** Configuración

**DEF** Definición

**DES** Descripción

**DISP** Dispositivos

**FUN** Funcionalidad

**HOG** Hogar

**INTF** Interfaz

**OVS** OpenvSwitch

**REND** Rendimiento

**REQ** Requisito

**SIS** Sistema

**SWEB** Servidor Web

**USU** Usuario



# Bibliografía

[1] Web del proyecto Django

<https://www.djangoproject.com>

[2] Web del proyecto PostgreSQL

<https://www.postgresql.org/docs/>

[3] Web del proyecto Python

<https://www.python.org/doc/>

[4] Web del proyecto Bootstrap

<https://bootstrapdocs.com>

[5] Web de Apache

<http://httpd.apache.org/docs/2.2/es/>

[6] Web estándar w3c

<http://www.w3c.es/estandares/>

[7] Grupo de Sistemas y Comunicaciones - Universidad Rey Juan Carlos.

<http://gsyc.urjc.es>

[8] Web del proyecto VNX de la Universidad politécnica de Madrid

<http://vnx.dit.upm.es>

[9] Web del proyecto open vSwitch

<http://openvswitch.org/support/>

[10] Web del proyecto Frenetic

<https://github.com/frenetic-lang/pyretic/wiki>

[11] Instituto Nacional de Estadística

[http://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica\\_C&cid=1254736176741&menu=ultiDatos&idp=1254735976608](http://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176741&menu=ultiDatos&idp=1254735976608)

[12] Web del proyecto LaTeX

<https://latex-project.org/guides/>

[13] Web del Django: User Authentication

<https://docs.djangoproject.com/es/1.9/topics/auth/default/#user-objects>