Trabajo Fin de Grado

Desarrollo de módulos para la obtención de información de switches Cisco y presentación en dashboard

Development of modules for information retrieval of Cisco switches and its dashboard representation

Autora

Aimar Paola García López

Director

Jorge Marco Poza

Ponente

José García Moros

Escuela de Ingeniería y Arquitectura / Universidad de Zaragoza

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Aimar Paola García López,

con nº de DNI 73134761M en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Desarrollo de módulos para la obtención de información de switches Cisco y presentación en dashboard,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 3 de febrero de 2017

Fdo:

Agradecimientos

Me gustaría en primer lugar agradecer a Nologin Consulting S.L. la oportunidad que me han concedido al permitirme colaborar con ellos para el desarrollo de este TFG y por la ayuda prestada.

En segundo lugar, a todas las personas que he ido conociendo durante el transcurso de la carrera, ya que han contribuido a mi desarrollo educativo y personal.

Finalmente, y sobre todo, quiero agradecer a todas las personas que me han apoyado desde que me conocen. A mi familia, amigos y novio por todo el cariño que me han transmitido y que ha permitido forjar en mí la capacidad de sobreponerme ante las dificultades y de seguir siempre para delante.

Muchas gracias a todos.

Aimar Paola García López
1 de febrero 2017

Desarrollo de módulos para la obtención de información de switches Cisco y presentación en dashboard

Resumen

En la actualidad, la tecnología forma parte de nuestro día a día. Esto es aún más cierto para las empresas, que deben disponer de una infraestructura tecnológica funcional y eficaz para llevar a cabo las tareas diarias con éxito. La dificultad que conlleva gestionar estos aspectos ha propiciado la creación de empresas de servicios profesionales que se dedican a administrar las infraestructuras TI (Tecnologías de la Información) de otras entidades. Esto permite a quienes les confían esta tarea reducir costes en sus departamentos TI y la tranquilidad de dejar en manos de expertos este trabajo.

El trabajo fin de grado (TFG) aquí presentado se ha llevado a cabo en la empresa aragonesa Nologin Consulting S.L., siendo uno de sus servicios la gestión y administración de infraestructuras tecnológicas. El trabajo desarrollado se enmarca en su proyecto "Work On Data", diseñado para mejorar la calidad de los servicios gestionados de los clientes, y cuya función principal consiste en el desarrollo de un producto de administración remota que permite la obtención de información de forma automática de los sistemas administrados. Dicho producto se materializa en un portal web (www.workondata.com), donde los clientes disponen de varios *dashboard* en los que se presenta información sobre sus "assets", es decir, los elementos gestionados.
Work On Data constituye pues una novedosa herramienta personalizada basada en el desarrollo de módulos específicos para cada tipo de asset que se desea gestionar. Por lo tanto, se requiere una gran inversión de tiempo y esfuerzo para generar suficientes módulos lo cual permita gestionar el máximo número de assets.

El objetivo de este TFG ha sido el de colaborar en su proyecto mediante el desarrollo del módulo de conocimiento para los switches CiscoCatalyst (producidos por la empresa norteamericana Cisco Systems). Es decir, de la implementación de las herramientas que permiten obtener información sobre assets CiscoCatalyst, y de la presentación en el portal web de los datos obtenidos, de forma coherente con el resto de los módulos ya existentes y operativos. Se ha usado como apoyo para ello el trabajo ya desarrollado hasta el momento y la infraestructura existente.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS Y TABLAS

LISTADO DE ACRÓNIMOS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
CDP	Cisco Discovery Protocol
CLI	Command Line Interface
CSS	Cascading Style Sheets
DNS	Domain Name System
DOM	Document Object Model
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDE	Integrated Development Enviroment
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MTV	Model Template View
MVC	Model View Controller
OID	Object Identifier
OSI	Open System Interconnection
PAM	Port to Application Mapping
PKI	Public Key Infrastructure
RSH	Remote SHell
SNMP	Simple Network Management Protocol
SO	Sistema Operativo
SSH	Secure SHell
TFG	Trabajo Fin de Grado
TI	Tecnologías de la Información
UML	Unified Modeling Language
URL	Uniform Resource Locator
WWW	World Wide Web

Capítulo 1

INTRODUCCIÓN

En este capítulo introductorio se ofrece una visión global sobre el trabajo desarrollado, contextualizándolo, presentando la motivación y los objetivos. Además, se detalla la línea temporal de trabajo que se ha seguido y la estructura del documento.

1.1 Contexto

Este Trabajo Fin de Grado (TFG) forma parte del proyecto Work On Data, de la empresa Nologin Consulting S.L. con la finalidad de contribuir a aumentar el inventario de módulos de conocimientos desarrollados. Concretamente, el aquí presentado, es el de los switches CiscoCatalyst modelos 3560X-24 y 3750X-48 pertenecientes a clientes de Nologin y gestionados remotamente por ésta.

Nologin es una compañía aragonesa que lleva más de 15 años colaborando con otras empresas de todo el mundo para guardar, gestionar, organizar, proteger y mantener su más preciado activo: la información. Ayudan a las empresas y proveedores de servicios a diseñar, poner en marcha y hacer evolucionar sus infraestructuras tecnológicas a la par que su negocio [1].

Uno de los beneficios del desarrollo de este trabajo dentro del equipo de la empresa, no sólo es el contar con expertos a los que consultar por razones técnicas, sino también el poder hacer uso de su infraestructura ya consolidada (bases de datos, servidores web, balanceadores…).

1.2. Motivación del TFG

La tendencia de hoy en día de las empresas se dirige hacia el *outsourcing* (subcontratación) de servicios y plataformas en el ámbito tecnológico, ya que esto les permite no sólo reducir costes, sino también el poder realizar sus funciones principales más eficientemente, liberándose de las preocupaciones que conllevan mantener y gestionar la infraestructura tecnológica que sustenta su trabajo.

En el ámbito de la gestión de sistemas existen ya en el mercado numerosas herramientas que facilitan esta tarea. En el caso de los productos Cisco, por ejemplo, la propia empresa ofrece numerosas soluciones para su gestión [2]. Sin embargo, en las instalaciones de las empresas coexisten equipos de distintos proveedores, y cado uno cuenta generalmente con su propia herramienta de gestión. Así es complicado obtener una visión global sobre el estado de todos los equipos que conforman la infraestructura completa. Para poner solución a este problema se ha desarrollado la herramienta del portal web del proyecto Work On Data.

El producto principal del proyecto Work On Data es el portal web accesible desde el exterior, que permite a cada usuario registrado consultar en cualquier momento y en cualquier lugar aspectos relacionados con los proyectos gestionados en los cuales participa, tales como:

1. El estado actual de las solicitudes asociadas al proyecto.
2. El estado en tiempo real de los chequeos de monitorización.
3. Documentación interna del proyecto.
4. El estado y configuración de cada uno de los elementos (hardware y software).
5. La colección de informes del proyecto.
6. La información contractual del proyecto.

De este modo se logra una solución global que concentra de forma estructurada la gestión de todos los sistemas y servicios del cliente en una sola herramienta.

Cabe destacar que la herramienta no proporciona la funcionalidad de modificación de los parámetros mostrados. Esto queda a cargo de las propias herramientas inherentes al equipo dispuestas para este fin.

1.3. Objetivos

Los objetivos se enuncian en el título del trabajo, y presentan dos fases claramente definidas: el desarrollo del módulo para la obtención de información de switches CiscoCatalyst (módulo de conocimiento), y el desarrollo de las vistas del dashboard en las que se presentan los datos recogidos por el primero. Haciendo referencia a los distintos servicios del portal web enumerados en el apartado anterior (1.2.), el trabajo realizado se corresponde con el cuarto punto, ya que los datos que se recogen en el módulo y que finalmente son mostrados en el dashboard son relativos al estado y la configuración de, en este caso, los switches CiscoCatalyst.

En primer lugar, para el desarrollo del módulo de conocimiento de los switches CiscoCatalyst, modelos 3560X-24 y 3750X-48, se ha trabajado con varios de ellos controlados remotamente. Esta primera fase engloba la familiarización con el entorno de trabajo, las herramientas y las tecnologías empleadas, así como el estudio de la información posible de obtener, su análisis para seleccionar únicamente la más útil, y finalmente la implementación del módulo de conocimiento y la obtención de una salida correcta y lista para usarse en la siguiente parte del trabajo.

La segunda fase, la generación de las vistas del dashboard, comienza por la puesta en producción del módulo desarrollado, de manera que comience a recolectar y almacenar la información deseada de manera automática. A partir de entonces, con los datos disponibles, se procede a trabajar con las tecnologías web para presentar esos datos de forma adecuada y siguiendo la línea de los demás módulos de conocimiento ya visibles en el portal.

1.4. Cronograma de trabajo

A continuación se presentan los cronogramas con la división en fases del trabajo realizado (figuras 1 y 2), para cada una de las dos partes del proyecto. La redacción y las correcciones del TFG se han incluido en el cronograma de la segunda fase.

El proceso de desarrollo ha sido iterativo. Debido a la relación entre las dos partes del proyecto, las modificaciones indicadas en el cronograma de la figura 2 engloban las correcciones de ambas partes.

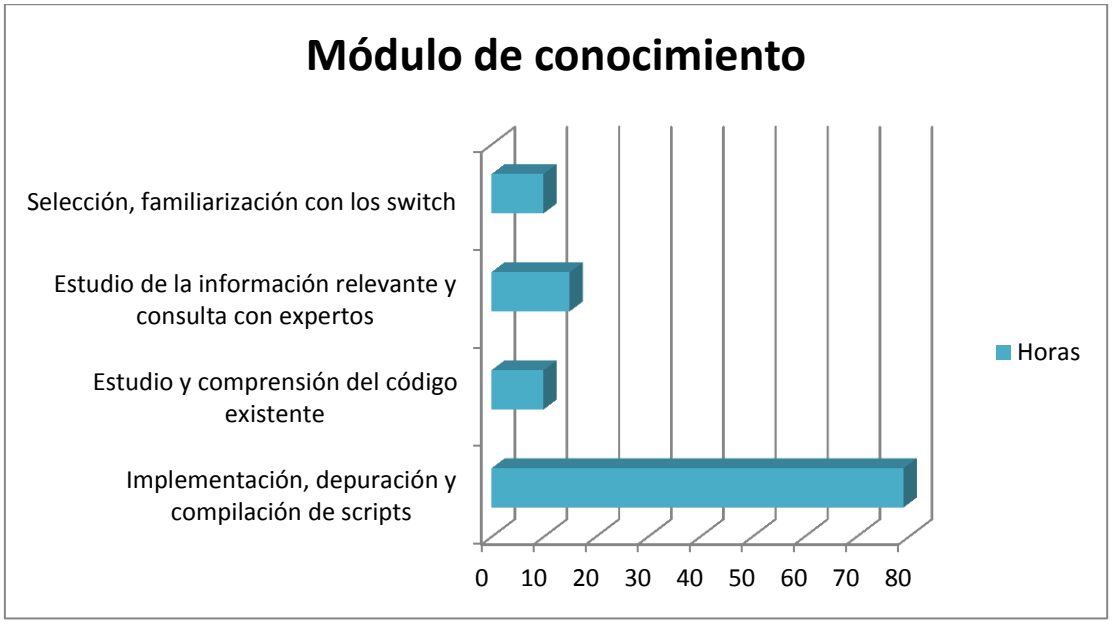

Figura 1 – Cronograma del desarrollo del módulo de conocimiento

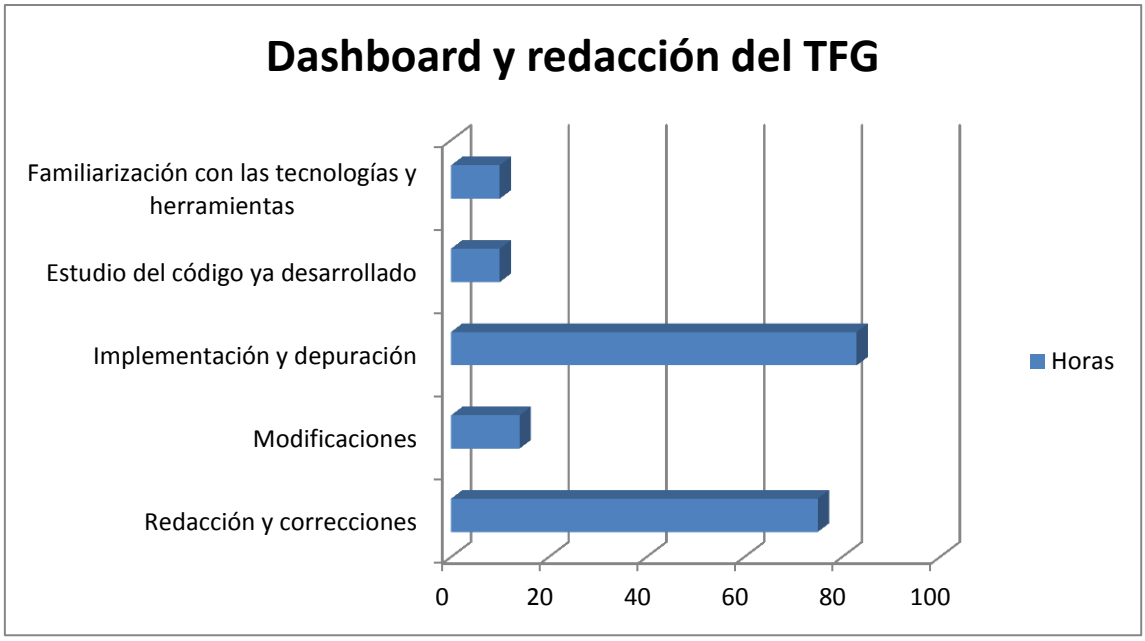

Figura 2 – Cronograma del desarrollo de las vistas del dashboard y de la elaboración del TFG

1.5. Estructura del documento

El documento de la memoria de este TFG consta de cuatro capítulos más un anexo, cuyos contenidos se describen a continuación (excepto el del presente):

- **Capítulo 2 - Desarrollo del módulo de conocimiento:** incluye todos los aspectos referentes a la primera parte del TFG, el desarrollo del módulo de conocimiento para switches CiscoCatalyst.

- **Capítulo 3 - Presentación en dashboard:** incluye todos los aspectos referentes a la segunda parte del TFG, el desarrollo de las vistas de los dashboard de la página web.

- **Capítulo 4 - Conclusiones y líneas futuras:** enuncia las conclusiones abstraídas tras la finalización del trabajo realizado, y posibles líneas futuras a considerar.
- **Anexos A y B**: contienen elementos aclarativos acerca de las cuestiones abordadas en los capítulos 2 y 3, respectivamente.

Capítulo 2

DESARROLLO DEL MÓDULO DE CONOCIMIENTO

En este apartado se presenta todo lo relativo al desarrollo del módulo de conocimiento CiscoCatalyst.

2.1. Conceptos previos

A continuación se presentan los conceptos previos imprescindibles que se mencionarán durante el resto del texto. Así mismo, estos son especificaciones previas del proyecto.

Asset

Dentro del contexto de la gestión de configuraciones, se denomina "asset" a cada uno de los elementos de configuración mínimos e indivisibles que requieren de un trabajo de administración. Un asset puede ser físico (por ejemplo, un servidor), lógico (un sistema operativo instalado en el servidor) o conceptual (un servicio que proporciona el servidor).

En este caso, los assets son cada uno de los switches con los que se ha trabajado, y que se presentarán más tarde. Todos estos assets tienen como elemento común el módulo de conocimiento, conjunto de herramientas mediante el cual se obtiene información sobre cada uno de ellos.

El estado de cada asset define su comportamiento, tanto en la ejecución de los satélites (definido más adelante) para determinar si debe o no debe realizarse la ejecución de su módulo de conocimiento asociado, como a la hora de consultar la información obtenida a través del portal web, mostrando u ocultando el acceso a su visualización.

Base de datos de assets

Para un correcto seguimiento de los assets existentes, sus estados y en general de toda la información necesaria asociada, se requiere tenerla almacenada adecuadamente. Esto lo hace posible la base de datos de assets, encargada de guardar colecciones de assets de diferentes clientes. Cada cliente dispone de una única colección, separada e independiente de los demás. Dentro de su colección se encuentran los diferentes de assets que le pertenecen. Por ejemplo, un cliente puede tener en su colección 3 assets CiscoCatalyst para unos switches a, b y c. Además puede contener assets de otro tipo, cuya información sea obtenida gracias a otro módulo de conocimiento.

La base de datos assets emplea el motor noSQL MongoDB [3]. Esta permite establecer roles y permisos a nivel de colección, pudiendo proteger así el acceso de cada cliente únicamente a las colecciones en las que tiene permisos.

En la figura 3 a continuación se muestra un ejemplo ilustrativo de la base de datos assets, y de un documento JSON contenido en ella.

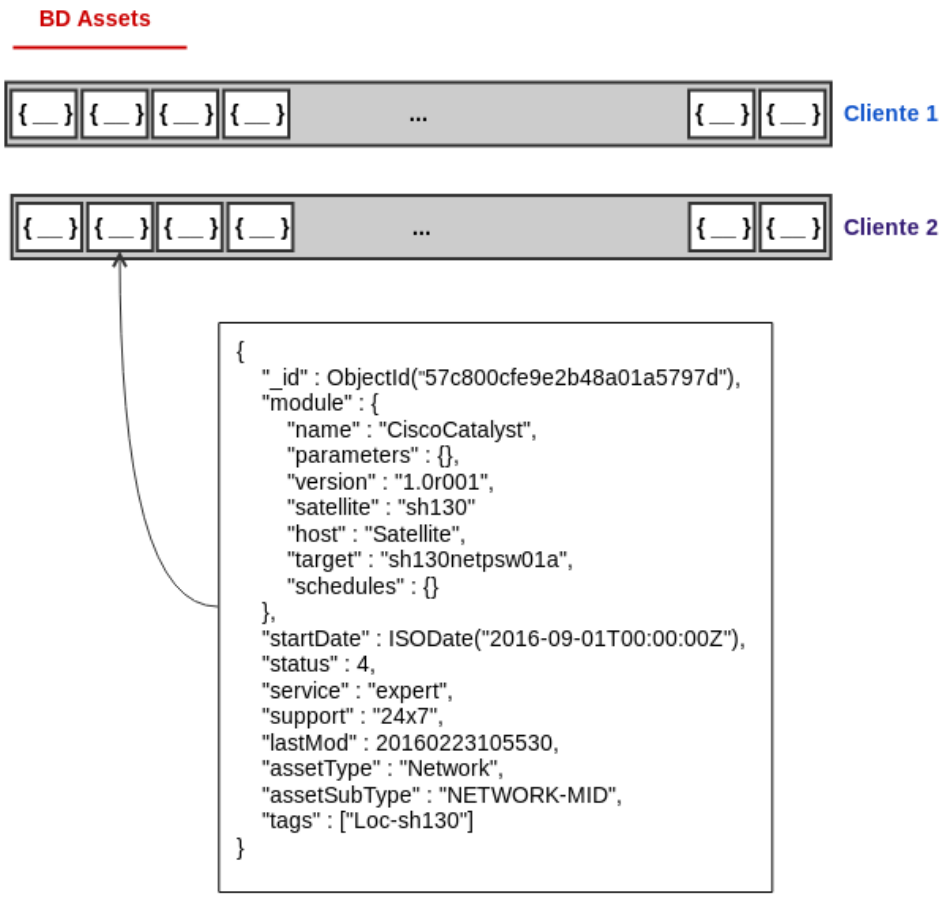

Figura 3 – Esquema de la Base de Datos de assets para un asset ejemplo“sh130netpsw01a”

Base de Datos de gestión de configuraciones (BD Data)

Como ya se ha mencionado, el propósito del desarrollo de módulos de conocimiento de diferentes assets es la obtención de información de configuración y de estado, para posteriormente mostrarla en el portal web. Esto conlleva por tanto el almacenamiento de la información recogida de algún modo.

La base de datos de gestión de configuraciones ("BD data") es, por tanto, la base de datos que sirve para almacenar información detallada recopilada para cada asset. Esta información se recoge repetidamente a lo largo del tiempo.

La base de datos de gestión de configuraciones se almacena mediante el motor de base de datos noSQL MongoDB (al igual que la base de datos de assets), dado que, entre otras características, guarda sus datos en documentos tipo JSON [4] con un esquema dinámico. Dada la fuerte heterogeneidad existente entre las diferentes salidas producidas por cada uno de los submódulos de conocimiento resulta imposible emplear un motor de base de datos SQL con esquema fijo.

La "BD data" es única para cada cliente, y cada colección se corresponde con los resultados recogidos por parte de todos los módulos de conocimiento para un día del año concreto (dAAAAmmdd). Esta organización permite, gracias a la herramienta de agregación de MongoDB Aggregation Framework explotar la información obtenida mediante distintos submódulos de conocimiento para un mismo día en particular.
Al igual que ocurre en la base de datos de assets, cada cliente solo tiene permisos de acceso a su BD Data.

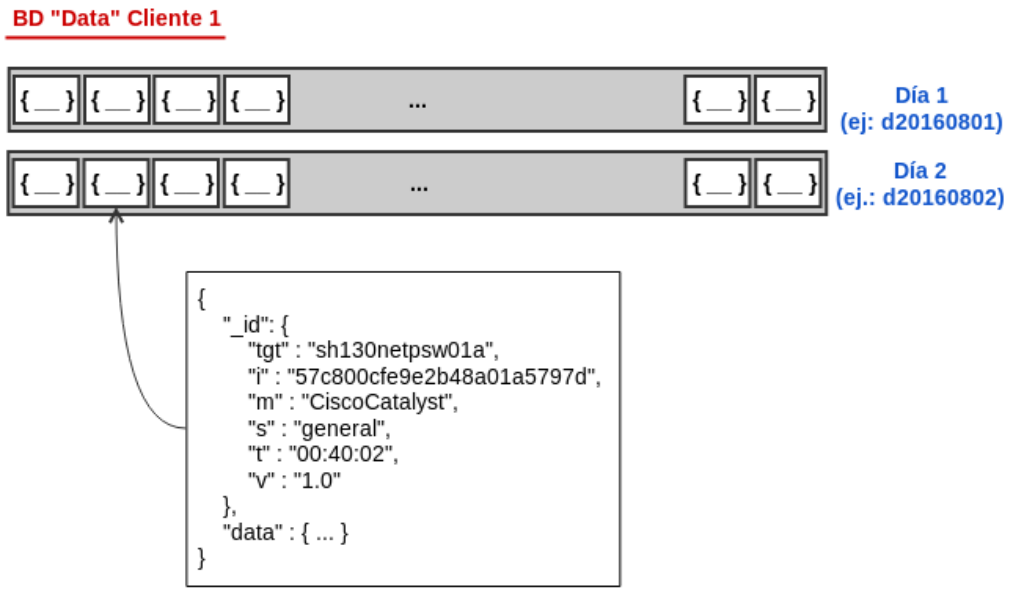

Figura 4 – Esquema de la Base de Datos Data y vista simplificada de una colección concreta

Módulo de conocimiento

El módulo es el punto de partida para organizar y estructurar no solo la ejecución de los futuros binarios sino también la disposición de la información obtenida en el portal web. Se compone de cada uno de los scripts que implementa la recolección de información acerca del elemento para el cual se desarrolla (y por tanto, que se desea gestionar), en este caso, de switches CiscoCatalyst.
El módulo de conocimiento de switches CiscoCatalyst debe ser capaz, por tanto, de recolectar información de todo switch Cisco del mismo modelo que el de los switches de trabajo, que se presentan en la sección 2.2.1. del documento.

La creación del módulo de conocimiento consiste, por tanto, en desarrollar herramientas personalizadas para obtener información de cada asset.

Cada módulo de conocimiento se identifica mediante su nombre y su versión. Pueden existir diferentes versiones de un mismo módulo con modificaciones ligeras o sustanciales, tanto en su contenido como en su programación. En este trabajo se ha desarrollado la primera versión del módulo de conocimiento de los switches CiscoCatalyst.

La versión desarrollada del módulo de conocimiento (y en general, para todas las que se desarrollen), cuenta con:

- Un binario (compilado según plataforma destino) para cada submódulo y albergado en la carpeta "src".
- Un fichero de metadata en formato JSON ("CiscoCatalyst.json").
- Un fichero de texto descriptivo ("readme.txt").

La metadata es la información que describe el funcionamiento propio de cada módulo de conocimiento. Es en dicho fichero donde se define la relación de pertenencia entre módulo y submódulos. También es allí donde se define la programación de la ejecución ("*schedule*") establecida por defecto para cada uno de los submódulos. El contenido de este fichero está expresado en formato JSON. Para consultar su contenido, ver el anexo A.1.

El fichero de texto, generalmente denominado "readme.txt", simplemente muestra información acerca del módulo (nombre, versión y lista de submódulos) y del desarrollo de dicho módulo (fecha y autor). Su propósito principal es aportar información básica referente al módulo de conocimiento de un modo sencillo y claramente legible. Para consultar su contenido, ver el anexo A.2

La carpeta src contiene todo el código fuente del módulo de conocimiento, ordenado por submódulos.

Submódulo de conocimiento

Cada módulo de conocimiento se compone de submódulos de menor tamaño y propósito concreto. Gracias a esta división en dos niveles, se puede realizar un reparto más eficiente y minimizar así el tiempo de respuesta de cada submódulo. Además, aporta sencillez a la hora de añadir futuras mejoras y nuevas funcionalidades y produce ejecutables finales más ligeros.

Cada submódulo puede ser programado con una periodicidad de ejecución diferente pudiendo aumentar el número de consultas para aquellos submódulos que obtengan información más dinámica y reducirlo para otros cuya información recogida sea más estática.

Los submódulos se traducen en un binario ejecutable final. Este binario es el encargado de recoger información acerca de un área concreta y debe producir una salida JSON y de tan solo 1 línea con el siguiente formato:

```
{
      "asset": <assetId>,
      "date": "AAAAMMDD",
      "host": <hostname>,
      "module": <moduleName>,
      "submodule":<submoduleName>,
      "time": "HH:MM:SS",
      "version": "X.X",
      "data": {
            "section1":{..JSON..},
            "section2":{..JSON..},
                  ...
            "sectionN": {..JSON..}
      }
}
```

Figura 5 – Formato de la salida de un submódulo

En el anexo A.1 se encuentra el resultado de la ejecución del submódulo General.

Sección de conocimiento

Dentro de cada submódulo de conocimiento se engloban varias secciones de contenido similar o complementario. Por ejemplo, el submódulo “ip”, contiene la secciones: ”routes”, “arp”, “portmap” y “sockets”.

Cada una de estas informaciones se corresponde con una sección indivisible ubicada en un fichero Python separado que será importado en el fichero global de cada submódulo, desde el cual se llamarán a las funciones en él definidas.

2.2. Infraestructura

Esta sección pretende presentar la infraestructura de trabajo, los switches que se han utilizado en el estudio de los parámetros que debe recoger el módulo de conocimiento, y explicar la interacción entre los elementos que forman parte de la infraestructura global.

2.2.1. Infraestructura de trabajo para el desarrollo del módulo de conocimiento

A continuación en la figura 6 se presenta un esquema con los elementos necesarios para el desarrollo del módulo de conocimiento y que se definió al comienzo del proyecto Work On Data.

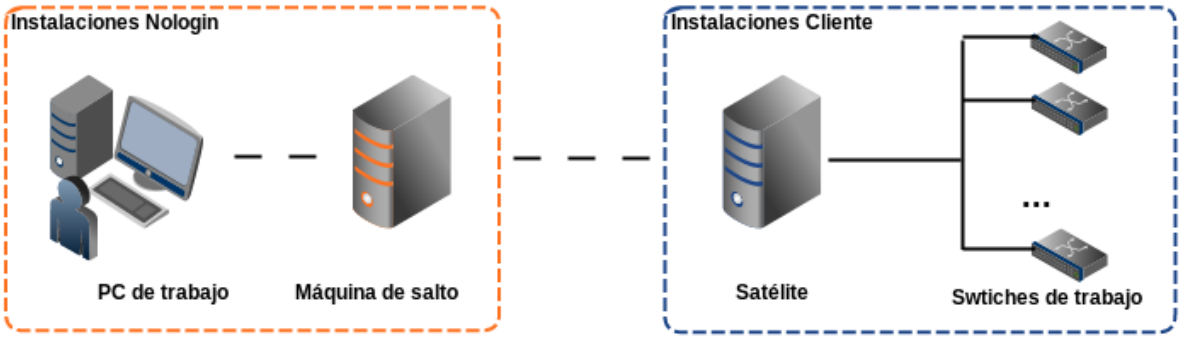

Figura 6 – Esquema de Infraestructura de trabajo

El PC de trabajo se refiere al equipo desde el cual es posible acceder al resto de los elementos presentes en el esquema. Este se encuentra en las instalaciones de la empresa administrada (cliente), y desde el cual se han elaborado los scripts que conforman el módulo de conocimiento.

El equipo "máquina de salto" es aquella conectada a los distintos satélites desplegados para cada cliente y por tanto que permite el acceso. A esto máquina se tiene acceso desde la máquina de trabajo mediante el protocolo SSH.

Satélite es la máquina (física o virtual) que, ubicada en la infraestructura del cliente, se encarga de obtener información acerca de la configuración y el estado de los assets pertenecientes al proyecto en cuestión. Es decir, lleva a cabo la ejecución de los módulos de conocimiento pertinentes y recoge los resultados obtenidos para cada asset, almacenándolos en la BD Data propia de cada o proyecto. Por tanto, el satélite actúa como "*host*", y el asset al que se dirige la ejecución del módulo de "*target*" (objetivo).
A esta máquina se accede desde la máquina de salto, también usando el protocolo SSH.

Por último, el "*target*" es en cada caso el switch CiscoCatalyst al que se destina la obtención de información llevada a cabo por el módulo de conocimiento. Por tanto, estos switches pertenecen al cliente y se encuentran en alguna de sus instalaciones.
Estos elementos de red tienen instalado uno de los sistemas operativos de Cisco, y como tal, son manipulados mediante comandos definidos por éste. En todo caso se utilizaron comandos de tipo "*show*" [5] (es decir, para mostrar información ya existente, que no modifican el estado de las variables).

Se ha trabajado (de forma remota) con los siguientes modelos de switches:

Figura 7 – Switch CiscoCatalyst 3560, 24 puertos

Figura 8 – Switch CiscoCatalyst 3750, 48 puertos

2.2.2. Interacción entre elementos de la infraestructura

Una vez familiarizados con la infraestructura que ha permitido el desarrollo y correcto despliegue del módulo de conocimiento, a continuación se procede a explicar de forma más detallada la interacción entre los elementos de la infraestructura y su papel en la ejecución del módulo de conocimiento. El siguiente esquema (Figura 9) tiene ese fin.

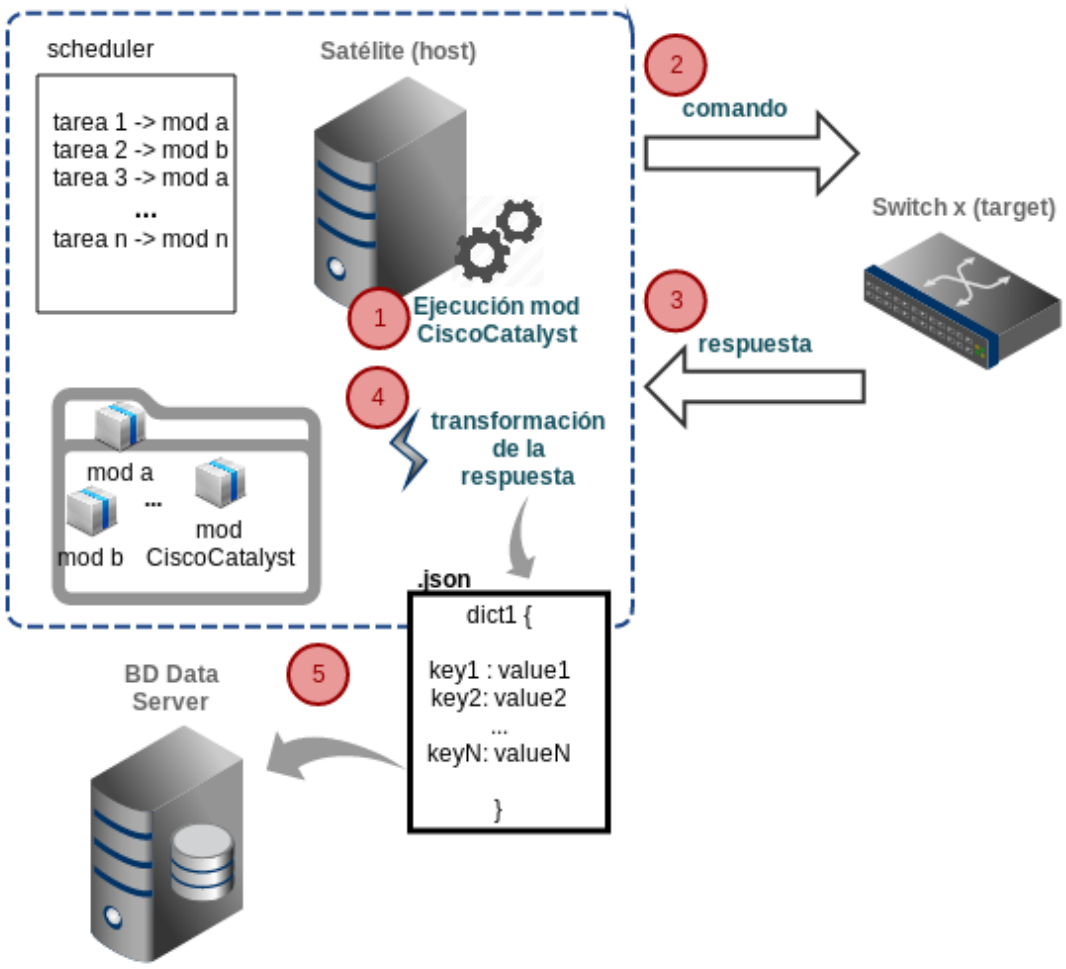

Figura 9 – Ciclo de funcionamiento del módulo de conocimiento

Como se ha explicado, el satélite es el responsable de la ejecución y el almacenamiento de los datos de cada módulo de conocimiento. Las tareas que debe ejecutar el satélite están especificadas en un fichero determinado ("*scheduler*"). Cada una de ellas define en la ejecución programada de los submódulos pertenecientes al módulo de conocimiento asociado a tal asset activo.

Llegado el momento determinado por el scheduler de lanzar la ejecución, el satélite se conecta en primer lugar al switch especificado en la tarea, el target "switch x" de la figura 9 mediante el protocolo SSH. Una vez queda establecida la

conexión, en los scripts programados en Python que componen cada submódulo, se lleva a cabo el envío de comandos al switch interpretables por él. La respuesta se transforma y almacena de forma adecuada, y finalmente se obtiene el resultado final del submódulo expresado en formato JSON. Esta información se almacena temporalmente en el satélite, hasta que llegue el momento en el que intenta conectarse al servidor que alberga y gestiona las bases de datos mencionadas en el punto 2.1. En caso de éxito, vuelca los datos en la base de datos BD Data, en la colección correspondiente.

2.3. Herramientas y tecnologías

A continuación se presentan las herramientas y tecnologías empleadas en el desarrollo del módulo de conocimiento.

Python

El lenguaje de programación predefinido y en el que están desarrollados todos los módulos de conocimiento es Python (versión 2.7), y por tanto el módulo de conocimiento CiscoCatalyst no es excepción.

Pexpect

Para la ejecución de comandos Cisco remota se ha elegido Pexpect [6], la implementación para Python de Expect. Esta herramienta es muy útil, ya que permite automatizar la creación (en éste caso concreto) del canal SSH entre el satélite y el target sin necesidad de interacción humana (lo cual es requisito indispensable para el proyecto). Esto se lleva a cabo definiendo diferentes comportamientos ante los diversos estados que se pueden dar durante la creación, mantenimiento y destrucción de la sesión (petición de la contraseña, aceptación del *fingerprint*, finalización de la sesión…).

Para la misma tarea se estudiaron otras posibles herramientas RSH y SNMP.

RSH (Remote SHell) [7] es un programa de consola que permite ejecutar comandos en ordenadores remotos. En el caso que se trata, bastaría con habilitar RSH en la máquina correspondiente y permitir a determinados usuarios ejecutar comandos en cierta máquina identificada por su IP. La ventaja es que RSH funciona en la mayoría de sistemas operativos UNIX y Linux, sin embargo, el hecho de que la información se transmita en texto claro hizo que se descartara la posibilidad de uso.

SNMP es un protocolo para el intercambio de información de administración entre dispositivos de red. Está soportado por casi cualquier equipo de red, incluido

los switches Cisco de trabajo. Permite tanto conocer como modificar los parámetros incluidos en el la estructura de árbol que los alberga como objetos identificados por OIDs (Object IDentifier). Sin embargo, estos parámetros están predefinidos, lo cual limita la información que es posible obtener.

Al conectarse directamente mediante SSH, con la ayuda de la API (*Application Programming Interface*) Pexpect, no existe limitación en la información obtenida, siempre que haya definido un comando Cisco que la muestre. Además al tratarse de SSH, se garantiza una transmisión segura de toda la información enviada entre ambos equipos.

Pycharm

Para facilitar el desarrollo de los scripts en Python se hizo uso del IDE (*Integrated Development Enviroment*) Pycharm. [8]

Pep8

Para comprobar la adhesión a los estándares sintácticos especificados por Python, se hizo uso del programa Pep8 [9].

PyInstaller

La compilación de los scripts que conforman los submódulos se llevó a cabo en una máquina destinada a ese fin, con el sistema operativo CentOS 6.7, y utilizando el programa PyInstaller [10] para la obtención de los ejecutables.

2.4. Proceso de desarrollo y resultados

2.4.1. Elección y familiarización con los switches de trabajo.

Existe un gran número de tipos de assets los cuales son susceptibles de contar con un módulo de conocimiento propio para explotar todo el potencial de gestión que ofrece el portal de Work On Data.

Es indudable la gran presencia de equipos de red Cisco en tareas de encaminamiento y enrutamiento [11] en entornos que tengan como objetivo garantizar la conectividad de forma segura de equipos de red. Esto hace que la inclusión en la herramienta de gestión Work On Data de los elementos que comercializan tenga prioridad, ya que serán numerosos los clientes que podrán beneficiarse de dicha incorporación.

Este factor, junto con el hecho de la existencia de switches CiscoCatalyst entre los pendientes por abordar en el proyecto, llevaron a la decisión de trabajar con ellos.

El primer paso a llevar a cabo tras la selección del módulo a desarrollar, fue conocer más a fondo los assets de trabajo. Inicialmente, un aspecto relevante fue averiguar la versión y el modelo. Esto se consigue fácilmente estableciendo conexión remota con ellos y usando el CLI (Command Line Interface o Interfaz de Línea de Comandos) para ejecutar el comando "show version". La siguiente tabla muestra para cada switch, su versión del software y modelo.

Nombre del switch (asset)	Versión SO	Modelo
sh130netpsw01a	C3560E Software 12.2(55) SE3	WS-C3560X-24
sh130netpsw01b	C3560E Software 12.2(55) SE3	WS-C3560X-24
sh130netmsw01a	C3560E Software 12.2(55) SE3	WS-C3560X-24
sh130netcsw01	C3750E Software 12.2(58) SE3	WS-C3750X-48
sh130netcsw02	C3560E Software 12.2(58) SE2	WS-C3750X-48

Tabla 1 – Versión y modelo de los switches de trabajo

Distinto hardware y distinto software pueden implican ciertos cambios en la existencia de los comandos y del formato de su resultado. Si el resultado obtenido y recibido en el satélite tiene un formato diferente al que se usó para desarrollar los scripts de Python que analizan dicho resultado, puede que al ser tratado se obtenga información extraña o que se almacene como valor de un parámetro con el cual no se corresponda (se explicará el funcionamiento de los scripts de Python más adelante.)

El módulo de conocimiento CiscoCatalyst es compatible (al menos) con aquellos switches con versiones del SO (Sistema Operativo) C3560E 12.2(55)SE3 y C3750E 12.2(58)SE2.

2.4.2. Especificación del modelo de información no formal de los switches

Una vez caracterizado el switch con su versión de Sistema Operativo y modelo, se procede a identificar la información que se puede y desea obtener del switch. Esto se denomina modelo de información.
En este caso no se ha considerado necesario generar tal modelo de información en ningún formato concreto (como UML, por ejemplo), ya que no se necesita especificar ni el tipo de los parámetros (son todo cadenas obtenidas del análisis sintáctico – *parse,* de la información devuelta por la ejecución de un determinado comando en el switch), ni establecer la relación entre dichos parámetros. He ahí la razón por la cual se le ha denominado modelo de información "no formal".
Sin embargo, el modelo de datos, que define la estructura y las restricciones de los datos que se almacenan, sí tiene un formato concreto, que viene impuesto por la base de datos data, y es JSON.

Para la especificación del modelo de información de los switches de trabajo, se elaboró una lista con los comandos que ofrecen información relevante de estado y configuración. Se contrastó además la existencia de dichos comandos en ambos modelos de switches (3560 y 3750), y si existía alguna diferencia en el formato de la salida. En tal caso se anotaron tales diferencias para tenerlas en cuenta posteriormente en el desarrollo de los scripts.

Una vez hecho esto, se procedió a consultar con expertos de la administración de equipos de red dentro de Nologin si el modelo de información incluía los aspectos con los que más tratan en su día a día, ya que el objetivo final del proyecto Work On Data consiste en poder ofrecer información que resulte útil a la hora de gestionar los equipos. Tras su aprobación, se estableció de forma concreta la estructura, organización, y contenido de los submódulos (con los nombres en inglés, ya que es el lenguaje por defecto de trabajo). En la Tabla 2 siguiente se muestran los submódulos y cada una de las secciones que los componen, además de los comandos usados en cada sección para obtener la información.

Nombre del submódulo	Secciones	Comandos
General	CDP	show cdp neighbors detail
	General Details	show version
	Licenses	show license detail
Ip	Arp	show ip arp
	Portmap	show ip port-map
	Route	show ip route
	Sockets	show ip sockets
Ports	Port Details	show interfaces show mac address-table dynamic
Security	Certificates	show crypto pki certificates
	Public Keys	show crypto key mypubkey rsa
Vlan	Vlan Details	show vlan brief

Tabla 2 – Submódulos, secciones y comandos

Submódulo General

Este submódulo aporta información de carácter general, y sobre aspectos que contribuyen a obtener una visión general del el switch y los temas que le afectan.

- Sección CDP: el acrónimo CDP (Cisco Discovery Protocol, o protocolo de descubrimiento de Cisco) [12] es un protocolo de red propietario de nivel 2, desarrollado por Cisco Systems y que utilizan la mayoría de sus equipos para compartir información sobre otros equipos Cisco directamente conectados, tal como la versión del sistema operativo y la dirección IP, mediante la difusión de anuncios a una dirección de destino. En esta sección se muestra información relativa a los "vecinos" del switch *target*.

- Sección General Details: información relativa a versiones del Sistema Operativo, el procesador, la memoria, los interfaces, el modelo etc.

- Sección Licenses: las licencias con las que cuenta el switch en cuestión. Éstas permiten acceder a determinadas prestaciones especiales en el switch después de instalarlas [13].

Ip

Los switches, pese a que operan principalmente a nivel 2 del modelo OSI (Open System Interconnection), algunos pueden ofrecer funcionalidades de capa 3. Este es el caso de los switches de trabajado, por tanto cobra sentido incluir este submódulo.

- Sección ARP: presenta la tabla ARP (Address Resolution Protocol) que tiene almacenado el switch.

- Sección Portmap: presenta información sobre los mapeos entre puerto y aplicación y el tipo de mapeo, es decir, presenta la configuración del PAM (Port to Application Mapping) [14].

- Sección Route: esta sección, aunque inicialmente tenía como cometido obtener la tabla de rutas almacenada en el switch y el *gateway* por defecto, debido a que los switches con los que se ha trabajado no contienen ninguna tabla, no se ha podido terminar esta implementación. Por tanto, de esta sección únicamente se obtiene el gateway por defecto.

- Sección Sockets: presenta información sobre los *sockets*. Para cada protocolo que tiene un socket activo, indica la dirección local y remota, el puerto local y remoto.

Ports

- Sección Port Details: ofrece información relativa a todos los interfaces (puertos, vlans, y port-channels), como la MAC (*Media Access Control*), el tipo de transmisión, medio, velocidad...

Security

Los switches, aunque operen mayormente en un ámbito local, requieren de medios para garantizar la seguridad y la autenticidad del contenido que manejan. Para ello, el dispositivo cuenta con una PKI (*Public Key Infrastructure*), lo que conlleva la generación de certificados y claves públicas.

- Sección Certificates: muestra información sobre todos los certificados propios y recibidos almacenados en el switch.

- Sección Public Keys: muestra las claves públicas almacenadas y otros aspectos como el tipo, la fecha de generación, el uso, ubicación de almacenamiento...

Vlan

- Vlan Details: información acerca de las Vlans existentes y otro tipo de información distinto del mostrado por la sección Port Details.

2.4.3. Desarrollo de los scripts de los submódulos

En este apartado se presenta la estructura y los aspectos más relevantes de los scripts desarrollados para el módulo de conocimiento CiscoCatalyst.

La estructura de los scripts que componen cada submódulo de conocimiento es similar para todos. Cada submódulo, para estar completo y generar el ejecutable final correctamente, debe contener:

- Un script "CiscoCatalyst.py". Este es completamente idéntico y común a todos los submódulos.

- Un script get<nombre_del_submódulo>.py . Tiene la misma estructura en todos los submódulos pero parámetros con valor distinto según el submódulo.

- Un script por sección llamado get<nombre_de_la_sección>.py con al menos una función, llamada check<nombre_de_la_sección>.

Script CiscoCatalyst.py

En el anexo A.3 se puede ver el código de este script.

En él se define la clase CiscoCatalyst, cuyos métodos son un constructor, un destructor, y un gestor de comandos, que se encarga de enviar comandos al switch, y devolver el contenido de su respuesta.

El elemento principal de este script es el módulo Pexpect de Python, concretamente, el objeto "pxssh". Como ya se explicó, Pexpect permite automatizar tareas interactivas. En este caso, la tarea es crear un canal SSH entre el satélite

(donde se ejecuta este script) y el switch, sin que sea necesario la supervisión de un humano (es decir, que se automático). Por eso importamos el objeto "pxssh" de entre todos los del módulo Pexpect.

A continuación se comentan ciertos aspectos interesantes sobre el contenido del script (disponible en el anexo A.3.).

- Para el proceso de conexión del satélite (quien ejecuta el submódulo, o host) con el switch (target) se requieren unos credenciales, "user" y "password", que son argumentos que toma el módulo de conocimiento. Estos se obtienen del diccionario "parameters" albergado en la base de datos assets, en el documento del target.

  En la siguiente imagen se muestra la colección de uno de los assets de trabajo, el "sh130netpsw01a", tal y como se ve a través Robomongo (interfaz gráfico que integra el motor de base de datos MongoDB). En la figura 10 se observa el diccionario (objeto) "parameters", que contiene 5 elementos de tipo *array* (uno por cada submódulo). Cada índice de cada array contiene o bien el nombre o el valor del parámetro (4 elementos en total), en el orden en el que se tienen que ejecutar.

Key	Value	Type
(1) ObjectId("57c800cfe9e2b48a01a5797d")	{ 10 fields }	Object
_id	ObjectId("57c800cfe9e2b48a01a5797d")	ObjectId
module	{ 7 fields }	Object
name	CiscoCatalyst	String
parameters	{ 5 fields }	Object
general	[4 elements]	Array
[0]	--user	String
[1]	admin	String
[2]	--password	String
[3]	somepass	String
ip	[4 elements]	Array
ports	[4 elements]	Array
security	[4 elements]	Array
vlan	[4 elements]	Array
version	1.0r001	String
satellite	sh130	String
host	Satellite	String
target	sh130netpsw01a	String
schedules	{ 0 fields }	Object
startDate	2016-03-01 00:00:00.000Z	Date
status	4	Int32
service	full	String
support	24x7	String
lastMod	20161115134400	String
assetType	Network	String
assetSubType	NETWORK-MID	String
tags	[1 element]	Array

Figura 10 – Contenido de la colección del target "sh130netpsw01a" en la BD Assets

- En el constructor de la clase, una vez se realiza la conexión con el switch, se le envía el comando "terminal length 0". En caso de no mandarlo, el contenido "mostrado" por el switch, y por tanto el que se recibiría en la sección en cuestión, sería de unas pocas líneas (la información en los *shell* se muestra por defecto paginada, y no toda de una vez). Esto significaría que para obtener la secuencia completa de la salida habría que gestionar el envío del comando "more" (barra espaciadora). Además, esto causaría la inserción de caracteres no imprimibles entre la salida, la cual dificultaría la tarea de análisis sintáctico. Con "terminal length 0" se evita todo esto y se obtiene en un solo envío el resultado de la ejecución del comando.

- En cuanto al "prompt", el carácter o conjunto de caracteres que se utiliza para delimitar el final del contenido devuelto por el comando y que marca la espera de uno nuevo, se consideran dos casos, que sea "#" o ">" (el nombre del switch seguido de uno de los dos caracteres). El primero indica que el modo de ejecución de los comandos es privilegiado, y el segundo, en modo usuario. Para asegurar el poder tener permisos para ejecutar todos los comandos, si se recibe un prompt con ">", se envía el comando "*enable*", que permite pasar a modo privilegiado, tras introducir la contraseña ("Password:") correcta.

- El método que gestiona los comandos ("cmd"), funciona de la siguiente manera. Primero se envía al switch el argumento pasado al método, y tras detectar la secuencia del prompt (nombre del switch + #) que marca el fin del contenido del comando y la espera de recibir uno nuevo, el método devuelve todo lo recibido hasta detectar el prompt (incluído), exceptuando la primera y la última línea, que siempre contienen el comando enviado y el prompt final, respectivamente. Es decir, devuelve únicamente el contenido "útil" del resultado del comando.

- Es importante quedar a la espera del nombre del switch + #, y no únicamente del carácter "#", ya que se puede dar en algún caso (y así ocurre) en el que este caracter aparece en el contenido devuelto por el switch, lo cual truncaría la información, ya que sólo se retornaría el contenido hasta el caracter "#" y se omitiría todo lo que fuera después.

Script get<nombre_del_submódulo>.py

En el anexo A.4 se encuentra el script getGeneral.py a modo de ejemplo.

- La estructura de este script es idéntico al de los demás submódulos, y en general, de todos los módulos de conocimiento.

- Lo único que se ha modificado con respecto a los otros, son los nombres de todas aquella variables y funciones que hagan referencia al submódulo, (por ejemplo, "getGeneral", "GeneralDictionary"), o a las secciones, el valor de la clave "submodule" del diccionario, el cual es esencial para estructurar la base de datos correctamente, los nombres de las funciones check<nombre_de_la_sección>, y además, la inicialización del objeto CiscoCatalyst (la instancia remoteSwitch).

- El resto consiste en el método *main* y el tratamiento de los argumentos y los *threads*. También se hacen las modificaciones necesarias para que el diccionario devuelto cumpla estrictamente el formato JSON. Esta implementación ya venía impuesta, ya que es la misma para todos los módulos.

Script get<nombre_de_la_sección>.py

En el anexo A.5 se encuentra a modo de ejemplo el script getGeneralDetails.py

En estos scripts se manipula la información devuelta por los comandos envíados al switch y se construye el modelo de datos.

Para el desarrollo de estos scripts se han tenido en consideración los siguientes requisitos impuestos por distintos factores:

- El formato final de la información recopilada (el modelo de datos) debe ser en formato JSON. Esto viene impuesto por la base de datos no relacional empleada y la manera en la que organiza la información en colecciones.

- La jerarquía de la información en el modelo de datos debe estar compuesta por listas de diccionarios (*arrays* de diccionarios), exceptuando el primer nivel de la jerarquía (el diccionario del submódulo), el cual contendrá dos diccionarios: "_id", con parámetros identificativos de la colección y "data", que alberga la información en sí de un determinado submódulo.
  Esta organización viene impuesta por la máquina de búsqueda del portal web, que necesita esta estructura para buscar según distintos campos.

En la figura 11 se muestra el diccionario del submódulo General con la finalidad aclarar los elementos comentados.

Figura 11 – Vista en Robomongo del diccionario General del asset "sh130netpsw01a"

- Todo diccionario contenido en una lista tendrá obligatoriamente la clave "_parent" (al mismo nivel que el resto de las claves del diccionario), que tendrá el valor del elemento un nivel superior en la jerarquía.
  Por ejemplo, si se observa el contenido que devuelve el submódulo General, (Figura 11) es un diccionario que contiene 2 diccionarios (permitido por ser el primer nivel de la jerarquía): "_id" y "data". El primero contiene 6 claves con sus respectivos valores (pareja clave-valor o *key-value pair* en inglés), mientras que "data" contiene tres listas: "licenses" y "cdp" y "general". Las listas "cdp" y "general" solo contienen un diccionario (un índice), y por lo tanto son de tamaño uno, mientras que "licenses" contiene dos. Ambos contienen las mismas claves, pero algunos o todos sus valores pueden ser distintos. En la figura 11 se observa que el valor para la clave "licenses_feature", por ejemplo, es distinto, mientas que para "licenses_version" es el mismo.

- Ninguna clave puede contener caracteres con significado reservado en HTML. Es decir: '<', '>', '#', '.' y ':'.

- Claves con mismo nombre deberán contener siempre valores del mismo tipo en todos los escenarios posibles.

El procedimiento general para el desarrollo de los scripts de cada sección, con nombre get<nombre_de_la_sección>.py es:

- Cada sección cuenta con un diccionario. En él se almacena toda la información obtenida tras el análisis sintáctico de la respuesta del switch al comando enviado. Para gestionar el envío de dichos comandos se crea una instancia de la clase CiscoCatalyst. Éste es el denominado en el script "remoteSwitch".

- La información que se recibe del switch a través del método "cmd" del objeto "remoteSwitch" es un array de cadenas (*strings*). Éste se "lee" línea a línea, y se analiza su contenido, almacenando en el diccionario aquellos elementos que se desean guardar. De este modo se va rellenando el diccionario con pares clave-valor, o creando listas de diccionarios si la misma información se repite con valores distintos.

2.4.4. Resultado del módulo de conocimiento

El objetivo de la ejecución de los submódulos que globalmente conforman el módulo de conocimiento, es obtener toda la información especificada en los scripts, o en su defecto, la máxima posible. Debido a fallos en la comunicación con el switch, o de errores a la hora de almacenar la información recibida por éste, los scripts están programados de tal forma que el resultado final de la ejecución del submódulo cumpla siempre el formato JSON (por ejemplo, un diccionario vacío, en caso de no poder establecerse la comunicación).

Dejando al margen las situaciones en las que, por diversas razones, se obtenga como resultado un diccionario vacío o con campos incompletos, la ejecución de cada submódulo generará en el satélite un documento en formato JSON compuesto por todas secciones. En el anexo A.6 se encuentra el resultado de la ejecución del submódulo General para un asset determinado.

Capítulo 3

PRESENTACIÓN EN DASHBOARD

La obtención de información de estado y configuración de switches CiscoCatalyst tiene objetivo final el poder presentarlos a los clientes de manera visual y atractiva. Esto se lleva a cabo mediante los diversos dashboards accesible en la página web del proyecto.

En este capítulo se abordan todas aquellas cuestiones que giran en torno al dashboard y al trabajo realizado.

3.1. Conceptos y elementos básicos de tecnologías web

La arquitectura sobre la cual se basa en su mayoría el intercambio de datos en la *World Wide Web* (WWW) es la de cliente – servidor. El cliente es el navegador web que muestra la página web al usuario, y el servidor, aquel que almacena tales contenidos y los "sirve" al cliente. La comunicación entre estos dos entes es posible gracias al protocolo de nivel de aplicación HTTP (HyperText Transfer Protocol), que se basa en el envío de comandos y respuestas en texto ASCII.

Estos dos elementos tan claramente diferenciados de la arquitectura desarrollan sus tareas mediante lenguajes distintos. Por ello, se habla de tecnologías del lado del cliente y del lado del servidor, según el ente que las ejecute o interprete. En el caso del cliente estos son principalmente HTML junto con otros lenguajes que aportan dinamismo e interactividad a la página web, como Javascript o CSS. En el servidor, los lenguajes utilizados son múltiples, por ejemplo C, Perl, PHP o Python, entre otros.

El intercambio de datos en la web consiste pues en un cliente que solicita páginas web y elementos concretos dentro de ésta al servidor. Éste a su vez, debe ser capaz tanto de gestionar las peticiones como de proveer al cliente esos contenidos. También debe ser capaz de interactuar con bases de datos u otros elementos que almacenen datos consultados por el servidor.

Todo desarrollo o sistema de alta complejidad que se precie se basa en abstracciones sobre las que se estructuran las distintas funcionalidades y elementos

presentes en él. En el caso del diseño de aplicaciones web, se construyen entorno al patrón de diseño "Modelo-Vista-Controlador", MVC (*Model-View-Controler* en inglés) [15]. Los componentes del modelo son:

- Modelo: se refiere a la lógica relacionada con los datos con los que trabaja el cliente. Provee un nivel de abstracción con la base de datos, sin necesidad de conocer los entresijos de su funcionamiento.

- Vista: es lo que se ve en el navegador del cliente. Sirve también como interfaz de entrada para la recogida de datos del cliente. Se encarga de toda la lógica entorno al interfaz del usuario.

- Controlador: ejerce control entre el modelo y la vista. Manipula los datos usando el Modelo e interactúa con las Vistas para ofrecer el resultado final.

La creación de una página web cuenta con una serie de pasos, los cuales son comunes a todas. Por ejemplo, encapsular la información en paquetes HTTP, establecer equivalencias entre URL y vista, autenticar a un usuario que intenta registrarse en su cuenta, conectarse a bases de datos para obtener determinada información… Sería demasiado ineficiente y costoso que cada programador tuviera que programar desde cero todas estas funcionalidades. Con el fin de solventar este problema existen frameworks como Django [16], que consiste en un conjunto de librerías con funciones que facilitan la creación de páginas web complejas. Todas las partes del framework están programadas en Python.

Todo el back-end del proyecto está programado en Python usando librerías Django.

Django sigue el modelo MVC, pero utilizando otra lógica de implementación. Así, el modelo MVC se redefine en Django como MTV, "Model Template View" (Modelo, Plantilla, Vista):

- Modelo: igual que en MVC, es decir, todo aquello sobre cómo acceder a los datos, validarlos, establecer relaciones entre ellos…

- *Template* (plantilla): se corresponde con la Vista. Establece la forma de visualizar el contenido. Las plantillas están escritas en HTML y también pueden llevar "etiquetas y filtros Django" (Django *tags and filters*) integrados entre el código HTML.
  Los templates sirven para aislar la presentación de la página web de sus datos.

- Vista: establece la lógica de acceso al modelo y remite a los templates apropiados. Sirve como enlace entre el modelo y los templates. Equivalente al Controlador

A continuación se procede a comentar de qué manera se incorpora el patrón de diseño MTV en el desarrollo del portal web de Work On Data.

Modelo

El modelo está definido mediante varios scripts programados en Python, usando distintas librerías, y la API "Pymongo". Ésta contiene una serie de herramientas para interactuar con las bases de datos MongoDB.

Estos scripts se encargan de tareas relacionadas con la conexión a las bases de datos, en un puerto y a una colección concreta, con un usuario y contraseña correspondientes y que deben ser autenticados, funciones para hacer operaciones sobre la base de datos (obtener una información concreta, filtrar assets según un cierto criterio…), realizar operaciones sobre elementos del módulo de conocimiento (obtener información sobre un módulo de conocimiento concreto, convertir un diccionario en una tabla...), aplicar operaciones de encriptado sobre la información (apoyándose de librerías ya creadas con ese fin), definir operaciones relativas a la cuenta del usuario (cambiar la contraseña, obtener sus proyectos), etc.

Templates

Los templates están escritos en HTML, Django, y algo de Javascript (aunque la mayoría del código Javascript está definido en un script propio con extensión .js). En ellos está definida la estructura del contenido de la página web. Como los datos obtenidos por el módulo de conocimiento forman parte del contenido de la página, y por tanto no son constantes definidas en el momento de creación de los templates, es necesario que esas variables estén marcadas con *tags* (etiquetas) Django. Esto requiere pues obtener los valores concretos antes de enviar el template al cliente.

Además, Django provee a HTML de funcionalidades de las que carece, como el uso de sentencias condicionales e iterativas. Lo primero es útil en casos en los que se desee mostrar un elemento de una forma en caso de que una variable tenga un valor, y de otra en caso de que tenga otro valor. Lo segundo permite repetir la misma estructura para varios elementos, sin tenerla que definir dos veces manualmente.

Otra funcionalidad de Django aplicada a los templates son los *filters* (filtros), que permiten modificar o hacer comprobaciones predeterminadas o personalizadas (*custom filters*) sobre variables (por ejemplo, convertir la primera letra de un string a mayúscula con el filtro "capFirst").

Vistas

Las vistas en Django se componen de dos scripts diferentes: "views.py" y "urls.py".

La función de "views.py" es la encargada de recibir un HTTP *Request* (petición) y devolver al cliente un HTTP *Response* (respuesta), idealmente con el contenido solicitado, o sino una redirección o un código de error. Para ello se apoya de varias librerías Django, entre ellas la de "http".

Para poder enviar el mensaje de respuesta, se hace uso de las distintas funciones definidas por el Modelo para obtener los datos concretos necesarios para insertarlos en la Vista correspondiente. Por ejemplo, para devolver la vista del menú principal tras introducir un usuario y contraseña correctos. Esto se denomina obtener el "contexto", es decir, especificar los valores concretos de las variables definidas en el template en cuestión, accediendo a bases de datos (a través del Modelo) en caso necesario. Una vez la vista pasa el contexto, llama al método *render()* sobre el template determinado y con el contexto adecuado. Esto devuelve un template en formato cadena (*string*) con las variables y los template tags evaluados conforme al contexto.

El script "urls.py" es fundamental, ya que establece la correspondencia entre una URL (que sirve de localizador de una página web) y las funciones de la Vista que se deben llamar cuando se pide dicha URL. Esto se consigue definiendo tal correspondencia en este fichero mediante expresiones regulares. Así, cuando llega una petición a una URL concreta, no hay más que comprobar para cada una de las entradas definidas en el fichero si alguna coincide (*match*) con alguna expresión regular.

A continuación se muestra un esquema que engloba todos los elementos que conforman el MTV en el servidor y la relación con el cliente.

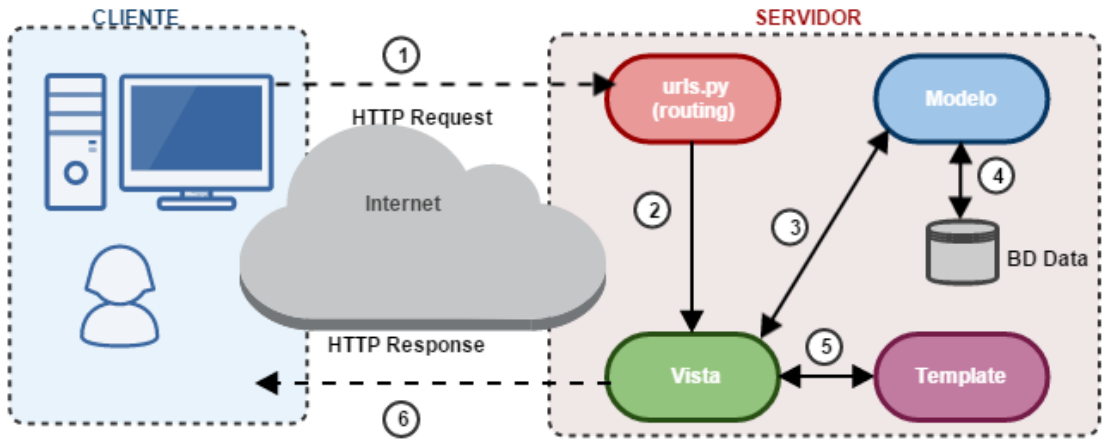

Figura 12 – Elementos del MTV y relación con la arquitectura cliente-servidor

3.2. Tecnologías y herramientas

En este apartado se comentan las tecnologías con las que se ha trabajado para el desarrollo de las vistas del dashboard.

Lado del servidor

Como ya se ha mencionado, los scripts del servidor están desarrollados en Python y Django.

Lado del cliente

HTML

HTML es la tecnología web indispensable para definir la estructura y posición concreta de los elementos que se desean mostrar dentro de cada vista de la página web.

Javascript

Javascript [17] es un lenguaje de scripting cuyo código se puede insertar en páginas HTML y de ejecución en el navegador, lo cual lo hace rápido en responder a las acciones del usuario. Javascript aporta dinamismo a la página web. Se ha utilizado por ejemplo, para lograr iconos desplegables, los cuales al hacer click sobre ellos, se expanden para mostrar información oculta, y con otro click más, se pliegan y vuelven a ocultarla.

En definitiva, su uso más común consiste en definir funciones que interactúan con los elementos definidos en el documento HTML, añadiendo, modificando, o eliminándolos.

JQuery

La biblioteca más utilizada de Javascript es JQuery, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM (*Document Object Model*)[18], manejar eventos, desarrollar animaciones y agregar interacción con el servidor gracias a la técnica AJAX [19].

Su sintaxis está definida de la forma: $(selector).acción(). El selector es aquello que se busca o sobre lo que recae la acción.

AJAX

AJAX (*Asynchronous JavaScript and XML*) [17] también contribuye a la creación de webs interactivas. Las aplicaciones se ejecutan en el cliente, pero pueden comunicarse asíncronamente con el servidor. Los datos se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. Así, el contenido de la página se actualiza sin necesidad de volver a cargarla.

CSS

Para describir el estilo (fuente del texto, colores, fondos…) de los elementos definidos en los templates HTML se ha usado CSS (Cascading Style Sheets). En los ficheros CSS se establecen una serie de reglas que describen la forma en la que se van a mostrar determinados "selectores" (los elementos en sí) mediante propiedades CSS (nombre: valor).

Bootstrap

Es un framework o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basados en HTML y CSS, así como extensiones de JavaScript opcionales adicionales [20].

Una de las muchas ventajas de usar los elementos de Bootstrap, es que son adaptativos (*responsive*), lo cual es una característica fundamental para obtener una correcta presentación de la página web en dispositivos de distinto tamaño.

Datatables

Es un *plug-in* para la librería JQuery de Javascript. Ofrece la posibilidad de aportar a las tablas un estilo distinto y con variedad de funcionalidades, como la búsqueda de elementos, la paginación de los contenidos en distintas "páginas", pudiendo especificar el número de entradas en cada una, etc.

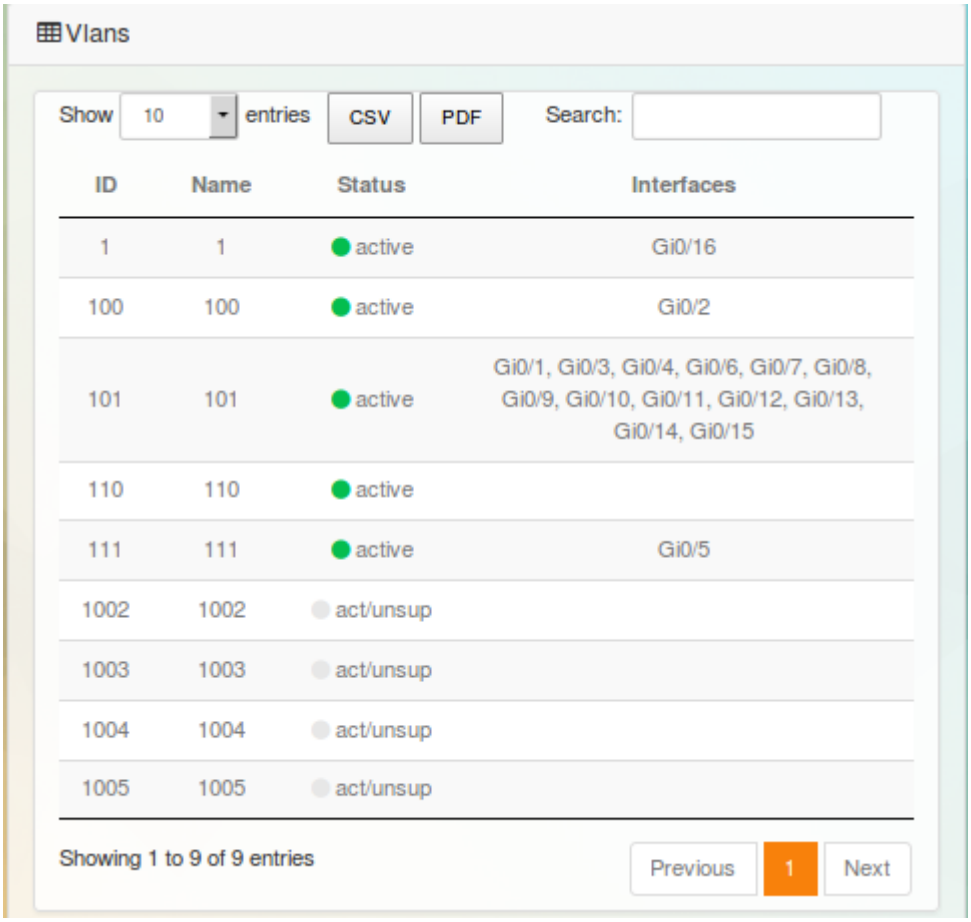

Figura 13 – Uso del plug-in Datatables en el submódulo Vlan.

Font Awesome Icons

Los iconos personalizables de todo tipo de categorías ofrecidos por este *toolkit* de CSS se incluyen en el código HTML fácilmente. Así se consigue enriquecer y hacer más expresiva la información visualizada.

Los iconos verdes y grises de la figura 13 (la de arriba) pertenecen a este toolkit.

Robomongo

Robomongo es la interfaz gráfica para el manejo del motor de base de datos MongoDB. Facilita la tarea de conexión con las bases de datos, y el acceso a las colecciones que contienen. Cuenta con un buscador que permite establecer diferentes reglas a seguir al ejecutar la búsqueda.

Esta herramienta es muy útil para comprobar la correcta población de las bases de datos con el resultado de los módulos de conocimiento y obtener de un vistazo toda la información necesaria.

3.3. El portal web

La herramienta final presentada al cliente es el portal web con URL www.workondata.com .

En este apartado se presenta el portal web y los dashboards (panel de control o cuadro de mando), la navegación por las ventanas principales, y la relación de éstas con los templates en los que se definen. Esto permitirá contextualizar y delimitar el trabajo propio desarrollado, abordado en el apartado 3.5.

En la siguiente figura 14 se puede ver un esquema de la navegación básica de las ventanas del portal.

Figura 14 – Esquema de navegación de las ventanas principales del portal

La navegación comienza naturalmente accediendo a la página principal del portal web en la URL mencionada (1). Ahí se encuentra la pestaña de *login* (ingreso).

Presionando sobre ella se llega a la ventana con la plantilla del login (2), en la que se deben introducir los credenciales (usuario y contraseña). Estos deberán ser

validados por el servidor, quien a su vez consultará con el servidor LDAP (este proceso se verá en el apartado siguiente).

Una vez estos datos son validados, se accede a la página principal del dashboard (3) en la que se da la bienvenida al portal. Cuenta con una barra superior, que se mantiene en el resto de ventanas del dashboard, en la que se encuentran el logo del proyecto que se está visualizando, los distintos servicios del dashboard que han sido contratados para dicho proyecto (monitorización, infraestructura, documentación…), y otras pestañas para la gestión de la cuenta, cambiar de proyecto y de idioma, entre otros.
Como se comentó en apartados anteriores, cada cliente tiene acceso únicamente a aquellos proyectos en los que tiene asignados permisos (que figuran en el servidor de LDAP, como se verá más tarde).

Ya que el trabajo realizado ha sido en el servicio de infraestructura, se van a omitir las ventanas que tienen que ver con los demás.

Si se presiona sobre el icono de infraestructura, se llega a su ventana principal (4), en la que se muestra un resumen de los assets gestionados y un buscador que filtra los assets según los parámetros indicados.

Se selecciona uno de los assets disponibles, y esto redirige al dashboard del mismo, en el cual se visualiza la información obtenida por el módulo de conocimiento establecido para dicho asset (5). El contenido está organizado por submódulos, seleccionables mediante un panel lateral con el nombre de cada uno de ellos y de sus secciones. Es en el desarrollo de estas ventanas para assets CiscoCatalyst en lo que se ha colaborado en este TFG. Esto se abordará en el apartado 3.5.

3.4. Infraestructura

La infraestructura consta de dos partes claramente diferenciadas: la del cliente (*front-end*), gestionada por el navegador web, y la del servidor (*back-end*), que se encarga de obtener todos los elementos solicitados por el cliente. Ambos se comunican usando peticiones y respuestas HTTP(S), como se ha explicado en la sección 3.1.

A continuación se presenta un esquema con las máquinas y las tecnologías que intervienen en el proceso de navegación en el portal.
Este es un esquema simplificado, que obvia elementos también relevantes como la consulta de servidores DNS, intercambio de certificados, etc, pero que exceden el alcance de este trabajo.

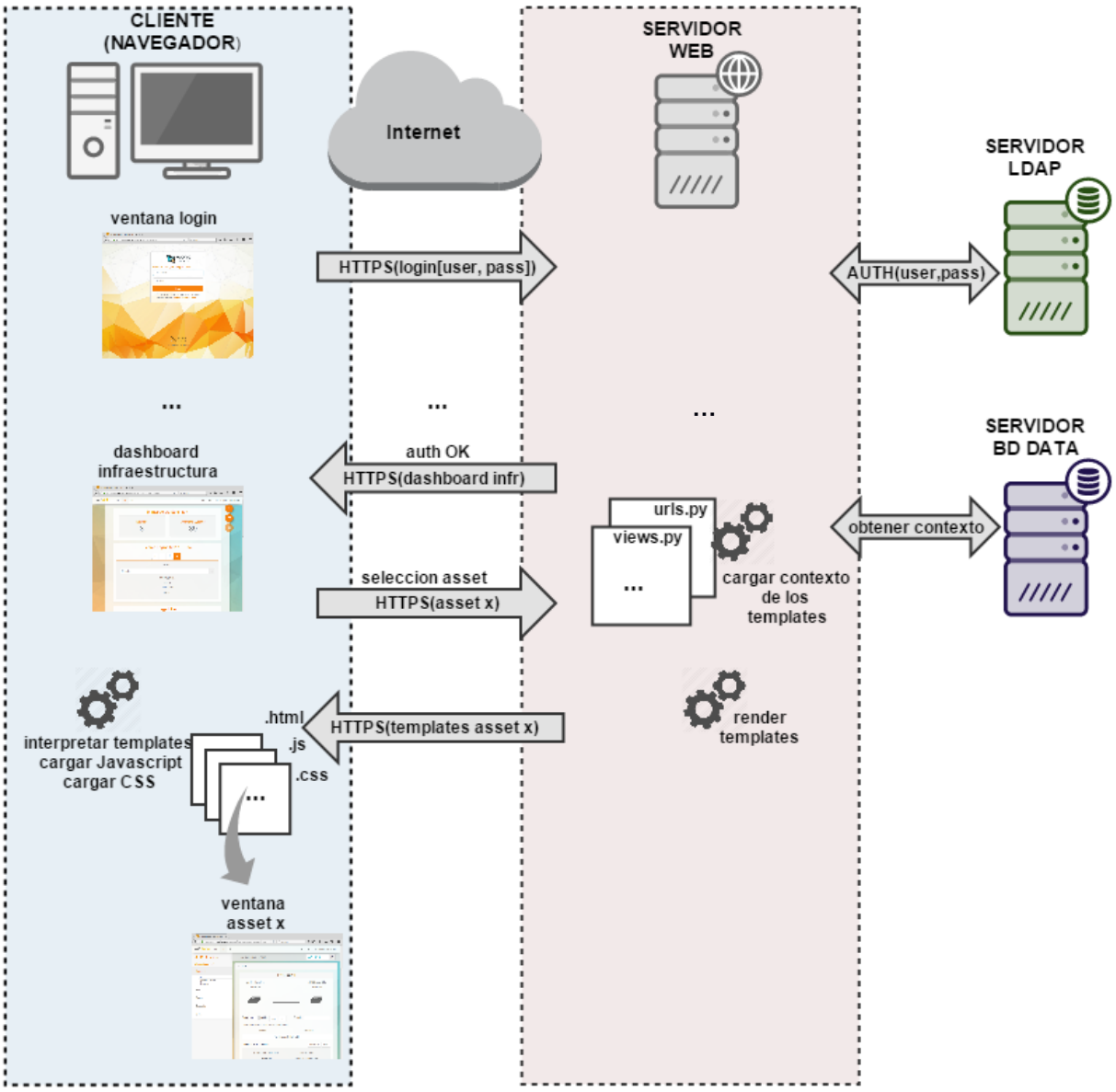

Figura 15 – Diagrama de interacción de distintos elementos en durante la navegación en el portal, en su apartado de infraestructura.

A continuación se procede a explicar de manera más detallada este proceso (aunque en la figura no aparece esquematizado el intercambio de comandos con AJAX, ni la petición de la ventana principal del portal, por simplicidad).

Una vez en la página de inicio, se requiere insertar los credenciales, esencial para obtener acceso al contenido del proyecto propio de cada cliente. El servidor web procesa la petición y accede a la base de datos LDAP [21] en el servidor correspondiente, quien se encarga de comprobar la autorización (a qué colecciones tiene acceso el usuario) y la autenticación (del usuario y la contraseña) introducidos.

Una vez se tiene acceso a la cuenta propia y al proyecto autorizado, si el objetivo es visualizar el dashboard de un asset concreto, se envían peticiones HTTP(S) al servidor web, el cual devuelve en cada caso los templates solicitados. Como se explicó en la sección 3.1, esto requiere del control de la Vista, quien gestiona este proceso. Además, en caso de mostrar los contenidos de los módulos de conocimiento, o saber qué servicios mostrar en la barra superior (que depende del tipo de contrato del cliente), se realizan accesos a las bases de datos correspondientes a través de la abstracción proporcionada por el Modelo.

El servidor web es el encargado de "presentar" (*render*) los templates HTML que están almacenados en él al cliente, tras evaluar el código Django incrustado entre el HTML.

Una vez se carga e interpreta la respuesta del servidor en el cliente, también se ejecuta el código Javascript y las hojas de estilo CSS. Las interacciones dinámicas corren a cargo de Javascript, sin necesidad de interactuar con el servidor web para ello.

Otro momento en el que se necesita de la interacción con el servidor web es tras seleccionar una fecha distinta de los datos mostrados, mediante un calendario desplegable habilitado para ello. Esto requiere consultar la base de datos Data y actualizar los valores. Para hacer esto de manera más eficiente se usa la técnica AJAX. Así, se logra modificar el contenido de la página sin necesidad de cargarla de nuevo completamente, ya que la estructura de la información mostrada es la misma (o en su mayoría) y varían el valor de las variables que se consultan en la base de datos.

3.5. Desarrollo de templates y scripts

En el anexo B.1. se encuentran las imágenes de las ventanas del dashboard desarrollados en este TFG.

Como se mencionó en el apartado 3.3, el trabajo asignado en esta segunda parte del TFG consiste en desarrollar las ventanas del dashboard que muestran la información obtenida previamente por el módulo de conocimiento CiscoCatalyst.

Previo a presentar los scripts desarrollados, conviene aclarar primero el proceso que permite a los script HTML tener acceso al valor de las variables, que como se ha explicó en el apartado 3.1., están definidas en lenguaje Django entre el código HTML.
El Modelo (dentro de la arquitectura MTV) en el módulo de conocimiento CiscoCatalyst, está implementado en un script Python con el mismo nombre

(CiscoCatalyst.py – que es totalmente independiente del script llamado igual presente en el propio módulo de conocimiento). Éste define una clase, también con el mismo nombre, y unos métodos llamados get<nombre-del-submódulo>Context(), uno para cada submódulo. Estos métodos se encargan de comunicarse con la base de datos, y obtener los datos que conforman el contexto, para finalmente procesarlo (*renderizarlo*) al script determinado (el que contenga tales variables).

3.5.1. Ficheros predefinidos

Para contextualizar el trabajo desarrollado con los templates, hay que presentar en primer lugar los ficheros ya existentes sobre los que se han construido los propios.
La vista del dashboard para el servicio de infraestructura de un asset determinado está compuesta por los siguientes templates predefinidos e idénticos para todos los módulos de conocimiento.

- "base.html": es el esqueleto principal de la ventana. Define la barra superior en la que aparece el logo de la empresa, los servicios, y el contenido de los iconos desplegables.
  Los demás templates lo "extienden", es decir, construyen sobre lo definido en él.

- "asset.html": extiende el template base.html. Define la barra lateral en la que se presentan los enlaces a cada submódulo, con el nombre del asset, del módulo y su versión.

- filters.html: define la barra gris en la que aparece la fecha de la última actualización (de los datos) y en el extremo derecho, el calendario que permite seleccionar la fecha de recogida de los datos visualizados.

Sobre este último, filters.html, se extiende cada uno de los templates que llevan el nombre del submódulo (por ejemplo, "general.html"), que se activan dinámicamente según se haga click sobre uno u otro submódulo en el panel lateral. Por defecto, al cargar la vista del asset, se carga el primer submódulo de la lista del panel lateral (en este caso general).

En la figura 16 se puede ver la ventana del submódulo general de un asset CiscoCatalyst y su composición de ficheros html.

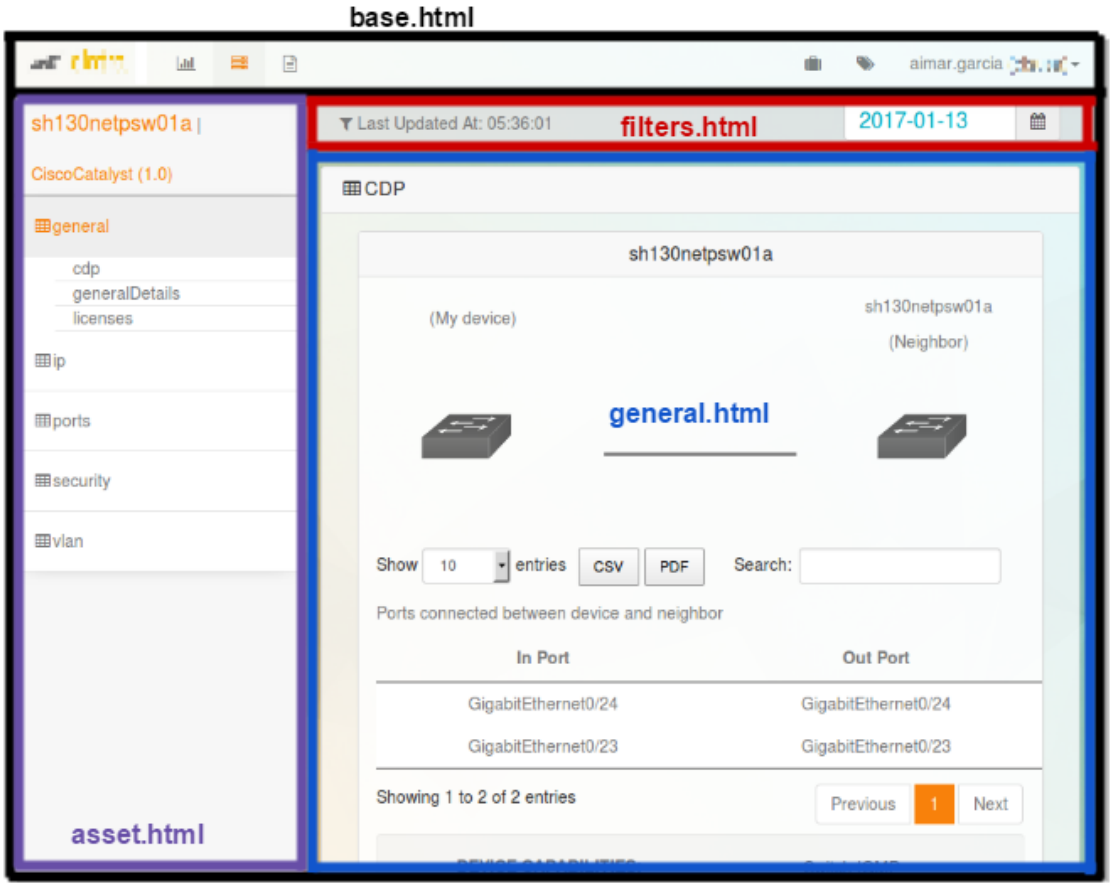

Figura 16 – Descomposición en ficheros .html de la ventana del submódulo general de un asset

3.5.2 Ficheros editados y desarrollados

A partir de esta estructura previa, se han desarrollado los siguientes ficheros o scripts, cuyo contenido define la estructura que debe tomar la información obtenida por el módulo de conocimiento.

La estructura de estos ficheros está organizada de manera modular. A continuación se presentan los templates y scripts creados o editados para la obtención del resultado final, el dashboard de los assets CiscoCatalyst.

Template <nombre-del-submódulo>.html

Éste es un template genérico para cada submódulo. En él se definen varias "divisiones", elementos "div" HTML, que establecen una división (un espacio) dentro del documento.

Una de ellas es obligatoriamente la sección identificada como "noDataSection". El propósito de esta sección es visualizarse en caso de que no se haya recogido información sobre el submódulo seleccionado. Esto puede ocurrir, por ejemplo, cuando para la fecha seleccionada no se recogieron datos, o no se han recogido todavía pero sí se hará más adelante durante el día (todo depende de la periodicidad de ejecución del submódulo).

El template genérico de cada submódulo también define tantas divisiones como secciones tenga el submódulo en cuestión. El contenido de esas secciones no está directamente definido en este template, sino en el template llamado <nombre-del-submódulo>-<nombre-de-la-sección>.html, con la finalidad de no "sobrecargar" el template con partes de distinto contenido. Asignándole un template diferente a cada sección hace la tarea de mantenimiento del código mucho más eficiente y simple.

Un ejemplo de código de este tipo (general.html) se puede encontrar en el anexo B.2.

Template <nombre-del-submódulo>-<nombre-de-la-sección>.html

Este template contiene la definición de la estructura de los elementos que deben aparecer en una sección concreta.

Aquí se hace uso de todas las etiquetas (*tags*) HTML necesarias para construir el esqueleto de la sección, además de etiquetas y filtros Django para hacer referencia a variables y hacer uso de elementos iterativos y condicionales.

Existen situaciones en las cuales varios elementos se desean representar de la misma manera (éste es el caso de la estructura de listas de diccionarios explicada). Por contener los mismos campos, se desea que todos estos elementos presenten una estructura idéntica. Por tanto, en lugar de definir varias veces la misma disposición de los elementos, se incluye otro template que contiene dicho esqueleto, dentro del actual. Con esto se evita "plagar" el código de líneas que definen lo mismo y dificultarían la comprensión y el mantenimiento global del template.
Este es el caso del template "general-licenses.html", que incluye el fichero "general-licenses-license.html" dos veces. En el anexo B.3. se puede revisar el contenido de ambos documentos.

Script Javascript <nombre-del-submódulo>.js

Cada submódulo cuenta con su propio fichero .js. En él se definen las instrucciones tanto en Javascript como con la librería Jquery y las llamadas AJAX. Una parte del contenido son funciones predefinidas comunes a todos los módulos de conocimiento, y el resto específicas a cada submódulo del módulo CiscoCatalyst.

Pese a que se puede incluir código Javascript entre las líneas de un template HTML, tal y como se puede apreciar en el código de los anexos B.2. y B.3., el sobrecargar el código HTML con muchas líneas de Javascript dificultaría enormemente la tarea de depurar y mantenerlo.
Además, separando el código Javascript en ficheros distintos según el submódulo sigue la línea de modularidad del resto de scripts y facilita el saber dónde mirar cuando algo en la vista de un submódulo no se está ejecutando correctamente.

Algunas de las funciones realizadas en los ficheros .js son:

- Controlar la activación del plug-in DataTables al desplegar un elemento con tabla, y la desactivación al ocultarlo. Con esto se consigue disminuir el tiempo inicial de carga de la vista, en aquellas vistas que contienen muchos elementos que siguen este patrón de funcionamiento.

- Los paneles de las secciones que están definidos con la clase HTML "hidden" (oculto). Una vez cargado el documento HTML en el navegador, pero justo antes de mostrarlo, se ejecutan unas líneas Javascript que comprueban la longitud del diccionario que contiene la información que se muestra en esa sección. Si no está vacío, entonces con una sentencia Jquery se elimina la clase "hidden" de dicha sección, para que sea mostrada por el navegador.
  Sólo en caso de que todas las secciones de un submódulo contengan la clase hidden, entonces se mostrará el "noDataPanel".

- Realizar las llamadas AJAX necesarias para cargar los nuevos datos que deben mostrarse tras seleccionar una nueva fecha en el panel y actualizar los datos.

En el Anexo B.4. se encuentra el fichero Javascript del submódulo general.

Script CSS <nombre-del-submódulo>.css

Al igual que ocurre con Javascript, los scripts CSS también se encuentran separados en distintos ficheros según el submódulo.

Pese a existir varios ficheros CSS genéricos que definen distintos estilos para elementos genéricos y que se repiten en varias ventanas (paneles, fuente y tamaño de la letra, color…), en cada vista concreta puede interesar sobrescribir alguno de esos estilos.

Cada fichero CSS define tantos apartados como secciones contenga el submódulo en cuestión, más un apartado para común para todas ellas.

CSS permite cambiar desde el tamaño y color de la fuente, del fondo, la posición de la letra u otros elementos, los márgenes, el relleno…

En el anexo B.5 se puede observar el fichero CSS definido para el submódulo general.

Capítulo 4

CONCLUSIONES Y LÍNEAS FUTURAS

4.1. Conclusiones

En este trabajo se ha presentado la contribución con el módulo de conocimiento CiscoCatalyst y el desarrollo de los correspondientes dashboards en la herramienta de gestión de assets, Work On Data.
Debido a que la aportación ha sido para el servicio de infraestructura (que muestra el estado y configuración de los assets), se han dejado de lado los demás: estado actual de solicitudes asociadas al proyecto (*ticketing*), estado en tiempo real de la monitorización, documentación interna, colección de informes e información contractual del proyecto.

El servicio de infraestructura (y también el de monitorización) ofrecido por la herramienta Work On Data tiene un fin común, siempre la de *visualización*, de forma centralizada, de diversos aspectos de los assets gestionados para el proyecto de cada cliente. Sin embargo, para no se permite la *modificación* de los parámetros gestionados.
El aspecto de modificación se delega a aquellas herramientas o métodos que cada asset cuenta ya para ese fin. Por ejemplo, para el caso de los switches CiscoCatalyst, se pueden modificar directamente aquellos parámetros de tipo *write* (escritura) mediante comandos definidos en su sistema operativo.

Esto puede constituir una ventaja, en la medida en que centraliza la información de los aspectos relacionados con cada proyecto, y cede la tarea de manipulación del estado a herramientas inherentes a cada equipo en cuestión. Esto es en principio siempre posible, ya que si no implicaría que el equipo estaría configurado por defecto y no podría ser modificable (*read-only*). En tal caso no se precisaría por tanto de herramientas para cambiar el estado de sus parámetros.

Este aspecto de no alterabilidad del estado de los parámetros gestionados constituye una clara diferencia con respecto a los protocolos de gestión de redes estudiados durante el Grado y dentro de la especialidad de Telemática, esencialmente SNMP y NETConf. Ambos definen en su modelo de comunicación primitivas para la modificación de la información gestionada. Por tanto, en ese aspecto, la aplicación Work On Data se asemeja más a Cacti, herramienta con la que

también se trabajó durante el Grado, y que ofrece métodos de recopilación de datos y generación de gráficas de los elementos de red monitorizados. Work On Data, además, permite la generación de informes automáticamente para mucha de la información mostrada, una herramienta para la gestión de incidencias o ticketing y un espacio para la documentación interna y contractual del proyecto.

En definitva, en el módulo de conocimiento CiscoCatalyst se ha desarrollado un "pseudoprotocolo" para la obtención del estado de parámetros determinados de los CiscoCatalyst. Éste se basa en el uso de una máquina (el satélite), la cual establece conectividad con el switch mediante SSH, implementado apoyándose en la librería Pexpect de Python. Una vez comunicados, el satélite (el gestor), envía "peticiones" al switch (el agente), que no son más que comandos Cisco que es capaz de interpretar, y éste último le devuelve una "respuesta" con la información solicitada. Este procedimiento aporta gran flexibilidad y nivel de personalización a la aplicación, ya que la información gestionable está solamente restringida por la cantidad de parámetros que permita mostrar al usuario el equipo en sí.
Esta información, tras darle forma y almacenarla correspondientemente, se ofrece de manera visual al cliente. La página web es, por tanto, la interfaz gráfica visible finalmente por el usuario.

4.2. Líneas futuras

Los módulos de conocimientos están sujetos a multitud de modificaciones. Algunas razones pueden ser la actualización del software del equipo y por tanto la posible modificación de los comandos y sus resultados, o el deseo de los clientes de incluir nuevos o distintos parámetros de gestión en el modelo existente.

Además, se ha de tener en cuenta que el módulo de conocimiento desarrollado, se ha comprobado para los modelos presentados al comienzo, lo cual no excluye que funcione correctamente para otras *releases* o versiones, o quizá únicamente tras la modificación parcial de alguno de los parámetros, sin necesidad de generar un módulo de conocimiento propio para tal versión.

En la versión siguiente del módulo CiscoCatalyst, urge completar en el submódulo de Ip, la sección *routes,* en la cual, por falta de la tabla de rutas en todos los switches de trabajo, no se ha podido elaborar el modelo de información de dicha tabla.

Finalmente, cabe resaltar que globalmente, el proyecto Work On Data tiene como fin disponer de un repositorio de módulos de conocimiento tan amplio y variado como sea posible, para poder proporcionar así su servicio al máximo

número de clientes. Esto implicará un largo y continuo trabajo de expansión y mantenimiento.

BIBLIOGRAFÍA

[1] Nologin Consulting: sitio web, página principal - *http://www.nologin.es/es/index*
Última visita: 22 Enero 2017.

[2] Cisco Systems, sitio web, productos de gestión de nube y sistemas - *http://www.cisco.com/c/es_es/products/cloud-systems-management/product-listing.html*
Última visita: 12 Enero 2017.

[3] MongoDB, entrada de Wikipedia - *https://es.wikipedia.org/wiki/MongoDB*
Última visita: 22 Enero 2017.

[4] JSON, entrada de Wikipedia - *https://es.wikipedia.org/wiki/JSON*
Última visita: 22 Enero 2017.

[5] Referencia de comandos para los switches Catalyst 3750-X and 3560-X, Release 12.2(53)SE2, sitio web - *http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3750x_3560x/software/release/12-2_53_se/command/reference/3750xcr.html*
Última visita: 22 Enero 2017.

[6] Documentación API Pexpect, sitio web - *https://pexpect.readthedocs.io/en/stable/*
Última visita: 23 Enero 2017.

[7] RSH, entrada de Wikipedia - *https://es.wikipedia.org/wiki/Rsh*
Última visita: 23 Enero 2017.

[8] Pycharm, sitio web - *https://www.jetbrains.com/pycharm/*
Última visita: 13 Enero 2017.

[9] Documentación Python del paquete Pep8, sitio web - *https://pypi.python.org/pypi/pep8*
Última visita: 13 Enero 2017.

[10] PyInstaller, sitio web - *http://www.pyinstaller.org/*
Última visita: 13 Enero 2017.

[11] Informe sobre la presencia de equipos Cisco en el mercado tecnológico en 2016, sitio web - *https://www.srgresearch.com/articles/cisco-continues-roll-punches-switching-router-market*
Última visita: 15 Enero 2017.

[12] Protocolo CDP, entrada de Wikipedia, sitio web -
https://es.wikipedia.org/wiki/Cisco_Discovery_Protocol
Última visita: 15 Enero 2017.

[13] Licencias de equipos Cisco, sitio web -
http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/CLIConfigurationGuide/Licensing.html
Última visita: 15 Enero 2017.

[14] Definición de Port to Application Mapping (PAM), sitio web de Cisco -
http://www.cisco.com/c/en/us/td/docs/ios/12_2/security/command/reference/fsecur_r/srfpam.html
Última visita: 15 Enero 2017.

[15] MVC, The Django Book, sitio web - *http://djangobook.com/*
Última visita: 17 Enero 2017.

[16] Framework Django, entrada de Wikipedia, sitio web -
https://es.wikipedia.org/wiki/Django_(framework)
Última visita: 17 Enero 2017.

[17] Introducción a tecnologías web UCM, sitio web -
https://www.fdi.ucm.es/profesor/jpavon/web/11-Introduccion-TecnologiasWeb.pdf
Última visita: 20 Enero 2017.

[18] Document Object Model (DOM), sitio web w3schools -
http://www.w3schools.com/js/js_htmldom.asp
Última visita: 20 Enero 2017.

[19] Definición de JQuery, entrada de Wikipedia, sitio web -
https://es.wikipedia.org/wiki/JQuery
Última visita: 20 Enero 2017.

[20] Bootstrap, entrada de Wikipedia, sitio web -
https://es.wikipedia.org/wiki/Twitter_Bootstrap
Última visita: 20 Enero 2017.

[21] LDAP, entrada de Wikipedia, sitio web -
https://es.wikipedia.org/wiki/Protocolo_Ligero_de_Acceso_a_Directorios
Última visita: 21 Enero 2017.