Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

JunCo: Una aplicación web para automatizar configuraciones de switches, routers y equipos de red

---

Alumno: Enrique García Nieto

Tutor: Yania Crespo González-Carvajal

A todas las personas que han hecho de mi camino un agradable paseo

Agradecimientos

A mi padre, por haber hecho todos los esfuerzos que me han permitido llegar al final de esta carrera.

A mis amigos, por haber estado siempre apoyando y haber sido uno de mis pilares de apoyo en todos estos años, tanto en esta situación como en muchas otras.

A José Luís y Mamen, por haber sido mi segundo hogar durante tantos años.

A Daniel Barba, por ser una motivación académica y un espejo donde poder mirarme en muchos aspectos.

A Javier Alaguero, cuyas indicaciones y paciencia me han sacado de más de un apuro durante la carrera, y por haberme enseñado el interesante mundo del front-end.

A mi equipo de trabajo, quienes desde el primer día se han esforzado en integrarme y en enseñar, pese a lo complicado de la situación del teletrabajo.

A todas las personas que han supuesto para mí más un hogar que una amistad y a aquellas que me han ayudado en cada parte de este camino.

Finalmente a Irene, mi principal apoyo en esta vida, por estar siempre ahí cuando más lo he necesitado y a quien durante muchos, muchos años, espero poder devolver todo lo que ha hecho por mí.

Resumen

El objetivo de este proyecto es facilitar la gestión, administración y aplicación de configuraciones en dispositivos de red, ya sean routers, switches o firewalls. El proyecto está enfocado a los administradores de redes y sistemas que día a día tienen que operar con estos dispositivos, para facilitar así su trabajo, aportando una interfaz para operar con estos dispositivos, además de añadir múltiples funcionalidades que pueden ser de gran utilidad como aplicar una misma configuración a varios dispositivos de manera automática, mantenimiento de una base de datos de los dispositivos utilizados, además de otras operaciones como búsqueda de tablas ARP o ver la configuración interna de cada dispositivo sin tener que conectarse directamente a éste a través de línea de comandos.

Por último, la motivación a la hora de desarrollar este proyecto es ahorrar tiempo a la hora de realizar tareas repetitivas que se pueden automatizar, aumentando la productividad del personal encargado de administrar estos equipos y aumentando la facilidad de uso de éstos a través de una interfaz intuitiva.

El proyecto se ha construido utilizando el framework Vue.js, Javascript, HTML5 y CSS para el frontend, Python para backend y MySQL para la capa de persistencia. En cuanto a la metodología para el seguimiento y el control del proyecto, se ha realizado adaptando el marco de trabajo para desarrollo ágil Scrum al contexto de este proyecto.

Abstract

The objective of this project is to facilitate management, administration, and application of configurations in network devices, whether these are routers, switches, or firewalls. The project is focused on network and system administrators who, commonly, operate with these devices, thus easing their work by providing a graphical interface to manage these devices. Moreover, the project aims to deliver a set of useful functionalities such as running batch configuration among a number of devices automatically as well as maintaining a device database, performing ARP table searches, or recovering remote configuration without the need for a CLI interface mechanism.

The motivation for this project comes from the nature of this kind of device operations, which are usually repetitive and time consuming. CLI interfaces tend to be rough and hard to master, also implies some risk, whereas graphical interfaces reduce complexity by hiding some procedures behind an easy to use application and an intuitive interface.

The project has been developed using the following technology stack. Frontend side uses the Vue.js framework, Javascript, HTML, and CSS. Backend has been coded in Python. Finally, for the persistence layer the project uses MySQL as DBMS and is interfaced with the backend using SQL Alchemy as ORM. The project has been developed using an adapted Scrum agile development framework for this context.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

El principal objetivo de este trabajo de fin de grado es crear, desarrollar y poner en funcionamiento una aplicación web dedicada a la gestión de configuraciones, obtención de datos y automatización de tareas de equipos de redes, tanto switches como routers o firewalls. Está destinada a administradores de dispositivos de red, quienes podrán disfrutar de varias ventajas al utilizar esta aplicación web en lugar de configurar equipos u obtener la información necesaria por línea de comandos. Si nos fijamos en los puntos favorables que podemos obtener al utilizar esta herramienta, podemos destacar (1) la seguridad, al tener software que valide la configuración antes de aplicarse a uno o varios dispositivos, (2) la comodidad y la automatización de tareas, que permite realizar una misma configuración en un lote de varios dispositivos de red, lo que facilitará el trabajo reduciendo el tiempo de tareas rutinarias y comunes, además de evitar problemas como, por ejemplo, actualizar un dispositivo con una configuración que provoque un mal funcionamiento del mismo.

Por otro lado, si nos fijamos en las cuestiones relacionadas con las configuraciones de equipos, la aplicación utilizará la API de `Junos Space` y, para cuestiones de autenticación y autorización, la API de la plataforma `ClearPass`. También utilizamos APIs para conectarnos a dispositivos Juniper a través de `PyEZ`, mientras que los dispositivos de Juniper utilizan `Netmiko` para conectarse a sus dispositivos. Estas últimas APIs nos permiten conectarnos a los equipos a través de SSH o Telnet y realizar llamadas para obtener información sobre sus configuraciones.

Por último, mencionar que tanto el enfoque ágil como las tecnologías empleadas se ajustan al contexto del Trabajo de Fin de Grado, junto a la elaboración del presente documento a modo de memoria [60].

1.2. Motivación

La principal idea de este proyecto surgió al ver las tareas que realizaban mis compañeros del grupo de trabajo. El grupo de trabajo donde me encuentro actualmente se dedica al mantenimiento, administración y soporte de dispositivos de redes, tales como switches, routers o firewalls. Ellos son los que se encargan de modificar la configuración de la red a petición de otros usuarios, comprobar incidencias, realizar mantenimiento de todos los equipos, cambios en las topologías de red y realizar un seguimiento del funcionamiento de todos estos equipos. Tras trabajar unos cuantos meses con ellos y adquirir algo de conocimiento sobre el dominio de redes y dispositivos de red, identifiqué varias tareas que son muy mecánicas, requiren gran cantidad de tiempo y, sobre todo, son repetitivas a lo largo del tiempo. Esto quiere decir que, periódicamente, se emplea una cantidad significativa de tiempo en realizar estas tareas y, por lo tanto, se puede ofrecer una solución software para facilitar las tareas a mi equipo de trabajo no sólo en cuanto a tiempo invertido, sino también respecto a otros aspectos como la facilidad o la seguridad a la hora de realizar este tipo de trabajos.

1.3. Público objetivo

Al no existir alternativas y al estar concebido para uso único y exclusivo dentro del marco empresarial de mi equipo de trabajo, el público objetivo se compone de los administradores de equipos de red que a diario reciben peticiones para aplicar configuraciones y modificaciones en dichos dispositivos. Debido a esta situación, el tratar día a día con los potenciales usuarios de esta aplicación me va a permitir tener una retroalimentación constante en aspectos como la interfaz de usuario, la experiencia de usuario o decisiones sobre el diseño de las herramientas de la aplicación o del propio diseño de ésta.

1.4. Aplicaciones similares

Al ser una aplicación con un propósito específico, si hubiera una aplicación que pudiese cubrir estas necesidades ya habría sido adquirida. Aun así, esta aplicación sí se nutre de funcionalidades de aplicaciones ya existentes, además de APIs de otros programas y aplicaciones como los siguientes:

- **SecureCRT**: propiedad de la empresa de software VanDyke, SecureCRT es un cliente de terminal tanto de SSH como de Telnet, con el cual podemos acceder a la terminal de dispositivos de red de manera remota. SecureCRT permite la introducción de comandos por terminal para ver, modificar o eliminar datos y configuraciones, lo cual también está incluido como funcionalidad inicial en mi proyecto [61]. Además de lo anterior, también permite guardar las sesiones del usuario para conectarte a los dispositivos (usuario y contraseña), de tal manera que permite ver todos los dispositivos a los que puedes acceder.

- **Junos Space**: es la herramienta de los equipos Juniper que permite administrar, ver detalles y aplicar configuraciones remotas en los dispositivos. También cuenta con una API sobre la que podemos hacer peticiones REST y operar con ella, de modo que parte de sus servicios los podemos integrar en nuestra aplicación [37].

1.5. Objetivos

En esta sección voy a desarrollar por una parte los objetivos a nivel de requisitos que quiero cumplir con este proyecto y, por otro, qué espero aprender personalmente en mi carrera profesional realizándolo.

1.5.1. Objetivos técnicos

El primero de los objetivos técnicos que se intenta cumplir con esta aplicación es facilitar el realizar varias tareas de manera automática a los administradores de redes. Hay multitud de tareas que se repiten en el tiempo y que conllevan mucho tiempo que se pueden realizar de una manera más simple, permitiendo al usuario introducir los parámetros necesarios para realizar la configuración deseada. La automatización de tareas suele traducirse en reducción de costes operacionales, aumento de productividad, disponibilidad, fiabilidad y rendimiento, por lo que atacar a este objetivo puede llevarnos a conseguir éxito en varios de los campos anteriores [26].

Junto a esto, otro objetivo que se cumple al reunir todas las características propuestas es la centralización de configuraciones y extracción de datos. En vez de utilizar distintas herramientas para realizar configuraciones, mostrar datos o añadir o eliminar dispositivos, con esta aplicación se podrá acceder de una manera más cómoda a todas las funcionalidades, aportando comodidad a los usuarios. Esto va a facilitar tanto la consulta como la ejecución de tareas desde un mismo lugar, ahorrando tiempo y añadiendo facilidad de uso.

Por último, también se añade seguridad debido a la facilidad a la hora de realizar tareas. Al evitar que el usuario tenga que introducir parámetros por consola directamente en los terminales, además de las herramientas de validación extras que podemos incluir, se reduce considerablemente el fallo humano [42]. Junto a lo anterior, también se añade una autenticación frente al servidor de ClearPass, lo que añade un factor extra de seguridad. En el servidor de ClearPass se encuentran registrados los grupos de usuarios y los permisos de cada uno de ellos, de modo que si un usuario se registra, podemos decidir qué permisos tendrá a la hora de realizar peticiones o ejecutar cierto tipo de tarea.

En resumen, los objetivos principales de nuestra aplicación se centran en la **facilidad de uso**, **centralizar las funcionalidades** que antes estaban recogidas en otras plataformas, **automatizar** algunas tareas y, por último, añadir un factor extra de **seguridad**.

1.5.2. Objetivos personales

Para finalizar esta sección deseo mencionar los objetivos personales que he tenido en cuenta y han influido en varias de las decisiones a la hora de desarrollar este proyecto. En primer lugar, algunas de las decisiones a la hora de qué tecnología utilizar han sido de libre elección, mientras que en otros aspectos estas decisiones han venido dadas por las necesidades de los dispositivos de red y las plataformas utilizadas. Uno de los principales objetivos que me he propuesto en los puntos donde tenía libertad de decisión ha sido salir de la zona de confort de los lenguajes que he utilizado durante la carrera, y aprender a utilizar otros entornos y frameworks como, en este caso, Flask y Vue.js.

Con este proyecto he querido también conseguir desarrollar una aplicación web de manera completa, desarrollar tanto las vistas y toda la lógica del front-end, como levantar un servidor back-end que procese las peticiones, en el cual definir también los mensajes de respuesta HTTP, comunicarse con la capa de persistencia y devolver correctamente los datos solicitados por la interfaz.

En cuanto al entorno de trabajo en el que estoy, al ser un equipo de administración y mantenimiento de equipos de redes y sistemas, mi objetivo ha sido integrar los conocimientos adquiridos durante mi desempeño con lo aprendido en la carrera para, así, aportar utilidad en un escenario empresarial real, para ver cómo se comporta un producto desarrollado por mí en un entorno digital.

Además de lo mencionado anteriormente, un objetivo que quise cumplir es ver si he conseguido identificar necesidades de los usuarios y, en consecuencia, dar una solución a esas necesidades a través del software. El constante feedback que he tenido de los miembros del equipo me han ayudado mucho para cumplir este objetivo.

Para finalizar, un objetivo que he tenido en cuenta para este proyecto es comenzar, desarrollar y finalizar un artefacto software siguiendo un proceso de desarrollo ágil. Nunca había utilizado este tipo de proceso, por lo que he querido aprender experimentando de forma práctica, desde el comienzo a la hora de planificar y documentar, hasta el final, cuando el producto está acabado.

1.6. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 2 Requisitos y planificación: En este capítulo se presenta la planificación inicial del proyecto, la planificación a través del marco de desarrollo Scrum y su adaptación a nuestro caso concreto. También se incluyen las Historias de Usuario que aportan funcionalidad y su previsión a lo largo del tiempo de desarrollo.

Capítulo 3 Análisis: Se presenta el análisis de los requisitos. El análisis hará referencia a los modelos y diagramas que describen conceptualmente el proyecto, ya sean el modelo

de dominio o bien, aspectos de análisis del funcionamiento de algunas partes importantes de la aplicación. Este capítulo describirá el comportamiento de la aplicación, su composición y también se explicarán algunos elementos importantes para la correcta comprensión del proyecto.

Capítulo 4 Tecnologías utilizadas: Aquí vamos a abordar las tecnologías, herramientas, lenguajes de programación y plataformas que se utilizan en el desarrollo y la implementación del código del proyecto. Se definirá, entre otras cosas, en qué consisten, en qué parte del proyecto se han utilizado o de qué tecnologías o lenguajes están compuestos los elementos utilizados.

Capítulo 5 Diseño: A lo largo de este capítulo vamos a ver las decisiones que se han tomado en lo que a diseño de interfaces se refiere. Se muestran los bocetos iniciales y su posterior implementación como vistas, etc. También hablaremos del diseño de los componentes y alguno de los patrones utilizados a la hora de programar el software. En este capítulo también se exponen los diagramas de despliegue, modelo relacional y otros como pueden ser el de despliegue o el diagrama de componentes.

Capítulo 6 Implementación y pruebas: Como su propio nombre indica, aquí se tratarán cuestiones relativas a la implementación: qué decisiones se han tomado, qué estructuras de paquetes tendrá nuestro proyecto, qué cambios ha habido respecto al primer diseño, etc. También se tratará qué tipo de pruebas se han realizado y qué metodología que se ha aplicado para realizarlas.

Capítulo 7 Seguimiento del proyecto: Durante la realización del proyecto, se ha realizado un seguimiento de las actividades mediante periodos fijos de tiempo llamados sprints. En cada sprint, durante su planificación, se han recogido los requisitos y funcionalidades a desarrollar y, al final de cada sprint, se documenta qué porcentaje se ha completado de las tareas previstas inicialmente, además de otras tareas que se hayan podido añadir a lo largo de ese sprint.

Capítulo 8 Conclusiones: En el último apartado, se aportan unas conclusiones tratando los aspectos más generales del proyecto, además de plasmar las posibles mejoras y el ámbito en el que se pretende seguir ampliando el proyecto tras su presentación, o las funcionalidades que se podrán añadir posteriormente.

Anexo A Manuales: Anexo para incluir manuales de mantenimiento, de instalación, despliegue, y de ejecución; útiles especialmente para poblar las bases de datos y comprobar periódicamente cambios en algunas tablas de los dispositivos.

Anexo B Resumen de enlaces adicionales: Incluye enlaces de interés sobre el proyecto, como el repositorio donde se aloja el código.

Capítulo 2

Requisitos y Planificación

2.1. Marco de desarrollo ágil

Como vimos en el resumen inicial, este proyecto se ha desarrollado adaptando el marco de trabajo para desarrollo ágil Scrum al presente contexto. Se trata de un marco de trabajo (framework) basado en un modelo de proceso Agile que tiene como finalidad la entrega de valor o funcionalidad en periodos significativamente cortos. Gracias a este proceso, se reduce la complejidad en la elaboración del proyecto.

2.1.1. ¿Qué es Scrum?

Scrum es un marco de trabajo para desarrollo ágil de software que surgió en la década de los 80 de la mano de Hirotaka Takeuchi y de Ikujiro Nonaka. Se creó debido a que algunas empresas como Canon, Honda, Epson o Hewlett-Packard necesitaban desarrollar productos exitosos en menor tiempo que el que utilizaron para desarrollar productos anteriores [40].

En cuanto a una pequeña definición de Scrum, podemos decir que es un marco de trabajo ligero que provee a las personas, equipos y organizaciones a generar valor añadido a través de soluciones flexibles y adaptativas para problemas complejos. Scrum se apoya sobre los siguientes bloques generales [57]:

- Plantear una estrategia de desarrollo incremental, en vez de la planificación completa tradicional. Esta tarea la lleva a cabo el *Product Owner*, quien es el encargado de organizar el trabajo necesario para resolver un problema complejo e incluirlo en el *Product Backlog*.
- Adaptar el proceso de desarrollo y solapar las fases de desarrollo, frente a realizar las fases de desarrollo de manera secuencial. Esta tarea consiste en seleccionar qué cantidad

de trabajo que genere valor añadido se va a llevar a cabo durante el sprint[1], y lo realiza el *Scrum Team*.

- Detectar inconvenientes o fallos durante el desarrollo gracias al conocimiento de los recursos humanos involucrados en el progreso de cada iteración. Los responsables de inspeccionar los resultados y ajustar posibles cambios para el siguiente sprint es el *Scrum Team* y los *Stakeholders*.

2.1.2. Roles en Scrum

Scrum tiene tres roles principales con responsabilidades bien definidas y diferenciadas. Estos roles están delimitados para garantizar una gestión efectiva, aumentar la productividad y alcanzar los objetivos dentro del tiempo definido. La unidad fundamental de Scrum es un pequeño equipo de personas llamado *Scrum Team*. Los Scrum Teams son equipos multifuncionales con las habilidades necesarias para aportar valor añadido en cada Sprint. Ellos son los encargados de decidir de manera interna qué hacer y qué no y, sobre todo, cuándo y cómo hacerlo. Este equipo está compuesto por un *Scrum Master*, un *Product Owner* y un *Development Team*. A continuación vamos a ver en qué consiste cada rol y algunos de los aspectos más relevantes de cada uno de ellos.

- **Product Owner** o propietario del producto: es el encargado de maximizar el valor del produccto final desarrollado por el *Development Team*. Entre algunas de las tareas que lleva a cabo este rol, encontramos:

  - Desarrollar y comunicar de manera clara los objetivos del producto.
  - Crear y comunicar los puntos del Product Backlog[2].
  - Organizar los puntos del Product Backlog.
  - Asegurarse que el Product Backlog sea claramente entendible, transparente y visible para todos los miembros del Scrum Team.

  El Product Owner puede encargarse personalmente de las tareas enumeradas anteriormente o, por el contrario, delegar estas tareas en otros miembros del equipo. Sin embargo, tome la decisión que tome siempre será el responsable, directa o indirectamente. Al contrario que en otros roles, el Product Owner debe ser una única persona, no varias, y es el representante de las necesidades de los *Stakeholders* (es decir, de las partes interesadas en el desarrollo del producto).

- **Developement Team** o equipo de desarrollo: son las personas encargadas de crear, desarrollar e implementar cualquier aspecto a la hora de entregar un producto usable y con valor al final de cada sprint. Las habilidades necesarias para el equipo de desarrollo varían según las necesidades del proyecto. Sin embargo, los desarrolladores están a cargo de actividades tales como:

  - Crear una planificación para el sprint (*Sprint Backlog*).

[1] El concepto *sprint* se define más adelante, en el apartado 2.1.3
[2] El término *Product Backlog* se define más adelante, en el apartado 2.1.4

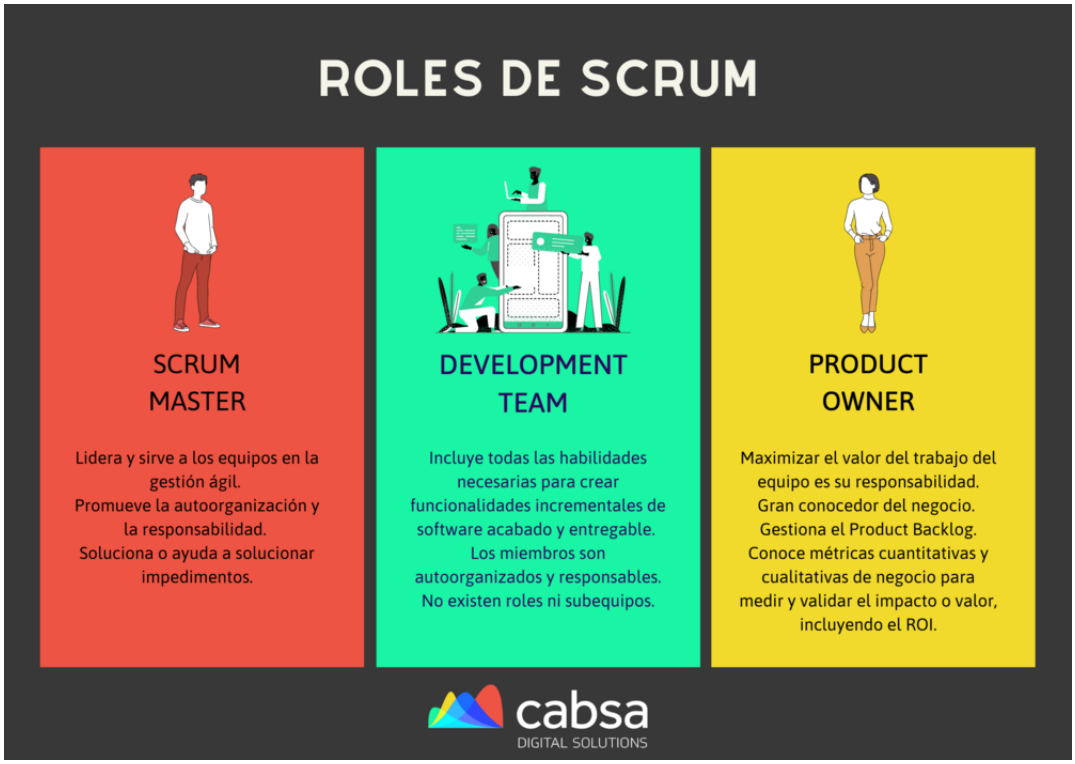


Figura 2.1: Roles en Scrum [11]

- Dotar de calidad al producto ciñéndose a la definición de "Terminado", es decir, aportar valor añadido aportando un producto con una o varias características terminadas al final de cada sprint.
- Adaptar el plan de trabajo diario para alcanzar los *Objetivos del sprint*.
- Responsabilizarse mutuamente como profesionales.

- **Scrum Master** o facilitador: es el responsable de que se sigan las pautas descritas en la guía de Scrum. Es el encargado de la eficacia y de cumplir los objetivos que se proponen para todo el Scrum Team, además de ayudar a encontrar soluciones a los problemas que se puedan presentar. En otras palabras, es el líder que se encarga de dirigir al Scrum Team.

2.1.3. Eventos de Scrum

Cada evento en Scrum es una oportunidad para inspeccionar y transformar los artefactos Scrum. Dentro de este marco de trabajo, estos eventos se utilizan para aportar regularidad y minimizar las reuniones que serían necesarias si no se utilizara este marco de trabajo [57].

Los eventos de Scrum son los siguientes:

- **Sprint**: un sprint es un bloque de tiempo en el cual se desarrolla el trabajo en sí mismo. Como recomendación, la duración de éstos debe ser constante y definida por el equipo o por el Scrum Master. Al finalizar el sprint, el equipo debe presentar la situación actual del proyecto, el progreso de éste así como el resultado obtenido. Dicho resultado consiste en un incremento de producto que se puede considerar como "terminado", que se puede entregar al cliente.

  Durante el sprint se recomienda no agregar objetivos adicionales a no ser que la ausencia de éstos pongan en peligro la funcionalidad del proyecto entregable. El sprint debe, en la medida de lo posible, mantener unos requisitos predefinidos y unos objetivos claros e inmutables durante su desarrollo. Si se quiere añadir algo, debe intentar incluirse en otro sprint posterior, aunque puede hacerse en el mismo.

  El sprint está limitado a una duración no mayor de un mes. En nuestro caso, la duración de cada sprint se ha fijado en una semana.

- **Sprint planning**: esta planificación del sprint consiste en organizar el trabajo a realizar durante el tiempo de duración del sprint. Este trabajo lo lleva a cabo el Developement Team. Durante esta fase también tiene lugar la asignación y estimación de carga de trabajo a cada tarea. Dicha asignación se denomina *Punto de historia (Story Point)*, denominado con las siglas *PH* o *SP*. Cada punto de historia equivale a una cantidad determinada de horas que el equipo de desarrollo debe emplear para realizar todas las tareas del punto de historia en cuestión.

- **Story Points** o Puntos de historia: como hemos visto antes, un punto de historia es una unidad de esfuerzo asignada a cada historia de usuario. Los puntos dan idea del tamaño y el esfuerzo que se necesita para realizar dichas tareas dentro de un sprint. Debido a esto, son una buena manera de expresar el esfuerzo en términos de tiempo de trabajo que puede llevar cada tarea.

  Los puntos de historia se pueden fijar con varios métodos. Uno de ellos es la escala lineal, en la cual un punto equivale a un número predefinido de horas, y si la cantidad de puntos de historia para una historia de usuario aumentan, lo harán de manera proporcional las horas de trabajo asociadas. En este proyecto he creído conveniente establecer la serie de Fibonacci, donde el siguiente número en la escala de los PH es la suma de los dos anteriores. De esta manera, una distribución de puntos de historia siguiendo la sucesión de Fibonacci queda así reflejada como se muestra en la Tabla 2.1:

- **Daily Scrum o Daily Standup**: llamado también *Scrum diario*, es una pequeña reunión entre cinco y quince minutos con el motivo de mantener informados al resto de miembros del equipo de desarrollo. De esta manera, ellos sabrán qué problemas se han encontrado desde el último scrum diario, qué se espera encontrar o en qué está trabajando cada miembro del equipo. Si se requiere ampliar un tema, se debe hacer posteriormente tras el scrum diario. En resumen, esta reunión pone en común el progreso del equipo y los esfuerzos se centran en sincronizar a todos los miembros del equipo para alcanzar correctamente la meta del sprint. En nuestro caso, como el Development Team está formado por una persona, no se ha creído necesario realizar esta Daily Standup.

- **Sprint Review** (revisión del sprint): el propósito de esta reunión es que, al final de cada sprint, el equipo realice tanto la revisión del sprint como su retrospectiva. En la

P.H.	Nº de horas
1	1
2	2
3	3
4	5
5	8
6	13
7	20
8	33
9	53

Tabla 2.1: P.H. asociados a número de horas (Fibonacci)

reunión dedicada a la revisión, se presentan los trabajos completados en ese lapso de tiempo. El tiempo recomendado no debe superar las 4 horas para un sprint de un mes.

Durante este evento, el Scrum Team y los stakeholders implicados deben inspeccionar qué ha cambiado en el entorno del proyecto y, basándose en esta información, se planifica qué hacer para el siguiente sprint. Consecuentemente, el Product Backlog debe ser ajustado para dar cabida a estos nuevos cambios

- **Sprint Retrospective** (retrospectiva del sprint): al igual que la revisión vista anteriormente, se lleva a cabo al finalizar el sprint. En la retrospectiva, todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado como, por ejemplo, qué problemas han tenido o sugerencias de mejoras para optimizar el trabajo.

  El Scrum Team debe identificar los cambios más significativos para mejorar su eficacia. Las mejoras que puedan suponer un mayor cambio positivo son las que primero deben adoptarse para los siguientes sprints.

  Esta sesión da por concluído el sprint. Debe limitarse a un máximo de tres horas para un sprint de un mes, siendo más corto para sprints con menor duración.

- **Sprint Goal** (objetivo del Sprint): el objetivo del sprint es una meta establecida para dicho sprint. A medida que el equipo trabaja, debe hacerlo con la meta o el objetivo del sprint siempre en mente para así intentar satisfacer el objetivo aportando la funcionalidad necesaria. El objetivo del sprint puede desglosarse en *User Stories (U.S.)* o *Historias de Usuario (H.U.)* del Backlog.

2.1.4. Artefactos de Scrum

Dentro del marco Scrum, los artefactos son aquellos que generan valor al producto. Representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia en el desarrollo. Los artefactos relevantes dentro de Scrum son los siguientes: [57]

- **Product Backlog** (lista de producto): lista ordenada de todos los requisitos del producto. Es la única fuente de requisitos para cualquier cambio que deba realizarse en el

producto. Comienza con un Backlog inicial, donde se ponen las tareas que se pretenden realizar a lo largo del ciclo de desarrollo de un proyecto. Estas tareas deben citarse posteriormente en el Product Backlog. El Product Backlog empieza con una visión inicial del producto y crece y evoluciona durante el desarrollo del producto. Debido a esta naturaleza, el Product Backlog es cambiante y siempre está sujeto a modificaciones. Esta lista de producto nunca está completa. Su desarrollo más temprano sólo refleja los objetivos más generales y más fácilmente comprensibles. A medida que se hacen los sprints y se obtiene retroalimentación, esta lista se va ampliando y los requisitos son más largos y exhaustivos. Las personas encargadas de trabajar en el Product Backlog son los integrantes del Development Team.

- **Sprint Backlog**: registra los requisitos desde el punto de vista de los desarrolladores. Es la lista de tareas pendientes durante un sprint para llevar a cabo el incremento deseado, es decir, el Sprint Backlog es un subconjunto del Product Backlog, puesto que el primero está compuesto de elementos del segundo que se han seleccionado para el sprint. Al contrario que el Product Backlog, el Sprint Backlog es estático no se modifica mientras el sprint se encuentra en desarrollo. El sprint backlog permite ver las tareas necesarias para alcanzar el Sprint Goal.

- **Incremento**: es el resultado de añadir todos los elementos del Product Backlog que se han llevado a cabo en el sprint. Al final de un sprint, el nuevo incremento debe estar terminado, es decir, debe estar disponible y operativo para ser utilizado.

2.2. Planificación

2.2.1. Adaptación de Scrum al proyecto

El marco general de Scrum debe ser aplicado al contexto de este Trabajo de Fin de Grado. De esta manera, debemos ver cómo podemos aplicar Scrum a las características de éste.

En primer lugar, vamos a hablar de los **roles de Scrum**. En nuestro caso, las personas que desempeñan los distintos roles están claramente diferenciadas.

- **Scrum Master**: este rol lo desempeña Yania, mi tutora de TFG, encargándose de tareas como que se cumplan los plazos, que cada iteración añada una funcionalidad útil para el Product Owner, que la documentación sea adecuada, etc. También es ella quien se encarga de fijar las reuniones (tanto Sprint Planning como Sprint Review, además de la Sprint Retrospective aunque, al ser sprints de una semana, estas reuniones se unifican realizándose todas el mismo día) y dar algo de retroalimentación en cuanto al porcentaje del proyecto completado y aspectos generales del proyecto. No obstante, en cuanto al Product Backlog, es el alumno quien se encargará de completarlo, siendo ella la que simplemente supervise este artefacto de Scrum.

- **Product Owner**: en este caso, como el proyecto se realiza bajo un entorno empresarial para el uso por parte de un equipo de administradores de redes, el Product Owner ha

sido el coordinador del equipo donde he estado trabajando. De esta manera, he podido tener retroalimentación constante de las funcionalidades, tanto de aquellas que fueron expresamente pedidas por él como otras propuestas por mí. Al igual que con el Scrum Master, acordamos al inicio del proyecto fijar un día por semana para revisar cómo se desarrollan las funcionalidades, ver nuevas posibilidades y posibles ampliaciones y cambios en la funcionalidad.

- **Development Team**: por último, como es obvio, el Development Team está compuesto únicamente por una persona, es decir, por mí. Al ser un proyecto personal, soy el único encargado de desarrollarlo. También he sido el responsable para redactar la lista de historias de usuario del Product Backlog. Para el desarrollo del código, he decidido utilizar Git como sistema de control de versiones. Como los datos que trato en mi TFG son considerados información sensible, no he podido tener todo el repositorio subido de manera inicial. Al finalizar el proyecto, he subido la versión final para poder consultarlo, por lo que simplemente he utilizado Git de manera local hasta la versión final. Por otro lado, para las historias de usuario, he utilizado Jira como herramienta para seguimiento de tareas. Al ser también una herramienta de dominio empresarial, en cada tarea ponía como supervisor al Product Owner, es decir, a mí coordinador. De esta manera, he ido anotando el desarrollo de las tareas, el tiempo que me han llevado, así como algunas tareas que eran suficientemente grandes como para dividirse en dos, incidencias que surgían, impedimentos, etc. En definitiva, como único integrante del Development Team, he utilizado Git como herramienta de desarrollo para llevar un seguimiento y un control de versiones, y Jira para controlar el progreso de cada tarea o historia de usuario. Para ver estas tecnologías con más detalle además de otras utilizadas durante el proyecto, consultar el Capítulo 4.

2.2.2. Planificación inicial de los Sprints

La realización de la asignatura Trabajo de Fin de Grado tiene asignados 12 créditos ECTS, los cuales suponen aproximadamente 300 horas lectivas [60], las cuales he decidido junto a mi tutora dividirlas en 10 sprints de 30 horas cada uno. El sprint está configurado para durar una semana, de lunes a domingo, por lo que la carga de trabajo será también 30 horas semanales. Además, dos de los sprints serán de refuerzo, siendo cercanos a la fecha de finalización del proyecto.

Esto no quiere decir que este número de sprints sea fijo. Quizá en algunas semanas se pueda meter más carga de trabajo o, por el contrario, se puedan añadir más sprints si el tiempo no es suficiente o si se añade más funcionalidad, hay algún problema con las historias de usuario, etc.

La distribución inicial de los sprints queda reflejada en la Tabla 2.2. Como vemos en ella, la distribución de tiempo se divide en 10 sprints como hemos comentado antes. A su vez, se dividen en 8 sprints normales y se dejarán 2 sprint de refuerzo, que suelen utilizarse para tareas de refactoring, corrección de bugs, o prevención de la posible repercusión en el desarrollo de cualquier aparición de problemas a lo largo del proceso.

Sprint	Comienzo	Finalización	Detalles
Sprint 1	08/02/2021	14/02/2021	
Sprint 2	15/02/2021	21/02/2021	
Sprint 3	22/02/2021	28/02/2021	
Sprint 4	01/03/2021	07/03/2021	
Sprint 5	08/03/2021	14/03/2021	
Sprint 6	15/03/2021	21/03/2021	
Sprint 7	22/03/2021	28/03/2021	
Sprint 8	29/03/2021	04/04/2021	
Sprint 9	05/04/2021	11/04/2021	Posible sprint de refuerzo
Sprint 10	12/04/2021	18/04/2021	Posible sprint de refuerzo

Tabla 2.2: Distribución inicial de sprints

2.2.3. Descripción de historias de usuario

Las historias de usuario (H.U.) se abordarán más detenidamente a lo largo del seguimiento de los sprints, en el Capítulo 7. No obstante, podemos ver en la Tabla 2.3 todas las H.U. que componen el Product Backlog. También se dan detalles a modo de resumen que muestran en qué consiste cada H.U.

2.2.4. Plan de riesgos y estimación de coste

Para hablar de riesgos, primero necesitamos definir qué es un riesgo. El riesgo se puede definir como un evento o condición incierta que, en caso de ocurrir, puede afectar (positiva o negativamente) al proyecto [19]. Los riesgos se pueden dividir en:

- **Riesgos de proyecto**: relacionados con aspectos inherentes al proyecto, como no conseguir los objetivos propuestos o no alcanzar una cota adecuada de satisfacción de requisitos.

- **Riesgos de negocio**: relacionados con temas económicos y de mercado. Aquí se incluyen riesgos de ventas, popularidad del producto, recepción de éste, etc.

Para poder controlar en medida de lo posible estos riesgos, existen una serie de pautas y acciones a realizar para medir, estimar y reducir el impacto de los riesgos. En primer lugar habría que identificar los riesgos y, una vez identificados, analizar qué aspectos son prioritarios frente a otros. Una vez hecho esto, se deben medir los riesgos mediante una escala predefinida para, posteriormente, monitorizar y controlar los riesgos.

H.U.	Nombre	Detalles
01	Ver dispositivos Juniper disponibles.	Como usuario quiero ver los dispositivos Juniper disponibles en Junos Space para ver a cuáles puedo aplicar configlets.
02	Ver configlets disponibles.	Como usuario quiero consultar qué Configlets hay disponibles para poder aplicarlos a dispositivos Juniper.
03	Ver datos de un configlet.	Como usuario quiero ver contenido del configlet para ver qué se va a aplicar en un dispositivo.
04	Login.	Como usuario quiero hacer login en la aplicación para poder utilizar sus funciones.
05	Tablas ARP.	Como usuario quiero consultar la ubicación de una dirección IP o MAC para ver en qué lugar físico se encuentra.
06	Aplicar configlets.	Como usuario quiero aplicar uno o más configlets a un dispositivo Juniper para modificar su configuración.
07	Añadir dispositivo.	Como usuario quiero añadir un dispositivo de cualquier fabricante para que esté disponible en la base de datos.
08	Eliminar dispositivo.	Como usuario quiero eliminar uno o varios dispositivos de la lista para que no aparezcan dispositivos obsoletos.
09	Selección de idioma.	Como usuario quiero elegir un idioma entre los dos disponibles (Español e Inglés) para comprender mejor el texto de la aplicación.
10	Ver config. dispositivo Juniper.	Como usuario quiero ver la configuración de un dispositivo Juniper de Junos Space para ver qué configuración refleja la API.
11	Seleccionar configlet.	Como usuario quiero ver la lista de configlets disponibles para aplicársela a uno o varios dispositivos Juniper disponibles.
12	Eliminar configlet.	Como usuario quiero seleccionar un configlet disponible y eliminarlo para que no se pueda aplicar a ningún dispositivo.
13	Logout.	Cpmo usuario, una vez logueado, quiero finalizar sesión para que sus datos sean borrados del LocalStorage.
14	Editar dispositivo.	Como usuario quiero editar la información de uno de los dispositivos mostrados para actualizarla.
15	Mostrar todos los dispositivos.	Como usuario quiero ver todos los dispositivos de todos los fabricantes para comprobar sus datos.
16	Mostrar la configuración de un dispositivo.	Como usuario quiero ver la configuración de un dispositivo de cualquier fabricante para saber algunos parámetros determinados.
17	Validar configlet.	Como usuario quiero ver si la configuración de un configlet es válida para poder aplicarla.

Tabla 2.3: Descripción de historias de usuario
H.U.: Historia de Usuario

Riesgos encontrados

En nuestro caso, todos los riesgos encontrados han sido riesgos de proyecto, pues el producto desarrollado no se ha concebido como un producto comercial del que se vayan a esperar beneficios con su salida o venta en el mercado. Como se mencionó anteriomente, es necesario asignar a cada riesgo una probabilidad de ocurrencia y un impacto en una escala numérica. De esta manera, la escala será una escala numérica, y cada valor equivalen a una valoración cuantitativa como podemos ver en la Tabla 2.4:

Valor	Probabilidad de ocurrencia	Impacto
1	Muy poco probable	Muy leve
2	Poco probable	Leve
3	Probabilidad media	Medio
4	Probable	Importante
5	Muy probable	Crítico

Tabla 2.4: Tabla de puntuación de riesgos

A continuación vamos a ver qué riesgos hemos encontrado a la hora de planificar el proyecto. De la Tabla 2.5 a 2.13, podemos ver su descripción, probabilidad de ocurrencia, impacto, plan de mitigación y, por último, plan de contingencia.

Riesgo	Falta de conocimiento de tecnologías de red
Probabilidad	5
Impacto	2
Plan de mitigación	Formación y aprendizaje gradual
Plan de contingencia	Preguntar dudas a mi equipo de trabajo

Tabla 2.5: Riesgo 1

Riesgo	Falta de conocimiento de tecnologías de desarrollo
Probabilidad	3
Impacto	3
Plan de mitigación	Seleccionar tecnologías sobre las que tenga más conocimiento a la hora de desarrollar
Plan de contingencia	Preguntar a un experto o consultar en manuales y foros

Tabla 2.6: Riesgo 2

Riesgo	El proyecto se alarga más de lo debido
Probabilidad	3
Impacto	4
Plan de mitigación	Reevaluar en cada sprint planning los puntos de historia y ver la situación en la que se encuentra el desarrollo del proyecto
Plan de contingencia	Redefinir las Historias de Usuario y realizar una nueva estimación de Puntos de Historia

Tabla 2.7: Riesgo 3

Riesgo	No me puedo comunicar con los equipos
Probabilidad	2
Impacto	4
Plan de mitigación	Revisar los logs en las configuraciones para ver si se ha aplicado una nueva configuración recientemente
Plan de contingencia	Hablar con el responsable que ha aplicado esa nueva configuración recientemente

Tabla 2.8: Riesgo 4

Riesgo	Apagón en la sede donde se aloje la máquina virtual
Probabilidad	1
Impacto	5
Plan de mitigación	Sólo se puede intentar evitar dejar mucho trabajo para días donde no haya personal de emergencia en estos casos
Plan de contingencia	Esperar a que el personal de emergencia solucione la incidencia

Tabla 2.9: Riesgo 5

Riesgo	Imposibilidad de conectarme a la máquina virtual
Probabilidad	2
Impacto	4
Plan de mitigación	Intentar trabajar con la máquina virtual siempre que haya un administrador de redes disponible
Plan de contingencia	Contactar con el administrador de las máquinas virtuales que esté disponible en ese momento

Tabla 2.10: Riesgo 6

Riesgo	Retraso en el desarrollo de tareas
Probabilidad	3
Impacto	2
Plan de mitigación	Planificar mejor el sprint a medida que se va adquiriendo conocimiento y fijar sprints de refuerzo
Plan de contingencia	Realizar las tareas en el siguiente sprint

Tabla 2.11: Riesgo 7

Riesgo	Mi estancia en la empresa finaliza antes de poder desarrollar el proyecto
Probabilidad	3
Impacto	4
Plan de mitigación	Consultar mi posibilidad de continuación y mi situación actual
Plan de contingencia	Preparar la información para poder realizar mocks en los dispositivos que simulen una comunicación real

Tabla 2.12: Riesgo 8

Riesgo	Cambio en la política de Login
Probabilidad	1
Impacto	4
Plan de mitigación	Eliminar en medida de lo posible la dependencia entre este componente y el resto
Plan de contingencia	Documentarme y adquirir la información sobre el nuevo protocolo para implementarlo en mi proyecto

Tabla 2.13: Riesgo 9

Riesgo	Cambio de fabricante de dispositivos
Probabilidad	1
Impacto	3
Plan de mitigación	Realizar la arquitectura del proyecto lo más modular posible para generar menos dependencias
Plan de contingencia	Adaptar el código para obtener la misma información pero de otro tipo de fabricante

Tabla 2.14: Riesgo 10

2.3. Estimación de costes del proyecto

Para estimar el coste que puede tener este proyecto, lo primero que se ha realizado es consultar el sueldo de un programador analista. Según el Boletín Oficial del Estado (BOE),

un ingeniero de software corresponde a la categoría Analista programador; Diseñador pagina web (Grupo III) [43]. Según el BOE, a un trabajador perteneciente a esta categoría, le correspondería una cuantía de 22.993,74 euros anuales, lo que supone un total de 1.438,08 euros mensuales. Suponiendo que la empresa contribuye a la Seguridad Social con un importe que constituye un 30 % del sueldo del trabajador [12], el coste total para la empresa por la contratación de un trabajador de esta categoría supone 29.891.862 euros anuales.

Suponiendo que las horas anuales de un trabajador de esta categoría se ajustan a un máximo de 1.800 horas anuales [43] por lo que, dividiendo el sueldo anual total entre las horas totales anuales, nos da un resultado de 16.60 euros/h que debería cobrar un trabajador de este tipo. Si suponemos las 300 horas que se estiman en la guía docente del TFG [60] como tiempo previsto en el que se realizará el presente proyecto, el coste de contratación de un trabajador de esta categoría asciende a 4.980 euros durante la duración de este proyecto.

Como la situación actual de pandemia ha resultado en un teletrabajo como forma diaria de operar en esta empresa, no se computan gastos fijos ni variables a la empresa. Además, todas las licencias para el desarrollo de este proyecto son o bien gratuitas o bien libres. El GitLab donde se alojará la versión final del código del proyecto es una versión para la Escuela de Ingeniería Informática asociada a la Universidad de Valladolid, por lo que la empresa tampoco paga ninguna licencia de este tipo.

De esta manera, el coste por este proyecto, si se realizara en las 300h estimadas en un principio, sería de 4.980,00 euros. No obstante, he preferido estimar las horas en 450 frente a las 300 horas iniciales y, para evitar problemas derivados de sobrecostes, la nueva cuantía asciende a 7.470 euros.

2.4. Modificaciones finales respecto a la planificación inicial

Por último, antes de dar por finalizado este capítulo, vamos a comentar qué aspectos respecto a la planificación se han cambiado que hayan tenido el suficiente impacto en el desarrollo de este proyecto.

El primero de ellos es el número de sprints. Ya sea por algunos problemas a la hora de conectarme a algunos dispositivos, como por estimación insuficiente del número de horas para realizar algunos aspectos de las historias de usuario, como la ampliación de la funcionalidad de la aplicación, el número inicial de 10 sprints ha sido insuficiente, llegando a alcanzar 17 sprints para finalizar el proyecto.

Otro de ellos ha sido las historias de usuario. Inicialmente se pensó las historias de usuario para trabajar con dispositivos Juniper, los cuales tienen su propia plataforma que nos permite automatizar tareas. Sin embargo, al ir trabajando día a día con estos equipos, me hizo coger soltura y pude añadir otras tareas útiles sugeridas por el Product Owner, tales como mostrar la configuración de cualquier tipo de dispositivo, o llevar un inventario de todos los dispositivos que se vayan anotando en un fichero común, con las acciones correspondientes (añadir, editar o eliminar un dispositivo). Este cúmulo de eventos ha hecho que

hayan sido necesarios más sprints y el número de historias de usuario también lo haya hecho en consecuencia.

Si en el capítulo anterior, se fijó una cantidad estimada de 7.470 euros por 450h de proyecto, ahora podemos comprobar que esa cantidad es mucho menor que la cantidad de horas empleadas finalmente. En total, se han empleado 630 horas, como se puede ver en el Capítulo 7.2. Si anteriormente se calculó que el coste de una hora de trabajo era 16.60 euros/hora, el montante final asociado al coste del trabajador se calcula nuevamente, con un resultado de 10.458 euros, lo que supone una diferencia de 2.988 euros frente a la previsión inicial.

Visto esto, queda finalizado el capítulo de requisitos y planificación. Hemos hablado del proceso ágil que hemos seguido para realizar el proyecto, cómo lo hemos adaptado, qué artefactos y roles se han aplicado y, por último, qué cambios sustanciales ha habido desde la planificación hasta la conclusión del proyecto.

Capítulo 3

Análisis

En este capítulo, vamos a ver todos los aspectos relacionados con el análisis de los requisitos y las funcionalidades, una explicación de los algoritmos que necesitamos desarrollar y, también, el modelo de dominio de la aplicación.

3.1. Análisis general de la funcionalidad

En este proyecto podemos dividir la funcionalidad general en tres secciones. Vamos a ver en qué consisten para, posteriormente, ver el diagrama de dominio asociado y explicarlo brevemente.

1. **Inventariado de dispositivos**: Uno de los aspectos con mayor utilidad para mi equipo ha sido diseñar una base de datos con los dispositivos en uso del equipo de redes de la empresa. Sobre estos dispositivos, se podrán realizar acciones CRUD. Las siglas CRUD basan su significado en las operaciones *Create, Retrieve, Update* y *Delete* [33], es decir, Crear, Leer, Actualizar y Borrar. De esta manera, podremos realizar estas acciones en nuestra base de datos para mantener un invetario actualizado de los dispositivos. Podremos añadir nuevos dispositivos introduciendo los datos necesarios en un formulario, editar estos mismos datos a través del mismo formulario de un dispositivo ya existente, refrescar la lista de dispositivos (y obtener la configuración de éstos) y, por último, eliminar dispositivos existentes en la base de datos.
2. **Aplicación de configlets**: mediante a la API de Junos Space, de la cual hablaremos en el Capítulo 4, podemos aplicar configuraciones de manera remota en dispositivos Juniper que estén registrados en esa plataforma. De esta manera, facilitamos el *push* de configuraciones de una manera más sencilla y a nivel remoto, sin tener que entrar en la plataforma o en la terminal del equipo o los equipos en cuestión.
3. **Algoritmo para búsqueda de IPs-MAC a través de tablas ARP**: aquí llegamos al punto más "interesante" de la aplicación. He desarrollado un algoritmo para descubrir

a qué máquina final (es decir, un nodo que puede ser desde una máquina virtual, impresora o cualquier dispositivo end-point que esté conectado a un dispositivo de red que tengamos registrado hasta otro dispositivo de red que no pertenezca a nuestro dominio) pertenece una dirección, ya sea MAC o IP. Esta labor es de gran importancia para mi grupo de trabajo, pues con una simple búsqueda pueden identificar cuál es el dispositivo que se corresponde a esa IP/MAC, de qué dispositivo físico depende (ya sea un router o un switch) y, por tanto, saber en qué sede y ubicación física se encuentra para, en caso de emergencia, poder hablar con la persona responsable y poder solucionar el problema de manera presencial si fuera necesario.

3.1.1. Aplicación de configlets

Un configlet es una plantilla de configuración sobre la que podemos modificar datos para, posteriormente, realizar un *push* de esa misma configuración al dispositivo o a los dispositivos que lo soporten [15]. Los configlets, además de un nombre, ID, y tipos de dispositivos a los que se puede aplicar, contienen dentro de ellos un fragmento de código con parámetros. Estos parámetros pueden ser predefinidos por el usuario, calculados en tiempo de ejecución o estar vacíos para que el usuario introduzca en ellos los datos que desee aplicar.

Así pues, en la aplicación habrá un apartado donde se podrá mostrar al usuario la lista de configlets disponibles con sus correspondientes datos, y el usuario podrá decidir cuál de ellos aplicar y a qué dispositivos. También, se permitirá seleccionar un dispositivo y ver qué configlets se pueden aplicar a esa familia de dispositivos. En ambas opciones, se muestra un formulario con los parámetros del configlet para poder validarlo y mostrar el fragmento de código real que se va a aplicar al dispositivo, antes de que el usuario confirme definitivamente la aplicación de este configlet, siempre y cuando la validación tenga éxito.

3.1.2. Búsqueda de IPs-MAC a través de tablas ARP

Para comprender cómo funciona esta búsqueda, es necesario explicar ciertos conceptos previamente:

- **Interface**: una interface (interfaz en español) es la capa que formatea los datagramas IP de la capa de red para que las tecnologías de red puedan interpretarlos y transmitirlos [30]. Estas interfaces están conectadas a un controlador que permite la comunicación entre el propio dispositivo y el dispositivo conectado a esa interfaz. Coloquialmente hablando, también se llama interfaces a los puertos lógicos que tiene un dispositivo de red, aunque el puerto simplemente sea el adaptador conectado a esa interfaz. De esta manera, un dispositivo tiene múltiples interfaces que pueden estar, conectadas o no, a otros dispositivos, ya sean dispositivos de red o dispositivos finales. Además de estas interfaces físicas, es decir, las que se corresponden con un puerto físico, tenemos interfaces virtuales (llamadas en algunos sitios *subinterfaces* que no se corresponden con ningún puerto físico pero también existen para la comunicación del equipo entre dos o más redes de área local virtual (conocidas como VLANs).

En un dispositivo de red, podemos comprobar qué interfaces tiene ese dispositivo y la descripción de éstas, cuya definición está editada por un administrador. Esto resulta de gran utilidad, pues da al administrador, en caso de consulta, información relevante acerca de qué hay conectado en esa interfaz.

- **ARP**: ARP, de las siglas `Address Resolution Protocol`, es un protocolo de comunicaciones perteneciente a la capa de enlace de datos. Se encarga de encontrar la dirección física (es decir, la MAC) del hardware vinculada a una IP [4]. Dicho de otra manera, cada entrada de una tabla ARP de un dispositivo de red es una relación entre una IP y su dirección física MAC. En nuestro caso, cada vez que se haga una búsqueda, como se introduce una IP o una MAC, se puede saber dicha asociación preguntando a todos los routers y firewalls, pues a ellos se correctan el resto de dispositivos de la red. Es por ello que tanto los routers como los firewalls, aunque no tengan directamente conectada la máquina o dispositivo que estamos buscando, almacenan en su configuración ese registro ARP, lo que nos lleva al siguiente paso.

  En los dispositivos de red, esta tabla nos da información relevante, pues cada entrada nos indica qué IP está asociada a qué MAC, y el nombre de la interfaz a través de la cual ese dispositivo ve al dispositivo buscado.

- **Ethernet-Switching**: la tabla Ethernet-Switching nos amplía la información que hemos visto en el punto anterior. Dicha tabla muestra, por cada dispositivo, todas las direcciones MAC que ve, a través de qué interfaz están conectadas y a través de qué VLAN las está detectando. En otras palabras, esta tabla es un registro de las direcciones MAC conectadas a las interfaces de ese dispositivo.

- **Neighbors**: por último, los Neighbors (vecinos en español) son los dispositivos conectados a cada una de las interfaces. Dentro de la configuración de los dispositivos de red, existe una tabla que muestra qué hay conectado al otro lado de cada interfaz, y una descripción de esto. Al otro lado puede haber otro dispositivo de red o un dispositivo final. Si está conectado a otro dispositivo, en la columna `device_id` aparecerá la interfaz a la que está conectada y, en la información del puerto, aparecerá el nombre del dispositivo. Si no fuera así y estuviera conectado a un dispositivo final (por ejemplo, a una máquina virtual o a un ordenador), el contenido de la columna `device_id` mostrará otro tipo de información; además, la columna de la información del puerto mostrará el nombre de dicho dispositivo final (si así lo ha configurado previamente el administrador de esa máquina). En resumen, esta tabla nos da información sobre lo que está conectado directamente a las interfaces de un dispositivo.

Una vez explicados brevemente los conceptos clave, vamos con la explicación del algoritmo. Para ello, vamos a describir un escenario imaginario.

Supongamos que a un administrador de redes le llega una incidencia que describe que un dispositivo falla. No llega la señal, no hay manera de conectarse a él, pero no sabemos ni dónde está ni qué es. Sólo tenemos o bien su dirección MAC o, por el contrario (lo que suele ser más común), su dirección IP. Queremos saber su ubicación (por si acaso hubiera habido algún problema con el cable o si no está bien conectada la máquina que le da soporte), su nombre, y cuándo fue la última vez que un dispositivo principal (router o firewall) lo vio en

su tabla ARP por última vez. El flujo que sigue nuestro algoritmo desde que introduces una IP o una MAC y devuelve un resultado son los siguientes:

1. **Comprobar si es IP o MAC**: este paso quizá es el más obvio, pero es necesario explicarlo. Mediante expresiones regulares, se hace un match con la cadena de texto introducida por el usuario y comprobamos, entre otras cosas, si el formato es correcto, si contiene espacios, etc. Es necesario mencionar que el formato de las direcciones MAC cambia entre fabricantes [44]. Mientras que Juniper utiliza el formato de separación con dos puntos cada 2 dígitos, en Cisco utilizan varios indistintamente. Es por ello por lo que necesitamos hacer una comprobación sobre si estamos introduciendo una IP o una MAC y, en caso de haber introducido una MAC, pasarlo a un único formato y comprobar que es correcto.

2. **Buscar si existe esa IP/MAC**: indistintamente de lo que hayamos introducido, debemos buscar en la tabla ARP de los routers y firewalls para ver si existe un registro ARP que asocie la IP con la MAC o viceversa. Si existe (ya sea en uno o varios) significa que se puede encontrar pero, si no, significa que no hay ninguna MAC o IP en nuestra red con ese valor.

3. **Buscar en las tablas Ethernet-Switching**: una vez que hayamos comprobado que existe, nos interesa el valor de la MAC. Ahora, en vez de buscar en los routers y firewalls, buscaremos en los switches, más concretamente en sus tablas Ethernet-Switching. Si en esa tabla existe un valor con esa MAC, nos dirá a qué interfaz está conectada y la VLAN por la que ve esa conexión. En este punto es importante destacar que podemos ver varios dispositivos que ven esa dirección MAC, pero debemos saber qué dispositivos de ellos están conectados directamente y cuáles no, pues sólo nos interesa el dispositivo final que está conectado de manera directa a la dirección buscada. Si sólo ha encontrado la dirección MAC en un único dispositivo, podemos deducir que es el dispositivo final. Si no fuera así, pasamos al siguiente punto.

4. **Detectar si el dispositivo es un dispositivo final**: aquí entran en juego varias tablas y recursos. Lo primero sería comprobar si la interfaz a través de la que un dispositivo ve esa dirección es una interfaz física final. Antes hemos comentado que también se llaman interfaces a los puertos de un dispositivo, pero una interfaz también puede ser lo que denominamos un *agregado*, que es una interfaz virtual que agrupa varias conexiones. De nuevo, tenemos una bifurcación del algoritmo:

   a) **La interfaz es una interfaz lógica**: comprobamos si el nombre de la interfaz se corresponde con un puerto gracias a una expresión regular. Si se corresponde con el puerto, sabemos con certeza que la MAC que hemos buscado está conectada a través de esa interfaz al dispositivo en el que hayamos consultado la tabla Ethernet-Switching del punto anterior.

   b) **La interfaz no es una interfaz lógica**: puede ser que la interfaz corresponda con un agregado o una interfaz virtual. Si esto ocurre, podría darse el caso de que el dispositivo resultado, pese a no estar directamente conectado al dispositivo que estamos analizando, sí sea una dirección final y, por tanto, de utilidad a la hora de mostrarlo en los resultados. Esto puede darse en el caso de buscar, por ejemplo, una IP de un switch conectado a los equipos de red de mi equipo de trabajo pero,

en esta ocasión, dicho switch será de otro dominio empresarial que no pertenece a mi área de trabajo. En este ejemplo, como se puede saber qué hay conectado a ese switch (puesto que no es de nuestra propiedad y por motivos de seguridad no se podría), ese dispositivo se debería incluir como resultado de nuestra búsqueda, pese a no ser un dispositivo final. Para saber cómo se puede resolver este caso, debemos fijarnos en el siguiente punto.

c) **Consultamos las tablas de vecinos**: si tenemos una dirección que cumpla las condiciones del punto anterior, debemos comprobar si es o no una dirección asociada a un dispositivo final. Aquí es donde entran en juego las tablas de vecinos (*Neighbors*). En estas tablas obtenemos información sobre los dispositivos conectados a un dispositivo en cuestión. Así, comprobamos si lo que está viendo nuestro dispositivo a través de esa interfaz es un dispositivo final o no. La manera de comprobarlo es, de nuevo, con una expresión regular. Si el nombre del dispositivo que hay al otro lado de la interfaz coincide con uno de los dispositivos almacenados en nuestra base de datos, significa que lo que estamos viendo al otro lado es un dispositivo de red (un firewall, un router o un switch), por lo que no debemos incluir este dispositivo en el resultado. Si no fuera así, significa que lo que hay al otro lado no es otro dispositivo de red, lo que significa que es un dispositivo final válido para incluirlo en el resultado.

Podemos ver el workflow del algoritmo de búsqueda desglosado en las imágenes 3.1 y 3.2. En ellas, lo mostrado en verde equivale a métodos que muestran información al usuario; los métodos rojos, errores que se muestran en un componente Alert para informar al usuario qué ha ido mal; en azul, lógica interna (asignaciones, incrementos, etc); los condicionales los representamos con rombos rojizos; y, por último, en amarillo tenemos los métodos generales que incluyen lógica y llamadas a la base de datos. El algoritmo completo lo podemos encontrar en la Figura 3.3.

Una vez que hemos visto detalladamente cómo funciona nuestro algoritmo principal de la búsqueda ARP, debemos mencionar una serie de anotaciones a modo de "disclaimer" a tener en cuenta:

- A veces puede haber más de un resultado. La única manera que hay de saber si un dispositivo cumple con las condiciones para incluirlo como "dispositivo final" es con expresiones regulares gracias a las descripciones de los datos en las configuraciones de los dispositivos. Si un administrador, por error o porque no ha podido actualizar ciertos datos, no incluye la descripción adecuada en el dispositivo, ésta se mostrará de distinta manera en la tabla correspondiente. De esta manera, podría incluirse alguna dirección no-final o no deseada. No obstante, el administrador de dispositivos puede saber fácilmente qué direcciones descartar en caso de que haya más de un resultado debido a la información que se muestra en los campos de búsqueda gracias a la información mostrada como resultado final.

- Otro de los motivos por los que puede haber más de una dirección es por las distintas configuraciones entre fabricantes. Mientras que Juniper pone a su disposición unas herramientas bastante útiles a través del paquete PyEz mencionado en el Capítulo 4,

Cisco tiene distintos formatos entre versiones, formatea sus tablas de distinta manera y permite una serie de formatos que Juniper no permite. La única solución ha sido extraer la información mostrada como texto y darle el formato deseado, por lo que a veces puede ser insuficiente con las reglas que he creado. Contar con una batería de pruebas proporcionada por el Product Owner en estos casos ha sido de gran utilidad, pues ha permitido detectar algunos dispositivos que tenían versiones distintas de sistema operativo y, por tanto, mostraban las tablas de manera distinta.

- La tabla Neighbors muestra informaciones diferentes al ejecutar el comando para obtenerla entre los distintos fabricantes. Esto ha supuesto que se haya tenido que crear una tabla en la base de datos adaptada a los campos que se necesitan.

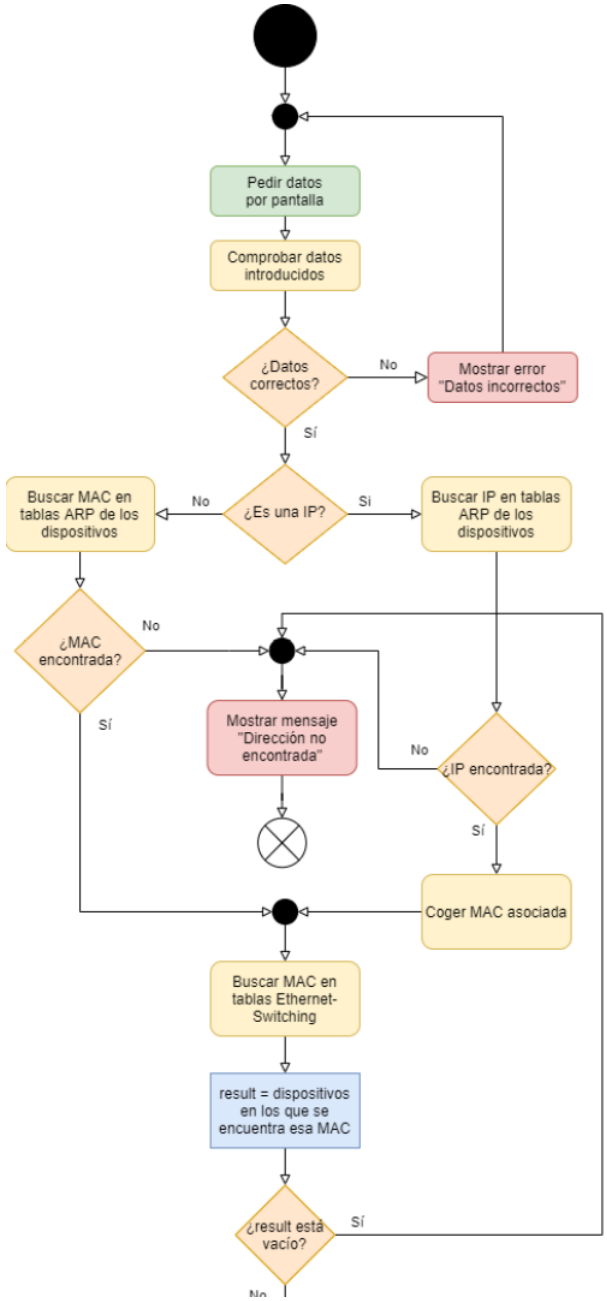


Figura 3.1: Workflow del algoritmo de búsqueda ARP (1/2)

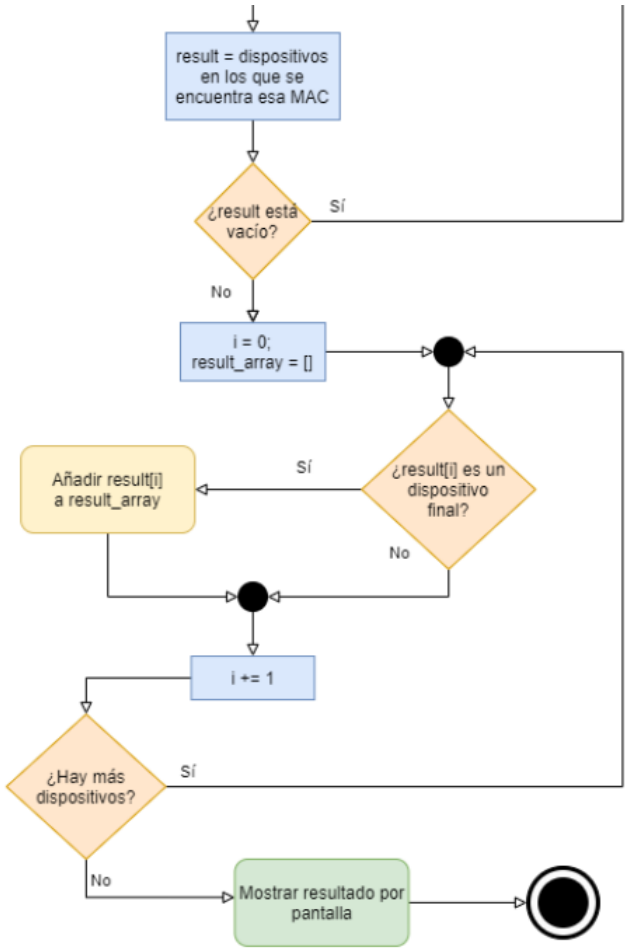


Figura 3.2: Workflow del algoritmo de búsqueda ARP (2/2)

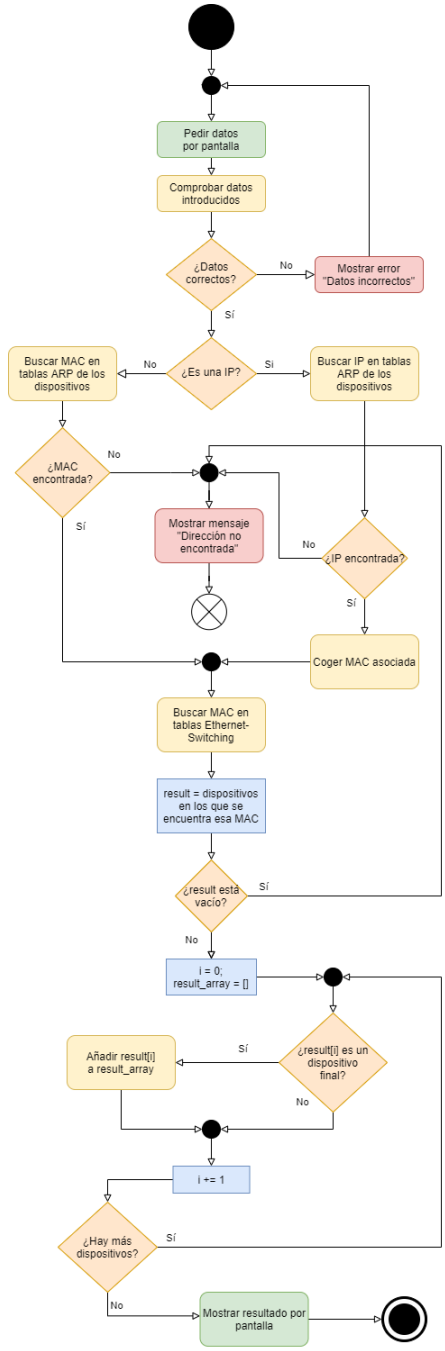


Figura 3.3: Workflow completo del algoritmo de búsqueda ARP (detalle por partes en las Figuras 3.1 y 3.2)

3.2. Modelo conceptual del dominio

En la Figura 3.4 podemos ver el modelo conceptual del dominio. Ha sido realizado tras analizar bien las Historias de Usuario del Backlog, recoger suficiente información sobre cómo funciona la búsqueda a través de entradas ARP y tener la experiencia necesaria para realizar estas consultas con una conexión a la terminal de los dispositivos. En la imagen se muestran las entidades necesarias para cumplir todas las necesidades anotadas en el backlog. Podemos ver una clase principal, llamada Devices, sobre la cual giran el resto de clases. He representado también las tablas mencionadas anteriormente en la búsqueda ARP a la hora de explicar el algoritmo de búsqueda, que están conectadas a estos dispositivos. A su vez, también he incluido enums para saber el fabricante y el tipo de conexión del propio dispositivo. Por último, destacar que el tipo de dispositivos es una generalización de la clase Device del tipo *disjoint*, lo que significa que un dispositivo sólo puede ser de la clase de una de las instancias a la vez. Esto tiene sentido, pues un dispositivo es un router, un switch o un firewall y, aunque algunos dispositivos puedan hacer parte de las funciones de otros, no son el mismo tipo de dispositivo.

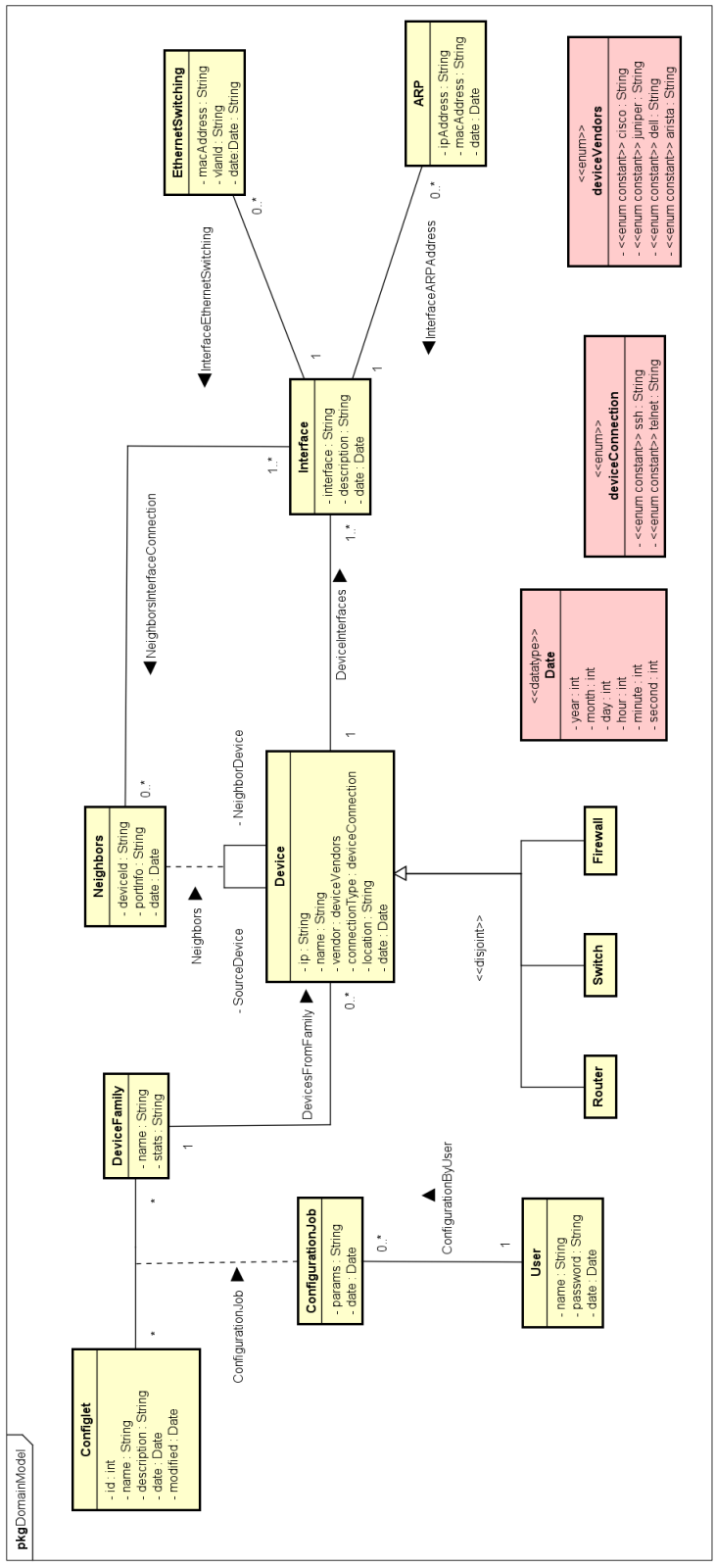


Figura 3.4: Modelo de dominio final

Capítulo 4

Tecnologías utilizadas

En este capítulo se presenta un resumen de las tecnologías utilizadas tanto para la gestión del proyecto, como para el desarrollo. Comenzaremos viendo las tecnologías utilizadas principalmente para gestionar y documentar el proyecto, además de las necesarias para llevar día a día un seguimiento de éste. Continuaremos hablando de los lenguajes de programación utilizados y, por último, comentaré algunas de las librerías importantes que han sido necesarias para llevar a cabo las tareas más críticas del proyecto.

4.1. Herramientas y programas utilizados

4.1.1. Jira

Comencemos hablando de Jira. Jira es una herramienta creada por Altassian orientada a la planificación de proyectos ágiles. Permite planificar, supervisar, crear flujos de trabajo y categorizar tareas, además de añadir *issues* en éstas. En cada tarea, se puede llevar a cabo un seguimiento por parte del usuario que tiene el rol de encargarse de ésta, ofreciendo la opción de poner un informador y un supervisor cada tarea, además de añadir comentarios e ir cambiando el porcentaje completado a medida que se avance en dicha tarea [35]. También permite permite integración y entrega continuas pero, en este caso, no se ha hecho uso de estas capacidades de Jira.

En mi caso personal, he utilizado Jira para crear una tarea por cada H.U. aunque, también, he utilizado esta plataforma para otras tareas utilizadas en mi grupo de trabajo no relacionadas con este proyecto, lo cual me ha permitido comprender mejor esta herramienta. Una vez que la historia de usuario se completa, se cambiaba el porcentaje de completitud de la tarea a 100 % para, después, mover la tarea a la columna de completados. Esta acción notifica al supervisor, cuya tarea es comprobar que la documentación es correcta y ver que esa tarea, efectivamente, ha sido realizada y funciona correctamente.

4.1.2. Git

Para hablar de Git, comencemos hablando de los Sistemas de Control de Versiones (comúnmente conocidos como *Version Control System* VCS). Un VCS es un sistema que permite realizar un seguimiento de los cambios realizados en el código de un conjunto de archivos seleccionados por un usuario de tal manera que se cree un repositorio para que el usuario pueda volver atrás y recuperar versiones anteriores de ese mismo código [6]. Aunque suene simple, se ha desarrollado tanto y de tal manera que permite una lista inmensa de posibilidades, además de integración con otras plataformas y herramientas que amplían todavía más las capacidades de esta tecnología.

Hay múltiples sistemas de control de versiones. De todos ellos, el más popular a nivel mundial es Git. Desarrollado por Linus Torvald en 2005 [2], el creador de Linux, este VCS gratuito de código abierto se ha convertido en el más popular [18] y el que más crecimiento ha tenido en el mundo de la informática, como podemos ver en las Figuras 4.1 y 4.2.

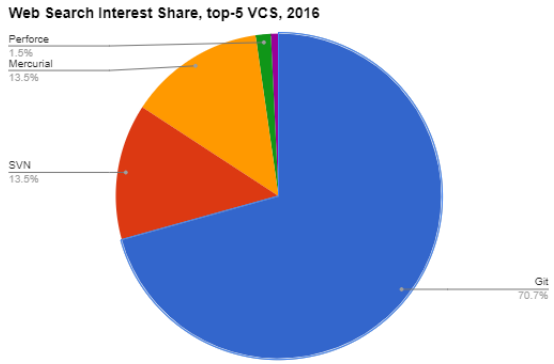


Figura 4.1: VCS más usados [63]

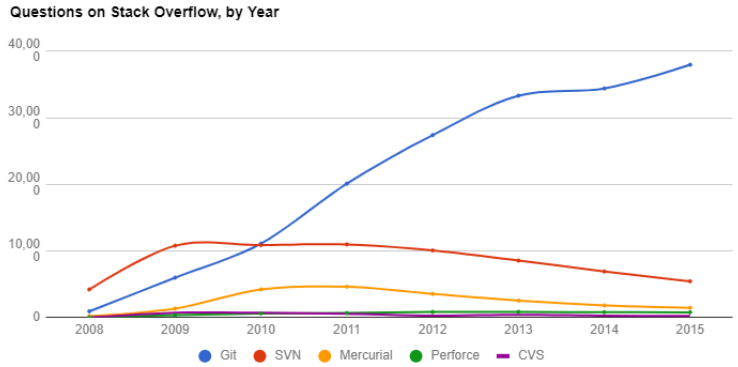


Figura 4.2: Evolución de las preguntas en Stack Overflow [63]

Git opera bajo los siguientes principios [55]:

- **Copias instantáneas, no diferencias**: en vez de manejar los cambios que se han hecho en cada archivo, Git maneja sus datos como un conjunto de imágenes hechas en un instante. Cada vez que se confirma un cambio y se guarda, se genera una imagen de ese archivo y guarda la referencia a esa imagen generada, de tal manera que si el archivo no se ha modificado, Git no almacena el archivo de nuevo, sino que mantiene el enlace anterior que apunta a la imagen generada.
- **Casi todas las operaciones son locales**: Git no necesita una plataforma remota para operar, puesto que se puede utilizar de manera local. No obstante, cuenta con plataformas donde puedes almacenar repositorios de manera remota, como GitHub o Gitlab.
- **Integridad**: antes de almacenar cada archivo, Git verifica éste previamente mediante una suma de comprobación. De esta manera, no se puede perder información durante la transmisión o sufrir corrupción de datos sin que Git pueda detectarlo.
- **Git generalmente solo añade información**: al añadir información, es muy difícil que una acción no se pueda deshacer o volver a un estado anterior. Esto permite trabajar de una manera segura, pudiendo siempre revertir estados no deseados en los proyectos.
- **Los tres estados**: aquí reside el núcleo de la filosofía de Git. Este VCS funciona a través de tres estados principales en los que se puede encontrar uno o varios archivos. Los estados son *Commited*, *Modified* y *Staged*, haciendo referencia, en castellano, a confirmado, modificado y preparado, respectivamente. Cuando un archivo dentro del directorio de trabajo (*Working Directory*) se modifica, su estado cambia. Aquí podemos añadir el archivo para la siguiente confirmación, lo cual cambia su estado a Staged y, por último, cuando esté todo ya hecho, si confirmamos los cambios, todos los archivos modificados que se hayan marcado para incluirlos en la próxima confirmación, pasan al estado confirmado. En la Figura 4.3 podemos ver este proceso de una manera más gráfica.

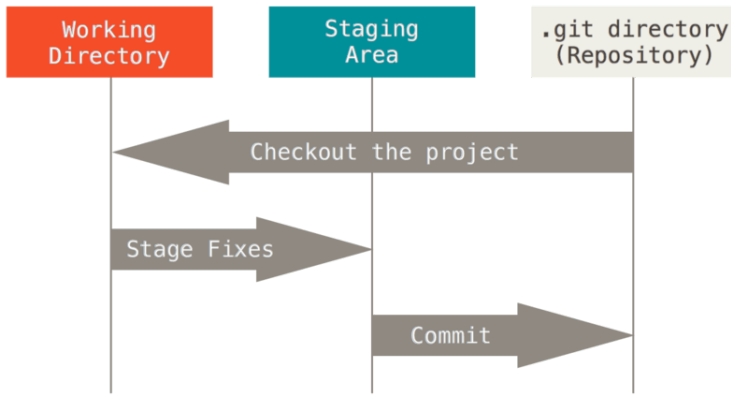


Figura 4.3: Los tres estados de Git [55]

Visto esto, hemos utilizado Git para llevar un seguimiento de nuestro proyecto. La manera de trabajar ha sido abrir una rama *Branch* por cada H.U. La gestión de branches en Git permite realizar bifurcaciones en el flujo del código. Si quiero trabajar de manera paralela en una funcionalidad para, una vez esté acabada, incorporarla al proyecto general, puedo crear una nueva rama para ese propósito. De esta manera, cada H.U. se ha gestionado con una rama diferente y cada vez que se finalizaba el código y se probaba que fuera correcto, se fusionaba (lo que se conoce como *merge* en Git) en la rama principal, llamada *Master*. Esta característica es de gran importancia, pues permite trabajar simultáneamente en varias historias de usuario sin tener que preocuparte por la integración de éstas o, lo que suele ser más común, evita tener que depender de que no haya cambios en otros segmentos de código que necesitas para que funcione lo que quieres modificar [23].

Podemos ver, en la Figura 4.4, un ejemplo gráfico de cómo funcionan estas *branches* en Git.

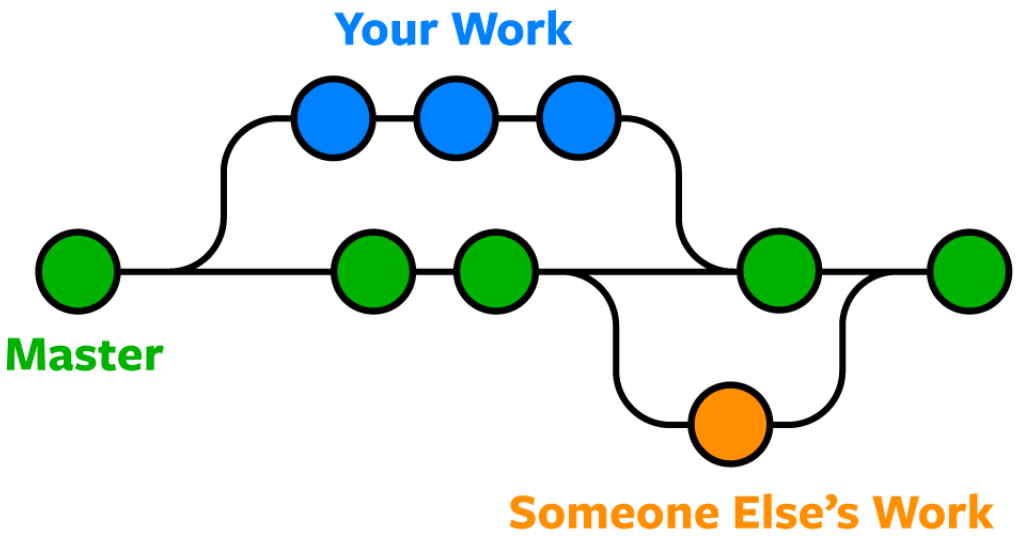


Figura 4.4: Ramas en Git (fuente: NobleDesktop)

Por último, aunque nos podríamos extender mucho en las múltiples bondades que ofrece Git, una parte importante es, como hemos comentado, la gestión de repositorios de manera remota en plataformas como Gitlab o Github. Esto amplía los horizontes para poder compartir código, además de permitir almacenar y descargar de manera remota, lo que potencia la accesibilidad y el trabajo simultáneo de varias personas en un mismo proyecto. A su vez, este tipo de plataformas amplían las funcionalidades de Git, elevándolo como VCS y haciendo que su utilización sea más que recomendable para cualquier tipo de proyecto, tanto grande como pequeño.

4.1.3. Balsamiq Wireframes

Balsamiq Wireframes es una herramienta para realizar bocetos, dibujos rápidos o mocks visuales para tener una idea de cómo se quiere maquetar una interfaz [9]. En mi caso, ha sido de gran utilidad por su facilidad de uso y su rapidez a la hora de realizar bocetos, lo cual ha permitido establecer y ordenar ideas a la hora de diseñar una interfaz web que luego he tenido que maquetar.

Sustituyendo al método tradicional, donde siempre ha sido común equivocarse con el lápiz, borrar, tachar, repetir el dibujo, etc., esta herramienta permite diseñar de manera muy intuitiva las vistas, y cuenta con componentes muy útiles que simulan los componentes más comunes de cualquier interfaz visual.

Podemos encontrar los bocetos de las vistas realizados en el Capítulo 5, donde se detallan qué bocetos se realizaron para plasmar las vistas iniciales. Posteriormente, a medida que la historia de usuario que incluya esa vista se vaya realizando, la vista definitiva puede cambiar respecto al boceto inicial.

4.1.4. Junos Space

Como vimos anteriormente, Junos Space es una herramienta que tiene funcionalidades muy similares a las que da soporte nuestra aplicación. Diseñada para operar con dispositivos de la marca Juniper, permite multitud de acciones, ya sea modificar configuraciones de dispositivos, monitorización, control de usuarios y seguridad, etc. [37]. Aunque la idea de este proyecto es centralizar las acciones para poder operar con varios dispositivos desde una misma plataforma, la API de Junos Space no permite realizar todas las operaciones deseadas, por lo que he tenido que aprender cómo utilizarla y llevar a cabo las acciones necesarias para poder comunicarme con ella a través de la API y, de esta manera, integrar sus funcionalidades en mi proyecto. El aspecto más importante que no nos permite hacer a través de llamadas a su API es la creación de plantillas de configuración de dispositivos llamadas Configlets. Mientras que gran parte de la funcionalidad de mi aplicación se centra en la aplicación de estas plantillas en los dispositivos deseados, comprobar si la información que se ha introducido está bien formulada y, por último, ver si el dispositivo lo valida, no ha sido posible implementar la creación de manera remota de estas plantillas debido a que su API no lo permite, lo cual obliga a crearlas desde la propia plataforma y, una vez creados, aparecerán en la vista correspondiente para poder aplicarse de manera remota.

4.1.5. Astah UML

Herramienta para trabajar con diagramas en el lenguaje de modelado UML. Permite la creación de múltiples diagramas como, por ejemplo, Diagramas de flujo, CRUD, Modelos de Dominio, de secuencia, etc.[7] Es una opción que estaba clara, debido al uso durante muchas de las asignaturas de la carrera. Por este motivo y al estar familiarizado con esta herramienta, ha sido la opción escogida para realizar algunos diagramas como, por ejemplo, el modelo de

dominio (Figuras 6.1, 6.2, 6.3, 6.4) aunque también se puede encontrar en otros diagramas realizados en este proyecto.

4.1.6. MySQL Workbench

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, desarrollada por Oracle, que permite crear, administrar y desarrollar de manera gráfica bases de datos. Para este proyecto, se ha utilizado tanto para desarrollar un pequeño esquema funcional y hacer pruebas poblando las tablas desde el software desarrollado, como para depurar errores, además de permitir realizar cambios a través de la interfaz gráfica evitando así tener que reestructurar y recompilar las tablas a través de comandos. Por último, es importante destacar su función para crear esquemas relacionales de bases de datos. Esta utilidad ha permitido diseñar las tablas utilizadas para mostrar el modelo relacional (Figuras 6.5, 6.6, 6.7, 6.8).

4.2. Lenguajes de programación y Frameworks utilizados

4.2.1. Javascript

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web. Entre sus ventajas, podemos mencionar que es ligero, interpretado o compilado *just-in-time*. Se usa en muchos entornos fuera del navegador (como `Node.js`, como veremos más adelante). En cuanto al paradigma, podemos decir que es un lenguaje multiparadigma, dinámico, monohilo y con programación imperativa y declarativa [34].

En cuanto a su función, se introdujo en 1995 para poder añadir programas o fragmentos de código a las páginas web HTML [51]. Es por ello por lo que ha ganado gran popularidad, debido a su uso, principalmente, en entornos de desarrollo web, además de multitud de Frameworks. Dentro de una página web, podemos diferenciar tres capas: capa lógica, que contiene la funcionalidad de la web; capa de *layout* que se encarga de maquetar y diseñar los elementos que va a contener la web; y la capa de estilo, quien decide cómo se muestran los elementos, la organización de estos en la pantalla, el tamaño, color, etc. Javascript se encarga de la capa lógica, pues permite realizar programas, métodos y funciones dentro de una vista en el propio navegador.

Para el proyecto, he utilizado Javascript para toda la lógica front-end de la aplicación, desde clases y funciones hasta llamadas `Axios` para comunicarme tanto con APIs externas como con la API del back-end.

4.2.2. Python

Python es un lenguaje de programación multiparadigma, interpretado, cuya filosofía destaca por la facilidad a la hora de leer el código [5]. Nace a finales de la década de los ochenta, creado Guido van Rossum en los Países Bajos [59]. A lo largo de los años ha ido evolucionando de tal manera que se puede incorporar en múltiples campos de la informática, desde programas matemáticos hasta páginas web, pasando por scripting, aplicaciones gráficas, inteligencia artificial, etc. [50]

En este proyecto, toda la lógica de back-end y login se ha realizado con Python, más concretamente, con un framework escrito en Python llamado *Flask*, del cual hablaremos más adelante. La decisión a la hora de elegir Python como lenguaje para el back-end vino determinada por las librerías tanto para comunicarse con los dispositivos, como para realizar el login a través del protocolo `Tacacs+`. Al tener ya las librerías desarrolladas por el propio fabricante para comunicarme con los dispositivos, además de la imposibilidad de realizar el login utilizando Tacacs+ al no haber módulos disponibles en Javascript, junto a conocer bastante el lenguaje Python al haberlo utilizado para desarrollar algún bot, decidí implementar el back-end utilizando esta tecnología.

4.2.3. SQL

Denominado *Structured Query Language*, es decir, Lenguage de Consulta Estructurada, es un lenguaje que surge en los laboratorios de IBM cuando se creó el nuevo software de base de datos System R. A lo largo de los años ha ido evolucionando, convirtiéndose prácticamente en el estándar de los lenguajes relacionales de base de datos. Utilizado como lenguaje en los principales gestores de base de datos, en 2019 se utilizó en un 60 % de las bases de datos de todo el mundo, frente a menos de un 40 % que no utilizaban este tipo de lenguaje[1].

En cuanto a los sistemas de gestión de bases de datos relacionales, en este caso nos hemos decantado por MySQL. Este gestor, desarrollado por Oracle [46], es el más utilizado y, al haber trabajado previamente con él, he decidido utilizarlo. En la Figura 4.5 podemos ver un gráfico con los sistemas de gestión de bases de datos más usados y el porcentaje de utilización que tienen a nivel global.

4.2.4. HTML y CSS

Para finalizar con los lenguajes, vamos a hablar de HTML y CSS. Como hemos comentado antes, Javascript forma parte de nuestra parte lógica de las páginas web. Pues bien, HTML y CSS son los lenguajes con los que implementamos la capa de layout y de estilo, respectivamente.

HTML, las siglas de *HyperText Markup Language*, es un lenguaje de marcado que permite elaborar páginas web [28]. Es un estándar para estas páginas, y supone el esqueleto de una página web. HTML no es un lenguaje de programación; utiliza marcas para etiquetar sus

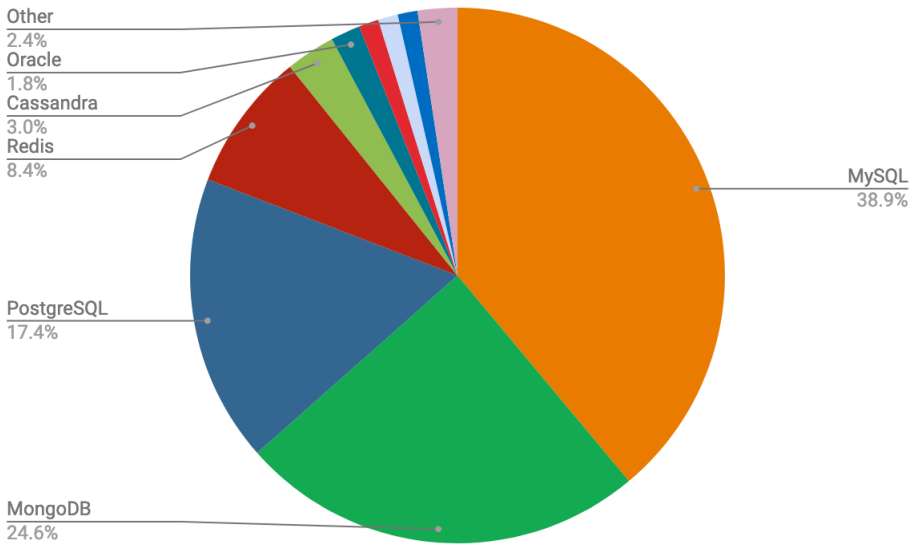


Figura 4.5: Sistemas gestores de bases de datos más usados (2019) [1]

elementos, ya sean tablas, texto, párrafos o contenedores. Pese a que las nuevas versiones incluyen aspectos relativos a la apariencia (como dar color al texto, formato a las tablas o sombreado a los párrafos, es CSS generalmente el encargado de realizar estas modificaciones estéticas.

CSS, Hojas de Estilo en Cascada (del inglés *Cascading Style Sheets*) describe cómo debe renderizarse el código HTML que se ha desarrollado previamente. Haciendo una analogía básica: se puede tener el elemento "ojos" en la estructura html, pero el CSS se encargará de decir si son grandes, pequeños, azules, marrones, rasgados, etc. De esta manera, con CSS damos el formato visual deseado y gestionamos la ordenación de elementos de nuestra web.

4.2.5. Vue.js

Vue.js es un framework progresivo muy potente utilizado para construir interfaces de usuario. Frente a otros frameworks monolíticos, Vue nos permite desarrollar el front-end de aplicaciones web del tipo SPA (*Single Page Application*, lo que nos permite descomponer un proyecto en múltiples vistas pero siempre bajo la misma página, sin necesidad de refrescar la dirección web del navegador para cambiar entre ellas [53].

En otras palabras, Vue.js es un framework basado en componentes, los cuales podremos desplegar donde necesitemos. Los componentes, a su vez, son de un solo archivo *Single File Components*, al igual que vimos con las SPA. Como cada componente está compuesto por un único archivo, éste tiene que tener dentro de sí las tres capas de las que hablamos anteriormente. De esta manera, cada componente tiene encapsulado los siguientes fragmentos, todos

ellos etiquetados con etiquetas HTML de apertura y cierre:

- **Template**: es la estructura HTML del componente. En ella describimos los elementos que queremos que tenga nuestro componente los cuales pueden a su vez ser otros componentes.

- **Style**: aquí podemos aplicar un estilo a los elementos definidos en el HTML. Podemos hacerlo con CSS, como es habitual y hemos mencionado anteriormente, o con SASS, un preprocesador de CSS que nos permite generar, de manera automática, hojas de estilo, añadiéndoles características que no tiene CSS [52].

- **Script**: es la parte lógica del componente. En ella se encuentran las variables, propiedades, declaraciones de componentes, métodos, etc. Vue.js permite que su lógica sea desarrollada tanto en Javascript como en Typescript, una variante tipada de Javascript.

  Dentro de este bloque, encontramos varios elementos importantes. A continuación, se enumeran los más importantes y los que más he usado a la hora de realizar las vistas [25]:

  - **name**: nombre del componente.
  - **components**: aquí declaramos los componentes que importamos para implementarlos en la vista.
  - **props**: son las propiedades del propio componente. Estas props deben pasarse al llamar al componente desde un punto externo; debe evitarse modificar las props desde el propio componente.
  - **data**: equivalen a los atributos de estado del propio componente. Al igual que las variables del props, las variables del data pueden llamarse desde el `Template` (es decir, desde el HTML) para mostrarlos y utilizarlos en la vista.
  - **methods**: aquí se definen los métodos y la lógica de la vista. Pueden ser utilizados por elementos del `Template` o, también, ser llamados por otros métodos.
  - **computed**: variables que requieren un cálculo, comprobación o modificación previa a su uso.
  - **watch**: observador que permite realizar acciones según si ocurre algún cambio en una variable ya sea del `prop`, del `data` o del apartado `computed`.
  - **mounted**: aquí se definen las acciones que se deben realizar al crear el componente. Pueden ser desde declarar un valor a una variable o llamadas a métodos.

Cada aplicación de Vue se comienza creando una nueva Instancia de Vue. El diseño de Vue, aunque no está estrictamente asociado con el patrón MVVM (Model-View-ViewModel)[39], está muy inspirado por él. Cada uno de los componentes que conforman una aplicación Vue es una instancia. Esto quiere decir que, cada componente, desde su creación hasta su destrucción, tiene un flujo prefijado en el cual se decide por qué etapas, métodos y pasos va a avanzar esta instancia. Estos métodos pueden ser utilizados en el `Script` del componente. Para ver de manera visual el ciclo de una instancia, debemos fijarnos en la Figura 4.6.

Antes de terminar de hablar de Vue.js y, en especial, de su filosofía Single File Components, debemos mencionar que cada sección de éstos (HTML, CSS o lógica) pueden mantenerse en ficheros externos y, desde el componente, hacer referencia a este fichero. No obstante, no es del todo recomendable para no romper en la medida de lo posible esta filosofía.

4.2.6. NPM

Node Package Manager, más conocido por sus siglas NPM, es un gestor de paquetes desarrollado en el lenguaje Javascript, del que hemos hablado antes. NPM nos permite agregar dependencias de forma simple, gestionar y distribuir paquetes en un entorno de directorios y controlar y administrar los módulos de un proyecto. En nuestro caso, NPM ha permitido instalar algunas dependencias necesarias para, entre otras cosas, desarrollar test unitarios, test end-to-end, paquetes de lenguaje, o instalar librerías de componentes [45].

Para hablar de los paquetes de NPM utilizados, podemos consultar el apartado siguiente. Allí hablaremos de paquetes, APIs, librerías, etc., que hemos utilizado en nuestro proyecto.

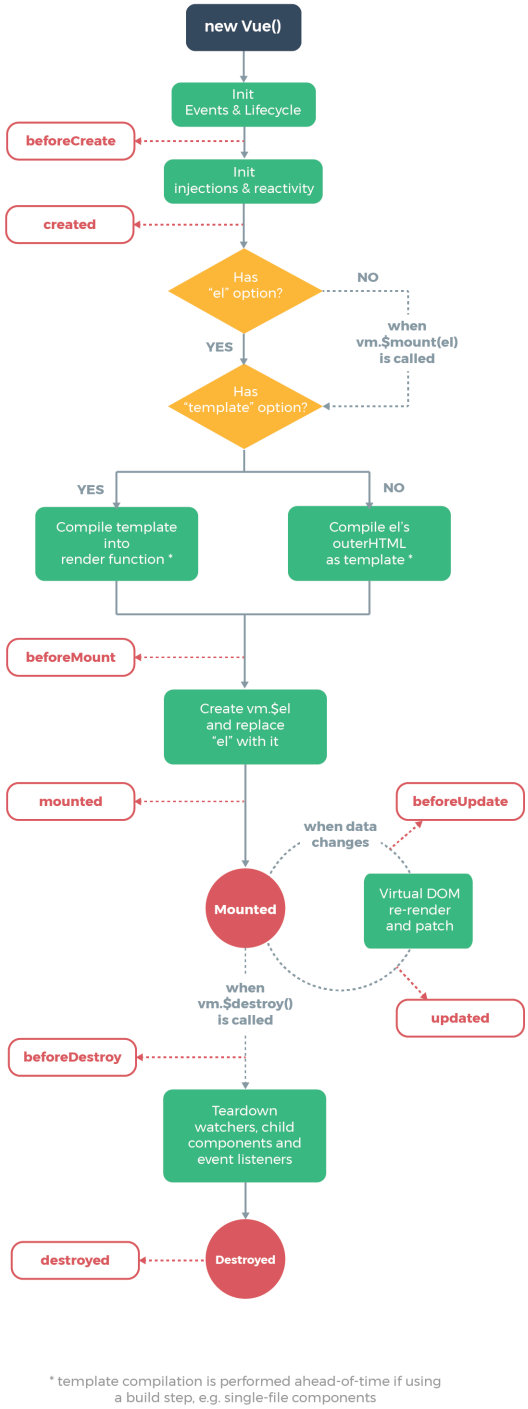


Figura 4.6: Ciclo de vida de una instancia en Vue [39]

4.2.7. Flask

Flask es un framework escrito en Python que permite crear aplicaciones web conectadas a un back-end con una cantidad de líneas de código muy baja respecto a otros frameworks [21]. Como nuestro proyecto necesitaba algunas librerías en Python, la elección del lenguaje para el back-end estaba condicionada por esto y, como Flask permite realizar las vistas en Vue y conectarse con la API del back-end mediante llamadas Axios, escogí este framework para el desarrollo del back-end.

4.2.8. SQLAlchemy

Para hablar de SQLAlchemy, primero habría que definir qué es un ORM. Un **ORM**, de las siglas *Object Relacional Mapping*, es una utilidad que permite interactuar con una base de datos como si fuera un objeto [56]. De esta manera, se pueden ejecutar queries y otras acciones sobre la tabla de datos sin utilizar el lenguaje habitual SQL (aunque podemos hacer llamadas con ese lenguaje si lo necesitamos).

La ventaja de un ORM es que es independiente del motor de base de datos a utilizar. SQLAlchemy es un ORM que se puede implementar en Python y permite gestionar todos los aspectos de la base de datos, desde la declaración de las tablas de dominio, querys, la conexión con ésta, etc. Entre las ventajas que ya hemos comentado, podemos añadir que proporciona un nivel de abstracción adicional al usuario, sobre todo en cuanto a la gestión del pool de conexiones, además de aportar mucha flexibilidad al trabajar con la base de datos. También permite utilizar un propio lenguaje para las consultas a la base de datos en vez de utilizar SQL nativo, gestionando de una manera más "natural" para el programador a la hora de utilizar filtros en las consultas. Otra de las grandes ventajas que tiene es que devuelve los resultados en modo de objeto, pudiendo manipularlo de una manera más sencilla y ampliando las posibilidades que trae por defecto cualquier motor SQL.

Así pues, gracias a SQLAlchemy, podemos disponer de una mayor flexibilidad y comodidad al tratar las tablas de la base de datos como objetos, lo que facilita el tratamiento y uso de estos datos, ya sea para obtenerlos o para inyectarlos en la propia base de datos.

4.3. Paquetes, APIs, librerías, etc.

En este capítulo, una vez hemos hablado de los lenguajes, frameworks y tecnologías utilizadas, vamos a hablar de otros elementos también importantes a lo largo del desarrollo de nuestro proyecto.

4.3.1. APIs

En primer lugar, debo mencionar la API de la plataforma Junos Space [38]. Esta API nos permite extraer datos de los dispositivos, configlets y, además, realizar llamadas con los

métodos HTTP (Post, Get, Put, Delete...). De esta manera, facilitamos al usuario realizar acciones remotas desde nuestra web, implementando una interfaz más amigable. Aunque esta API permite multitud de operaciones, tiene algunas limitaciones. Por ejemplo, no se permite crear configlets con peticiones a la API al ser una tarea demasiado compleja y requerir múltiples comprobaciones. Aun así, si un usuario creara un configlet dentro de la plataforma Junos Space, este configlet se podría borrar o aplicar en uno o varios dispositivos desde la aplicación gracias a las opciones que permite esta API.

4.3.2. Paquetes, librerías y protocolos

Tacacs_plus

Tacacs_plus (o Tacacs+) es un protocolo derivado de TACACS (Terminal Access Controller Access-Control System) que permite controlar servicios relacionados con la seguridad, tales como la autorización y la seguridad[8]. Es un protocolo AAA, cuyas siglas significan *Authentication, Authorization and Accounting* [3].

Tacacs se implementa en los dispositivos de red para realizar una comprobación de la autenticación de un usuario contra unos servidores Active Directory (AD), quienes se encargan de comprobar quién es el usuario que se está autenticando, si tiene los permisos necesarios y si está introduciendo bien la contraseña. De esta manera, he creado un servicio mediante el módulo disponible en PyPi [58] que implementa una llamada con este protocolo a la IP del servidor AD que devuelve un mensaje HTTP 200 en caso de una autenticación válida, mientras que devuelve un HTTP 400 en caso de que el nombre del usuario o la contraseña sean incorrectos o, por otro lado, si son correctos pero el usuario no tiene permiso para acceder a estos servicios.

Paquetes de NPM

- **i18n**: este plugin de Node nos permite internacionalizar los textos en los idiomas que deseemos. Para ello debemos crear un fichero de Javascript por idioma y, en él, debemos definir las variables que incluirán la cadena de texto que deseemos mostrar. De esta manera, convenientemente en cada fichero tendremos el mismo número de variables pero el contenido de dichas cadenas estará en el idioma que deseemos representar [29].
- **mdi**: el paquete de Node MDI hace referencia a Material Design Icons, iconos utilizados en la aplicación, tanto en botones como en otras funcionalidades [41].
- **Vuetify**: también de la mano de Material Design, es un framework de componentes que simplifica la implementación de éstos en un entorno web. Evita que el desarrollador tenga que crear, por ejemplo, botones, sliders o selects, permitiendo importarlos con una funcionalidad y una estructura que podemos cambiar, pero facilitando enormemente la gestión de éstos [32]. En estos componentes se incluyen los MDI vistos anteriormente. Aunque Vuetify nos brinde la facilidad a la hora de implementar componentes, es el usuario quien decide en última instancia qué lógica debería tener el componente y, consecuentemente, implementar el código correspondiente.

- **Router**: uno de los plugins más importantes de Vue, pues facilita la creación y gestión de las rutas Vue [64]. Declarando las rutas en un archivo `router.js`, facilita la lógica de la aplicación al cambiar de rutas en la URL. Este plugin, a diferencia de otros, puedes decidir añadirlo al crear el proyecto.
- **Cypress**: este plugin es el que se ha usado para test end-to-end en la aplicación. Es el plugin que incluye Vue por defecto [17].

Pandas

Pandas es una librería de Python que nos permite utilizar estructuras de datos similares a los dataframes del lenguaje R. Se utiliza para tratamiento de datos, pues pueden representarse de manera tabular (con columnas, tipos, etc) o con series temporales [48].

Pandas es útil, pues permite leer y escribir datos en distintos formatos (por ejemplo, SQL, CSV, o Microsoft Excel), tratar estos datos y seleccionarlos en función de un valor o de la posición que ocupen, unir y separar datos, etc.

En mi caso, he utilizado Pandas debido a la facilidad de manejar tablas y manipularlas. Al leer tablas y datos de los dispositivos, Pandas tiene utilidades que permiten ir poblando tablas y, posteriormente, manipular estos datos para guardarlos en una base de datos.

4.3.3. Junos PyEz

Librería para Python del fabricante Juniper para conectar de manera remota con dispositivos Juniper [36]. Con ella podemos abrir conexiones y ejecutar comandos, ya sea para aplicar configuraciones o extraer tablas de datos. La ventaja frente a otras librerías es que permite extraer los datos encapsulados en un objeto, lo que facilita acceder a ciertos datos (por ejemplo, si queremos acceder a los datos de una de las columnas). Así pues, PyEz nos brinda bastantes facilidades a la hora de conectarse y poblar las tablas de la base de datos gracias a algunos métodos presentes en esta librería.

4.3.4. Netmiko

Al igual que hemos visto con Junos PyEz, necesitamos una manera de conectarse de manera simple a los dispositivos de Cisco y otros vendedores [47]. Pues bien, esta librería nos permite establecer conexiones con dispositivos de redes del fabricante Cisco. Pese a que es muy útil a la hora de permitirnos conectar con estos dispositivos, ya sea por el protocolo SSH o Telnet, no incluye funciones como el encapsulamiento de datos a la hora de hacer un `get` de una tabla del dispositivo, por lo que tenemos que inyectar nosotros a mano los datos de esa cadena de texto en una tabla que posteriormente será guardada en la base de datos.

Capítulo 5

Diseño

En el diseño arquitectónico de este proyecto se utiliza, principalmente, el patrón MVVM, desarrollando un diseño basado en componentes. A lo largo de este capítulo, vamos a presentar este modelo, además de otros elementos importantes como los bocetos de las vistas, las vistas finales de la aplicación o el modelo de despliegue, entre otros.

5.1. Model-View-ViewModel (MVVM)

MVVM, en español Modelo-Vista-Modelo de vista, es un patrón de arquitectura de software que consiste en la separación interfaz de usuario - lógica de la aplicación [54]. Este patrón fue descrito al público en 2005 por John Gossman [31] como variación del patrón MVC (Modelo-Vista-Controlador) ajustando a WPF (Windows Presentation Foundation). MVVM consta de tres elementos que describiremos a continuación:

1. Modelo (Model): se define como Modelo al conjunto de estructuras de datos que se utilizan para representar, entre otras cosas, el estado, el funcionamiento y la lógica de negocio de la aplicación.
2. Vista (View): elemento que se asocia con la interfaz de usuario. Esta capa se encarga de mostrar los elementos contenidos en la lógica al usuario y capturar los eventos que ocurran por parte del usuario con estos elementos.
3. Modelo de vista (ViewModel): ya hemos visto qué es el Modelo y qué es la Vista; pues bien, el modelo de vista se encarga de enlazar la interfaz de usuario con el modelo. Para ello, utiliza todos los métodos necesarios para comunicar al modelo la ocurrencia de una interacción o un evento capturado en la interfaz de usuario.

En la Figura 5.1 podemos comprobar cómo interactúan estos 3 elementos explicados anteriormente.

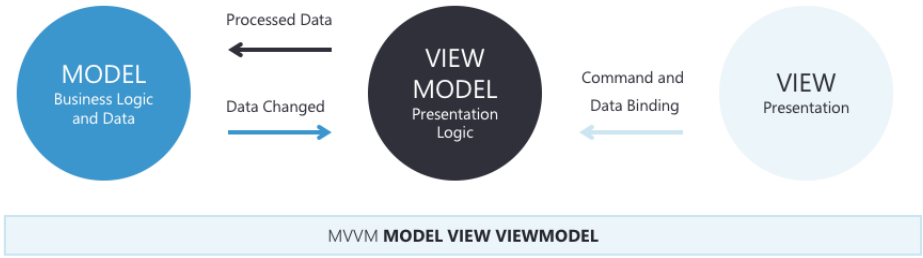


Figura 5.1: Elementos del patrón MVVM, obtenida en [10]

Como hemos visto, el patrón consiste en una separación entre 3 componentes. Esta separación también aporta algunas ventajas, entre ellas:

- Separar el desarrollo de la interfaz de usuario del resto de código.
- El desacoplamiento permite que la lógica pueda ser fácilmente testeada con test unitarios.

Estas ventajas hacen que el patrón MVVM sea útil cuando se estén desarrollando aplicaciones multiplataforma, pues nos aportará gran desacoplamiento para poder desarrollar con más facilidad. Una vez visto esto, podemos centrarnos en la pregunta ¿cómo se enlaza la vista con el Modelo?. Aquí entra el elemento estrella del patrón MVVM, el *binding*. El binding consiste en el enlace entre la Vista y el ViewModel. Gracias a este elemento, los datos del modelo se actualizan automáticamente cada vez que el usuario realice alguna acción que modifique dichos datos en la vista. Esto supone una mayor flexibilidad al tratar los cambios de estos datos.

5.1.1. Aplicación del patrón MVVM en este proyecto.

Como vimos en el Capítulo 4, el framework utilizado para desarrollar la interfaz de usuario ha sido Vue.js. Este framework facilita enormemente la aplicación del patrón MVVM ya que cada instancia de Vue está pensada para actuar como ViewModel. Así, se aprovecha el binding bidireccional que proporciona este patrón.

Para poder implementar este patrón, Vue.js pone a su disposición una serie de directivas que permiten relacionar los datos del modelo con elementos visuales de la interfaz de usuario, permitiendo (gracias al binding) manipular estos últimos mostrando los cambios en el modelo. A continuación, vamos a explicar algunas de las directivas más importantes que hemos utilizado en nuestro proyecto al aplicar este patrón:

- **v-if**: esta directiva condicional actúa como un controlador de flujo. Si el contenido que hay dentro de la directiva se cumple, los elementos visuales asociados a ella se

mostrarán en la interfaz de usuario; por el contrario, si no se cumple, todo lo que haya a continuación no se mostrará al usuario por pantalla, no incluyéndose en el DOM virtual de Vue. Esta directiva puede ir, opcionalmente, acompañada de las directivas `v-else-if` y `v-else`.

- **v-for**: equivale a un bucle for. Nos permite, en nuestro caso, rellenar algunas tablas de manera automática según los datos de un array existente en el `data` del modelo, estableciendo una relación de única dirección hacia esos datos.
- **v-bind**: esta directiva es la encargada de establecer una relación entre una variable lógica y un elemento de la interfaz de usuario unidireccional (es decir, la relación va desde el modelo a la vista). En nuestro caso lo hemos utilizado para realizar una asociación en elementos muy concretos y poder observar los cambios que suceden en ese elemento.
- **v-show**: similar a `v-if`. La diferencia reside en que, en esta directiva, el elemento asociado sí se incluye en el DOM, pero sólo se mostrará en la interfaz del usuario si se cumple la condición contenida en esta directiva.
- **v-model**: permite relacionar un elemento visual con una variable pero, a diferencia de `v-bind`, se hará de manera bidireccional.
- **double moustache binding**: esta directiva se representa con dos llaves de apertura y dos llaves de cierre. Dentro de ellas, irá la variable que queramos mostrar en la interfaz de usuario. Gracias a esta directiva, podemos realizar un binding a variables y mostrarlas junto a elementos HTML estáticos. De esta manera, aunque el HTML cambie, esta variable tendrá un comportamiento distinto

5.2. Diseño basado en componentes

Una arquitectura basada en componentes consiste en descomponer el diseño en elementos individuales llamados componentes. Estos componentes deben ser funcionales o lógicos, y deben exponer interfaces de comunicación bien definidas para proporcionar un nivel de abstracción mayor [49]. Entre los principios que definen este tipo de diseño, se encuentran:

- **Reusabilidad**: los componentes deben ser conceptualizados para poder ser utilizados en distintos escenarios. No obstante, puede darse la creación de componentes para un uso muy específico.
- **Sin contexto específico**: los componentes, como hemos visto antes, deben crearse para ser utilizados en distintos escenarios. Información determinante como puede ser el estado del componente, el nombre del componente o el comportamiento específico, deben ser pasados al componente a través de su interfaz en vez de permitir al componente acceder a ellos o modificarlos.
- **Extensible**: un componente puede ser extendido desde otro para definir un nuevo comportamiento.

- **Encapsulado**: los componentes deben mostrar una interfaz para permitir al programa o a otros componentes utilizar su funcionalidad sin revelar detalles o implementación interna.
- **Independiente**: se debe reducir al mínimo la interdependencia entre componente, para favorecer la reutilización.

Una vez que hemos visto un poco de información sobre los principios de este tipo de diseño, podemos enumerar algunas de las ventajas de este tipo de diseño. La primera de ellas es que, al ser encapsulados, favorece mucho la independencia entre los propios componentes, lo que facilita a su vez la reutilización y reduce al mínimo la interdependencia entre ello. En segundo lugar, como consecuencia lógica de lo explicado, esta independencia hace que la fase de testing y de pruebas sea más sencilla, pues permite aislar a los componentes y, así, aislar los errores que puedan existir. El conjunto de todo lo que hemos visto también se traduce en un mantenimiento del sistema más sencillo, pues si se necesita modificar una serie de componentes para que sean más escalables, modificar su comportamiento o, directamente, eliminarlos, estas características harán que el proceso sea mucho más sencillo.

5.2.1. Diseño basado en componentes en este proyecto

En este proyecto, se ha empleado Atomic Web Design como diseño para el desarrollo de componentes. Atomic Web Design surge en 2013 para explicar el diseño de una interfaz web descomponiendo en entidades más pequeñas (denominadas átomos) para reutilizarlos y combinarlos con el objetivo de generar entidades más grandes (composiciones) y potentes que den sentido a la funcionalidad del diseño [22].

Al utilizar el framework Vue.js en nuestro proyecto, lo que estamos haciendo es definir y diseñar componentes reutilizables con su propio estado y propiedades (denominadas *props*, como vimos en el Capítulo 4.2.5). Una vez diseñados estos componentes, se diseña la componetización para agregarlos y que generen una funcionalidad mostrada por la interfaz [62].

En Vue.js, como se explicó en el Capítulo 4.2.5, se utiliza la filosofía Single File Component (SFC), en la que un archivo de Vue está compuesto por el *Template*, el *Style* y, por último, *Script*. De esta manera, un archivo SFC es una composición de estas tres entidades, manifestándose así en un componente.

En el siguiente apartado vamos a ver el diagrama de componentes de nuestro proyecto y a comentar en qué consiste cada componente, además de comentar aspectos relevantes del propio diagrama y decisiones que se han tomado para realizarlo.

5.2.2. Diagrama de componentes

En la Figura 5.2 podemos ver el diagrama de componentes de este proyecto.

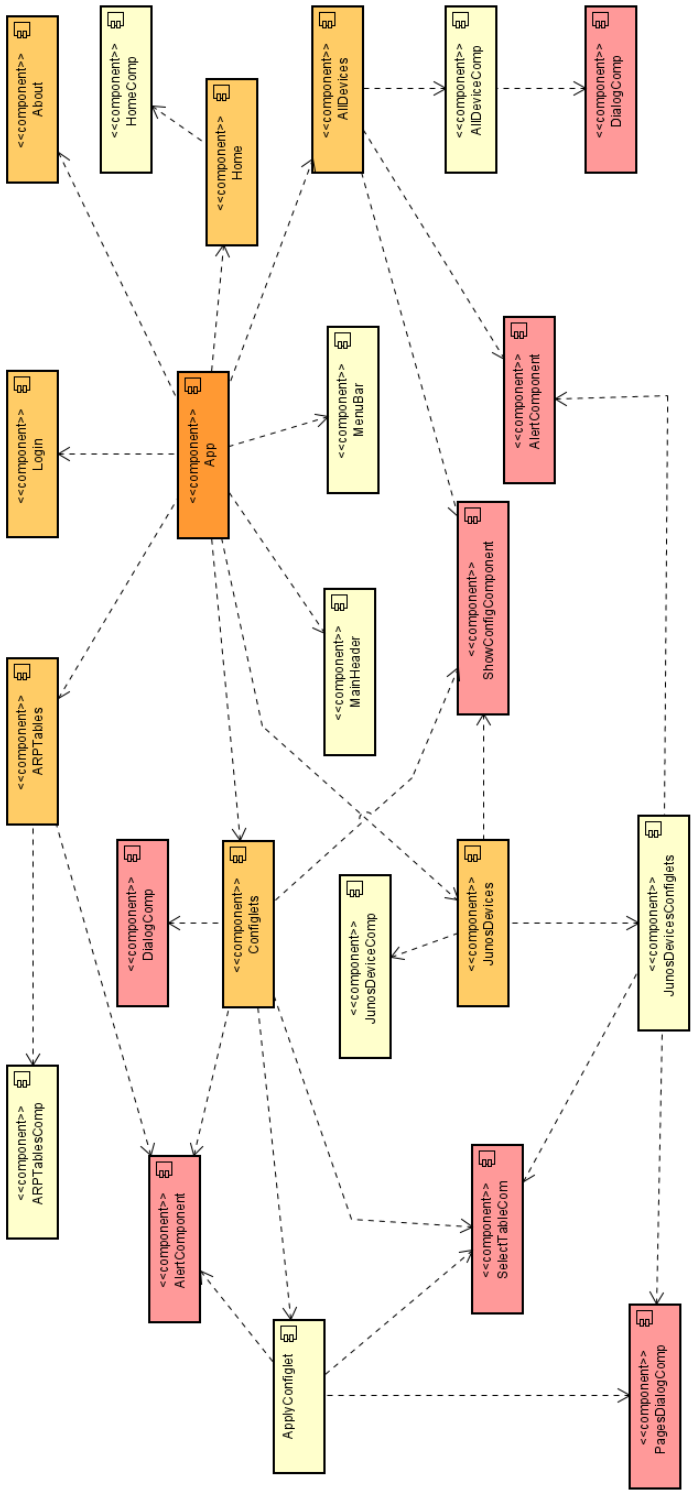


Figura 5.2: Diagrama de componentes.

Como se puede ver en el diagrama, se han utilizado cuatro tipos de colores para representar estos componentes. Para explicar en qué consiste cada componente, vamos a dividir la explicación según estos colores:

1. **Color naranja**: representa la aplicación principal. En él, se muestra únicamente un componente llamado `App`, el cual incorpora todos los componentes, pues contiene la aplicación completa.
2. **Color amarillo oscuro**: en el siguiente nivel tenemos los componentes que se encargan de las vistas principales, es decir, a las que se llega desde el punto de partida de la aplicación. Estos componentes, a diferencia de otros, pueden lógica de negocio como llamadas a la API, conexión con el back-end y llamadas a bases de datos. Dentro de este tipo de componentes podemos encontrar:
   - `ARPTables`: componente que se encarga de realizar las llamadas para la búsqueda ARP según la información introducida por el usuario.
   - `Login`: componente de inicio de sesión en la aplicación. Es el encargado de enviar la petición al back-end con las credenciales introducidas.
   - `About`: muestra la vista del About.
   - `Home`: muestra la vista inicial que ve el usuario una vez que inicia sesión.
   - `Configlets`: este componente realiza una petición a la API de Junos cuya respuesta contiene todos los configlets disponibles para el usuario.
   - `JunosDevices`: similar al anterior, con otra petición a la API de Junos obtiene la lista de dispositivos Juniper disponibles en la plataforma.
   - `AllDevices`: componente realiza una petición a la base de datos para obtener todos los dispositivos.
3. **Color amarillo claro**: son componentes secundarios. Aquí se incluyen componentes dependientes de las vistas principales. No suelen realizar peticiones a APIs o bases de datos, aunque puede haber excepciones.
   - `HomeComp`: componente de la vista Home. En él se muestra la información que se desee. Es un componente que será más útil en el futuro, cuando se incluyan otras funcionalidades a la aplicación.
   - `MenuBar`: componente que muestra el menú lateral, visible en cualquier vista de la aplicación (a excepción de Login). En él se encuentran rutas a algunas de las principales vistas del proyecto.
   - `MainHeader`: componente que actúa de cabecera de la aplicación. De igual manera que el componente anterior, muestra rutas a algunas de las vistas principales.
   - `AllDeviceComp`: componente que se encarga de mostrar en una tabla los dispositivos de todos los fabricantes. Además, contiene botones que permiten eliminar, modificar, editar o refrescar esta lista de dispositivos.
   - `JunosDeviceComp`: componente que muestra los dispositivos Juniper en una tabla.
   - `JunosDevicesConfiglets`: este componente muestra los configlets disponibles para aplicar en un dispositivo. Se encarga de obtener esta información y mostrarla a través de los componentes necesarios.

- `ApplyConfiglet`: es similar al componente anterior en comportamiento pero, en este caso, en vez de mostrar los configlets disponibles para un dispositivo, muestra los dispositivos a los que podemos aplicar un único configlet que hayamos seleccionado en la vista anterior. De igual manera, muestra la información correspondiente a través de distintos componentes.

4. **Color rosado**: por último, para finalizar, hemos agrupado los componentes de utilidad bajo este color. Estos componentes cumplen un propósito general y pueden ser acoplados por casi cualquier componente. Utilidad como mostrar datos en una tabla, mostrar un mensaje de alert o un cuadro que muestre un texto a modo de configuración se agrupan en este tipo de componentes:

   - `AlertComponent`: muestra un mensaje de tipo alert indicando éxito, información o error en una acción realizada por el usuario.
   - `DialogComp`: muestra un componente Dialog que, a su vez, puede incluir otros componentes como `SelectTableComp`. En él se muestra información relevante sobre una acción que desea realizar el usuario.
   - `PagesDialogComp`: componente que muestra, en varias páginas, información sobre configuraciones (Configlets, concretamente), que se desean aplicar. Se pueden desplazar las páginas para ver la configuración de cada item.
   - `SelectTableComp`: componente al que le pasamos unos datos estructurados y los muestra a modo de tabla.
   - `ShowConfigComponent`: componente para mostrar configuración de un elemento.

Como podemos ver los componentes, utilizados bajo la idea de Atomic Web Design, nos han facilitado enormemente la elaboración de este proyecto reutilizando código, facilitando la realización de éste y, por último, reduciendo la interdependencia. Si bien es cierto que utilizar este tipo de diseño dificulta en ciertos aspectos la elaboración del código (por ejemplo, necesita un diseño muy cuidado pues, si no, toca refactorizar en numerosas ocasiones para reducir la interdependencia), una vez que se domina es muy sencillo reutilizar cualquier tipo de componente.

Por último, creo que es necesario hacer hincapié en otra de las ventajas de este tipo de diseño: la escalabilidad. El modelo basado en componentes me ha permitido diseñar algunos elementos de tal manera que se vayan a poder reutilizar en el futuro. Las líneas de trabajo futuras, de esta manera, serán más fáciles de aplicar al utilizar componentes ya existentes, por lo que la escalabilidad se favorce con este tipo de diseño.

5.3. Diseño de la Interfaz de Usuario (IU)

Al tratarse de una aplicación concebida para el uso profesional por administradores de equipos de red, la interfaz no debe ser demasiado explicativa, aunque sí intuitiva. Debe ser fácil de usar, pero tampoco demasiado simplificada a costa de perder una funcionalidad más avanzada. Debido al concepto de aplicación, he optado por una interfaz muy minimalista,

limpia y que muestre en todo momento las configuraciones disponibles para el usuario, de tal modo que de un vistazo pueda saber en qué punto se encuentra y que una simple vista pueda mostrar la mayor cantidad de datos posible sin sacrificar la comodidad del usuario.

A continuación, vamos a ver los bocetos que he ido realizando a medida que tenía lugar la planificación de cada sprint y, posteriormente, veremos las vistas finales, con las explicaciones necesarias acerca de posibles cambios que haya habido entre el boceto y su versión final.

5.3.1. Bocetos de la IU

Para realizar los bocetos, he utilizado la herramienta Balsamiq Wireframes, la cual permite bocetar rápidamente aspectos básicos pero relevantes de las vistas planteadas. Antes de maquetar, he realizado unos pequeños bocetos con esta herramienta que me han ayudado a estructurar los componentes y el diseño de la página, además de detectar posibles errores de diseño antes de programar.

Para una mejor explicación, he puesto un rótulo en color rojo en cada vista nombrando cada componente o cada item importante para poder hacer referencia a él en la explicación.

Sprint 1 - Bocetos (ver Tabla 7.1)

Durante este sprint ha tenido lugar el comienzo de las historias de usuario 1 - Consulta de dispositivos y 13 - Ver configuración del dispositivo. El boceto de esta vista se puede consultar en la Figura 5.3. A la hora de preparar esta vista, surgen 2 componentes claros: la tabla de dispositivos y la configuración de cada dispositivo.

Antes de tratar en profundidad los componentes de estas H.U., podemos fijarnos que la aplicación tiene una Cabecera y un Menú lateral. La cabecera permite cambiar el idioma, desloguearse (si el usuario ya está logueado) e ir a la vista Home (haciendo click arriba a la izquierda en el logo de la app). En cuanto a la barra del Menú Lateral, ahí podemos hacer click en la acción que queramos realizar, desde ver los dispositivos, ver los configlets disponibles, las tablas ARP, etc.

En este caso, seleccionado la opción “Devices” (dispositivos) está seleccionada, pues es la que mostrará la vista de la H.U. 1 - Consulta de dispositivos. Una vez lo seleccionemos haciendo click sobre él, se mostrará la Tabla de Dispositivos y, en ella, saldrán todos los dispositivos disponibles. También se podrán eliminar y añadir dispositivos, pero no corresponden a esta H.U.. Si seleccionamos un único dispositivo (haciendo click sobre él), también podremos realizar la H.U. 13 y ver la configuración interna del dispositivo. La configuración se obtiene con una llamada a la API y se muestra en el cuadro de texto llamado “Config de cada dispositivo”, como podemos ver en la imagen.

Figura 5.3: Boceto de la vista "Devices"

Sprint 2 - Bocetos (ver Tabla 7.2)

Durante este sprint han tenido lugar las H.U. 2, 3 y 4, correspondientes a la consulta de configlets, aplicación de configlets y login, respectivamente. Debido a esto, contamos con las siguientes vistas:

En la Figura 5.4, podemos ver los configlets disponibles y seleccionar varios de ellos. Una vez seleccionados, si hacemos click sobre el botón "Select", se nos muestran los dispositivos sobre los que podemos aplicar los configlets. Esta nueva vista viene representada en la Figura 5.5, donde en el cuadro "Configlets a aplicar" muestra los configlets y el cuadro inferior, los dispositivos donde se puede aplicar cada uno de ellos. Además, en el "cuadro resumen de configuración" se puede ver el resumen de lo que se va a realizar en la siguiente vista (que se podrá ver más adelante). Por último, podemos comprobar en la 5.6 cómo es la vista del login inicial, donde se pide al usuario que introduzca su nombre y contraseña.

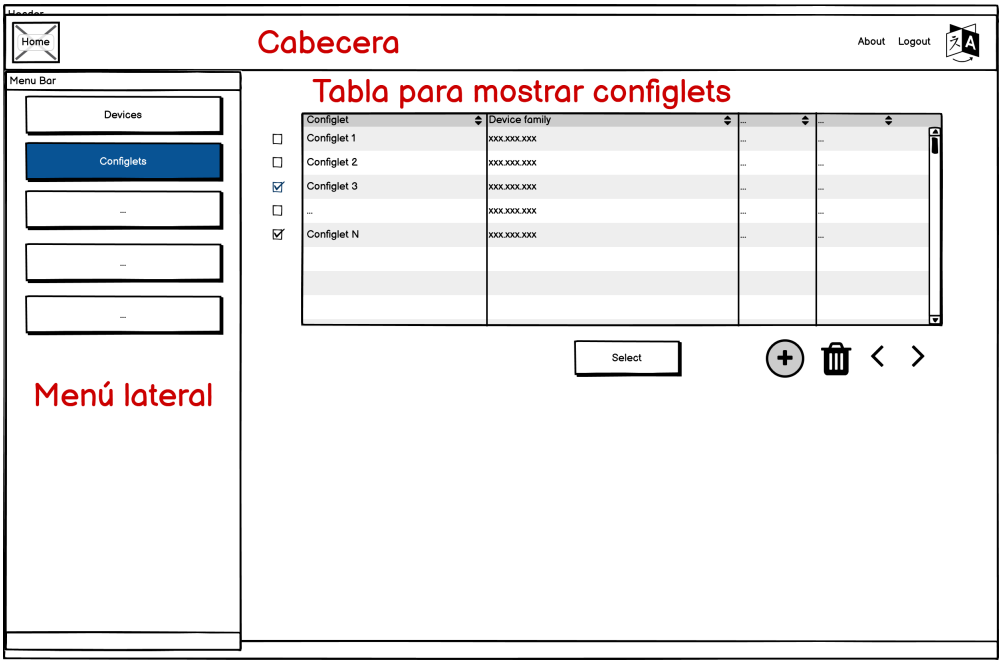


Figura 5.4: Boceto de la vista "Configlets"

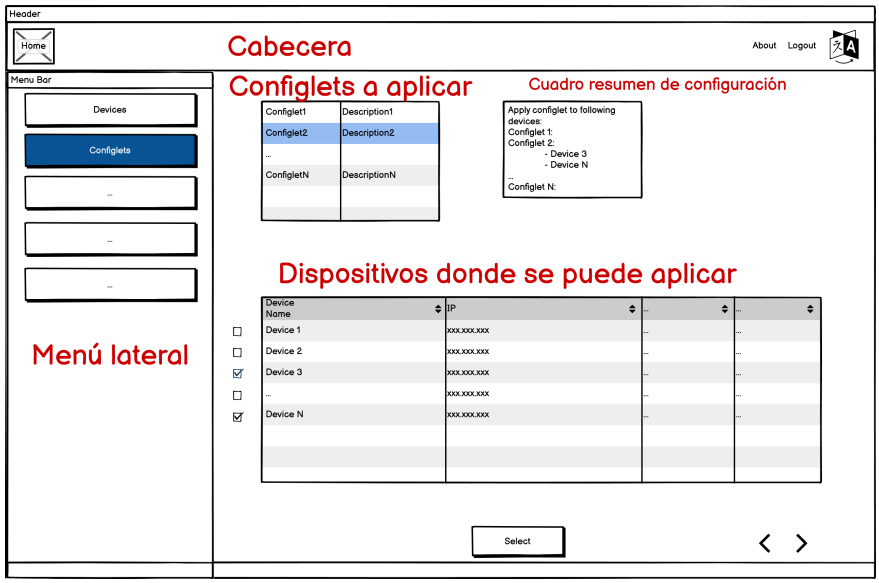


Figura 5.5: Boceto de la vista "Apply Configlets"

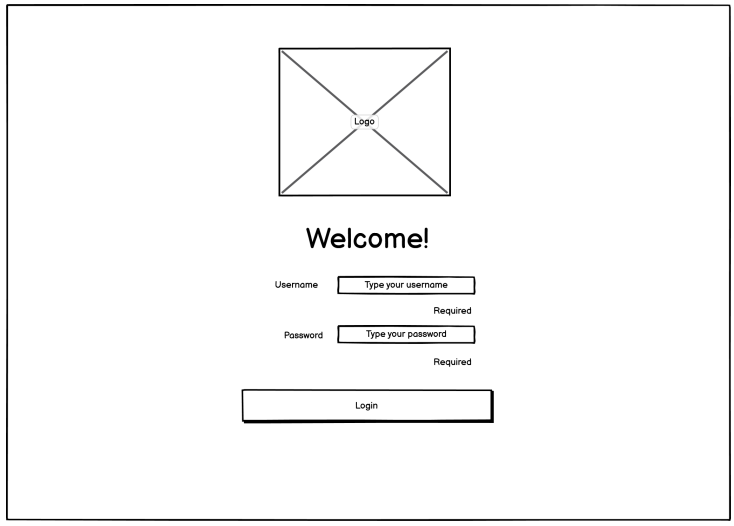


Figura 5.6: Boceto de la vista "Login"

Sprint 6 - Bocetos (ver tabla 7.6)

En este sprint se comienza con la H.U. 5, la cual permitirá hacer búsquedas ARP. Como se muestra en la Figura 5.7, se crea una nueva vista con una barra de búsqueda en la cual debemos introducir la dirección IP, la MAC o el nombre de la red que queremos buscar. Una vez introducido, debemos hacer click en el botón de búsqueda que vemos representado por una lupa y, una vez hecha la búsqueda, se mostrarán las tablas pertinentes si el dispositivo existe y la búsqueda se ha completado con éxito.

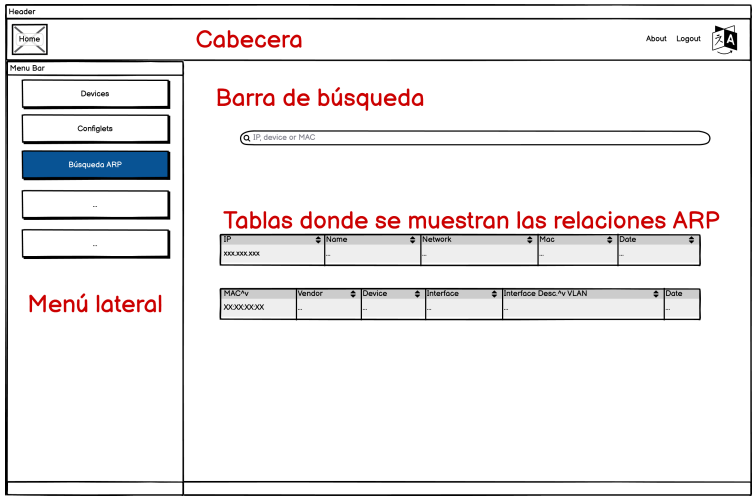


Figura 5.7: Boceto de la vista "ARP Search"

Sprint 8 - Bocetos (ver tabla 7.8)

En este sprint se comienza a desarrollar la implementación de las acciones CRUD de dispositivos. De esta manera, en este sprint se crea un componente Dialog que, según la opción seleccionada, muestre un formulario (vacío en el caso de añadir un dispositivo, con los parámetros actuales si deseamos modificarlo) o una tabla con los datos que se van a eliminar en el caso de querer borrar un dispositivo de la base de datos. Las distintas vistas de este componente las podemos ver en las tablas 5.8 y 5.9.

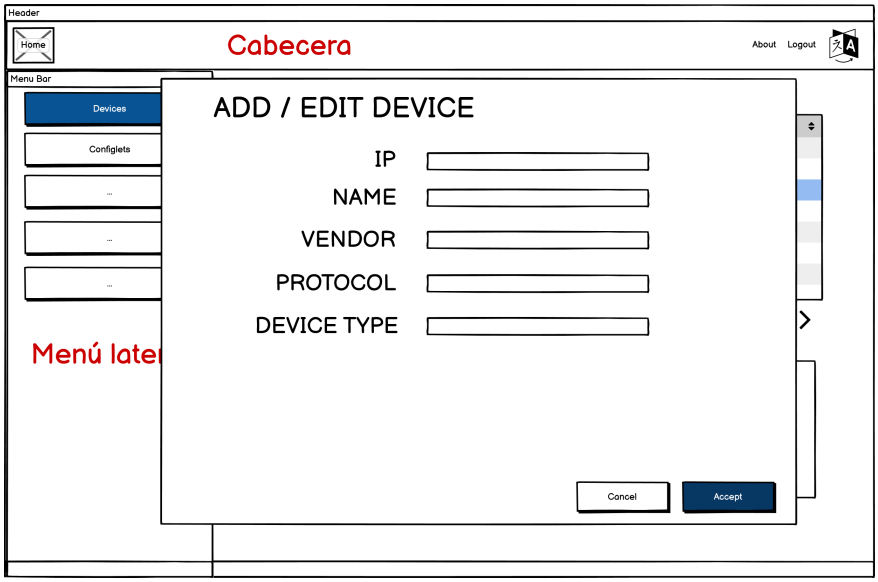


Figura 5.8: Boceto del componente Dialog en el caso de Añadir/Editar dispositivo

En el caso de añadir y editar un dispositivo 5.8, la vista es la misma, solo que los campos del formulario aparecerán vacíos en el caso de añadir, mientras que si editamos el dispositivo, los campos aparecerán rellenos con los parámetros actuales del dispositivo

Por otro lado, aunque se utilice en numerosas vistas (pues es un componente de uso general), el Alert se genera también en este sprint. El Alert informa de un evento que sucede en la base de datos o al realizar una llamada a la API. De esta manera, cada vez que el usuario realiza una acción que implica una llamada, el commponente Alert informa al usuario de qué ha sucedido, ya sea para informar de que la acción se realizó con éxito, que hubo un error o informar a modo de advertencia que los datos introducidos no son correctos. Podemos ver el boceto de este componente en la Figura 5.10. Aunque la vista de fondo se corresponde con la H.U. 05, este componente es el mismo para todas las vistas, por lo que el concepto puede entenderse de igual manera bajo esta vista.

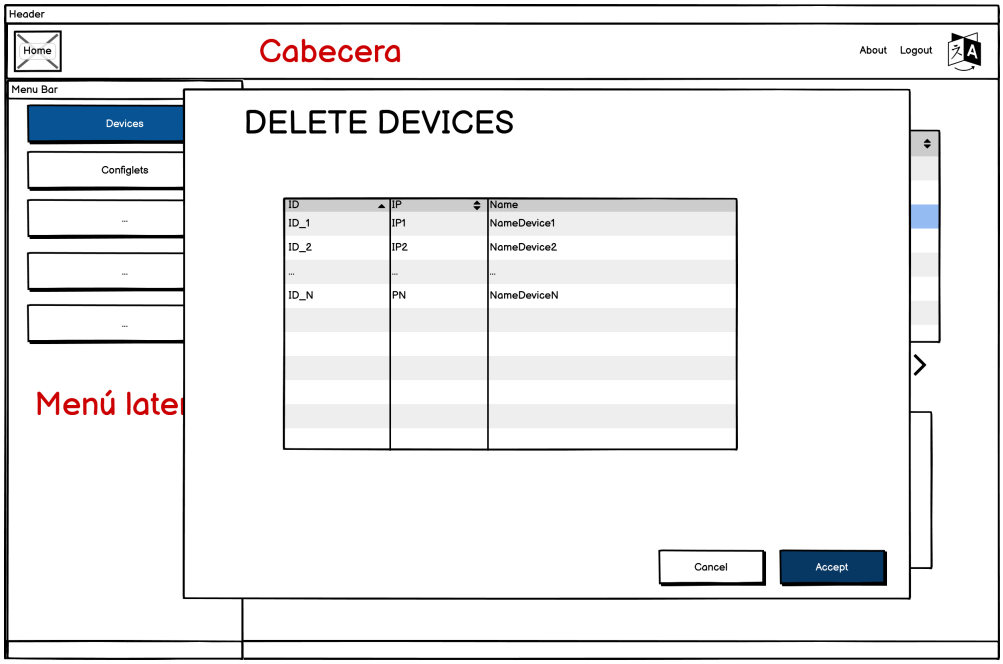


Figura 5.9: Boceto del componente Dialog en el caso de Eliminar dispositivo

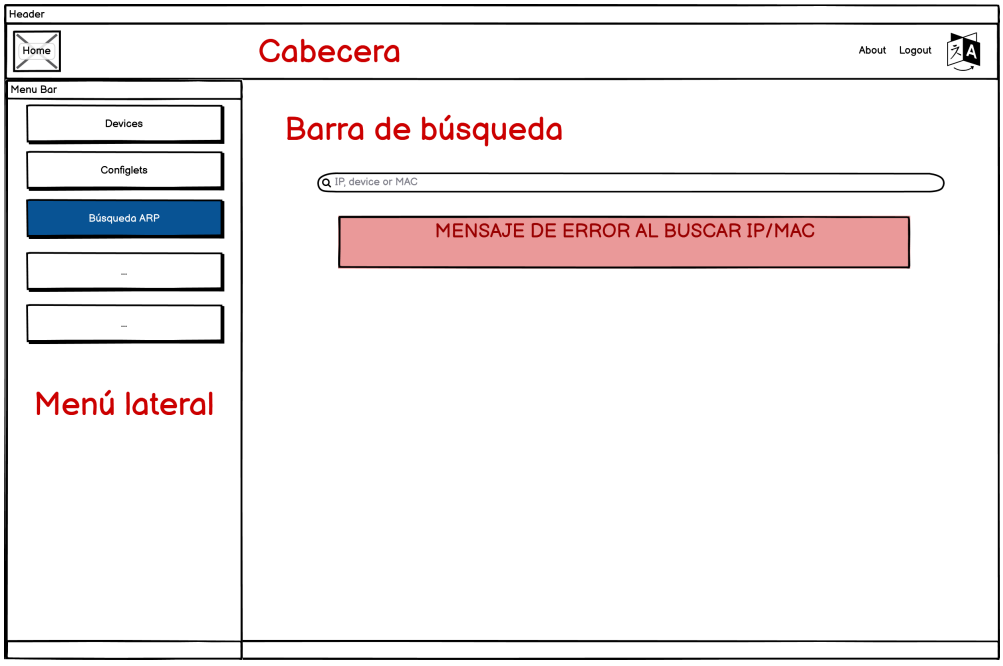


Figura 5.10: Boceto del componente Alert de propósito general

Sprint 13 - Bocetos (ver tabla 7.13)

Durante este sprint tienen lugar las H.U. 06: Aplicar configlets y H.U. 17: Validar configlet. En la H.U. 06 primero se selecciona un dispositivo y se muestran los configlets disponibles para aplicar y, en la segunda, se muestra la validación de todos los configlets que se quieren validar. Esta validación se muestra en un componente Dialog que mostrará un mensaje en verde por cada configlet válido y, en caso de una configuración no válida, mostrará un mensaje en rojo en el mismo componente. En el caso de seleccionar varios confilets para validar, se podrá pasar de uno a otro con flechas laterales.

Podemos ver, en la Figura 5.11, cómo se define esta vista. Lo primero que mostrará será la IP y el nombre del dispositivo donde se van a aplicar los configlets seleccionados. Para cambiar entre configlets, se pueden ver unas pestañas que tendrán el ID del configlet y su correspondiente formulario, donde el usuario tendrá que introducir los datos deseados. Una vez que el usuario haya introducido los datos necesarios, si hace click en el botón aplicar (*Apply* en la figura), podrá validar todos estos configlets y, dependiendo de si el resultado ha sido correcto o no, aparecerán las vistas que se pueden ver en las Figuras 5.12 y 5.13, respectivamente.

Como se puede ver, ambas vistas son idénticas solo que, en el caso del error, aparecerá un código en rojo que indicará la causa del error. En ambas vistas, el botón de aceptar se habilitará sí y sólo si todos y cada uno de los configlets han sido validados con éxito. Como hemos mencionado, al pulsar sobre las flechas laterales se cambiará entre configlets y se podrá ver el código que se va a inyectar en el dispositivo seleccionado.

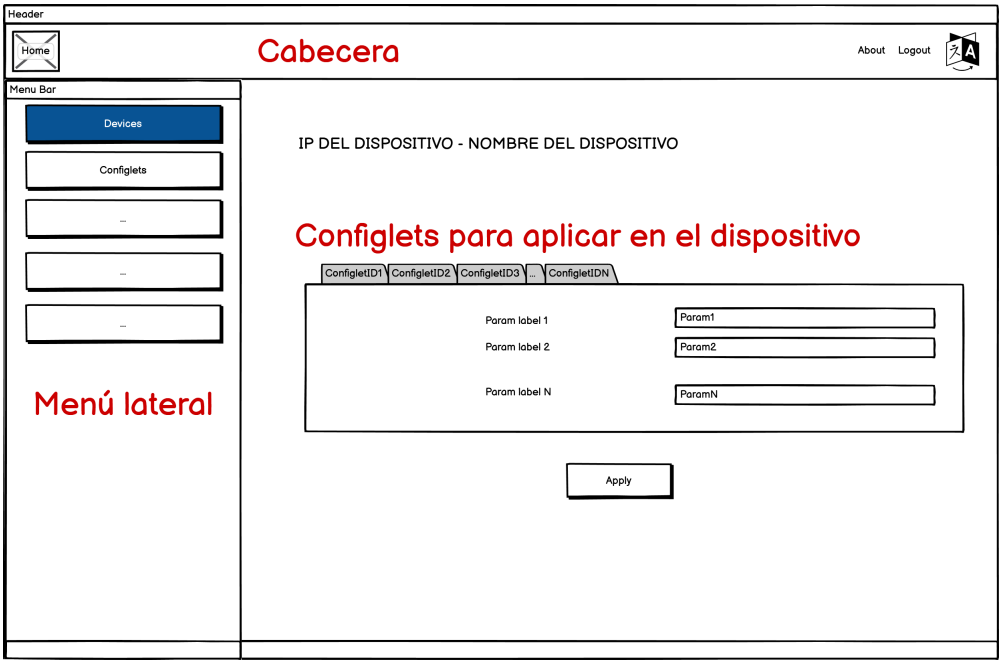


Figura 5.11: Aplicación de configlets a un dispositivo

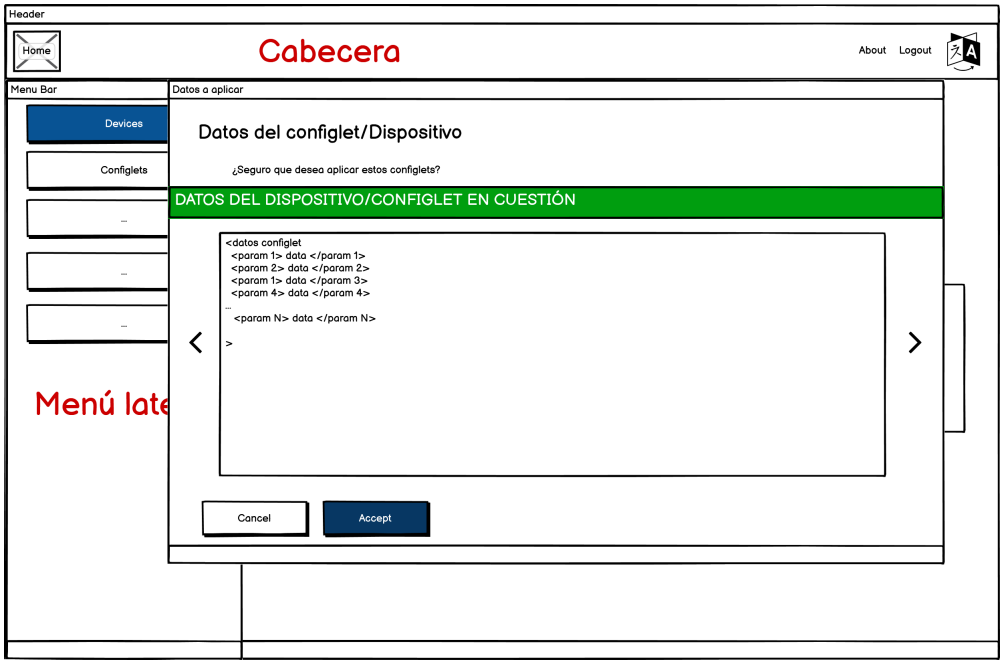


Figura 5.12: Validación de configlets correcta

Figura 5.13: Validación de configlets incorrecta

Sprint 14 - Bocetos (ver tabla 7.14)

En este sprint tiene lugar el bocetado de la última vista. En el sprint anterior vimos cómo se generaba una vista (Figura 5.11 donde, seleccionado un dispositivo, se le pudieran aplicar los configlets necesarios. Pues bien, en la H.U. 11: Seleccionar configlets, se ha implementado la opción contraria: en la lista de configlets, al seleccionar uno para aplicar, podremos ver los dispositivos a los que lo podemos aplicar. De esta manera, si antes se aplicaba varios configlets en un dispositivo, ahora es el caso contrario: se aplica un solo configlet en uno o varios dispositivos. La vista para aplicar un configlet a varios dispositivos la podemos encontrar en la Figura 5.14.

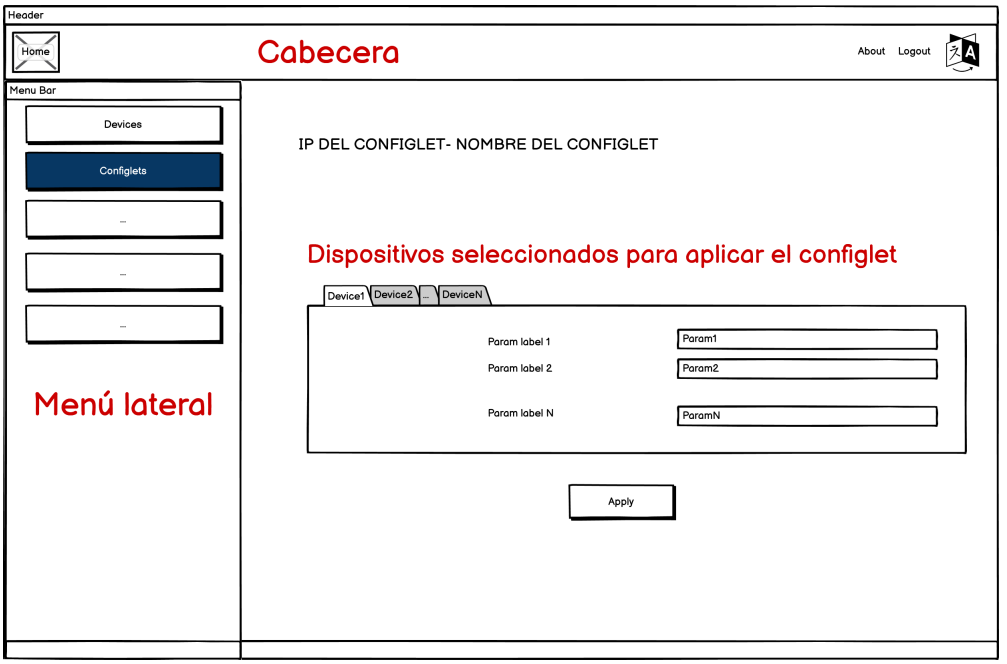


Figura 5.14: Aplicar un configlet a varios dispositivos

Como se puede ver, ambas vistas son muy similares. Sin embargo, en estas pestañas se cambia el dispositivo al que le aplicaremos los datos introducidos en el formulario correspondiente. El resto de la lógica se mantiene intacta: si hacemos click sobre el botón de aplicar, nos aparecerá un dialog indicando si la validación ha sido correcta o no, pues se utiliza el mismo componente para mostrar este paso (ya visto en las Figuras 5.12 y 5.13).

5.4. Arquitectura lógica del back-end

El servidor back-end también necesita un diseño de clases agrupadas por paquetes. El diagrama mostrará en medida de lo posible la estructura de dicho servidor para poder ha-

cernos una idea de la arquitectura dentro de este sistema. Se puede ver este diagrama en la Figura 5.15.

Durante el diseño del back-end, además se han tomado decisiones importantes a la hora de aplicar algunos patrones para mejorar el código y su reutilización. Entre ellos, el patrón fachada ha resultado útil en el módulo `management` del proyecto. Este patrón fachada (*facade*) se suele utilizar para reducir acoplamiento entre clases en un proyecto [20]. En nuestro caso, como el paquete `management` se comunica con un gran número de métodos y clases, se realiza una clase que actúa como interfaz simple para reducir la dependencia entre las clases del proyecto.

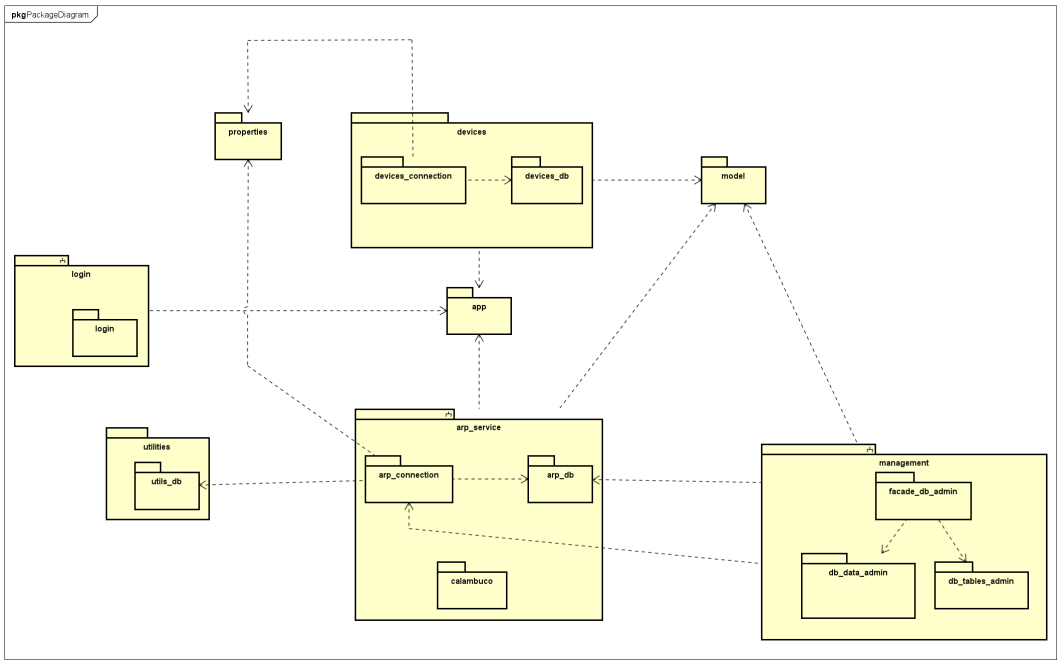


Figura 5.15: Diagrama de despliegue.

5.5. Diseño del despliegue

El despliegue se realizará en una máquina virtual ofrecida por mi entorno de trabajo. Para conectarme a ella, es necesario tener credenciales de usuario, pues necesita estar en un entorno autorizado para realizar las peticiones tanto a los portales utilizados como a los propios dispositivos. Actualmente, el proyecto se encuentra desplegado en la misma máquina pero, posteriormente, su despliegue se realizará por contenedores, utilizando otra máquina para realizar las peticiones de Login. Una vez que la aplicación entre en entorno de producción, posterior a la dockerización del Login, se podrá acceder a ella desde cualquier ordenador dentro de la VPN de la empresa. Así, podremos comunicarnos con esta aplicación, realizar las peticiones necesarias y aplicar los configlets deseados (ver Figura 5.16).

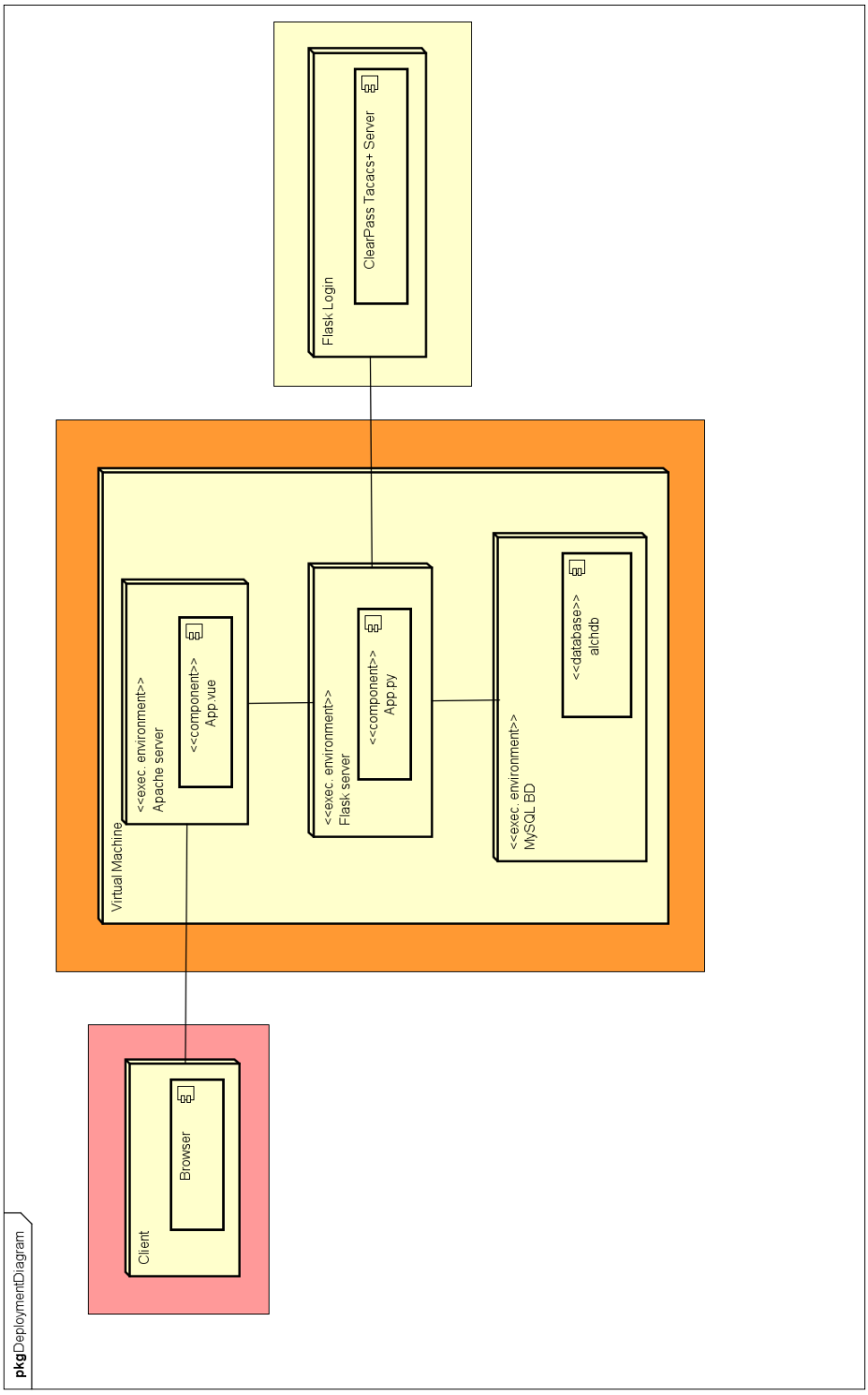


Figura 5.16: Diagrama de despliegue.

Capítulo 6

Implementación y pruebas

Tras haber visto el diseño de la aplicación, el siguiente paso lógico es hablar de la implementación. En este capítulo vamos a hablar de la estructura final del código en el servidor y en las vistas de la parte web. Además, también vamos a comentar aspectos relevantes como cambios en el modelo del dominio o en el diseño de las vistas, aspectos a tener en cuenta a la hora de modificaciones en la funcionalidad.

6.1. Estructura del código de la web

Aquí se muestra la estructura y jerarquía final del código. Una vez listados estos archivos, se procede a hablar de los más importantes para la parte del front-end. Es importante comentar que no se van a listar todos los archivos o carpetas, pues algunos son implementaciones internas del editor de código, configuraciones internas, y otros componentes no relacionados con el código

```
|   .browserslistrc
|   .eslintrc.js
|   .gitignore
|   babel.config.js
|   cypress.json
|   jest.config.js
|   package.json
|   README.md
|
+---cypress (módulo para realizar test end-to-end)
|
+---node_modules (módulos de Node usados, comentados anteriormente)
|
+---public
```

```
|   |   index.html
|   |   logo-junco.png
|   |   logo-junco-dark.png
|
+---server (carpeta que contiene la parte del servidor back-end,
            se comentará en el siguiente apartado)
|
+---src
|   |   App.vue
|   |   main.js
|   |
|   +---api
|   |   |   arp.js
|   |   |   devices.js
|   |   |   junos.js
|   |
|   +---assets  (se omite su contenido por brevedad)
|   |   |
|   |   +---configlets
|   |   |   |   apply-cli-configlet.json
|   |   |   |   validate-configlets.json
|   |
|   |   +---images (imágenes usadas en la app, se omite su contenido)
|   |
|   +---components
|   |   |
|   |   +---arptables
|   |   |   |   ARPTablesComp.vue
|   |
|   |   +---devices
|   |   |   |   AllDeviceComp.vue
|   |   |   |   DeviceComp.vue
|   |
|   |   +---global
|   |   |   |   HomeComp.vue
|   |   |   |   MainHeader.vue
|   |   |   |   MenuBar.vue
|   |
|   |   +---utilityComponents
|   |   |   |   alertComponent.vue
|   |   |   |   DialogComp.vue
|   |   |   |   PagesDialogComp.vue
|   |   |   |   SelectTableComp.vue
|   |   |   |   ShowConfigComponent.vue
|   |
|   +---lang
|   |   |
|   |   +---messages
|   |   |   |   en.js
|   |   |   |   es.js
|   |   |
```

```
|   |   |   i18n.js
|   |
|   +---plugins
|   |       vuetify.js
|   |
|   +---router
|   |       index.js
|   |
|   +---store
|   |   |   index.js
|   |   |
|   |
|   +---utils
|   |   |   constants.js
|   |   |   functions.js
|   |
|   +---views
|   |
|   |   +---arptables
|   |   |   |   ARPTables.vue
|   |
|   |   +---configlets
|   |   |   |   ApplyConfiglet.vue
|   |
|   |   +---devices
|   |   |   |   AllDevices.vue
|   |   |   |   JunosDevices.vue
|   |   |   |   JunosDevicesConfiglets.vue
|   |
|   |   +---global
|   |   |   |   About.vue
|   |   |   |   Home.vue
|   |   |   |   Login.vue
|   |
+---tests
|   |   e2e
|   |   unit
```

Los primeros ficheros de la lista hacen referencia a ficheros de configuración de Node, configuración de test (tanto unitarios como end-to-end), el `.gitignore` (fichero en el cual podemos incluir qué archivos no queremos *trackear*, es decir, ficheros que si cambian, Git no tiene ningún tipo de seguimiento sobre ellos y que no se añaden al repositorio) y, por último, el `README.md` del proyecto. En `.gitignore` hemos añadido, además de carpetas como `node_modules`, archivos con información sensible como, por ejemplo, la dirección IP de algún servidor con el que nos comunicamos, etc. De esta manera, si se sube el proyecto a alguna herramienta de Git (como Github), estos archivos no aparecerán. Ahora vamos a comentar de qué están compuestas el resto de carpetas.

- **public**: contiene el fichero index (el cual se utiliza como punto de entrada a la app) y logos de la aplicación utilizados en la pestaña del navegador.
- **server**: esta carpeta contiene el back-end de la aplicación, así como la capa de persistencia. Lo veremos en el siguiente apartado
- **src**: aquí se encuentran todos los archivos que ejecuta la aplicación front-end, desde vistas hasta utilidades, además de otros componentes:
  - **App.vue**: es el fichero principal de Vue, pues es la vista general de nuestra aplicación. En él también se pueden definir estilos o clases que se aplicarán al resto de la aplicación, si así se desea.
  - **main.js**: fichero Javascript en el que se declaran qué módulos y plugins se van a utilizar en nuestro proyecto. Muchos de los módulos comentados anteriormente se tienen que declarar aquí para que el proyecto sepa cómo utilizarlos.
  - **api**: contiene los archivos que incluyen el código para realizar tanto peticiones Axios al back-end creado como peticiones a la API de Junos Space.
  - **assets**: en la carpeta `images` se encuentran las imágenes utilizadas en la aplicación (por ejemplo, los iconos para representar el idioma) mientras que, en la carpeta `configlets`, se encuentran las plantillas JSON que se tienen que modificar en las vistas correspondientes para aplicar o validar configlets. Estas plantillas se modifican según los datos que haya introducido el usuario en los formularios y se envían mediante una petición Axios a la API de Junos Space
  - **components**: aquí se incluyen los componentes de la aplicación. Están divididos según funcionalidad:
    - **arptables**: componentes para mostrar resultados de las tablas ARP.
    - **devices**: componentes para mostrar al usuario las tablas que incluyen a los dispositivos y sus configuraciones (tanto todos los dispositivos como sólo Juniper para aplicar configlets).
    - **global**: componentes utilizados en partes globales de la aplicación, como la cabecera, barra lateral o la vista `Home.vue`.
    - **utilityComponents**: componentes de utilidad, muy específicos, como un Alert para avisar de componentes, Dialogs que se muestran para enseñar formularios o tablas, etc.
  - **lang**: en el fichero `i18n.js` se declaran los ficheros que se van a usar para guardar las cadenas de texto según el lenguaje y, en la carpeta `messages`, se declaran estos ficheros, cada uno con las cadenas adaptadas al lenguaje deseado.
  - **plugins**: las librerías de Node utilizadas durante el desarrollo. En mi caso, sólo he utilizado `vuetify`.
  - **router**: aquí nos encontramos el archivo `index.js`, que es el encargado de definir todas las rutas HTML de nuestra aplicación, además de los nombres de éstas y otros aspectos como si se pueden acceder según qué condiciones. En esta aplicación, por ejemplo, se ha definido que si el usuario no está logueado, no pueda acceder a ninguna vista, por motivos de seguridad.
  - **store**: aquí está el código necesrio para administrar el estado global de la aplicación a través de Vuex.

  - **utils**: carpeta para guardar utilidades. Aquí se exportan las constantes necesarias (en mi caso, expresiones regulares) en el fichero `constants.js`, mientras que en `functions.js` se han definido funciones comunes para ser usadas por cualquier vista como, por ejemplo, métodos para formatear una dirección MAC o manipular archivos JSON.
  - **views**: vistas de la aplicación. Por cuestiones de acoplamiento y cohesión, he decidido que sean las vistas las que se encarguen del grueso de la lógica además de las llamadas axios para que, en consecuencia, los componentes sean más reutilizables y simplemente se encargen de cuestiones como mostrar una tabla, devolver datos seleccionados, etc. La estructura es la misma que explicamos en la carpeta de `components`, por lo que no es necesario volver a comentar de qué se componen las carpetas bajo este directorio.

- **test**: aquí se incluyen los test realizados, tanto los unitarios como los end-to-end.

6.2. Estructura del código del servidor

En el apartado anterior vimos la estructura global del proyecto. Sin embargo, en la carpeta `server` se dejó sin comentar con la intención de comentarla en este apartado. Esta carpeta es la que contiene toda la lógica del servidor, conexión con base de datos y gestión de peticiones por parte de la interfaz front-end, además de encargarse de enviar los datos en cada consulta a la parte web. Veamos cómo se desglosa esta carpeta:

```
+---arp_service
|   |   arp_connection.py
|   |   arp_db.py
|   |   calambuco.pyh
|
+---data
|   |   lista-switches.csv
|
+---devices
|   |   devices_connection.py
|   |   devices_db.py
|
+---login
|   |   login.py
|
+---management
|   |   db_data_admin.py
|   |   db_tables_admin.py
|   |   facade_db_admin.py
|
+---model
|   |   model_db.py
|
```

```
+---properties
|
+---PRUEBAS (carpeta para realizar pruebas)
|
+---testu.egg-info (configuración del servidor)
|
+---utilities
|   +---cisco_helper (utilidades para dispositivos Cisco)
|   |   data_file.json
|   |   utils_db.py
|
+---venv (carpeta para el virtual environment)
|
|   app.py
|   dependencies.txt
|   setup.py
```

Vamos a comentar los aspectos más importantes de la estructura del servidor:

- **arp_service**
  - **arp_connection.py**: clase para obtener tablas de los dispositivos para nuestro algoritmo de búsqueda ARP. Esta clase, cuyo código se divide en dos clases internas según si el dispositivo es Juniper o Cisco, hace la conexión y la llamada pertinente según el tipo del dispositivo, y sus métodos devuelven estas tablas.
  - **arp_db.py**: esta clase es la encargada de guardar en la base de datos las tablas extraídas gracias al fichero `arp_connection.py`.
  - **calambuco**: esta clase representa al algoritmo principal de la búsqueda ARP. En él se ejecuta el flujo de código descrito en la Figura 3.3, devolviendo el resultado para que se muestre por pantalla.
- **data**
  - **lista-switches.csv**: archivo de datos que nos sirve para poblar la lista de dispositivos. El mantenimiento de este archivo corre a cargo de mi equipo de trabajo. Cada vez que se actualiza un dispositivo, se elimina o se añade otro nuevo, el cambio debe realizarse en este archivo también.
- **devices**: bajo este directorio se encuentran dos archivos que realizan acciones asociadas con información de los dispositivos.
  - **devices_connection.py**: en esta clase se encuentran métodos para obtener información interna de los dispositivos. Cada vez que queremos obtener datos o tablas de un dispositivo, nos conectamos gracias a las utilidades y los métodos definidos en esta clase.
  - **devices_db.py**: esta clase sí manipula los propios dispositivos en la base de datos. Se pueden eliminar, obtener datos de todos los dispositivos, comprobar su fabricante con llamadas a la base de datos, añadir o crear dispositivos, etc.

- **login**: carpeta que contiene la clase para realizar el login a través de una llamada a un servidor utilizando el protocolo Tacacs_plus.

- **management**: bajo esta carpeta se encuentran los archivos que gestionan cualquier aspecto de la modificación de la base de datos:

  - **db_data_admin.py**: clase que controla todo lo que ocurre relativo a la información contenida en las tablas de la base de datos. Contiene métodos que permiten poblar una tabla, borrar el contenido de ésta o cargar todos los datos de un fichero `.csv` en la base de datos.
  - **db_tables_admin.py**: si la clase anterior nos permitía manipular el contenido de las tablas de la base de datos, ésta nos permitirá controlar qué ocurre con la propia tabla en vez de con sus datos. Esta clase nos permite crear las tablas definidas en el modelo, borrar una tabla o todas las tablas de manera ordenada, para que no haya errores de consistencia.
  - **facade_db_admin**: clase para aplicar el patrón fachada a la gestión de la base de datos. Desde esta clase podemos llamar a los métodos que necesitemos a la hora de manipular la base de datos. Se utiliza introduciendo un número o una cadena de texto por terminal cada vez que se quiera utilizar para aportar comodidad cada vez que se desee manipular la base de datos o los datos contenidos en sus tablas.

- **model**: carpeta donde se almacena el modelo de la base de datos junto a sus tablas y la configuración de éstas.

- **properties**: carpeta de propiedades del proyecto, tales como constantes, ruta y motor de conexión para la base de datos o direcciones IP. Está vacía, pues sus archivos se han incluído en `.gitignore` debido a su contenido.

- **PRUEBAS**: carpeta donde se crean ficheros para realizar pruebas controladas. Se omite su contenido al ser volátil.

- **testu.egg-info**: información y configuración del servidor Flask.

- **utilities**: carpeta de utilidades.

  - **cisco_helper**: utilidad de conexión para ciertos dispositivos Cisco.
  - **data_file.json**: archivo que se importa para ser modificado en ciertos casos a la hora de realizar configuraciones. Actualmente está en desuso debido al cambio de conexión con los dispositivos.
  - **utils_db.py**: archivo de utilidad. En él se definen métodos y funciones para ser usados en cualquier parte del servidor, expresiones regulares, etc.

- **venv**: entorno virtual para el proyecto. Nos permite generar una configuración que sólo se aplicará en un entorno controlado de directorios. Gracias a él, podemos configurar una configuración personalizada en cuanto a dependencias, paquetes y configuración general de Python sin cambiar la configuración global de Python instalado en la máquina.

- **app.py**: clase principal del servidor back-end. En él se gestionan las llamadas axios realizadas desde el front-end, se comprueban datos, se establecen códigos HTTP en caso de éxito o error, y se llama a los métodos pertinentes según el tipo de petición recibida.
- **dependencies.txt**: dependencias Python del proyecto.
- **setup.py**: archivo de configuración del `venv` (virtual environment)

6.3. Decisiones tomadas a lo largo del proyecto

Este proyecto se ha desarrollado bajo el marco empresarial de un equipo de administradores de redes, por lo que los requisitos se han ido modificando según algunas necesidades y protocolos de la empresa. No obstante, gran pate de las decisiones las he podido tomar libremente.

La primera decisión a tomar es qué lenguaje o entorno iba a utilizar para desarrollar el proyecto. Decidí utilizar Vue.js debido a las facilidades que ofrece este framework y la cantidad de librerías que tiene, además de que Javascript es un lenguaje que permite implementar gran cantidad de componentes y métodos con facilidad. como también una parte del código de frontend se tenía que comunicar con la API de Junos Space, vi que con llamadas Axios se podía realizar este tipo de llamadas sin necesidad de back-end, lo que ha sido una ventaja.

En cuanto al back-end, mi primera idea para llevar un inventariado de equipos y realizar la búsqueda ARP de direcciones fue Node.js. No obstante, se presentó un problema serio al realizar la Historia de Usuario que incluye la vista de Login. Al estar dentro de un equipo de desarrollo, los usuarios se autentican contra un servidor Active Directory. Como explicamos en la sección correspondiente del capítulo 4, el protocolo necesario para enviar esta petición al servidor es Tacacs+. El problema viene a raíz de que las librerías para iniciar la comunicación con un server utilizando Tacacs+ utilizan una versión anterior y no son válidas para este caso. Una vez hechas las pruebas para comunicarse con este servidor, vi que Python sí que incluía librerías para implementar este protocolo con éxito. Esto, aunque pueda ser un problema, en realidad se podría haber solucionando dockerizando este servicio e implementarlo en otro componente, pero como las librerías para comunicarse tanto con los dispositivos Cisco como Juniper a través de SSH o Telnet también estaban en Python, vi más lógico implementar el back-end en este lenguaje, pues me ha permitido tanto realizar la base de datos como las búsquedas ARP, además del servicio de Login y Logout.

En cuanto a la elección de un sistema de control de versiones, el CVS que he elegido ha sido Git. No sólo es el más usado, sino que llevo varios años, tanto en la carrera como en desarrollo personal, usándolo. Además, en mi grupo de trabajo, una vez que el proyecto se ponga en producción, el código se subirá a su GitHub empresarial, por lo cual, la decisión en este caso sólo podía ser una.

Por último, para hablar de los datos del usuario (es decir, los datos a la hora de loguearse y con los que va a hacer las llamadas a las APIs y consulta de los dispositivos), se ha optado por realizar un almacenamiento de sesión del lado del cliente. Este almacenamiento (llamado

SessionStorage en Vue) permite almacenar las credenciales del usuario en el navegador y utilizarlas en posteriores llamadas[13]. He decidido utilizar este sistema puesto que es el más sencillo de implementar y más que suficiente para cubrir la funcionalidad de mi aplicación. Cuando el cliente hace logout, estos datos se borran del lado del cliente y se le obliga a iniciar sesión de nuevo si quiere realizar cualquier tipo de servicio que ofrezca nuestra plataforma.

6.4. Cambios importantes en el desarrollo del proyecto

En esta sección vamos a hablar en términos generales de la implementación de la lógica del proyecto y de cambios importantes junto con los procesos que han obligado a realizar estos cambios.

6.4.1. Cambios en las vistas de la web

Aunque sí que ha habido algunos cambios (por ejemplo, alguna vista que se ha dividido en dos para facilitar el código y la implementación), el concepto se ha mantenido prácticamente igual tanto en los bocetos como en las vistas. Al utilizar componentes, se ha intentado hacerlos lo más reutilizables posible, de modo que muchas de las vistas que incluyan una tabla de datos, un formulario o un Dialog, suelen ser similares, pues utilizan el mismo componente cambiando únicamente el contenido y la implementación de éste.

Aunque haya habido pocos cambios, en el Capítulo 5 se muestran tanto los bocetos realizados al inicio de cada sprint para como las vistas de la aplicación, con los comentarios correspondientes en caso de que se haya realizado un cambio con el impacto suficiente. El resultado final se muestra en el Manual de Usuario en el Anexo A.1.

6.4.2. Cambios en la implementación del Login

Al principio, cuando la funcionalidad era limitada, implementé un Login atacando a la API de Junos Space. En ese Login, el mecanismo era básico, pues enviabas tus datos a través de un mensaje HTTP y el servidor cambiaba tu estado como logueado. No obstante, la API no te devuelve ningún token ni te devuelve un `response` con el que puedas hacer algo. Además, necesitaba un sistema que pueda encargarse de la administración y gestión de usuarios para que miembros de la empresa pudieran gestionar estos usuarios, dar permisos o revocarlos, etc. Debido a esto, el login con Junos Space fue algo básico que se implementó al principio a modo de prueba, pero que se desechó por estos motivos.

Más tarde, la elección del Login fue a través de un protocolo LDAP. A través de Python encontré varias librerías que permitían acceder con LDAP de tal manera que haces una llamada con el usuario y contraseña, además de otros campos, y en la respuesta te dice si el usuario se ha autenticado con éxito. Al parecer, esta solución fue buena pero en mi grupo se cambió la autenticación de los dispositivos contra un servidor de Clearpass, quien utiliza el

protocolo AAA de Autenticación, Autorización y Registro [3]. Para comunicarnos con esta plataforma, tuve que implementar finalmente Tacacs+ como protocolo final, por lo que la versión definitiva del proyecto incorpora este tipo de Login con protocolo Tacacs+ frente al protocolo LDAP que implementé anterior a éste.

6.4.3. Cambios en el algoritmo de búsqueda ARP

Aunque el algoritmo era funcional con el diseño inicial, la depuración gracias a casos de uso y una batería de pruebas propuesta por algunos miembros de mi grupo de trabajo ha sido esencial para perfeccionar y pulir el algoritmo todo lo posible.

Comenzamos con la primera implementación, realizada en los primeros días de proyecto (Figura 6.1). Si nos fijamos bien, el diseño del modelo de dominio es muy simple: una clase Device que contiene atributos para representar a sus datos: nombre, IP, vendedor, tipo de dispositivo y conexión son los más importantes. De él salen dos clases con relación uno a muchos, que son las tablas ARP y EthernetSwitching. De esta manera, primero se buscaba la relación IP-MAC en routers y, posteriormente, se busca por qué interfaz se ve esta MAC en las tablas EthernetSwitching. Esta opción, aunque sencilla, mostraba la respuesta, pero había un problema: mostraba todas las respuestas que se encontraban en las tablas EthernetSwitching, respuestas que no eran el dispositivo final, pues en ellas se indica que se está viendo esa dirección a través de otro dispositivo que está conectado al dispositivo en cuestión.

Para mejorar los resultados que se muestran, se incluyó una nueva tabla llamada Interface. Antes, la interfaz era simplemente un atributo de cada tabla pero ahora hay una nueva tabla, que también incluye la descripción y el nombre de las interfaces de ese dispositivo, lo que significa que una vez encontrada la interfaz por la que se ve una MAC, también encontramos el nombre y la descripción de ésta. Este hecho, además de permitirnos obtener unos datos relevantes sobre la interfaz, nos permite ver si la interfaz es lógica (es decir, es un puerto del dispositivo) o es una interfaz virtual. Si el algoritmo detecta que no es una interfaz lógica (es decir, no es un puerto del dispositivo), no incluye este resultado en el array final que se mostrará por pantalla al usuario.

Por último, gracias a probar más ejemplos y casos, vimos que hay redes que, aunque no sean una interfaz física, pueden ser un resultado a mostrar. Estos resultados, aunque en vez de verse a través de una interfaz lógica se vean a través de una VLAN o de un agregado, también son necesarios, pues la información que muestran es crítica en caso de querer buscar su dirección IP o MAC. La pregunta era ¿cómo sabemos que una interfaz que no sea un puerto está asociada con un dispositivo final?

Aquí entra en juego la tabla de Neighbors, que muestra los dispositivos que hay conectados a un dispositivo. Si tenemos un switch, consultar esta tabla de Neighbors nos mostrará qué hay conectado por cada interfaz del dispositivo. Es una tabla que ha sido realmente útil, pues en la descripción está escrito (por los responsables que han configurado estas conexiones) información sobre el dispositivo al que está conectada cada interfaz. Así pues, si al otro lado vemos que el nombre es de un dispositivo de nuestra lista, sabemos que no es una dirección final, sino que simplemente está viendo esa MAC a través de otro dispositivo al que está

conectado. El diagrama final de dominio lo vemos en la Figura 6.4, que es ya la versión que se documenta en el Capítulo de Análisis (Figura 3.4).

6.4.4. Cambios en el modelo de dominio

Los cambios en el modelo de dominio han sido motivados tanto por los cambios en el algoritmo de búsqueda ARP como en mejoras para tener un modelo más consistente. Recordemos que el proyecto ha ido siguiendo un desarrollo ágil, abordando las diferentes historias de usuario sprint a sprint.

En primer lugar, como vemos en la Figura 6.1 podemos ver un modelo bastante simple. Al principio sólo se pensó en representar las clases necesarias para una búsqueda preliminar de las tablas ARP. Aquí tenemos sólo la figura del dispositivo y sus tablas.

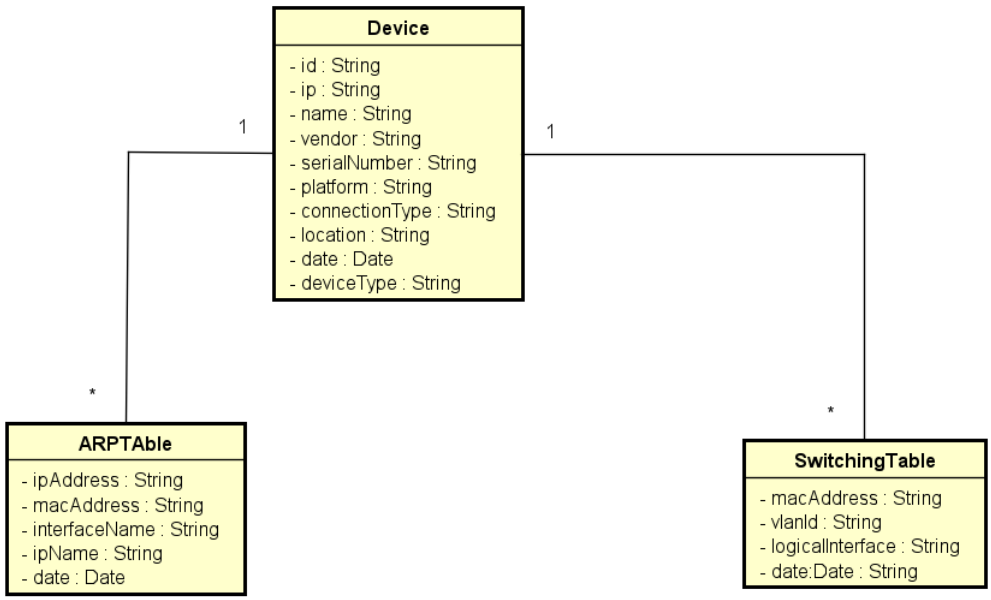


Figura 6.1: Modelo de dominio inicial

Más adelante, se empezaron a detectar ciertos cambios que se pueden implementar. Entre ellos, que el dispositivo sólo puede ser un tipo de dispositivo a la vez. Un router, aunque puede actuar con funciones de switch, no es un switch, por lo que tiene sentido añadir en Device una modelado de especialización/generalización.

También se incluye la tabla Interfaces que interactúa con ambas tablas existentes. Estos cambios se recogen en el modelo de dominio de la Figura 6.2.

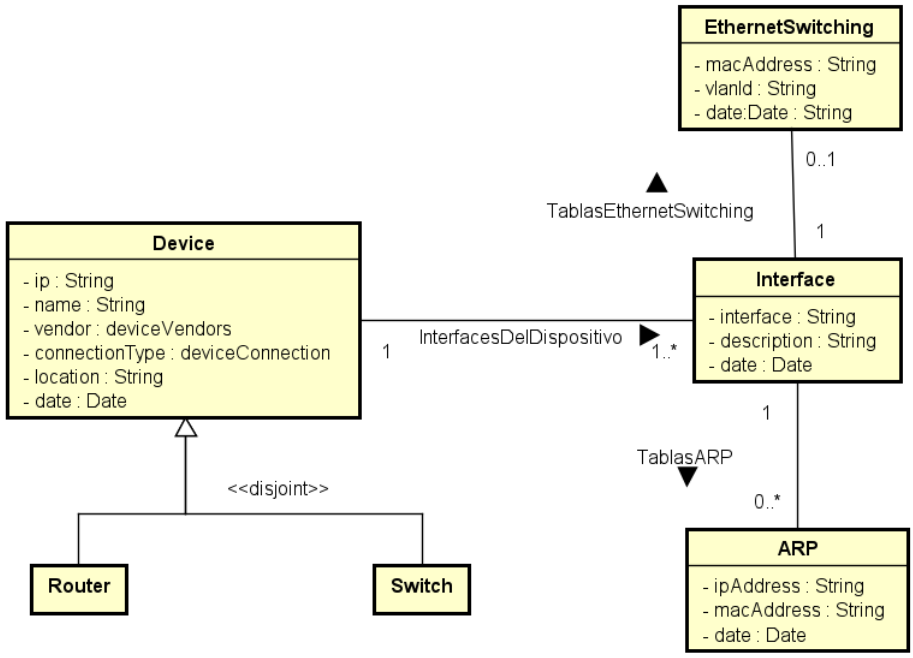


Figura 6.2: Modelo de dominio versión 2

En el momento que detectamos algunos casos de uso que no mostraban las interfaces finales porque no las veían por una interfaz lógica, fue necesario añadir Neighbors. De esta manera, en la tabla Neighbors se listan los dispositivos que están conectados al dispositivo seleccionado a través de sus interfaces. También, se utiliza la clase Interfaces para añadir información extra muy útil para el usuario, además de poder hacer comprobaciones a la hora de filtrar resultados no deseados.

Como también fue necesario añadir los Firewalls para tener sus tablas ARP, se añaden a la generalización de los dispositivos. Para ver los cambios reflejados, consultar Figura 6.3.

Por último, aunque no se había pensado en incluir los configlets y sus aplicaciones en el modelo de dominio al no ser parte del servidor back-end, he considerado incluirlos, al formar parte de toda la lógica del proyecto. Así pues, en el modelo de dominio final también aparece el objeto Configlet, relacionado con el dispositivo gracias a una clase de asociación llamada *ConfigurationJob*, que incluirá qué datos del configlet ha modificado el usuario para aplicarlo en uno o más dispositivos. Esta configuración la hará un usuario, por lo que la clase *User* también se incluye en el diagrama final (Figura 6.4).

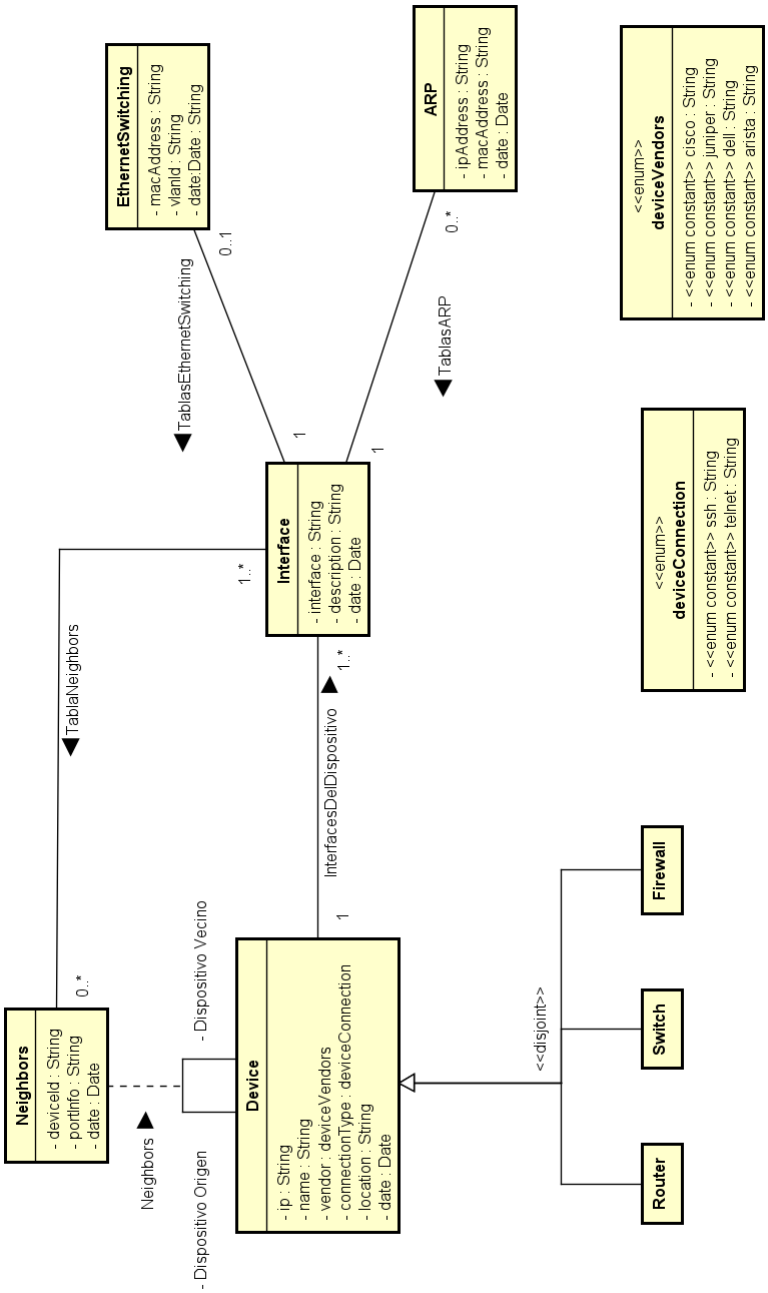


Figura 6.3: Modelo de dominio sin Configlets

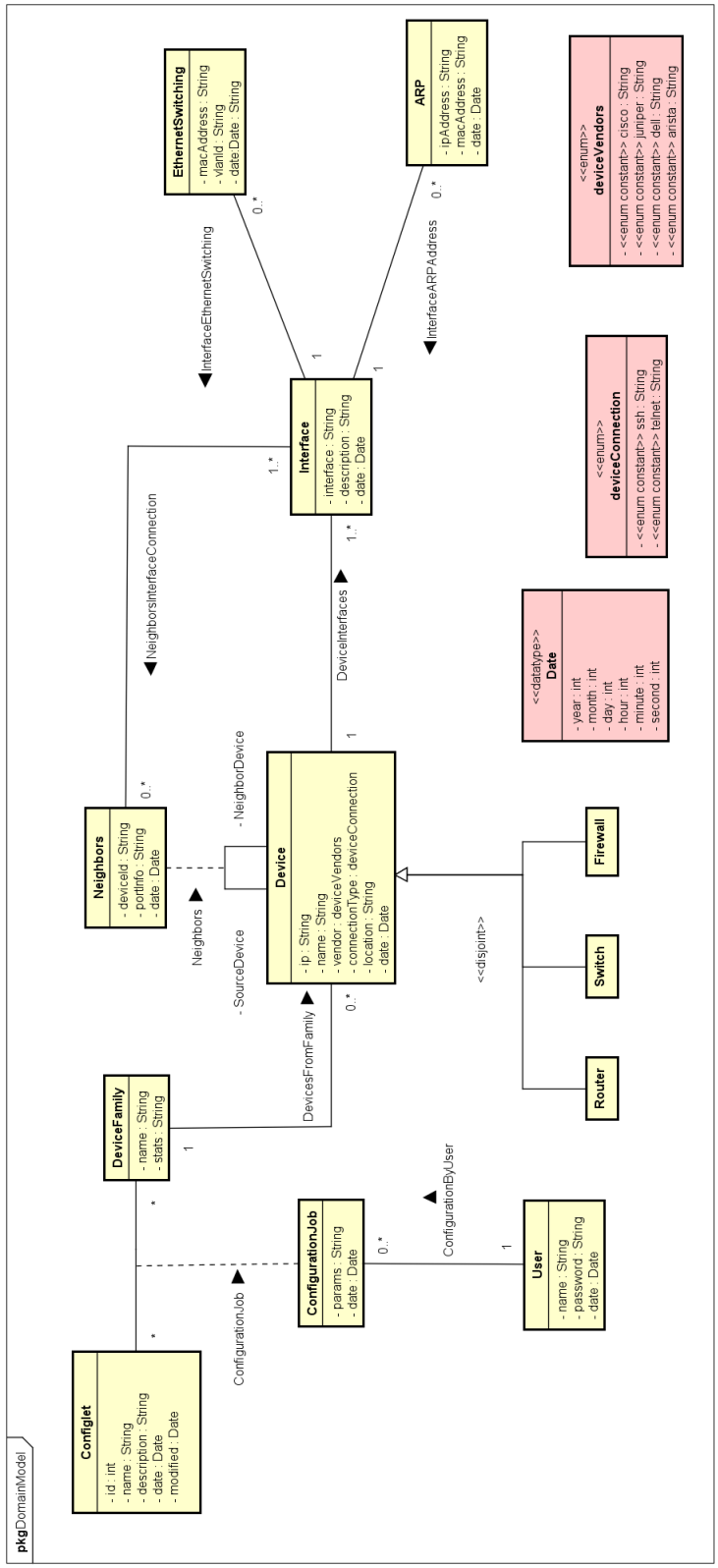


Figura 6.4: Modelo de dominio definitivo

6.4.5. Cambios en el modelo relacional de la base de datos

Al igual que sucedió con los modelos de dominio, cada cambio que se ha realizado en la web ha provocado que se actualizara el modelo relacional de la base de datos. En este subapartado vamos a ver cómo ha ido evolucionando la base de datos de nuestro proyecto a través de su modelo relacional y, a su vez, explicando qué cambios son los que han motivado estas modificaciones y el resultado de éstas.

El primer modelo relacional de la base de datos se puede observar en la Figura 6.5. En ella vemos la tabla `devices`, cuyas claves primarias son el ID del dispositivo y su dirección IP. A su vez, tenemos las tablas `arp_table` y `ethernet_switching_table`. Ambas tienen como clave foránea (que a su vez también es primaria) el campo `ip_addres`, que referencia al campo `ip` de la tabla `devices`.

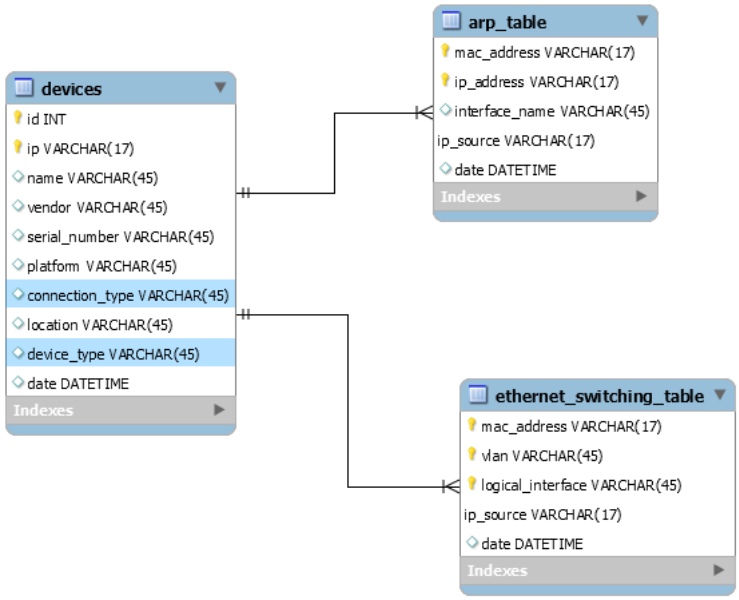


Figura 6.5: Modelo relacional de la base de datos (1)

Bajo este primer modelo se comenzó a desarrollar el algoritmo de búsqueda ARP. No obstante, surgieron dos problemas derivados de este modelo relacional inicial:

1. Algunos campos de la tabla `device` necesitan ser consistentes. Entre ellos, el tipo de conexión, el tipo de dispositivo y, por último, el fabricante. Estos campos son `device_type`, `connection_type` y `vendor`, respectivamente.

2. Para optimizar el algoritmo, necesitamos una tabla añadida que muestre las interfaces y sus descripciones.

Una vez analizadas las necesidades para desarrollar un nuevo modelo relacional, se cambia el modelo de la base de datos y la implementación de este. Así, surge el siguiente modelo relacional (Figura 6.6).

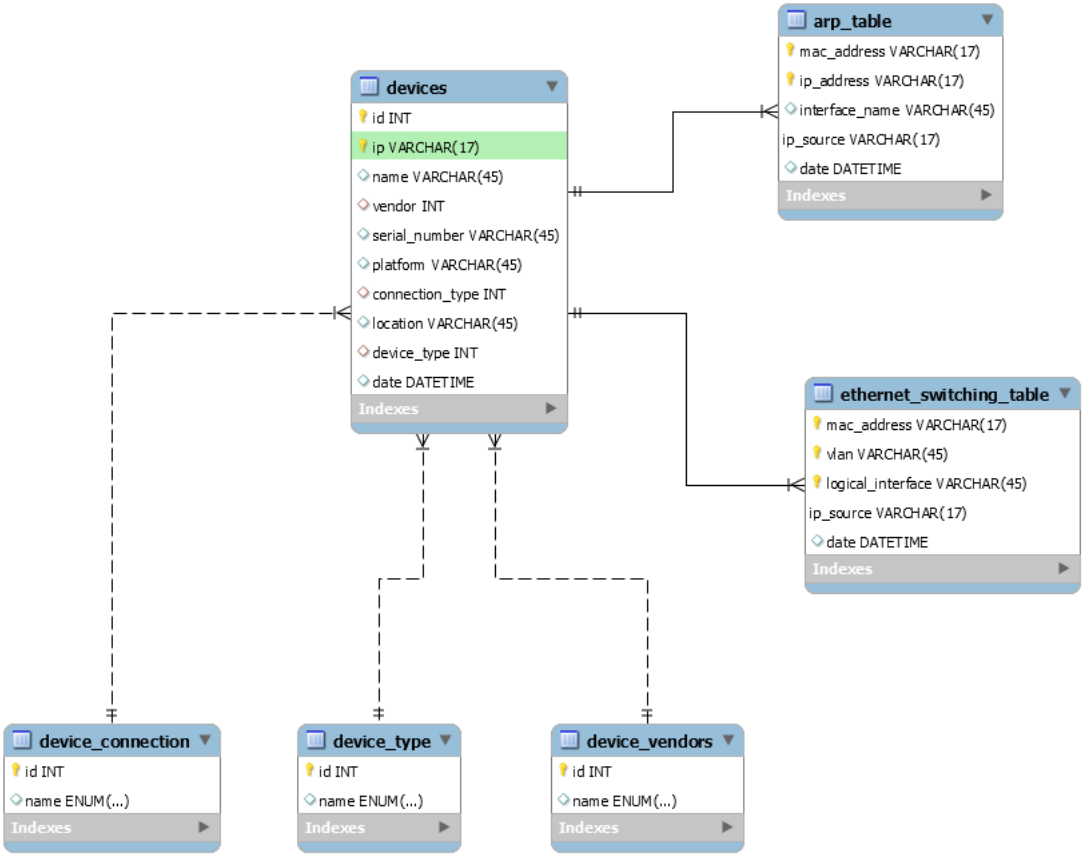


Figura 6.6: Modelo relacional de la base de datos (2)

Aquí podemos ver las correcciones sobre los dos problemas encontrados en la versión anterior del modelo relacional. Para garantizar consistencia, estos campos (que sólo pueden tener unos valores predefinidos por el dominio de redes sobre el que trabajo) se cambia a enum. De esta manera, la tabla `devices` obtiene el valor de estos campos comprobando el ID de la tabla `enum` correspondiente. En cuanto a la tabla de interfaces de cada dispositivo, aquí no se incluye, pues se encuentran problemas de compatibilidad en fabricantes a la hora de referenciar con claves foráneas debido a distintas nomenclaturas, lo que puede generar violaciones de la consistencia de la base de datos. Estos cambios se verán en el siguiente modelo (Figura 6.7).

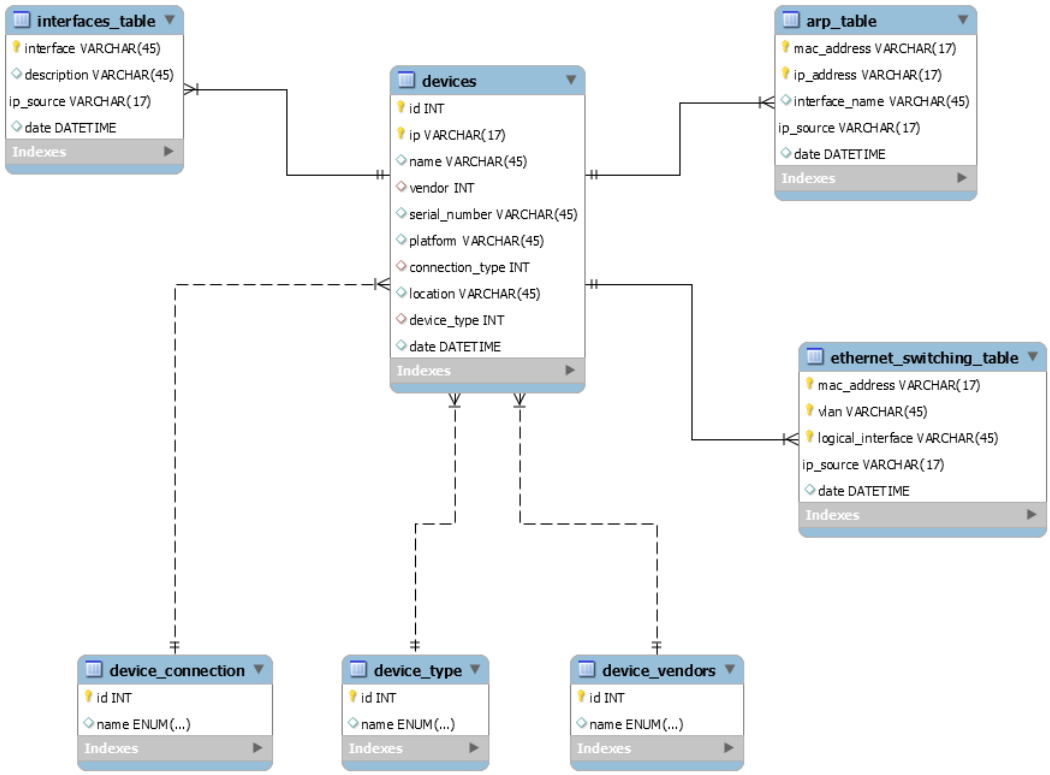


Figura 6.7: Modelo relacional de la base de datos (3)

En este modelo podemos ver cómo se añade la tabla `interfaces_table`. Al igual que `arp_table` y `ethernet_switching_table`, tiene una referencia a la dirección IP de la tabla `devices`. La solución que se ha encontrado es establecer esta tabla sin relación a las dos tablas mencionadas antes. Esta decisión es debido a que el formato entre distintos vendedores tiene una nomenclatura distinta, incluso se ha dado el caso que en Cisco, según el dispositivo y su versión, se nombran de manera distinta. Además, se toma esta decisión debido a que hay interfaces virtuales que no salen reflejadas en alguna tabla, por lo que esa clave foránea tendría que ponerse a `null`. Así pues, dejando esta tabla únicamente relacionada con `devices`, se evitan estos problemas.

Por último, como comentamos a la hora de modificar el modelo de dominio, hace falta ver los vecinos que tiene cada dispositivo para comprobar que el resultado no es un dispositivo dentro del dominio de mi grupo de trabajo. Así pues, en la Figura 6.8, podemos ver cómo se añade la tabla `neighbors_table` que, al igual que las anteriores, hacen referencia a la IP de la tabla `devices`. De esta manera, el modelo relacional de la base de datos evoluciona hasta dar como resultado el diseño que muestra esta figura.

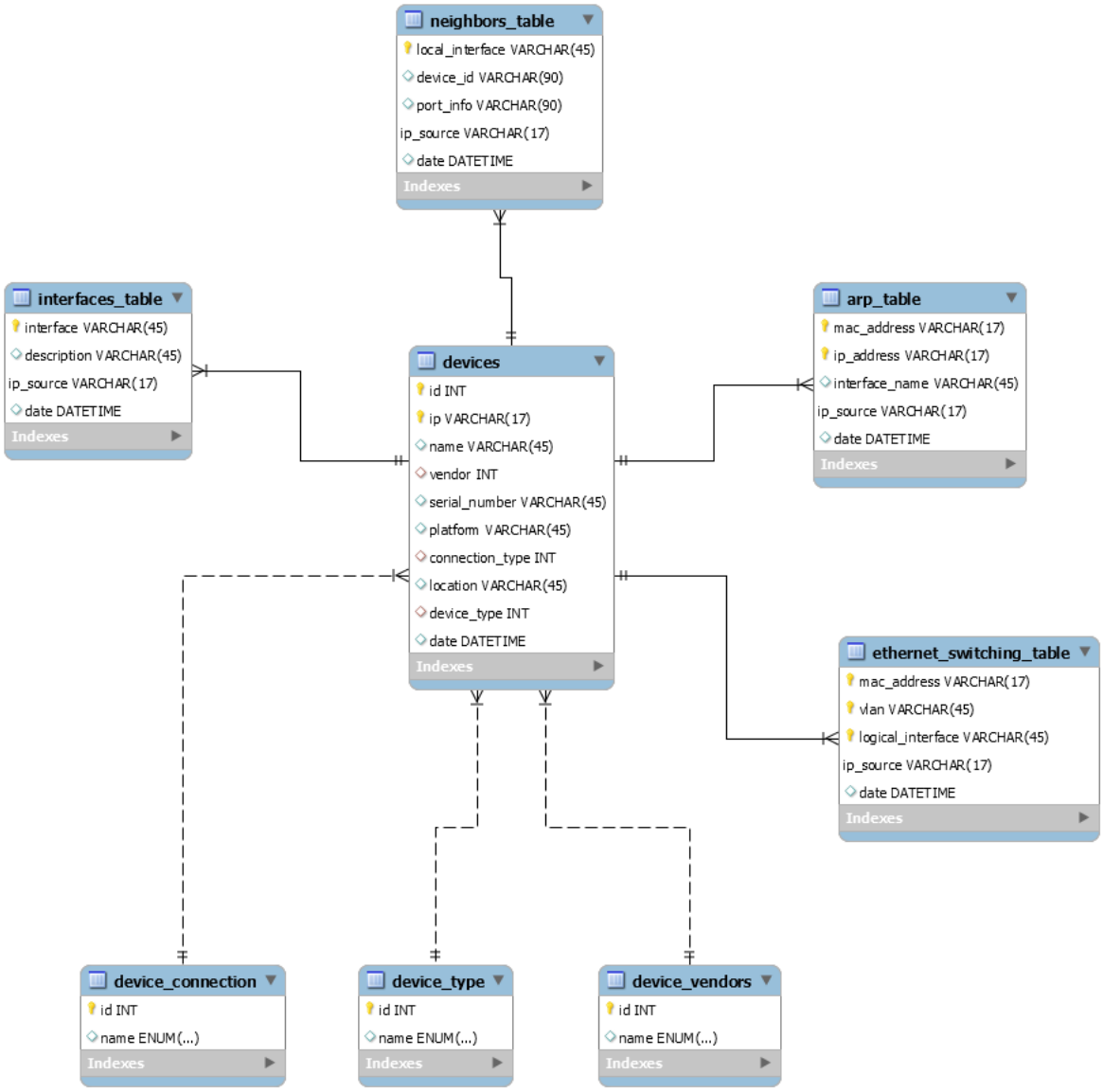


Figura 6.8: Modelo relacional de la base de datos final

6.5. Pruebas

En las secciones anteriores se han explicado tanto la estructura de ficheros (ya sea en el front-end como en el back-end) como las decisiones tomadas a lo largo del proyecto, así como cambios importantes que se han ido realizando durante desarrollo. Los cambios en la implementación, en especial los relativos al algoritmo de búsqueda ARP, se han dado gracias a seleccionar una batería de pruebas suficiente para detectar posibles fallos, sugerencias de implementación, etc. Esta batería de pruebas, por lo general, se ha ido aumentando y ampliando a medida que lo hacía la funcionalidad de la web y ha sido, en su mayor parte, confeccionada por el Product Owner del proyecto.

El método escogido para realizar el apartado de pruebas se define como desarrollo guiado por pruebas de software, más conocido por **Test-Driven Development (TDD)**. El desarrollo TDD es una práctica de ingeniería que consiste en diseñar primero las pruebas que se van a hacer para, posteriormente, escribir el código fuente (o las modificaciones a incluir en el código fuente), desarrollar una implementación que pasa las pruebas y hacer un refactoring para mejorar el código desarrollado [27]. El proceso de diseño de software, combinado con metodologías ágiles en nuestro caso, se define así:

1. El cliente elige una H.U..
2. Se escriben junto al cliente los criterios de aceptación de esta H.U. simplificándolos lo máximo posible.
3. Se escoge el criterio de aceptación más simple y se convierte a una prueba unitaria.
4. Se verifica que esa prueba falla.
5. Se escribe la implementación del código fuente.
6. Se ejecutan las pruebas automatizadas.
7. Se refactoriza el código.
8. Se actualiza la lista de criterios de la H.U. y se vuelve al punto 3.

El modelo de implementación que se ha escogido es la técnica triangular. Esta técnica se basa en tres pasos de manera cíclica [24]:

1. Se escoge el caso más fácil para resolver.
2. Se aplica el algoritmo del TDD.
3. Repetir los pasos 1 y 2 cubriendo nuevas casuísticas, esta vez más complejas.

Si bien es cierto que, por lo general, realizar una batería de pruebas para la mayoría de las historias de usuario ha sido una tarea bastante trivial, para la búsqueda ARP ha sido bastante más complejo, tanto en extensión como a la hora de seleccionar qué casos iban a ser interesantes. Entre las razones de su dificultad, podemos encontrar las siguientes:

1. El entorno de prueba es cambiante. Por ejemplo, hay casos muy concretos que pueden ser de utilidad pero en ese momento no se pueden realizar porque la configuración del dispositivo todavía no está disponible en la base de datos o, por ejemplo, si un dispositivo ha sido modificado recientemente, la información que se puede buscar está obsoleta. Una de las posibles soluciones hubiera sido mockear pero, debido a la naturaleza del Desarrollo TDD, no habría sido nada fácil detectar determinados casos que han ido surgiendo a medida que se modificaba el entorno.
2. Parte del algoritmo se basa en comprobar algunos campos de las tablas que son descripciones que ha realizado un administrador encargado de definir la configuración del equipo. Si esa definición es errónea, el usuario puede ver por pantalla resultados que no son los correctos. Aun así, es necesario mencionar que, en caso de mostrar varios resultados, el usuario tiene siempre el suficiente conocimiento para, gracias a la información final mostrada, descartar los dispositivos incorrectos.
3. Los diferentes fabricantes muestran formatos distintos en las tablas. Para sobrepasar este obstáculo, han sido necesarios bastantes casos de prueba para depurar la manera a la hora de almacenar la información.

A continuación, se describirá una tabla que ha permitido describir qué casos se han usado en cada versión del modelo de dominio. Como puede resultar algo complejo, primero explicaré qué razonamiento he seguido para escoger los casos de prueba y, posteriormente, las columnas de la tabla y el porqué de usar estas columnas para realizar las pruebas.

En primer lugar, se debe hacer mención a que, en cada nueva versión, el algoritmo iba siendo más restrictivo y seleccionando mejor las respuestas. Si bien al principio el resultado era siempre correcto (en caso de que existiera lo que se quería buscar), en el resultado incluía uno o varios resultados que había que "deshechar". En cada versión, los cambios introducidos (consultar Sección 6.4.4) se ha realizado con la idea de ver qué resultado era válido o, mejor dicho, qué condiciones había que tener en cuenta para eliminar resultados que no fueran el dispositivo buscando en cuestión. De esta manera, tanto añadir tablas que tuvieran en cuenta las descripciones de las interfaces o los vecinos que tiene cada dispositivo, así como tener en cuenta qué agregados son de tránsito, ha permitido que el algoritmo muestre únicamente el resultado deseado según el contenido de la base de datos.

Ahora, una vez explicado los cambios, vamos a ver cómo se compone la tabla de batería de pruebas que se ha realizado en cada nueva modificación del algoritmo. Esta tabla (Tabla 6.1) nos ha permitido añadir una columna en cada modificación para ver si el resultado cumplía con esa nueva restricción. Con cada cambio, se volvían a buscar estos resultados y, si alguna de las columnas anteriores cambiaba, significa que había que revisar el código, pues había algún problema que resolver en el nuevo código. Vamos a ver una explicación de las columnas:

COR **Correcto**: en otras palabras, la búsqueda ha tenido éxito, aunque la respuesta sea múltiple y haya que descartar datos o, por el contrario, la dirección no exista y en consecuencia no se pueda mostrar un resultado.

ENC **Encontrado**: en caso afirmativo, significa que la respuesta esperada se ha encontrado en la base de datos, por lo que se mostrará por pantalla. En caso contrario, significa

que se ha encontrado un registro ARP pero no se encuentra el dispositivo final al que pertenece. Esto puede deberse a que el dispositivo se ha eliminado pero se conserva el registro ARP en los routers/firewalls o, lo que es más común, el dispositivo final está directamente conectado a un switch que aún no hemos introducido en la base de datos (ya sea porque es nuevo, no se ha incluído todavía, no está visible, etc).

INT **Interfaz**: indica si el resultado es una interfaz física o lógica. Es la siguiente comprobación necesaria. Si es una interfaz física, el dispositivo se incluye como candidato a dirección final pero, si es virtual, deben realizarse comprobaciones adicionales.

NUM **Número**: el resultado que se ha encontrado es único (uno) o está compuesto de múltiples dispositivos (varios). Gracias a las interfaces y su descripción, se logra desechar varios dispositivos (en el caso de tener respuesta múltiple) pero no es suficiente con este nivel de comprobación.

VEC **Vecinos**: indica si ha sido necesaria la comprobación de vecinos del dispositivo. Para ello, se consulta la tabla `neighbors_table` indicada en la Figura 6.8. Esta tabla nos dice si la interfaz por la que vemos el resultado es un dispositivo de nuestra lista, lo que quiere decir que no sería una dispositivo final sino otra ruta por donde podemos acceder a él.

OUT **Output**: salida esperada para el usuario. Para simplificar la tabla, numeraremos los distintos resultados posibles que se muestran al usuario:

1. Dirección IP o MAC incorrectas, si el usuario ha introducido una cadena de caracteres no válida.
2. Dispositivo no encontrado. Aunque la dirección IP/MAC sea correcta, el dispositivo no se encuentra.
3. El dispositivo se encuentra en las tablas ARP pero no se encuentra un resultado final. Esto quiere decir que las tablas ARP de los routers y firewalls tienen ese registro en memoria, pero se ha borrado del dispositivo final o no se encuentra.
4. Se muestra el resultado final, aunque puede mostrar más de uno, además del correcto. En este caso, se mostraría el resultado en varias filas con los campos correspondientes. En los casos excepcionales en los que el algoritmo muestra más de una fila, es el usuario el que debe saber identificar cuál es la dirección final gracias a los parámetros de las filas. Para estos casos, comentaré en la columna de la descripción por qué se muestran varios.

DES **Descripción**: se añade una descripción para comentar casos muy concretos.

Una vez que hemos visto las columnas y su significado, la batería de pruebas se define finalmente en la Tabla 6.1.

Búsqueda	COR	ENC	INT	NUM	VEC	OUT	DES
10.95.132.153	-	-	-	-	-	1	El texto introducido no es ni una dirección MAC ni una IP
10.95.132.153	Sí	Sí	Física	Uno	No	4	Todo correcto
10.95.80.227	Sí	Sí	Física	Varios	No	4	Todo correcto
10.95.223.24	Sí	Sí	Física	Varios	Sí	4	Correcto. Primer caso en el que se aplican restricciones por vecinos
10.95.217.104	Sí	Sí	Física	Varios	Sí	4	Todo correcto
b8:af:67:2d:65:00	No	No	-	-	-	2	Esta MAC no existe en ninguna tabla ARP
10.95.132.178	Sí	Sí	Virtual	Uno	No	4	Primer caso tras incluir interfaces virtuales
10.95.133.102	Sí	Sí	Virtual	Uno	No	4	Todo correcto
10.95.55.74	Sí	Sí	Virtual	Varios	No	4	Todo correcto
10.95.80.20	Sí	Sí	Virtual	Varios	No	4	Correcto. Esta IP sólo se ve a traves de los Firewalls
10.95.100.79	Sí	No	-	-	-	3	Sólo existe el registro ARP de esta dirección, pero volverá a aparecer en el futuro
10.95.63.51	Sí	Sí	Virtual	Varios	No	4	Todo correcto
00:25:90:a9:ff:62	Sí	Sí	Física	Varios	Sí	4	Todo correcto
10.95.80.185	Sí	No	-	-	-	3	No lo encuentra debido a que esta IP corresponde a un switch y no a una dirección final
10.95.80.188	Sí	No	Sí	Sí	Uno	3	No lo encuentra debido a que esta IP corresponde a un switch y no a una dirección final
0c:c4:7a:66:ca:37	No	Sí	Sí	Varios	Sí	4	Es correcto, pero muestra 2 respuestas porque no tiene manera de saber cuál es correcta
10.95.80.131	Sí	No	-	-	-	3	No lo encuentra debido a que esta IP corresponde a un router y no a una dirección final
10.95.132.178	Sí	Sí	Virtual	Uno	No	4	Todo correcto
10.95.116.221	No	Sí	-	Varios	No	4	En este caso, se busca una dirección de un dispositivo. Como no está en la base de datos, el algoritmo no puede "saber" que esto es un dispositivo, y muestra todos los datos por donde lo ve
10.95.95.90	No	Sí	Sí	Varios	No	4	Aparecen varios resultados, pero el resultado debería ser el output 3. Salen varios dispositivos porque el algoritmo no tiene forma de saber que esto es un router por la descripción de las interfaces

Tabla 6.1: Batería de pruebas (Modelo de dominio 1)

6.6. Licencia

El presente proyecto se ha desarrollado bajo un entorno empresarial, por lo que se le ha dotado de una licencia privativa de software. Queda prohibido su uso, distribución, redistribución, copia, modificación o cesión de parte o la totalidad de este código.

Debido a ello, el repositorio donde se aloja es privado, y no se puede realizar ningún cambio ni ninguna actualización en él sin el consentimiento del autor.

Capítulo 7

Seguimiento del proyecto

7.1. Seguimiento de los sprints

En el presente capítulo se va a documentar cómo se ha desarrollado cada sprint, con las anotaciones pertinentes asociadas a cada uno de ellos. Los sprints, en este caso, tienen una duración semanal de 30 horas. Al final cada uno de estos periodos, se ha planificado una reunión con la tutora (quin cumple el rol de *Scrum Master*) para una actualización de la información, ver el estado en el que se encuentra el proyecto y preparar el comienzo del sprint siguiente.

Durante la planificación del sprint, que tiene lugar cada lunes, se deciden qué historias de usuario se van a abordar en el siguiente sprint, además del tiempo empleado en cada una de ellas. Para ello, recogemos en una tabla qué tareas estimamos realizar en cada sprint. La tabla consta de los siguientes apartados:

- **H.U.**: Representa el número de la historia de usuario en cuestión. Cada subtarea suele estar ligada a una historia de usuario que se pretenda realizar en el sprint, lo que ayuda a planificar los tiempos empleados en caso de que se desarrolle más de una historia de usuario por sprint. También puede ocurrir que la tarea sea de refactoring, corregir código general o afectar a varias historias de usuario. En ese caso, al no estar relacionada con una H.U., se indica con un guión (-).
- **Tarea**: número asignado a la tarea.
- **Descripción**: breve descripcción de la tarea.
- **Estimado**: tiempo estimado, en horas, asignado a la tarea.
- **Empleado**: tiempo empleado, en horas, asignado a la tarea.
- **Estado**: estado final, al acabar el sprint, de la tarea. Puede ser *Completado*, *No iniciado* o *En progreso*. Como caso especial, algunas tareas se declararán *En pausa* si están bloqueadas por otras tareas.

7.1.1. Sprint 1

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
01	T-1	Definir las tareas de backlog	2h	2h	Completado
01	T-2	Estructurar puntos de la memoria	1.5h	1h	Completado
01	T-3	Arquitectura y estructura del proyecto	5h	5h	Completado
01	T-4	Comunicación con las distintas APIs para obtener datos y realizar peticiones REST	8h	18h	Completado
01	T-5	Documentación del sprint	0.5h	1h	Completado
01	T-6	Bocetado de las vistas para la H.U. 1	2h	3h	Completado
01	T-7	Creación de las vistas necesarias para la página de dispositivos	1h	1h	Completado
01	T-8	Creación del componente para mostrar los dispositivos y poder aplicar filtros de búsqueda	2h	1.5h	Completado
01	T-9	Creación del componente de cabecera de la aplicación	1h	1.5h	Completado
01	T-10	Creación del componente barra lateral de la aplicación	2h	1.5h	Completado
01	T-11	Creación de las vistas necesarias para la página principal	2h	2.5h	Completado
09	T-12	Creación de ficheros necesarios para cambio de idioma	0.5h	0.5h	Completado
09	T-13	Añadir cadenas de texto de cada idioma	0.5h	0.5h	Completado
01	T-14	Revisión y documentación del código	1h	-	No iniciado
01	T-15	Test unitarios Consultad de dispositivos	2h	-	No iniciado
01	T-16	Test end-to-end (e2e) Consultad de dispositivos	2h	-	No iniciado
-	T-17	Documentación de la memoria	1h	1h	Completado
TOTAL			**34 horas**	**40 horas**	**14/17**

Tabla 7.1: Sprint 1: 08/02/2021 - 14/02/2021

En este sprint, tienen lugar las primeras tareas de bocetado, arquitectura y diseño del proyecto. Se ha elegido la primera Historia de Usuario (de ahora en adelante, H.U.), para intentar realizarla casi en su totalidad. De esta manera, se puede realizar un primer contacto con las tecnologías empleadas, ver cuánto se tarda tanto en maquetar como en desarrollar el código, cómo se comporta la API, etc. Por ello, se le ha asignado 7 Puntos de Historia (P.H.). Como vimos en la tabla 2.1, se corresponden unas 20 horas con esta puntuación. Por otro lado, la H.U. 09 tiene 5 P.H. pues, aunque se realiza durante varios sprints, son tareas cortas de añadir cadenas de texto.

Al ser la primera tarea, hay mucho trabajo en cuanto a leer documentación, atacar a una API para obtener datos y enviarlos, etc., por lo que una de las tareas críticas como la T-004 se estima que lleve bastantes horas. No obstante, una vez realizada, las siguientes tareas de comunicación con la propia API se estima que lleven mucho menos tiempo al entender cómo funciona y ver si ha habido problemas a la hora de realizar las peticiones.

En este sprint se realizarán las tareas que se describen en la Tabla 7.1.

Como se puede ver, se han tenido que emplear más horas de las previstas debido a que algunas tareas han llevado más tiempo de lo estimado. Las 10 horas aproximadas adicionales que se han tenido que emplear son, en gran parte, consecuencia de la tarea T-004, debido a que tuve problemas a la hora de comunicarme con la API debido a un error de CORS.

`CORS (Cross-origin resource sharing)` es un mecanismo que permite que se puedan solicitar recursos restringidos (como por ejemplo, las tipografías) en una página web desde un dominio diferente del dominio que sirvió el primer recurso [14]. Al desplegar el proyecto desde una máquina virtual, las peticiones a la API (que es un servidor Apache) fueron denegadas. Para solucionarlo tuve que mirar documentación en internet para ver qué significaba este error. La solución fue incluir unas líneas en un fichero del propio servidor para autorizar las peticiones de la IP donde se despliega el proyecto. De esta manera, todas las peticiones y el tráfico que lleguen desde la IP del proyecto se permitirán y podremos comunicarnos con la API a través de peticiones REST.

En cuanto al resto de tareas, no ha sido posible revisar y documentar el código, por lo que esa tarea se traslada al siguiente sprint. Tampoco se ha podido realizar los test. Requerían instalar ciertas herramientas y no he podido, por lo que se ha decidido pasarlo a otro sprint.

7.1.2. Sprint 2

En este sprint se van a incluir las tareas de Login (H.U. 4), consulta de configlets (H.U. 2) y aplicación de configlets (H.U. 3). He elegido continuar con la tarea de Login debido a que, para subir el código, necesito tener en funcionamiento la capacidad de loguearse para no subir datos o credenciales de usuarios, debido a las políticas de seguridad del entorno donde se va a usar esta aplicación. De esta manera, una vez que el usuario se registra, se almacena un token que se incluirá en la cabecera de las peticiones GET y POST. Los P.H. de las H.U. 2, 3 y 4 son 5, 8 y 9, respectivamente. La H.U. 5 tiene una menor puntuación debido a que, al realizar la H.U. 1, he adquirido el conocimiento suficiente para trabajar con la API de Junos de manera más eficiente.

Una vez realizado esto, el paso siguiente será poder ejecutar un configlet en uno de los dispositivos de la lista, con lo cual se completan las H.U. 2 y 3. No obstante, como la tarea del Login es muy laboriosa al tratar temas de seguridad y comunicación con API, está previsto que su finalización tenga lugar en el siguiente sprint. Por ello, se han añadido las historias 2 y 3, bastante más cortas en duración. Las tareas se pueden ver en la Tabla 7.2

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
04	T-18	Bocetado de vista Login	1h	1h	Completado
03	T-19	Bocetado de Configlet y componentes de Configlet	1h	1h	Completado
04	T-20	Bocetado de la aplicación de configlet y sus componentes	1h	1h	Completado
04	T-21	Maquetado de Login	4h	5h	Completado
02	T-22	Maquetado de Configlets y componentes	5h	7h	Completado
03	T-23	Maquetado de componentes de Configlets	3h	2.5h	Completado
04	T-24	Lectura de documentación de la API de ClearPass para autenticación	5h	7h	En progreso
02	T-25	Maquetado de Configlets	1.5h	1.5	Completado
04	T-26	Lógica de Login y envío de datos	1.5h	1.5	Completado
04	T-27	Securización y anonimización de credenciales	5h	-	No iniciado
02	T-28	Peticiones REST para mostrar lista de configlets	1h	0.5h	Completada
03	T-29	Envío de peticiones para aplicar un configlet en un dispositivo	1h	1h	Completada
03	T-30	Mocks para la plantilla de modificación de datos del configlet	0.5h	0.5h	Completada
09	T-31	Añadir cadenas de texto de cada idioma	1.5h	1.5h	Completada
02	T-32	Revisión y documentación del código	1h	1h	Completada
03	T-33	Revisión y documentación del código	1h	1h	Completada
04	T-34	Revisión y documentación del código	1h	1h	Completada
-	T-35	Documentación de la memoria	1h	1h	Completado
TOTAL			**36 h**	**35 h**	**16/18**

Tabla 7.2: Sprint 2: 15/02/2021 - 21/02/2021

Una vez comentada cuál ha sido la previsión de tareas, podemos comprobar que en este sprint surgieron problemas problemas con el propio proyecto relacionado con su árbol de dependencias. Esto ha imposibilitado realizar los test a tiempo, debido a que no ha permitido instalar la herramienta Cypress para los test end-to-end y los unitarios. Como consecuencia, en el sprint siguiente únicamente tendrá lugar la autenticación del login y la realización de los test pendientes.

La tarea de autenticarse a través de ClearPass está en progreso. La API es algo más compleja de lo que pensaba, aunque sí que había tenido en cuenta la complejidad de la tarea, pues es una de las que tienen más Puntos de Historia asociados.

Por último, al hablar de la H.U. 3, se hizo una pequeña prueba piloto con una configuración sencilla para ver que el dispositivo recibe la aplicación de ese configlet. Sin embargo, se deja para más adelante debido a su complejidad.

7.1.3. Sprint 3

Como comenté en la sección anterior, no pude realizar los test de las H.U. debido a un problema de dependencias. Al mover el proyecto y recompilarlo, el problema parece estar solucionado, por lo que se espera que, durante este sprint, se realicen todas estas tareas pendientes para así, por fin, dar por concluidas varias de las H.U. ya realizadas.

Se esperan finalizar completamente las H.U. 01, 02 y 03 y 10. Sin embargo, la 04 no se espera completar durante este sprint debido a su complejidad al tener que trabajar contra una herramienta externa. Para la tarea 10 se fija una cantidad de 3 P.H.. El desglose de las actividades se puede ver en la Tabla 7.3.

Como podemos ver, en este sprint no se ha podido iniciar los test debido a un problema: el Product Owner necesita que, si el usuario no se loguea correctamente, no se puede acceder a ninguna otra vista. Tras tener una reunión con el equipo de desarrollo, se ha decidido implementar el mismo mecanismo de login (mediante un Active Directory) que en el resto de aplicaciones de desarrollo que tienen lugar dentro de la división donde me encuentro desarrollando este proyecto. Puesto que la reunión tiene lugar la semana que viene, de momento las actividades relacionadas con el Login quedan paralizadas, pues dependen de esta actividad. Se espera que en el siguiente sprint, una vez se tenga la reunión se complete la actividad y, por consiguiente, los test del resto de historias de usuario.

En cuanto al resto del tiempo, me he dedicado a revisar documentación sobre tablas ARP y tareas de diseño, por ejemplo, cómo implementar las librerías que permitan conectarse a los dispositivos para, entre otras cosas, manejar una base de datos de más vendedores (no sólo Juniper) y, así, poder realizar las siguientes H.U. de manera más fluida.

7.1.4. Sprint 4

Durante este sprint se prevé finalizar la H.U. 4: Login. Durante esta semana tendrán lugar reuniones sobre qué política se va a aplicar a la hora de realizar el Login, tratar la manera en la

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
01	T-36	Test e2e de Consulta de dispositivos	4h	4h	Completado
01	T-37	Test unitarios de Consulta de dispositivos	4h	1h	En progreso
02	T-38	Test e2e de Consulta de configlets	3h	-	No iniciado
02	T-39	Test unitarios de Consulta de configlets	3h	-	No iniciado
03	T-40	Test e2e de Aplicación de configlets	2h	-	No iniciado
03	T-41	Test unitarios de Aplicación de configlets	2h	-	No iniciado
10	T-42	Creación del componente para ver config. del dispositivo	2h	2h	Completado
10	T-43	Lógica para mostrar la configuración del dispositivo	2h	2h	Completado
04	T-44	Comunicarse con la API de ClearPass a través de peticiones REST	8h	16h	Completado
04	T-45	Revisión y documentación del código	1h	2h	Completado
-	T-46	Documentación de la memoria	1h	1h	Completado
TOTAL			**32 h**	**24 h**	**4/11**

Tabla 7.3: Sprint 3: 22/02/2021 - 28/02/2021

que se van a gestionar las peticiones y ver el Active Directory contra el que se va a autenticar. No obstante, en el sprint anterior ya se realizó un proceso de autenticación contra Junos Space en el que se introducían las credenciales y se guardaban en el `sessionStorage` de la propia aplicación, por lo que se reutilizará parte de este código para realizar la autenticación contra el nuevo servidor.

A lo largo de esta semana se ha planificado finalizar esta tarea securizando y anonimizando las credenciales de usuario necesarias para el Login, además de realizar los test de esta H.U. con la consiguiente modificación del resto de tests. Esto es debido a que, si el usuario no se ha logeado, no se puede acceder al resto de vistas por motivos de seguridad.

Las actividades planificadas para este sprint se encuentran en la Tabla 7.4.

Como hemos visto en la tabla, se ha tenido que utilizar una mayor cantidad de horas debido a que en el anterior sprint no pude avanzar todo lo que me gustaría debido a que tuve que esperar para que el equipo de desarrollo se pusiera en contacto conmigo. Durante este sprint, completé los test de las anteriores Historias de Usuario y, además, implementé la H.U. del Login. No obstante, durante los últimos días se encontró un problema a la hora de autenticar contra los servidores, por lo que se paraliza la implementación actual. Aunque

la tarea está acabada, es posible que se tenga que modificar en el futuro debido a cambios externos en el servidor de autenticación (tarea sobre la que no tengo ninguna capacidad de operar). Debido a esto, se marcan con el estado "En pausa" aunque, como hemos visto, están ya finalizadas y son funcionales.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
04	T-47	Petición con éxito contra el nuevo AD	5h	3h	En pausa
04	T-48	Aplicación del protocolo elegido en la nueva autenticación	8h	5h	En pausa
04	T-49	Integración en la aplicación de las nuevas políticas de autenticación	8h	8h	Completado
04	T-27	Securización y anonimización de credenciales	5h	4h	En pausa
04	T-50	Test e2e de Login	2h	4h	En pausa
04	T-51	Test unitarios de Login	2h	4h	Completado
01	T-52	Modificación test e2e Consulta de Dispositivos	1h	1h	Completado
01	T-53	Modificación test unitarios Consulta de Dispositivos	1h	1h	Completado
02	T-38	Test e2e de Consulta de configlets	3h	1h	Completado
02	T-39	Test unitarios de Consulta de configlets	3h	3h	Completado
03	T-40	Test e2e de Aplicación de configlets	2h	1h	Completado
03	T-41	Test unitarios de Aplicación de configlets	2h	4h	Completado
10	T-54	Creación del componente para ver config. del dispositivo	2h	2h	Completado
10	T-55	Lógica para mostrar la configuración del dispositivo	2h	2h	Completado
-	T-56	Documentación de la memoria	1h	1h	Completado
TOTAL			**47h**	**44h**	**11/15**

Tabla 7.4: Sprint 4: 01/03/2021 - 07/03/2021

La conclusión es que, durante este sprint, se ha completado la tarea de testing de las anteriores H.U., además de redefinir la funcionalidad que debe tener la aplicación para la presentación final. Se ha completado la tarea de Login con un pequeño inconveniente debido a problemas externos, por lo que en el futuro quizá se tenga que volver a cambiar y, por último, se ha pensado en cómo desarrollar el resto del proyecto al haber un cambio en las H.U. del Product Backlog.

7.1.5. Sprint 5

Una vez acabada la H.U. del Login, podemos continuar con el resto de tareas que aportan una funcionalidad importante a la aplicación. Para este sprint, se ha decidido comenzar la H.U. 7: Añadir dispositivo. Al contrario que en las anteriores historias, aquí vamos a incluir también los dispositivos Cisco y de otros fabricantes, por lo que será necesario desarrollar también una base de datos que almacene los datos importantes de éstos y permita hacer búsquedas entre ellos. Para ello también será necesario estudiar qué lenguaje o qué tecnologías permiten conectarse a estos dispositivos y, una vez que esto se ha probado, desarrollar la BD e integrar todo en su conjunto con nuestra aplicación. Es una tarea muy extensa, pues es la primera que necesita arrancar la base de datos y comenzar la comunicación directa con los dispositivos. Para ello, además, es necesario desarrollar un back-end a través del cual podamos ejecutar consultas en la base de datos mediante una API. Debido a los factores comentados, se fija una puntuación de 9 P.H. para esta tarea.

Las tareas asociadas a este sprint son las que se describen en la Tabla 7.5.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
07	T-57	Integrar Flask en la aplicación	8h	10h	Completado
07	T-58	Conexión directa con los dispositivos a través de SSH/-Telnet	14h	10h	Completado
07	T-59	Diseño de las tablas de la BD con SQL	3h	2h	Completado
07	T-60	Conexión con la BD para mostrar datos de los dispositivos	8h	2h	En progreso
07	T-61	Conceptualizar cambios en el diseño y las vistas para implementar las nuevas funcionalidades	2h	2h	Completado
04	T-62	Finalización del Login	8h	6h	Completado
04	T-63	Volver a comprobar los test e2e del Login	1h	1h	Completado
-	T-64	Documentación de la memoria	1h	1h	Completado
TOTAL			**45h**	**34h**	**7/8**

Tabla 7.5: Sprint 5: 08/03/2021 - 14/03/2021

Según se muestra en la tabla, se han completado casi todas las tareas a excepción de una (tarea 65) por falta de tiempo. Este imprevisto ha sido causado por la H.U. del Login, puesto que ya pude finalizarla al saber de qué manera tenía que autenticar y contra qué servidor. Debido a ello, aunque no estaba previsto durante el Sprint Planning, se han añadido estas dos últimas tareas dentro del sprint y, con ellas, se finaliza del todo la H.U. 4.

Como la H.U. 7 queda en curso, se prevé que sea completada durante el siguiente sprint. Únicamente queda por completar la creación y conexión con la base de datos de los dispositivos y mostrarlos por pantalla a través de consultas SQL.

7.1.6. Sprint 6

En este sprint se introduce la H.U. 5: tablas ARP y, además, también se planifica finalizar la historia de usuario 7 que quedó incompleta en el anterior sprint debido a cambios en la política de Login de la aplicación. Al acordarse la política definitiva, esto va a permitir permite avanzar en otras historias de usuario. También se añade como posible tarea la H.U.8: Eliminar dispositivo en caso de que diese tiempo a realizarla.

Como la tarea 5 es, con diferencia, la más larga y compleja del proyecto, no se espera finalizar durante este sprint y se le asignan 9 P.H. Sin embargo, como la H.U. 8 reutiliza mucho código y ya tiene la conexión con la base de datos realizada, sus P.H. asociados se situan en 4. Las tareas asociadas a este sprint las podemos ver en la Tabla 7.6.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
07	T-65	Recogida y tratamiento de datos de un fichero excel/csv para poblar las tablas de la BD	4h	5h	Completado
07	T-60	Conexión con la BD para mostrar datos de los dispositivos	8h	11h	Completado
07	T-66	Ajustar cambios en la lógica de las vistas para mostrar los nuevos dispositivos	3h	4h	En progreso
08	T-67	Eliminar dispositivos	2.5h	-	No iniciado
07	T-68	Test e2e de Añadir dispositivo	1h	-	No iniciado
07	T-69	Test unitario de Añadir dispositivo	5h	-	No iniciado
08	T-70	Test e2e de Eliminar dispositivo	1h	-	No iniciado
08	T-71	Test unitario de Eliminar dispositivo	2h	-	No iniciado
05	T-72	Maquetación de la vista de las tablas ARP	1h	1h	Completado
05	T-73	Diseño de la lógica para las consultas ARP	4h	6h	Completado
-	T-74	Desarrollo de la memoria	6h	6h	Completado
TOTAL			**37.5h**	**33h**	**5/11**

Tabla 7.6: Sprint 6: 15/03/2021 - 21/03/2021

Durante la realización de este sprint ha surgido un problema que me ha impedido acceder a los dispositivos, por lo que se han podido desarrollar las H.U. 7 y 8. Sin embargo, como solución para seguir avanzando, he mockeado de forma local los posibles datos de los dispositivos para probar la base de datos a la hora de inyectar dispositivos en ella. Gracias a ello, he podido comprobar que el diseño de ésta es correcto y que los datos se introducen bien a través de un fichero `.csv` en el cual se pueden ir actualizando por cualquier miembro del equipo de trabajo. También hice pruebas para comprobar si las tablas de la base de datos son correctas y comprender mejor los módulos de Python con los que hacía las llamadas, puesto que los utilizaré para otras H.U. posteriores. El tiempo que no se ha podido emplear en realizar estas tareas, lo he utilizado para completar el documento de la memoria y ampliar algunas secciones.

Las tareas que no se han llevado a cabo se trasladan al siguiente sprint.

7.1.7. Sprint 7

Aquí, en el sprint 7, vamos a adaptar a nuestro código desarrollado localmente en el Sprint anterior a todas las tareas realizadas en otra máquina. Además, una vez acabada esta H.U. 7 (Añadir dispositivo), se pretende finalizar la H.U. 8 (Eliminar dispositivo) para poder eliminar dispositivos y, si fuera posible, comenzar la H.U. 14 (Editar dispositivo), finalizando así las acciones CRUD sobre los dispositivos.

Además, se ha creado una nueva H.U. para mostrar todos los dispositivos actualmente en uso, ya sean del fabricante que sean. Esta tarea es distinta respecto a la H.U. 1 (Consulta de dispositivos Juniper), pues esta última sólo muestra los dispositivos Juniper y, además, los muestra a través de una llamada a la API frente a la H.U. 15., que los obtiene con una llamada al backend, obteniéndolos así de una base de datos. Además, la H.U. 1 permite, a través de la interfaz correspondiente, aplicar los configlets disponibles según el dispositivo, pero no permite realizar acciones CRUD sobre ellos. A la H.U. 15 se le asignan 8 P.H..

El seguimiento de este sprint se puede ver en la Tabla 7.7.

Durante este sprint se han completado todas las tareas estimadas. No obstante, me he dado cuenta de que, muchas de ellas, se pueden mejorar. Por ejemplo, para el siguiente sprint, donde pienso realizar las H.U. 7 y 14 (Añadir y editar dispositivo, respectivamente), sería muy adecuado crear un componente "Dialog" para que muestre un mensaje al usuario preguntando si realmente desea eliminar esos dispositivos o, por otro lado, que muestre los datos del dispositivo que se desea modificar o una pequeña tabla donde puedes añadir los datos que va a tener un nuevo dispositivo. De esta manera, con el componente Dialog, el usuario podrá ver, antes de eliminarlos definitivamente, los dispositivos que se van a eliminar en una lista que se mostrará a modo de confirmación. Como hemos comentado, este componente se podrá reutilizar para las opciones de añadir y editar dispositivo, puesto que el Dialog se mostrará también al pulsar un botón, pero cambiará el contenido en función del botón que se pulse.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
15	T-75	Creación de la BD en el backend de la aplicación	2h	2h	Completado
15	T-76	Conexión de la BD con el frontend	6h	8h	Completado
15	T-77	Adaptar componentes existentes para mostrar los datos de todos los dispositivos	2h	2h	Completado
15	T-78	Creación de botones para acciones CRUD	2h	1h	Completado
15	T-79	Cambio en la lógica de componentes para las siguientes HU	6h	5h	Completado
15	T-80	Creación de la acción de refrescar la lista	2h	1h	Completado
-	T-81	Adaptación de la lógica de backend para mejor escalabilidad	5h	5h	Completado
08	T-82	Backend para query para borrar dispositivos	3h	2h	Completado
08	T-83	Selección de varios dispositivos para posterior borrado	1h	3h	Completado
09	T-84	Añadir cadenas de texto para mostrar según el idioma seleccionado	1h	1h	Completado
08	T-85	Conexión frontend-backend para borrar dispositivos	3h	2h	Completado
-	T-86	Documentación de la memoria	1h	1h	Completado
TOTAL			**34h**	**33h**	**12/12**

Tabla 7.7: Sprint 7: 22/03/2021 - 28/03/2021

7.1.8. Sprint 8

Como comentamos anteriormente, se llevarán a cabo las H.U. 7 y 14 (añadir y editar un dispositivo, respectivamente), además de añadir una pequeña modificación a la H.U. 8 (eliminar un dispositivo). Como la H.U. 14 se basa en varios componentes ya creados para la H.U. 7, se le asigna una puntuación de 7 P.H., puesto que aunque parte del código es muy similar, son necesarias muchas comprobaciones y cambios en la lógica de la vista.

También, se ha creído oportuno mostrar un mensaje de alerta a través de un Alert Component que permita informar al usuario si la operación se ha llevado a cabo con éxito o, si por el contrario, ha habido un error en la base de datos. Para ver lo que se ha llevado a cabo en este sprint, consultar la Tabla 7.8.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
08	T-87	Creación componente Dialog de propósito general.	8h	10h	Completado
08	T-88	Relleno del componente con los datos de los dispositivos a borrar	3h	3h	Completado
07	T-89	Adaptación del componente para la HU 7	5h	4h	Completado
07	T-90	Creación de la llamada axios para añadir el dispositivo	2h	1h	Completado
07	T-91	Creación del backend para añadir dispositivo	4h	5h	Completado
14	T-92	Adaptación del componente para la HU 14	4h	1.5h	Completado
14	T-93	Creación del backend para editar el dispositivo	3h	1h	Completado
14	T-94	Adaptación del frontend respecto a la HU 07 para editar dispositivo	2h	4.5h	Completado
09	T-95	Añadir cadenas de texto para mostrar según el idioma seleccionado	1h	1h	Completado
-	T-96	Aplicación del patrón controlador a todas las vistas creadas con botones	3h	3h	Completado
-	T-97	Documentación de la memoria	1h	1h	Completado
TOTAL			**36h**	**35h**	**11/11**

Tabla 7.8: Sprint 8: 29/03/2021 - 04/04/2021

Con este sprint, quedan realizadas todas las acciones CRUD sobre la lista de dispositivos. No obstante, hay algunos detalles que se irán perfeccionando más adelante en tareas de depuración, pues se necesita conectar directamente con los dispositivos utilizando un módulo Python distinto según el fabricante (`PyEz` para dispositivos Juniper, `Netmiko` para dispositivos Cisco).

También, para el siguiente sprint, se añadirá un componente Alert para poder informar al usuario si su acción se ha realizado o no. Como se han reciclado la mayoría de componentes, no he considerado necesario realizar bocetados, pues la idea al utilizar componentes es la reutilización y despliegue de éstos.

7.1.9. Sprint 9

En este sprint se espera implementar el componente Alert mencionado en el sprint anterior para mostrar si ha habido algún problema al conectar con la BD al realizar cualquier acción CRUD. Además, se continúa con la H.U. 5, puesto que es la tarea más compleja de nuestro proyecto. Para ver las tareas relacionadas a este sprint, consultar la Tabla 7.9.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
-	T-98	Aplicar patrón controlador en otras partes del código	2h	2h	Completado
-	T-99	Creación del componente Alert para feedback acciones CRUD dispositivos	3h	3h	Completado
-	T-100	Pasar código de error o acierto desde backend al alert component	3h	4h	En progreso
-	T-101	Corregido fallo al borrar el dialog de las acciones CRUD	2h	2h	Completado
05	T-102	Cambiar estructura BD para mejor mantenimiento	8h	9h	Completado
-	T-103	Mostrar mensaje de error personalizado que indique fallo de la BD	0.5h	0.5h	Completado
05	T-104	Cambiar métodos y clases del backend	4h	4h	Completado
05	T-105	Enlazar la petición de una IP con una llamada a un dispositivo de la lista	8h	2h	En progreso
05	T-106	Mostrar dinámicamente las tablas ARP	10h	-	No iniciado
09	T-107	Añadir cadenas de texto para cada idioma	1h	1h	Completado
-	T-108	Cambiar en backend para introducir SQLAlchemy	10h	12h	Completado
-	T-109	Documentación de la memoria	1h	1h	Completado
TOTAL			**52.5h**	**40.5h**	**9/12**

Tabla 7.9: Sprint 9: 05/04/2021 - 11/04/2021

Durante este sprint se ha ido cambiando mucha lógica tanto de back-end como front-end para conseguir una aplicación más robusta que obtenga los parámetros de una base de datos consistente. Para que esto tenga lugar, también se ha cambiado el diseño de la base de datos para garantizar dicha consistencia.

También se ha decidido cambiar las llamadas SQL para utilizar un ORM (SQLAlchemy en este caso) para facilitar tareas posteriores. Esto ha hecho que las tareas de enlazar la

petición de una IP y mostrar dinámicamente las tablas ARP se pasen al siguiente sprint por falta de tiempo. También, el cambio de mecánica al utilizar este ORM ha hecho que el tiempo estimado sea muy alto para este sprint debido a que ha desplazado a dos tareas bastante largas en cuanto a tiempo.

7.1.10. Sprint 10

Durante este sprint se continúa con la H.U. 5, además de ir cambiando paulatinamente algunos aspectos de la BD para hacerla más eficiente y mejor estructurada. Las tareas las podemos encontrar en la Tabla 7.10.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
-	T-110	Cambiar modelo de tablas a través de clases	5h	5h	Completado
05	T-111	Almacenamiento de tablas ARP para dispositivos Juniper	3h	3h	Completado
05	T-112	Almacenamiento de tablas Ethernet-Switching (ES) Juniper	3h	5h	Completado
05	T-113	Almacenamiento de tablas ARP para dispositivos Cisco	5h	6h	Completado
05	T-114	Almacenamiento de tablas ES para dispositivos Cisco	5h	8h	En progreso
05	T-105	Enlazar la petición de una IP con una llamada a un dispositivo de la lista	8h	6h	Completado
05	T-106	Mostrar dinámicamente las tablas ARP	10h	-	No iniciado
05	T-115	Gestionar mantenimiento para poblar/eliminar tablas de la BD	2h	1h	Completado
05	T-116	Documentar mediante diagrama de flujo del algoritmo	2h	2h	Completado
-	T-117	Añadir nuevos dispositivos y añadir Firewalls a los tipos de dispositivos	1h	1h	Completado
TOTAL			**44h**	**37h**	**8/10**

Tabla 7.10: Sprint 10: 12/04/2021 - 18/04/2021

Como podemos ver en la tabla, se han desarrollado los métodos para contactar con los dispositivos y poblar las tablas de la base de datos correspondientes. Con la tabla Ethernet-Switching ha habido un problema, puesto que según la versión del dispositivo Cisco, el output es distinto, por lo que capturar estos casos para inyectarlos en una tabla de la base de datos

ha sido complejo debido a la variedad de los dispositivos. No se ha completado esta tarea, por lo que se desplaza al siguiente sprint.

7.1.11. Sprint 11

En el anterior sprint también se continuó con la H.U. 5, por lo que en este sprint se espera finalizarla. Empezaremos con la tarea de poblar la tabla Ethernet-Switching de Cisco, para poder tener todo disponible a la hora de hacer una búsqueda de una dirección MAC desde cualquier dispositivo, ya sea Cisco o Juniper. Las tareas las podemos encontrar en la Tabla 7.11.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
05	T-114	Almacenamiento de tablas ES para dispositivos Cisco	5h	5h	Completado
05	T-118	Buscar dirección MAC física tanto desde IP como desde MAC	8h	8h	Completado
05	T-119	Crear tablas BD para las interfaces	1h	0.5h	Completado
05	T-120	Crear script para poblar las interfaces Juniper	2h	2h	Completado
05	T-121	Crear script para poblar las interfaces Cisco	2h	6h	Completado
05	T-122	Devolver datos del dispositivo final al frontend	15h	12h	En progreso
-	T-123	Documentación de la memoria	2h	2h	Completado
TOTAL			**35h**	**35.5h**	**6/7**

Tabla 7.11: Sprint 11: 19/04/2021 - 25/04/2021

Al finalizar el sprint, la H.U. 05 está casi completada. No obstante, se ha encontrado un fallo para casos muy, muy concretos que deberá ser revisado y, en consecuencia, el algoritmo debe ser modificado. También se ha encontrado un pequeño fallo al poblar las tablas ARP, Ethernet-Switching, pues si se añade un espacio al nombre no detecta el dispositivo. Esto se hará en una pequeña tarea de depuración en el sprint siguiente.

7.1.12. Sprint 12

En este sprint se espera finalizar la H.U. 05 y, además, completar las H.U 12: Eliminar Configlet y 16: Mostrar configuración de cualquier dispositivo. Para estas dos últimas se asignan 3 y 5 P.H. y, por otro lado, para la H.U. 05 se espera añadir casos excepcionales en el algoritmo general, poblar correctamente las tablas de Ethernet-Switching y terminar

de conectar back-end con front-end para poder buscar una dirección (MAC o IP) desde la interfaz web. Las acciones descritas, junto a otras, se pueden ver en la Tabla 7.12.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
05	T-122	Devolver datos del dispositivo final al front-end	6h	6h	Completado
05	T-124	Corregir almacenamiento descripcion de interfaces Juniper	2h	2h	Completado
05	T-125	Corregir almacenamiento tablas Ethernet-Switching de Cisco	3h	3h	Completado
05	T-126	Añadir casos excepcionales algoritmo ARP	2h	3h	Completado
05	T-127	Añadir Alert que informe cuando la búsqueda no se ha realizado con éxito y por qué	2h	1.5h	Completado
05	T-128	Cambio estilo CSS para ver bien las tablas al mostrar respuestas	0.5h	0.5h	Completado
12	T-129	Crear lógica para eliminar un configlet	5h	2h	Completado
12	T-130	Conectar lógica para eliminar un configlet con interfaz web	2h	4h	Completado
12	T-131	Cambio en la lógica de los configlets para reutilizar componentes	3h	3h	Completado
16	T-132	Configurar llamada para obtener configuración de dispositivos Juniper	2h	2h	Completado
16	T-133	Configurar llamada para obtener configuración de dispositivos Cisco	2h	2h	Completado
16	T-134	Mostrar configuración por pantalla	4h	3h	Completado
-	T-135	Cambio en la BD para poder hacer UPDATE y DELETE ON CASCADE	4h	3h	Completado
09	T-136	Añadir cadenas de texto para el lenguaje	0.5h	0.5h	Completada
-	T-137	Documentación de la memoria	1h	1h	Completado
TOTAL			**39h**	**36.5h**	**15/15**

Tabla 7.12: Sprint 12: 26/04/2021 - 02/05/2021

En este caso sí se han completado todas las tareas, pues eran tareas a priori más sencillas y utilizaban componentes y llamadas a APIs que ya se habían realizado anteriormente.

7.1.13. Sprint 13

Una vez acabada la tarea más compleja, vamos a llevar a cabo la H.U. 3 (Ver la configuración de un configlet) y el Logout, recogido en la H.U. 13, la cual cuenta con 4 P.H. asignados. Para consultar las tareas de este sprint, ver la Tabla 7.13.

También, se comenzará la H.U.6: Aplicar configlets, aunque intentaremos aplicar un solo configlet a un dispositivo para, posteriormente, poder aplicar varios. No se espera completar esta última H.U. durante el presente sprint. Esta H.U., al ser mucho más compleja, se le asignan 8 P.H.

Como podemos ver, hay una tarea de refactoring para reutilizar uno de los componentes que muestran las tablas. Para ello, se ha cambiado bastante lógica de tal manera que podamos utilizar el mismo componente para mostrar una tabla seleccionable con cabeceras y datos variables. No obstante, no nos sirve para la tabla que muestra todos los dispositivos, pues tiene a mayores opciones para editar éstos.

Al realizar la H.U. 06 de aplicar un configlet, se ha creado una nueva H.U. para validar dicho configlet, pues aunque no es necesario, puede ser muy útil para confirmar previamente si la configuración que ha introducido el usuario es correcta. De esta manera, cuando se carga el configlet y se rellena el formulario, el usuario puede comprobar si los datos son válidos para una posible aplicación de esa configuración en el dispositivo. Tanto si lo son como si no lo son, se muestra un alert por pantalla indicando al usuario del resultado. Esta H.U. se crea con el nombre H.U. 17: Validar configlet, cuyos P.H. se fijan en 6.

Como ha surgido otra H.U., la historia de aplicar uno o varios configlets queda para el siguiente sprint, además de la H.U. relativa al logout (H.U. 13)

7.1.14. Sprint 14

Para este sprint se trasladan las H.U. 06 y 13. Se esperan finalizar durante este sprint y, por último, finalizar la H.U. 11 que contará con 6 P.H. asignados. (ver Tabla 7.14).

Se ha corregido bugs en la H.U. 6, puesto que al deseleccionar algún dispositivo no se realizaba correctamente. El fallo residía al no copiar correctamente por valor un objeto, pues Javascript lo hace por referencia y, entonces, al modificar el destino, también se modificaba el origen.

También se han eliminado vistas y clases obsoletas que ya no se utilizan, dejando más limpia la estructura del proyecto.

Por último, se ha completado la H.U. 11. Es algo más compleja que la H.U. 6 aunque se han reutilizado algunos componentes y partes de código, lo que ha facilitado la adaptación

de éste a la nueva H.U.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
13	T-138	Borrar datos sessionStorage	2h	-	No iniciado
13	T-139	Borrar los datos al pulsar Logout en la interfaz	2h	-	No iniciado
13	T-140	Mostrar Logout siempre o redirigir a Login	1h	-	No iniciado
-	T-141	Cambio en la lógica del componente para mostrar tablas	6h	8h	Completado
03	T-142	Cambio en la lógica para la reutilización del componente de mostrar configuraciones	3h	5h	Completado
03	T-143	Llamada a la API de Junos Space para mostrar la configuración del configlet	2h	1h	Completado
03	T-144	Conectar lógica para mostrar la configuración del configlet por pantalla	2h	2h	Completado
06	T-145	Bocetado de la vistas tras seleccionar un configlet	2h	2h	Completado
06	T-146	Conectar rutas dinámicas de Configlets a ApplyConfiglets	2h	2h	Completado
06	T-147	Maquetado de la vistas tras seleccionar un configlet	6h	8h	Completado
06	T-148	Aplicación de configlets vía API	6h	-	No iniciado
09	T-149	Añadir cadenas de texto para el lenguaje	0.5h	0.5h	Completado
17	T-150	Bocetado de la vista para mostrar la validación de configlets	1h	1h	Compleatdo
17	T-151	Maquetado de la vista para mostrar la validación de configlets	3h	5h	Completado
17	T-152	Lógicas para las llamadas a la API para validar configlet	2h	2h	Completado
17	T-153	Conectar validación del configlet para posterior aplicación	2h	3h	Completado
-	T-154	Completar apartados de la memoria	3h	-	No iniciado
-	T-155	Documentación de la memoria	1h	-	No iniciado
TOTAL			**46.5h**	**39.5h**	**12/18**

Tabla 7.13: Sprint 13: 03/05/2021 - 09/05/2021

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
06	T-156	Añadir la opción de que se pueda copiar los parámetros a los otros dispositivos	7h	-	No iniciado
06	T-157	Cambiar scroll al componente	2h	1h	Completado
06	T-158	Comprobar bug al cerrar tabs	4h	2h	Completado
06	T-159	Mostrar alerts cuando se aplique (bien o mal) una config	2h	2h	Completado
06	T-160	Deshabilitar botón si algún configlet no está bien validado	3h	2h	Completado
06	T-161	Comprobar configlets que fallan al querer editarlos y convertirlos en editables	2h	2h	Completado
13	T-138	Borrar datos sessionStorage	2h	-	No iniciado
13	T-139	Borrar los datos al pulsar Logout en la interfaz	2h	-	No iniciado
13	T-140	Mostrar Logout siempre o redirigir a Login	1h	-	No iniciado
11	T-162	Maquetado para la H.U. 11	3h	3h	Completado
11	T-163	Adaptar componentes para la H.U. 11	3h	3h	Completado
11	T-164	Debug a la hora de seleccionar y montar componentes y objetos	3h	5h	Completado
11	T-165	Adaptar componentes para la H.U. 11	3h	3h	Completado
11	T-166	Llamada para aplicar uno o más configlets distintos en un dispositivo	5h	4h	Completado
01	T-167	Mostrar un botón verde o rojo según el estado del dispositivo	1h	1h	Completado
09	T-168	Añadir cadenas de texto para el lenguaje	0.5h	0.5h	Completado
-	T-169	Completar apartados de la memoria	3h	3h	Completado
-	T-170	Documentación de la memoria	1h	3h	Completado
TOTAL			**47.5h**	**34.5h**	**14/18**

Tabla 7.14: Sprint 14: 10/05/2021 - 16/05/2021

Para el siguiente sprint únicamente queda hacer la H.U. 13 (Logout) y tareas de refactoring, especialmente en el CSS para mostrar una interfaz de usuario mucho más adecuada. Adicionalmente, se intentará implementar la opción de copiar valor del formulario de uno a varios configlets, además de otras pequeñas mejoras en la funcionalidad. También se completarán diagramas para dejar documentadas las decisiones tomadas y otros apartados de la memoria, al estar hechas las historias de usuario.

7.1.15. Sprint 15

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
06	T-156	Añadir la opción de que se pueda copiar los parámetros a los otros dispositivos	7h	-	No iniciado
-	T-171	Añadir la opción de "deseleccionar todo" en varias vistas	3h	-	No iniciado
13	T-138	Borrar datos sessionStorage	2h	-	No iniciado
13	T-139	Borrar los datos al pulsar Logout en la interfaz	2h	-	No iniciado
13	T-140	Mostrar Logout siempre o redirigir a Login	1h	-	No iniciado
-	T-172	Cambiar CSS de las tablas para dejar columnas fijas	3h	-	No iniciado
-	T-173	Cambiar CSS de todas las vistas	4h	-	No iniciado
-	T-174	Refactoring de la BD para poderlo automatizar	7h	9h	Completado
05	T-175	Refactoring del input de las direcciones MAC para aceptar y convertir más formatos	2h	2h	Completado
05	T-176	Refactoring para las nuevas tablas de Neighbors en la BD	2h	2h	Completado
05	T-177	Poblar las tablas de Neighbors para los Juniper	2h	3h	Completado
05	T-178	Poblar las tablas de Neighbors para los Cisco	7h	10h	En progreso
-	T-179	Refactoring del código para hacerlo más reutilizable	7h	5h	En progreso
-	T-180	Diseño del nuevo logo y cambio de nombre	3h	-	No iniciado
-	T-181	Desarrollo de la memoria	15h	15h	En progreso
TOTAL			**67h**	**46h**	**4/15**

Tabla 7.15: Sprint 15: 17/05/2021 - 23/05/2021

Durante este sprint, se plantea simplemente finalizar la H.U. 13. Además, se intentará añadir algo de funcionalidad a la H.U. 06 para hacerla más cómoda a la hora de rellenar varios dispositivos para el mismo configlet. Por último, las tareas que se llevarán a acabo serán de mejorar funcionalidad, depuración y cambiar el CSS para mostrar las vistas de manera más clara o más estética. También se limpiará algo de código y se probará distintos escenarios.

Como podemos ver en la Tabla 7.15, se han dejado sin comenzar bastantes tareas relacionadas con CSS en pos de añadir una nueva tabla de base de datos, que nos permitirá todavía afinar un poco más la búsqueda ARP de los dispositivos. De esta manera, se pasan al siguiente sprint, que será únicamente de aspectos visuales y depurar el código, además de eliminar todo aquel que finalmente no se utilice para estructurarlo mejor.

El resto del tiempo se empleará en seguir desarrollando el documento. Durante este sprint se realizan varios puntos de desarrollo de la memoria lo que permitirá avanzar en paralelo la corrección de la tutora.

7.1.16. Sprint 16

Llegamos al que, según lo acordado, será el sprint final en cuanto a código. Debido a esto, se completarán las tareas anteriores y se rematarán otras nuevas, pero no se añadirá nueva funcionalidad a la aplicación. Si bien es cierto que durante el Sprint Planning se acordó que este es el último sprint, siempre se puede añadir otro a mayores. Podemos ver las tareas en la Tabla 7.16.

No se han incluido algunas tareas (por ejemplo, la opción de "deseleccionar todo" o la de copiar parámetros) por falta de tiempo y porque al consultarlo con el Product Owner indicó que tampoco era una funcionalidad esencial, pues no se suele dar el caso en el que llegue a ser útil. Al terminar este sprint, la aplicación ya es totalmente funcional y sólo queda acabar la memoria del proyecto, además de subirlo a la instancia GitLab *on premises* de la que dispone la Escuela, cambiar el logo por defecto y completar información relevante en el *About*.

7.1.17. Sprint 17

Como comentamos antes, el código de la aplicación está acabado. En este sprint, simplemente se llevarán a cabo acciones sobre la memoria, la documentación, cambiar el logotipo de la aplicación, redactar el *About* y otras tareas que puedan surgir al comprobar la batería de pruebas y funcionalidad. Las tareas se han recogido en la Tabla 7.17.

Como podemos ver, algunas tareas han llevado más tiempo del estimado. Por ejemplo, al cambiar el nombre del proyecto, hubo que recompilar todo el entorno virtual de Python. Por otro lado, se depuró y se añadieron restricciones al algoritmo que restringen todavía más los resultados mostrado por pantalla. También, se redactó el *About* de la página que muestra mis datos de contacto empresariales por si hubiese algún error, además de reflejar algo de información sobre las tecnologías usadas en el desarrollo.

Con esto, queda finalizado el proyecto incluida su documentación. Se han realizado en total 17 sprints, distribuidos a lo largo de 17 semanas.

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
06	T-156	Añadir la opción de que se pueda copiar los parámetros a los otros dispositivos	7h	-	No realizado
-	T-171	Añadir la opción de "deseleccionar todo" en varias vistas	3h	-	No realizado
13	T-138	Borrar datos sessionStorage	2h	2h	Completado
13	T-139	Borrar los datos al pulsar Logout en la interfaz	2h	0.5h	Completado
13	T-140	Mostrar Logout siempre o redirigir a Login	1h	2h	Completado
-	T-182	Cambiar CSS de las tablas para dejar columnas fijas	3h	1h	Completado
-	T-183	Cambiar CSS de todas las vistas	4h	4h	Completado
-	T-184	Cambiar CSS del caroussel	4h	2h	Completado
-	T-185	Cambiar CSS del dialog de los dispositivos	4h	1h	Completado
-	T-186	Cambiar CSS de los botones para hacerlos cuadrados	1h	2h	Completado
05	T-178	Poblar las tablas de Neighbors para los Cisco	5h	3h	Completado
05	T-187	Modificar algoritmo Calambuco para que muestre otros mensajes y tenga en cuenta los neighbors	5h	6h	Completado
-	T-179	Refactoring del código para hacerlo más reutilizable	7h	5h	Completado
05	T-188	Creación de logs por cada tabla de la BD al poblar	5h	5h	Completado
-	T-189	Diseño del nuevo logo y cambio de nombre	3h	-	No iniciado
TOTAL			**56h**	**33.5h**	**12 / 15**

Tabla 7.16: Sprint 16: 24/05/2021 - 30/05/2021

H.U.	Tarea	Descripción	Estimado	Empleado	Estado
-	T-190	Redactar información de la vista About	2h	1h	Completado
-	T-191	Testing y detección de errores	7h	12h	Completado
-	T-192	Depuración algoritmo búsqueda ARP	7h	10h	Completado
-	T-189	Diseño del nuevo logo y cambio de nombre	3h	5h	Completado
-	T-193	Completar la memoria del proyecto	18h	18h	Completado
-	T-194	Limpiar código y crear nuevo proyecto git para no añadir datos sensibles	3h	5h	Completado
TOTAL			**40h**	**51h**	**6/6**

Tabla 7.17: Sprint 17: 31/05/2021 - 06/06/2021

7.2. Resumen de ejecución del proyecto

Durante la realización del proyecto, se han realizado 17 sprints de una semana cada uno. En ellos, se han realizado en total 17 historias de usuario, cada una compuesta de sus tareas correspondientes. En este apartado, vamos a contrastar los números que reflejan las tablas del apartado anterior.

En primer lugar, se puede ver a lo largo de estos 17 sprints se han realizado 194 tareas. En total, estas tareas han supuesto un total de 632 horas efectivas, frente a 729 horas estimadas. La diferencia de casi 100 horas reside en que, al realizar los sprints, hay tareas que no se han podido realizar hasta el siguiente y, de esta manera, las tareas ligadas a esas horas no se realizan.

En cuanto al número de tareas según su estado, se han contabilizado:

- Tareas completadas: 165 tareas en total.
- Tareas en progreso: 12 tareas.
- Tareas en pausa: 4 tareas (todas se pueden encontrar en el sprint 4, Tabla 7.4).
- Tareas no iniciadas: 27 tareas que se han trasladado al sprint siguiente.
- Tareas no realizadas: 2 tareas finalmente no se aplicaron en el código de la aplicación (se encuentran en el último sprint, Tabla 7.17)

Para poder hacer una estimación visual sobre qué porcentaje de tareas a lo largo de todos los sprints quedaron en cada estado, podemos fijarnos en la Figura 7.1.

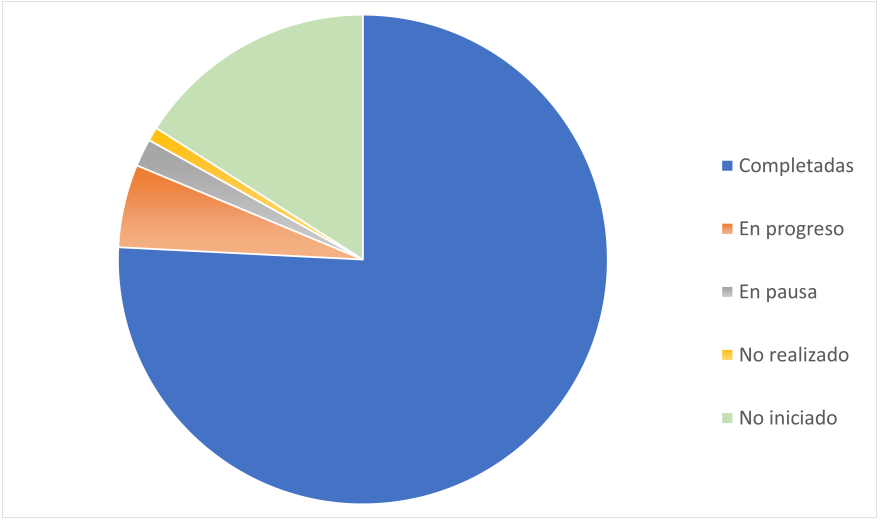


Figura 7.1: Tareas según su estado

También, creo que es interesante comentar cuántas tareas han sido necesarias para cada Historia de Usuario. En la Tabla 7.18, podemos ver todas las H.U. y, a su lado, el número de tareas que se han realizado asociadas a esa historia. Como se puede ver, la H.U. 5: Tablas ARP es la que más tareas ha necesitado. Esto es lógico pues, además de ser una de las tareas con más Puntos de Historia, sin duda ha sido la que más tiempo ha llevado, pues ha necesitado muchas horas para depurar el algoritmo y filtrar los resultados. A su vez, la que menos tareas han necesitado son las H.U. 12: Eliminar configlet, 14: Editar dispositivo, y 16: Mostrar dispositivo. La H.U. 12 es simplemente una llamada a la API enviándole en un formato específico qué configlet se quiere eliminar. Como en muchas tareas anteriores ya se había trabajado con la API, ya se tenía el conocimiento necesario para realizar estas tareas de forma eficiente. En el segundo caso, esta H.U. se realizó después de otras que abrieron camino a la hora de interactuar con la base de datos. De esta manera, el código y el conocimiento sobre cómo enviar y recibir peticiones ya estaba hecho. Debido a esto, el código ya estaba escrito, sólo había que reutilizarlo para la nueva H.U., por lo que el número de tareas es más corto. Por otro lado, en cuanto a la H.U. 16, es una historia de usuario muy concreta, por lo que el hecho de que tenga pocas tareas es una consecuencia lógica de sus características. En esta tabla, se debe mencionar también el caso de la H.U. 13: Logout. En este caso, aunque aparezcan 12 tarea asociadas, en realidad sólo tiene 2, pues la mayoría de ellas tienen como estado "no inciado", lo que significa que se pasaron a al siguiente sprint.

Una vez hemos hablado del número de tareas, pasemos a fijarnos en las horas empleadas y estimadas en cada Historia de Usuario. En la siguiente tabla (Tabla 7.19) podemos ver estos datos. Al igual que en la tabla anterior, podemos ver que las historias con un mayor número de tareas, también son las que ocupan un mayor número de horas. Esto también se aplica a las historias que menos horas ocupan, que se corresponden en general con las historias que menos tareas tienen asociadas.

H.U.	Nº de tareas
1	22
2	8
3	12
4	17
5	33
6	12
7	13
8	8
9	7
10	4
11	5
12	3
13	12
14	3
15	6
16	3
17	4

Tabla 7.18: Tareas por cada Historia de Usuario

H.U.	Horas estimadas	Horas empleadas
1	45	48
2	20,5	14
3	21,5	19
4	66,5	69,5
5	149,5	133,5
6	50	21
7	67	56
8	23,5	20
9	6	6
10	8	4
11	17	18
12	10	9
13	20	4,5
14	9	7
15	20	19
16	8	7
17	8	11

Tabla 7.19: Horas por cada Historia de Usuario

Capítulo 8

Conclusiones

8.1. Conclusiones

A lo largo de este proyecto hemos visto cómo aplicar soluciones software a un dominio de redes. Al realizar el proyecto he podido comprobar cómo se desarrollan las actividades de un Ingeniero de Software en un entorno real, con problemas y necesidades reales a través de las herramientas disponibles gracias a los conocimientos adquiridos. Además, creo que ha sido gratificante poder hacerlo de principio a fin, tomando decisiones a lo largo de todo el desarrollo del producto.

Personalmente, ha supuesto un reto para mí en cuanto a adquirir muchos conocimientos de equipos de redes que no tenía, sobre seguridad y, en especial, haber construido una aplicación desde el diseño hasta el producto final por primera vez.

Si nos centramos en los objetivos expuestos en el Capítulo 1.5.2, personalmente creo que he cumplido con el objetivo de aprender a utilizar entornos no conocidos anteriormente como Flask y Vue.js. En mis estudios como Ingeniero de Software nunca me había encontrado en esta tesitura de tener que montar un proyecto completamente. Ahora, visto con perspectiva, creo que ha sido un reto bastante motivador y, verlo acabado y funcional, gratificante. Creo que Vue.js es un framework muy potente y que, con bastante seguridad, utilizaré en el futuro para proyectos personales.

También he logrado comprender cómo se gestionan los elementos web, problemas e inconvenientes que pueden surgir a la hora de cambiar la disposición de éstos y qué decisiones tomar a la hora de hacer más amigable la aplicación de cara al usuario.

Aunque el conjunto de experiencia adquirida al realizar este proyecto ha sido claramente positiva, también he aprendido lo que son los errores difíciles de responder. En especial, los errores CORS mencionados en el Sprint 7.1. Esto me ha enseñado que siempre, por muy fácil que parezca una tarea, si implica comunicación entre entornos, dispositivos o equipos, puede complicarse. A su vez, estos contratiempos también me han enseñado a prever mejor

el tiempo de las tareas, y desenvolverme mejor a la hora de mirar documentación y buscar cómo solucionar un error, ya sea de este tipo o de otro.

Hablando de dificultades, también he notado que trabajar sobre un entorno tan cambiante como son los datos de un equipo de redes, pone de manifiesto aún más la importancia de la planificación, el análisis y, sobre todo, el diseño. En mi caso, lo he podido ver a la hora de desarrollar el algoritmo de las tablas ARP. Al principio pensé que utilicé demasiado tiempo a la hora de planificarlo pero, a medida que iban saliendo complicaciones y nuevos casos de uso que provocaban tener que ampliar las restricciones, vi que un correcto diseño inicial facilitó enormemente la modificación de su comportamiento. Lo mismo sucedió para la base de datos y el diseño basado en componentes: se hicieron numerosos cambios al principio al descubrir maneras más eficientes de implementarlo pero, en cuanto se fue refinando el diseño final, cualquier cambio que haya tenido lugar, se ha implementado con gran facilidad y rapidez gracias a un diseño que facilita estos cambios.

Creo que desarrollar este proyecto me ha hecho aprender también conceptos sobre seguridad. En este caso, me ha servido para ver cómo se puede implementar el protocolo de autenticación, dónde hacer las peticiones, configurar la plataforma que gestiona estas peticiones y, por último, comunicar este sitio con la aplicación. Al igual que algunos aspectos comentados anteriormente, nunca había tenido la oportunidad de realizar estas tareas por lo que, a mi modo de ver, conseguir ejecutarlas es otro reto que he conseguido en este proyecto.

Una vez que hemos hablado en rasgos generales de los objetivos cumplidos y los retos a los que me he enfrentado, quiero enumerar una serie de conclusiones personales que he ido elaborando a lo largo del tiempo empleado en crear este proyecto.

- La planificación de las tareas es sumamente complicada de hacer al principio, en especial si no conoces la tecnología con la que se va a trabajar o el dominio sobre el que se tiene que desarrollar. Debido a esto, las tareas de los primeros Sprints fueron algo caóticas y menos ajustadas en cuanto a tiempo, mientras que las últimas se fueron adaptando más a la estimación puesta. La planificación es muy importante a la hora de gestionar qué se debe implementar y en qué momento por lo que es crítico hacer una correcta planificación para un mejor desarrollo posterior, principalmente en cuanto a tiempo.

- Aprender a utilizar nuevas tecnologías y lenguajes amplía enormemente tus horizontes. En mi caso, comprender el funcionamiento de los dispositivos de redes, mirar documentación, manuales y comprender cómo comunicarme con los dispositivos me ha aportado una valiosa experiencia y mayor soltura a la hora de cambiar de un dominio conocido a otro bastante desconocido por mi parte.

- Un correcto modelado y diseño, ya sea conceptual del dominio, de la base de datos o de cualquier método o algoritmo es fundamental si se piensa trabajar sobre él y representa una parte importante del proyecto. Gracias a utilizar gran parte del tiempo en modelar el dominio y diseñar la base de datos, con sus posteriores cambios, las tareas fueron cada vez más sencillas, pues modificar, añadir o eliminar parte de esto fue sencillo gracias a un buen diseño, a lo que hay que añadir la importancia de un buen diseño de los componentes.

- Por último, destacar que toda tarea que pueda realizarse de una manera más cómoda, más eficiente o de manera más rápida conlleva grandes cambios para un trabajador. Gracias a la utilidad aportada en mi aplicación, he visto cómo algunas de las tareas de mi equipo de trabajo (y por tanto, algunas tareas asignadas a mí) se han resuelto de manera sencilla al tener una herramienta que permitiese acceder a cierta información de manera más cómoda a través de una interfaz, frente a los métodos anteriores que requerían más tiempo y entrañaban un mayor riesgo.

- Comunicar una aplicación front-end con una API back-end es algo bastante más complejo de lo imaginado. Debes establecer la manera en la que vas a enviar, tratar y devolver peticiones, datos incluidos en ellas, ver qué protocolo vas a utilizar, mensajes HTTP que vas a devolver y qué vas a hacer con ellos en la parte del front-end, etc. Para mí ha sido una de las mayores complejidades del proyecto, sin ninguna duda. Junto con la comunicación con APIs y resolver problemas de comunicación y seguridad, ha sido uno de los mayores retos que me han surgido a la hora de completar esta aplicación, lo cual también me ha aportado un bagaje a tener en cuenta a la hora de resolver problemas similares en el futuro.

- El proceso ágil (en especial, el marco de trabajo Scrum) me han proporcionado una ayuda incalculable a la hora de adoptar buenas costumbres y rutinas a la hora de documentar los avances, planificar el sprint siguiente, hacer una comparación de horas entre las H.U. y, sobre todo, llevar un control semanal que me ha permitido saber en todo momento en qué punto del desarrollo del proyecto me encontraba. Tener una perspectiva global cuando estás muy centrado en tareas concretas ayuda a no perder la referencia, y adoptar este marco de trabajo para realizar este proyecto ha sido muy útil en este sentido.

En resumen, este proyecto ha sido la conclusión de los conocimientos adquiridos durante la carrera de Ingeniería Informática pero, además, no sólo ha sido posible debido a estos conocimientos sino a las aptitudes desarrolladas a lo largo de mis estudios para poder desenvolverme en entornos tecnológicos desconocidos, resolver problemas, buscar soluciones y aportar valor en el software desarrollado. En el plano personal ha sido un gran reto debido al cambio de dominio sobre el que estoy acostumbrado a trabajar y, saber que he podido realizar de manera completa y funcional un proyecto que aporta soluciones sobre este dominio, es algo que ha sido, sin duda, gratificante.

8.2. Líneas de trabajo futuras

Este proyecto, al haber sido desarrollado bajo un marco empresarial, su uso está sujeto a cambios y, en especial, a posibles mejoras. El dominio de las redes es cambiante y tiene gran posibilidad de explotación, por tanto, siempre surgen posibilidades de mejoras, de añadir funcionalidades y de optimizar el código creado.

El primer aspecto en el que se podría trabajar en el futuro es el ejecutar scripts de manera periódica mediante Crontab (en Linux) o alguna herramienta similar en Windows,

según donde se vaya a alojar definitivamente la aplicación. Crontab es una herramienta que permite ejecutar scripts de manera periódica según lo que configure el usuario [16]. De esta manera, se podría fijar un tiempo (por ejemplo, una hora) para que, pasado ese tiempo, se vuelva a ejecutar de nuevo el script para poblar todas las tablas con los datos que existan en los dispositivos en ese momento. De esta manera, se tendrán las tablas lo más actualizadas posibles y tareas como, por ejemplo, la búsqueda por las tablas ARP, serán lo más precisas posible. Esta tarea se puede realizar debido a que se ha dotado al back-end con un archivo a modo de interfaz para poder realizar acciones directas sobre la base de datos. En esta interfaz se aplica el patrón fachada para minimizar dependencias. Así pues, al ejecutar este archivo, según el parámetro que se pase se realizará una acción u otra sobre la base de datos. Este archivo también ha sido útil a la hora de modificar la base de datos y realizar cambios pero, sobre todo, se ha diseñado con la idea de utilizarlo en un futuro para realizar acciones automáticas programadas.

En segundo lugar, una tarea que se puede realizar sería la modificación de interfaces físicas y virtuales de los dispositivos. Al igual que hacemos con otros datos (por ejemplo, con las tablas ARP o Ethernet-Switching), se podría mostrar al usuario de manera gráfica en un panel qué interfaces del dispositivo están habilitadas y cuáles no, junto con la opción de cambiar el estado (de UP a DOWN y viceversa). De esta manera, se facilitaría la visualización de interfaces de manera gráfica y en un solo panel sin meterse en la configuración del dispositivo y permitiría hacer el cambio de estado de una o varias interfaces de manera rápida y mucho más simple que a través de línea de comandos, añadiendo además un nivel de comprobación extra que aportaría seguridad adicional a la acción.

Por otro lado, otra de las líneas que se podrían trabajar en el futuro sería la creación de tareas más complejas que tengan utilidad para mi equipo de trabajo. Estas tareas no estarían sujetas a los configlets, por lo que se podrían realizar en cualquier tipo de dispositivo, aumentando así las posibilidades de la aplicación. Estas tareas, además, se podrían mostrar en la vista `Home`, de tal manera que se mostrarían las más utilizadas primero para facilitar su búsqueda.

Por último, quiero mencionar que también se pueden mejorar aspectos relativos al CSS. Esta aplicación está pensada para uso profesional, pero no se ajusta bien a tablets, el diseño de las tablas de la interfaz puede hacerse ser más responsive, etc. Este aspecto quizá es en uno de los que más he "sufrido" a la hora de realizar las vistas, pues es un aspecto del desarrollo web que no he trabajado mucho anteriormente. Es por esto por lo que, personalmente, creo que hay muchas posibles mejoras que se pueden hacer para que la interfaz fuera más amigable y, además, añadir mejoras como la posibilidad de un modo oscuro.

En resumen, las opciones que se muestran disponibles son enormes y aumentan a medida que surgen nuevos trabajos con los dispositivos de red. Identificar qué tareas pueden ser útiles y pueden facilitar el trabajo diario de los administradores de red es uno de los aspectos más importantes a tener en cuenta para implementar posibles mejoras

Apéndice A

Manuales

En este apéndice se van a tratar aspectos relativos al mantenimiento, despliegue e instalación y uso de la propia aplicación.

A.1. Manual de usuario

En esta sección vamos a ver cómo se utiliza la aplicación. Para ello, vamos a ilustrar con imágenes las diferentes vistas y vamos a realizar un breve manual para guiar al usuario a través de ellas, indicando qué se debe hacer y qué función tiene cada una de las vistas.

En primer lugar, suponiendo que el usuario no está autenticado y que es la primera vez que entra en la aplicación, la primera vista que aparecerá será Login (ver Figura A.1). En esta vista podemos observar que hay dos campos en los que el usuario debe introducir un texto: usuario y contraseña. Una vez que éstos se hayan rellenado (si no es así, saldrá un mensaje de error), se debe pulsar sobre el botón *Iniciar sesión*. Si las credenciales introducidas han sido válidas, el usuario iniciará sesión e irá a la vista `Home` (Figura A.2). Si, por el contrario, el usuario, la contraseña o ambos son incorrectos, saldrá un mensaje de error y el usuario debe introducir de nuevo estas credenciales.

A.1.1. Vistas principales

Una vez nos hayamos autenticado, accedemos a la vista `Home`, como hemos comentado. En esta vista vamos a comentar algunos elementos que serán comunes a cualquier vista de la aplicación, a excepción de la vista Login ya comentada. En primer lugar, podemos ver una barra lateral que nos permiten ir a algunas vistas para realizar ciertas acciones que explicaremos más adelante. También observamos una cabecera con los siguientes elementos:

- **Logo de la aplicación**: es el logo principal de nuestra aplicación. Si hacemos click sobre él, iremos a la vista `Home` (sobre la que nos encontramos actualmente).
- **Sobre esta página**: si hacemos click sobre este botón, nos conduce a la vista `About` (Figura A.3).
- **Cerrar sesión**: al hacer click sobre este botón, se borran las credenciales almacenadas del usuario, lo que redirige a la vista `Login` y el usuario debe iniciar sesión de nuevo.
- **Bandera con el idioma**: al hacer click sobre este icono, se despliega una lista con tantas banderas como idiomas disponibles. Actualmente sólo están disponibles el castellano y el inglés. Esta acción está representada en la Figura A.4, marcada con un círculo rojo.

Figura A.1: Vista de Login

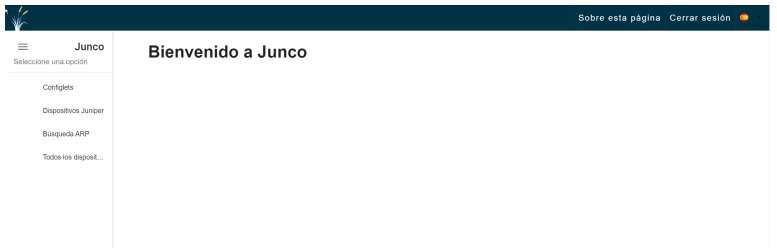


Figura A.2: Vista Home

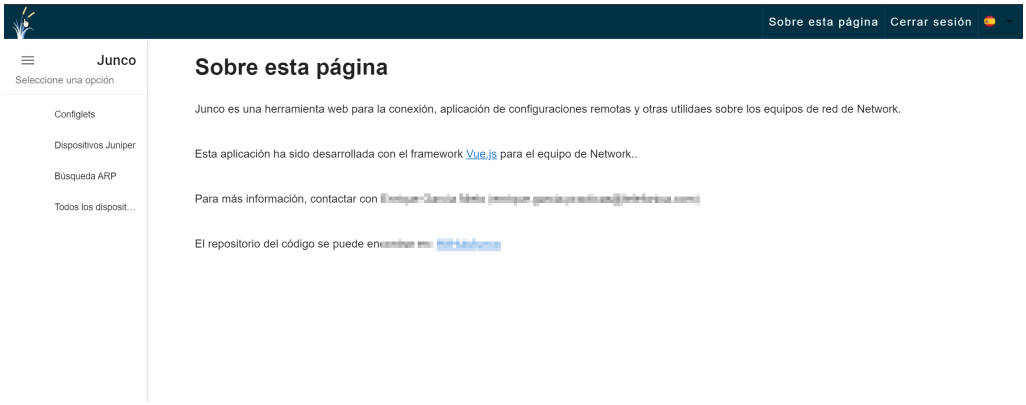


Figura A.3: Vista de About (Sobre esta página)

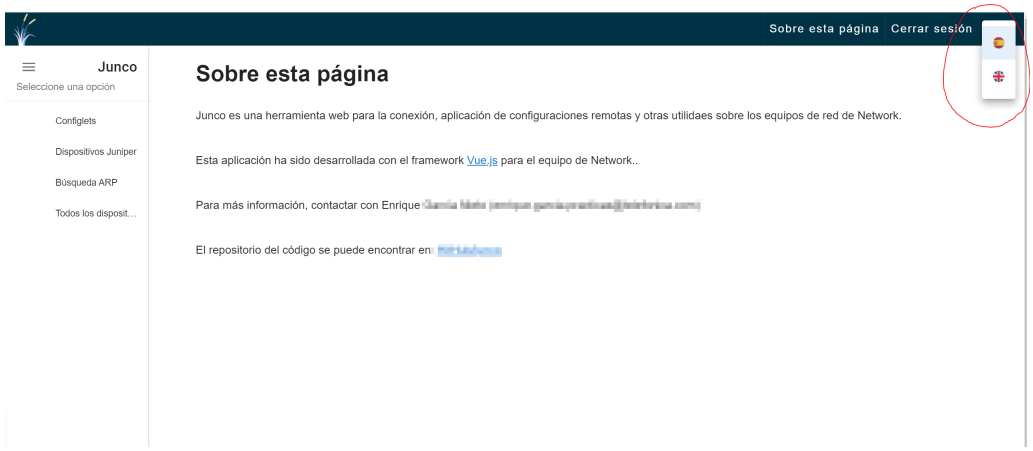


Figura A.4: Selección de idioma

Una vez que se ha hecho un breve recorrido por todas las vistas generales y que se pueden acceder haciendo click en los botones de la cabecera, vamos a hablar de las vistas a las que el usuario puede acceder haciendo click sobre las opciones de la barra lateral situada a la izquierda.

Estas vistas, como veremos en la siguiente sección, están relacionadas con acciones sobre los dispositivos y aplicación de configuraciones remotas. A su vez, pueden conducir a otras vistas o al despliegue de otros componentes, por lo que se explicará cada caso con detenimiento.

A.1.2. Vistas de acciones sobre los dispositivos

Supongamos que ahora, el usuario, quiere realizar acciones en los dispositivos. Por ejemplo, quiere ver la lista total de todos los dispositivos, sea cual sea el fabricante. Para ello, debe hacer click sobre la opción de *Todos los dispositivos*, en el menú lateral izquierdo. Una vez realizada esta acción, el usuario puede ver una tabla de todos los dispositivos disponibles. En esta vista, representada en la Figura A.5,

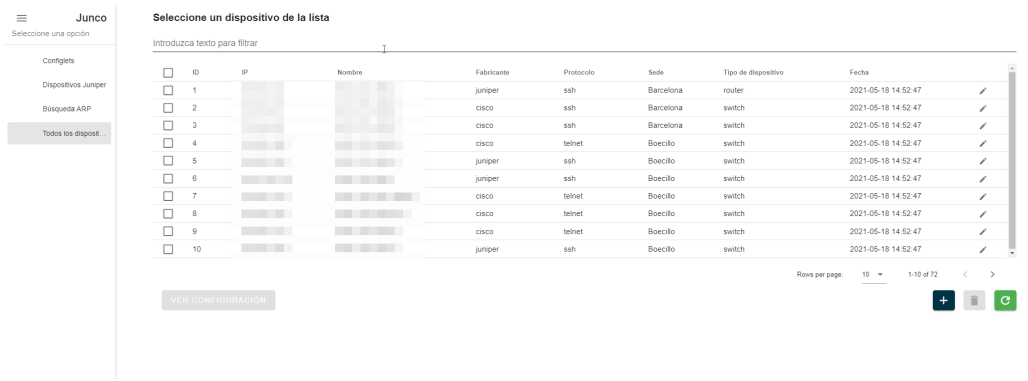


Figura A.5: Vista *All devices*, en la que se muestran todos los dispositivos

En esta vista, se pueden realizar las siguientes acciones:

- Buscar un dispositivo en la tabla. Para ello, simplemente tenemos que escribir un dato en la barra de búsqueda encima de la tabla y sólo se mostrarán los dispositivos que tengan algún parámetro cuyo valor coincida con el texto introducido.
- Ver la configuración de un solo dispositivo.
- Añadir un dispositivo (botón azul con un "+").
- Borrar uno o varios dispositivos (botón con una papelera como icono).
- Refrescar lista de dispositivos.
- Editar un dispositivo. Para ello, debemos hacer click en el lápiz a la derecha de la fila del dispositivo que queramos editar.

Para ver la configuración de un dispositivo, se tiene que cumplir la condición de haber seleccionado un solo dispositivo. Para eliminar uno o varios dispositivos, debe haber uno o varios dispositivos seleccionados. Como en la Figura A.5 no hay ningún dispositivo seleccionado, no se pueden realizar estas acciones, pues los botones aparecen deshabilitados. Sin embargo, en la A.6 vemos cómo hay un dispositivo seleccionado y, en consecuencia, los botones mencionados aparecen habilitados.

Si el usuario, por ejemplo, quisiera ver la configuración de uno de los dispositivos de la lista, debe hacer click en el botón *Mostrar configuración*. A continuación, tras un breve

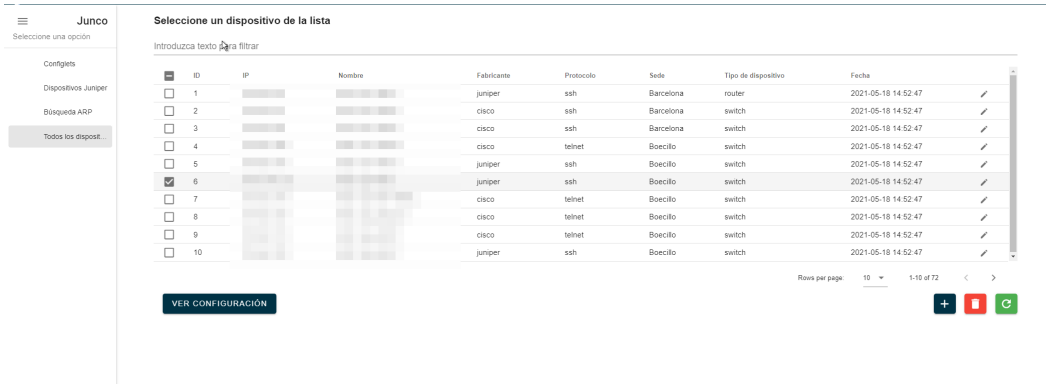


Figura A.6: Vista *All devices*, con un dispositivo seleccionado

tiempo de carga, aparecería un componente en la parte interior de la pantalla que mostraría una pestaña con una caja de texto donde aparece toda la configuración interna del dispositivo seleccionado. Si el usuario quiere cerrar esta pestaña, únicamente debe hacer click en el botón con una equis negra de la pestaña con el nombre del dispositivo. Si quiere ver la configuración de otro dispositivo, simplemente debe seleccionar otro en la tabla y repetir el patrón. Como esta acción es común en varias vistas, se explicará más adelante cómo mostrar una configuración de un dispositivo o configlet, pues en ambos casos se procede de la misma manera.

Supongamos ahora que el usuario quiere añadir un dispositivo a la lista. Para ello, hace click en el botón azul con un "+". Al realizar esta acción, sale un dialog pidiendo los datos del nuevo dispositivo a través de un formulario. El formulario tiene varios campos y los botones de *Cancelar* (se cierra el formulario y se vuelve a la vista anterior), *Limpiar* (vacía todos los campos del formulario en caso de que haya algo escrito en alguno de ellos) y Confirmar. Este Dialog aparece en la Figura A.7. Una vez que el usuario haya introducido los datos (y estos sean correctos pues, si no, aparecerá un mensaje de error en el propio dialog indicando qué campo es incorrecto), pulsa *Confirmar* y el dispositivo se incluirá en la base de datos (si no existe todavía en ésta).

Imaginemos que, por el contrario, lo único que quiere hacer es editar un dispositivo en vez de añadirlo. Quiere, por ejemplo, cambiar el nombre o la IP de uno de ellos. Lo único que debe hacer es click en el lápiz situado a la derecha de la fila del dispositivo y aparecerá el mismo Dialog que en el caso anterior pero, esta vez, los campos estarán rellenos con los valores actuales del dispositivo (Figura A.8. Una vez que el usuario quiera confirmar los cambios de uno o más parámetros, procede igual que en el caso anterior, haciendo click sobre *Confirmar*.

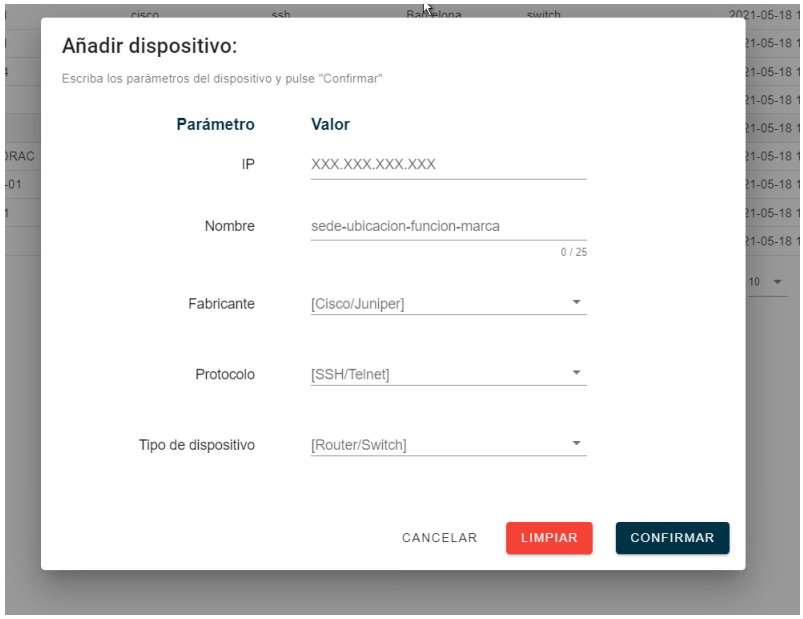


Figura A.7: Componente para añadir un dispositivo

Figura A.8: Componente para editar un dispositivo

Por último, imaginemos que el usuario quiere eliminar uno o varios dispositivos. Simplemente debe seleccionar los dispositivos deseados en la tabla y, cuando haya al menos uno seleccionado, se habilitará el botón rojo que tiene una papelera como imagen. Si pulsa sobre él, aparecerá el mismo componente pero mostrando una tabla con los dispositivos a eliminar,

a modo de mensaje de confirmación. Aquí, el usuario puede cancelar la acción o confirmarla, eliminándose así los dispositivos seleccionados. Esta acción se muestra en la Figura A.9.

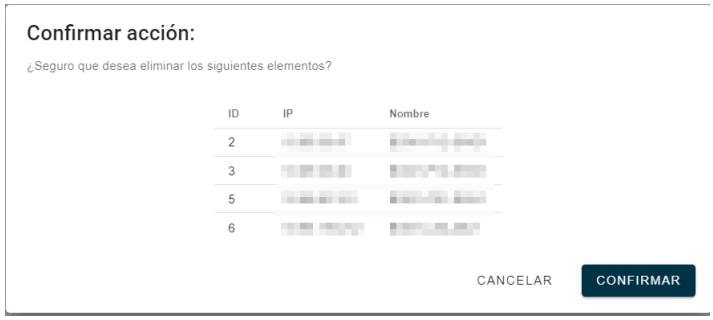


Figura A.9: Componente para eliminar dispositivos

Con esta última acción concluyen las acciones que puede realizar un usuario sobre los propios dispositivos.

A.1.3. Búsqueda en tablas ARP

Si el usuario quiere realizar búsquedas en las tablas ARP, debe hacer click en el botón de *Búsqueda ARP* de la barra lateral. Una vez realice esta acción, el usuario verá lo que se muestra en la Figura A.10. En esta vista, debe introducir una dirección MAC o IP en la barra de búsqueda, y pulsar el botón redondo azul con una lupa. Una vez realice esta acción, puede tener lugar uno de los siguientes resultados:

- **Resultado encontrado**: se muestra una tabla con los parámetros del resultado.
- **Resultado no encontrado pero el registro ARP existe**: se muestra un mensaje indicando que, efectivamente, el registro ARP existe pero no se encuentra en ningún dispositivo.
- **Resultado no encontrado**: si no se encuentra ningún registro ARP, se indica al usuario con un mensaje de error.
- **Formato incorrecto**: ocurre cuando el usuario introduce una cadena de caracteres inválida que no representa ninguna IP o MAC. Se muestra un mensaje indicando de la cadena incorrecta.

Si la búsqueda ha tenido éxito, la Figura A.11 representa cómo sería la tabla donde se muestra el resultado de la búsqueda. Si se quiere volver a realizar una búsqueda, se pulsa sobre el botón verde y la barra de búsqueda queda vacía, a la espera de introducir una nueva IP o una nueva MAC.

Figura A.10: Vista para realizar búsqueda ARP

Figura A.11: Vista que muestra un resultado de una búsqueda ARP

A.1.4. Configlets

Vamos ahora a hablar sobre las posibles acciones en lo referente a los configlets. Si un usuario quiere ver la lista de configlets disponibles y las acciones sobre ellos, debe hacer click sobre el botón *Configlets*, en el menú lateral. Una vez en la vista correspondiente, verá una lista con los configlets disponibles, además de los siguientes botones:

- **Ver configuración**: muestra la configuración del configlet.
- **Aplicar configlets**: permite aplicar la configuración de ese configlet a uno o varios dispositivos.
- **Borrar configlets**: botón con una papelera como icono. Permite borrar los configlets seleccionados.
- **Refrescar lista de configlets**: refresca la tabla de configlets.

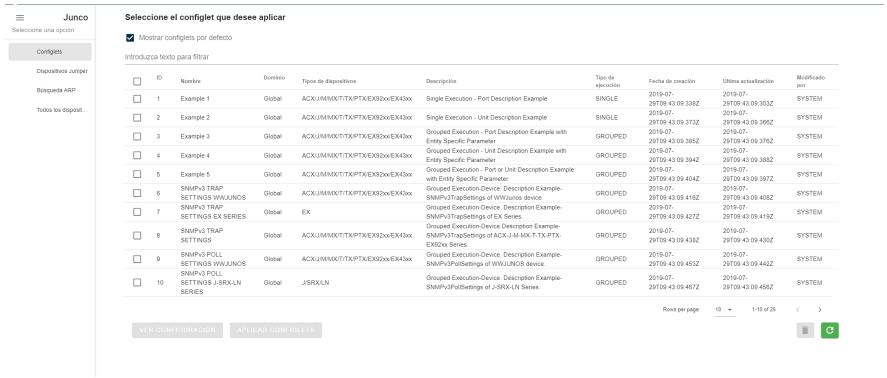


Figura A.12: Vista que muestra los configlets disponibles.

Los botones *Ver configuración* y *Aplicar configlets* sólo se mostrarán activos si hay un solo configlet seleccionado, mientras que el botón para eliminar configlets se mostrará activo si hay por lo menos uno o más configlets seleccionados.

Dicho esto, supongamos que el usuario quiere ver la configuración de un configlet. Al pulsar sobre *Ver configuración*, se mostrará bajo la tabla un componente que contiene el código que representa la configuración del configlet. De esta manera, el usuario puede hacerse una idea de lo que se podría aplicar en un dispositivo si desea aplicar ese configlet en cuestión. La Figura A.13 muestra este componente. Este mecanismo es igual que el explicado anteriormente, por lo que no nos detendremos más en explicar cómo funciona la acción de mostrar la configuración.

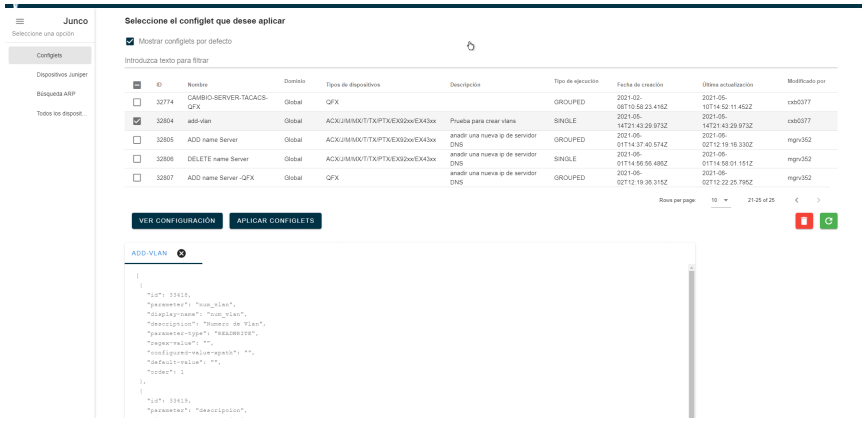


Figura A.13: Configuración de un configlet

Los botones de eliminar configlets y refrescar la lista de configlets tienen el mismo funcionamiento que en la vista de los dispositivos, comentada anteriormente. Tampoco nos detendremos en este punto.

Una vez comentadas estas acciones, nos queda por ver qué pasaría si el usuario quiere

aplicar uno de los configlets disponibles en la lista a los dispositivos que admitan dicha configuración. Una vez seleccionado el configlet en cuestión, el usuario debe pulsar el botón *Aplicar configlets*, lo que le llevará a la vista representada en la Figura A.14.

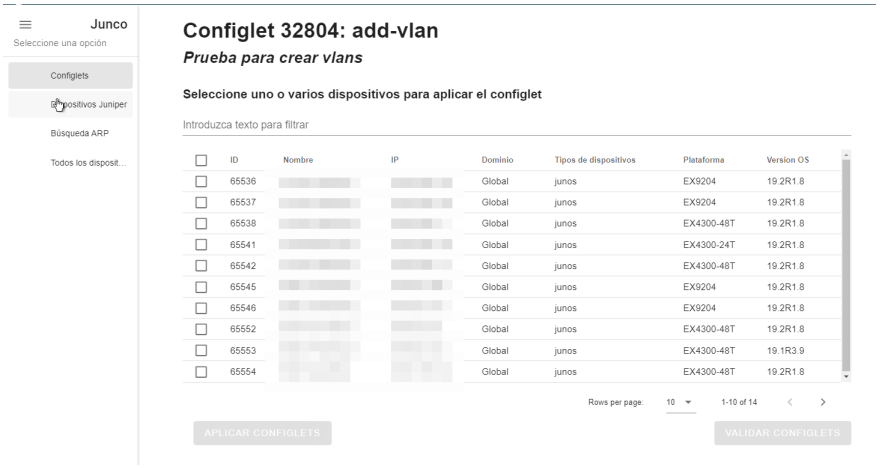


Figura A.14: Vista para aplicar un configlet a varios dispositivos

En esta vista aparecerá una tabla con los dispositivos que admiten la aplicación de ese configlet y, además, los botones *Aplicar configlets* y *Validar configlets*. El primero sólo se activará cuando se haya seleccionado, como mínimo, un dispositivo de la lista. Una vez que el usuario identifica en qué dispositivos quiere aplicar esa configuración, debe hacer click en dicho botón y, como muestra la Figura A.15, aparecerá un formulario con tantas pestañas como dispositivos seleccionados, cuyas pestañas representan, cada una, a uno de los dispositivos dentro de la selección. Las pestañas se pueden cerrar pulsando el botón circular con una equis. Si se desea realizar otra selección, se debe seleccionar las opciones deseadas en la tabla y volver a hacer click sobre *Aplicar Configlets*. De esta manera, en cada pestaña debe introducir los datos que quiera aplicar en la configuración de ese dispositivo en cuestión. Como vemos, al desplegarse este formulario, también se activa el botón de *Validar configlets*. Una vez que el usuario haya introducido todos los datos deseados en cada uno de los dispositivos, debe pulsar en el botón *Validar configlets* para que la plataforma Junos compruebe si la información introducida puede ser aplicada correctamente en cada dispositivo.

En este punto, se pueden dar dos casos:

1. Hay, al menos, una o más configuraciones incorrectas.
2. Las configuraciones seleccionadas para todos y cada uno de los dispositivos son correctas.

En el primer caso, se muestra un componente con varias páginas. Las páginas vienen redondeadas con un círculo azul, para ser más precisos, en la Figura A.16. Cada guión muestra un configlet que se ha validado y podemos hacer click encima de ellos para ir pasando de uno a otro. En la figura, se muestra cómo se representaría una validación errónea. En este caso,

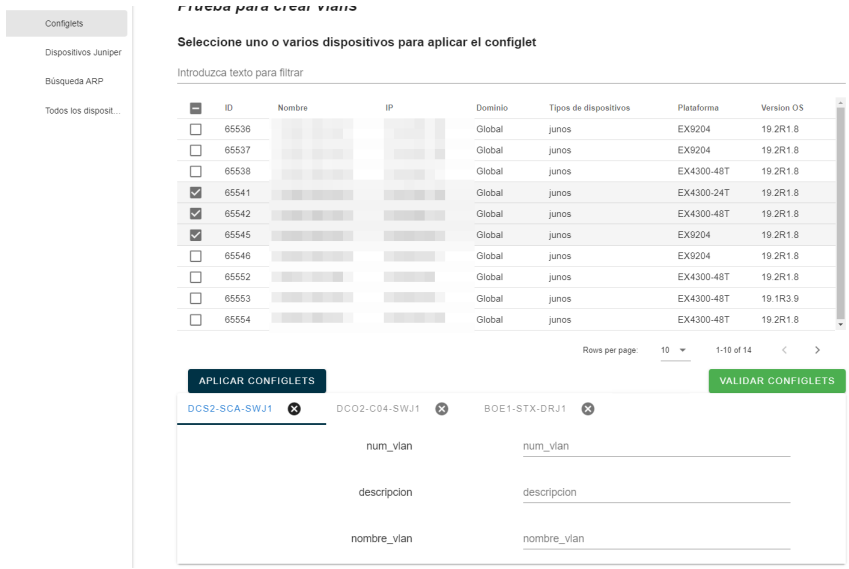


Figura A.15: Vista para introducir los parámetros en cada uno de los dispositivos seleccionados

los datos que ha introducido el usuario son incorrectos, por lo que se muestra en rojo un mensaje de error indicando cuál ha sido el fallo. Si pulsamos sobre otros configlets, podemos ver si ha habido errores o no. Como para aplicar los configlets su validación debe ser correcta, en este caso el botón *Confirmar* se muestra deshabilitado.

Si se da el caso en el que todos los configlets se han validado correctamente, el botón *Confirmar* se habilita y todos y cada uno de los resultados salen en verde como se ve en la Figura A.17, mostrando en este componente el código que se va a inyectar en cada dispositivo seleccionado. Al pulsar este botón, se muestra un mensaje al usuario indicando el resultado de la aplicación del configlet a todos los dispositivos.

De esta manera, queda explicada la sección sobre los configlets y las acciones que un usuario puede hacer sobre ellos. A continuación, vamos a ver la última sección, en la que se explicará qué puede hacer un usuario sobre los dispositivos disponibles en la plataforma Junos Space.

A.1.5. Dispositivos Juniper

Supongamos que el usuario quiere realizar la acción opuesta a la comentada anteriormente: si se ha explicado los pasos para aplicar un solo configlet en uno o más dispositivos, a continuación vamos a ver cómo aplicar uno o más configlets en un solo dispositivo. Al hacer click sobre *Configlets* en la barra lateral, iría a una nueva vista.

Como vemos en la Figura A.18, se muestran los dispositivos Juniper disponibles en la

Figura A.16: Validación incorrecta de, al menos, un configlet.

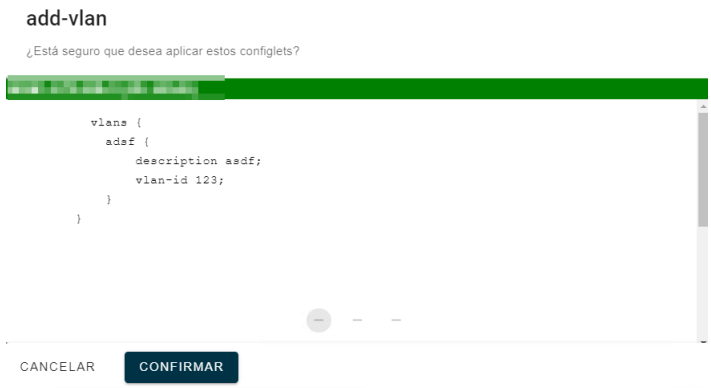


Figura A.17: Validación correcta de configlets.

plataforma Junos Space. Esta vista es muy similar a la descrita para todos los dispositivos, excepto por unas peculiaridades:

1. No se permiten realizar acciones (como añadir, editar, borrar) sobre estos dispositivos.
2. Los campos de las tablas son ligeramente diferentes.
3. Se muestra una columna representando el estado de ese equipo. El círculo verde muestra que el dispositivo está activado para operar sobre él, mientras que un círculo rojo muestra que el dispositivo no está operativo para realizar operaciones sobre él.

En esta vista, nos encontramos dos botones. El primero de ellos, *Ver configuración*, nos permite ver la configuración interna del dispositivo. Como lo hemos comentado tanto para las vistas de todos los dispositivos y configlets, vamos a continuar con el siguiente botón. *Aplicar configuración* es el botón que nos mostrará una lista para ver qué configlets están disponibles para aplicarse en un dispositivo concreto. Si, por ejemplo, un usuario quisiera

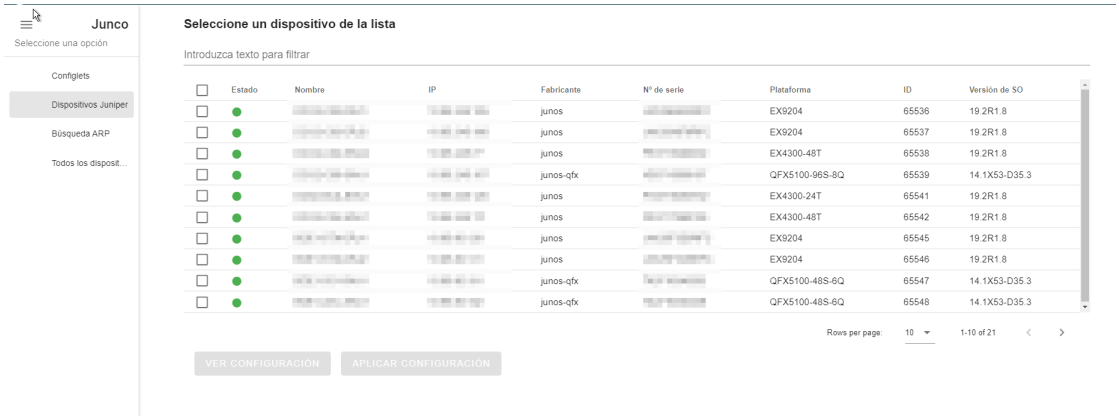
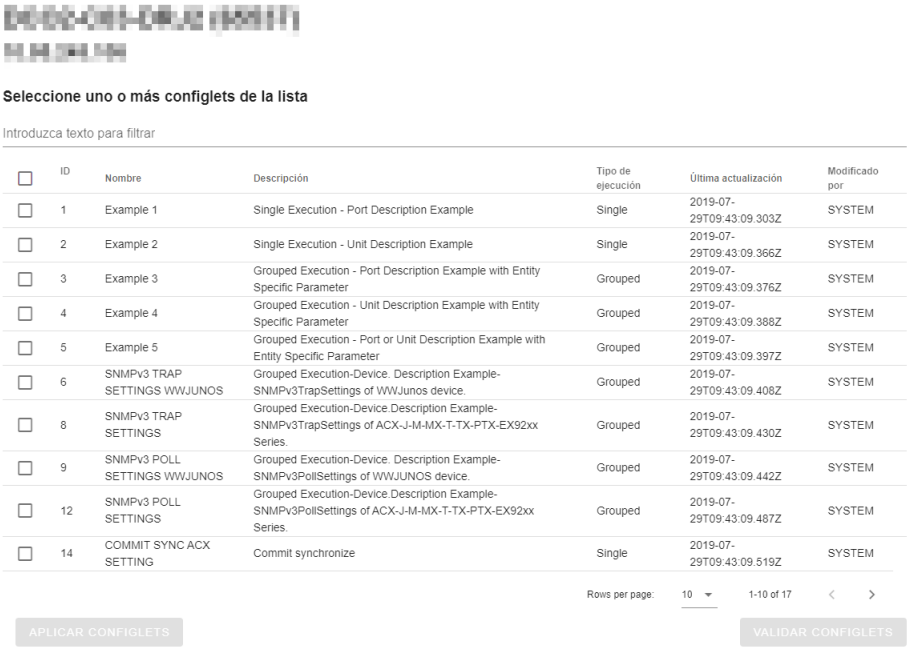


Figura A.18: Dispositivos Juniper

aplicar uno o más configlets en un dispositivo, tendría que venir a esta vista, seleccionar el dispositivo en cuestión (cuyo estado debe ser disponible, es decir, con el círculo de color verde). Cuando haya seleccionado uno y sólo uno de los dispositivos de la lista, se habilitarán los botones y, a continuación, debe pulsar el botón *Aplicar configuración*, lo que nos lleva a la vista mostrada en la Figura A.19.

La manera de proceder por parte del usuario, es casi idéntica al caso anterior. En este caso, en vez de mostrar una lista de dispositivos, se mostrará una lista de configlets susceptibles de ser aplicables al dispositivo seleccionado. Una vez que se seleccione uno o más configlets, si se pulsa *Aplicar configlets* se mostrará un formulario similar al visto anteriormente en la Figura A.15. Sin embargo, esta vez los campos del formulario cambiarán entre pestañas, pues antes las pestañas simbolizaban la aplicación del mismo configlet pero para distintos dispositivos, por lo que la configuración no cambiaba; sin embargo ahora, cada pestaña es un configlet, por lo que la configuración será distinta.

El resto del procedimiento es exactamente igual. Una vez que el usuario introduzca los datos deseados en cada pestaña, pulsará sobre *Validar configlets* y, según si la validación ha sido correcta o incorrecta, se mostrará un componente como el de la Figura A.17 o A.16, respectivamente. Una vez que los datos hayan sido validados correctamente, el usuario pulsará *Confirmar* y un mensaje mostrado por pantalla le informará si la aplicación de configlets ha tenido éxito.



Seleccione uno o más configlets de la lista

Introduzca texto para filtrar

	ID	Nombre	Descripción	Tipo de ejecución	Última actualización	Modificado por
☐	1	Example 1	Single Execution - Port Description Example	Single	2019-07-29T09:43:09.303Z	SYSTEM
☐	2	Example 2	Single Execution - Unit Description Example	Single	2019-07-29T09:43:09.366Z	SYSTEM
☐	3	Example 3	Grouped Execution - Port Description Example with Entity Specific Parameter	Grouped	2019-07-29T09:43:09.376Z	SYSTEM
☐	4	Example 4	Grouped Execution - Unit Description Example with Entity Specific Parameter	Grouped	2019-07-29T09:43:09.388Z	SYSTEM
☐	5	Example 5	Grouped Execution - Port or Unit Description Example with Entity Specific Parameter	Grouped	2019-07-29T09:43:09.397Z	SYSTEM
☐	6	SNMPv3 TRAP SETTINGS WWJUNOS	Grouped Execution-Device. Description Example-SNMPv3TrapSettings of WWJunos device.	Grouped	2019-07-29T09:43:09.408Z	SYSTEM
☐	8	SNMPv3 TRAP SETTINGS	Grouped Execution-Device.Description Example-SNMPv3TrapSettings of ACX-J-M-MX-T-TX-PTX-EX92xx Series.	Grouped	2019-07-29T09:43:09.430Z	SYSTEM
☐	9	SNMPv3 POLL SETTINGS WWJUNOS	Grouped Execution-Device. Description Example-SNMPv3PollSettings of WWJUNOS device.	Grouped	2019-07-29T09:43:09.442Z	SYSTEM
☐	12	SNMPv3 POLL SETTINGS	Grouped Execution-Device.Description Example-SNMPv3PollSettings of ACX-J-M-MX-T-TX-PTX-EX92xx Series.	Grouped	2019-07-29T09:43:09.487Z	SYSTEM
☐	14	COMMIT SYNC ACX SETTING	Commit synchronize	Single	2019-07-29T09:43:09.519Z	SYSTEM

Rows per page: 10 1-10 of 17

APLICAR CONFIGLETS VALIDAR CONFIGLETS

Figura A.19: Vista para aplicar uno o más configlets sobre un dispositivo concreto

Figura A.20

A.2. Manual de despliegue e instalación

Nuestro proyecto consta de dos partes diferenciadas: entorno front-end y servidor backend.

Para garantizar el despliegue del front-end de este proyecto, necesitamos las siguientes versiones de los siguientes paquetes:

- `npm`: versión 8 o superior.
- `Node.js`: versión 7.5.2 o superior.

Por otro lado, para desplegar el servidor back-end, necesitamos:

- `Python`: versión 2.7.0 o superior.
- `Python3`: versión 3.9.0 o superior.
- `Flask`: versión 1.1.2 o superior.

Se han utilizado dos versiones de Python distintas porque, debido a algunas librerías concretas, el entorno virtual ha necesitado al principio dos versiones. No obstante, la versión con la que se ejecuta tanto el servidor como los scripts es `Python3`. Una vez que tengamos disponibles e instalados estos elementos, procedemos a preparar el entorno para el posterior despliegue.

A.2.1. Preparación del entorno

Antes de realizar el despliegue, necesitamos generar artefactos que contengan los ficheros estáticos para el funcionamiento de la web de la aplicación.

Una vez descargado todo el proyecto (enlace disponible en el Anexo B), debemos abrir una terminal en la carpeta raíz del proyecto. Una vez estemos situados en esta raíz, si ejecutamos el comando `npm install`. Este comando se encargará de instalar todas las dependencias y paquetes necesarios para desplegar el proyecto.

Una vez hayamos ejecutado este comando y se hayan terminado de instalar todas las dependencias, el comando `npm run serve` se encarga de arrancar el proyecto front-end de nuestra aplicación. Esto nos permitirá abrir el proyecto en un navegador si pulsamos en uno de los enlaces mostrados por la terminal que se generarán una vez termine este comando de arrancar el servidor front-end.

Cuando ya tengamos instalado y en ejecución el servidor front-end para mostrar la web, necesitamos arrancar el servidor que se encargará de gestionar las peticiones web. Lo primero que debemos hacer, una vez tengamos Python instalado, es crear un entorno virtual (a partir de ahora, llamado `venv`, de *Virtual Environment*. Para instalar este entorno virtual, debemos ejecutar el comando, en Python, `pip install virtualenv`. Una

vez termine, debemos crear una carpeta que será nuestro entorno virtual, con el nombre que el usuario desee, y movernos a ese directorio creado. Una vez dentro, debemos ejecutar el siguiente comando `python3 -m venv nombre_del_venv_escogido`. Para activar este entorno virtual, debemos situarnos en la carpeta raíz de todo el proyecto y ejecutar `./nombre_del_venv_escogido/Scripts/activate` (en Windows). Esto ejecutará el script de activación, y estaremos dentro de nuestro entorno virtual.

El entorno virtual permite encapsular un entorno de trabajo, de tal manera que el sistema de directorios, dependencias y paquetes y librerías instalados, sólo afectarán a los proyectos dentro de dicho entorno virtual. Esto es útil si, por ejemplo, nuestro proyecto utiliza una versión concreta de Python u otra librería, pero no queremos cambiar la versión global de Python del sistema.

Con el entorno virtual ya instalado y con nuestro código desplegado, debemos ir a la carpeta `server` que cuelga del directorio raíz y ejecutar los siguientes scripts:

- `pip install -e .`: este script instala todo el árbol de dependencias de nuestro servidor, lo que permite realizar imports de clases, métodos, librerías, etc.
- `pip install -r requirements.txt`: al ejecutar este script, se instalan todas las librerías, paquetes y dependencias existentes en el fichero `requirements.txt`. Estas dependencias se instalan de manera recursiva.

Por último, para arrancar el servidor back-end, con el entorno virtual activado con el comando comentado anteriormente, debemos ejecutar el archivo `app.py` dentro de la carpeta `server` que, como hemos comentado, cuelga de la carpeta raíz del proyecto. Si nos situamos en la carpeta raíz, podemos ejecutar el siguiente comando `py ./server/app.py` para arrancar el servidor backend.

A.3. Manual de mantenimiento

Por último, tras hablar de cómo se debe utilizar la aplicación y cómo se debe instalar, en esta sección vamos a realizar unas breves anotaciones para un correcto mantenimiento de la aplicación.

En primer lugar, la tarea más crítica es la de los scripts para la base de datos. En éstos, se han contemplado solamente cuatro fabricantes: Dell, Cisco, Juniper y Arista. Si en el futuro se estima que se pueden incluir más dispositivos, se debe incluir una clase nueva por cada nuevo vendedor y, por cada tabla existente en la base de datos (ARPTable, EthernetSwitchingTable, InterfacesTable y NeighborsTable), se debe incluir un nuevo método que permita extraer la información de estas tablas de cada dispositivo según el fabricante. Una vez realizado esto, a la hora de poblar las tablas, también se debe incluir la llamada a los métodos de estos nuevos fabricantes. De esta manera, cada vez que se elija poblar las tablas, el script que se encarga de recorrer todos los dispositivos y obtener todas las tablas de cada uno de ellos, también tendrá en cuenta los nuevos dispositivos de los nuevos fabricantes y, en el momento de extraer su información, también tendrá disponibles los métodos para extraerlos.

Por otro lado, cabe mencionar que en el proyecto disponible en GitLab hay archivos que no se han subido al repositorio, pues contienen información confidencial. Estos archivos contienen, entre otras cosas, *secrets* para conectarse con los dispositivos, o las direcciones de algunas máquinas. Por ello, si alguien quiere mantener este código, debe aportar los siguientes datos, ya que en el proyecto no se van a encontrar:

- **properties.py**: este fichero utilizaod por el back-end debe contener la IP del servidor AD contra el que se realiza el Login, el *secret* de ese servidor y, además, la ruta del motor de la base de datos que se vaya a utilizar.
- **axiosinstance.js**: este fichero contiene, entre otras cosas, la llamada a la dirección URL de Junos Space. Además, es el fichero donde se encuentra la lógica para almacenar el `sessionStorage` que recoge las credenciales de autenticación del usuario.

A.3.1. Otros comandos útiles

Otros comandos útiles que nos pueden ayudar a realizar el mantenimiento de la aplicación son los siguientes:

- `npm run test:unit`: ejecuta los test unitarios. En nuestro proyecto, sólo hemos desarrollado test de las H.U. iniciales pero, si se desarrollaran test en el futuro, este es el script para ejecutarlos.
- `npm run test:e2e`: ejecuta los test end-to-end.
- `pip freeze >requirements.txt`: este comando genera un nuevo archivo `requirements.txt` con todas las dependencias necesarias para arrancar el servidor. Sirve para actualizar el fichero de dependencias pues, si se han añadido dependencias y no se ha ejecutado este comando, no se muestran en el fichero mencionado.

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código: `https://gitlab.inf.uva.es/enrigar/tfg_enrique.git`.

Bibliografía

[1] *2019 Database Trends*. ScaleGrid, mar. de 2019. URL: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/ (visitado 23-05-2021).

[2] *A Short History of Git*. URL: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git (visitado 22-05-2021).

[3] *AAA - AUTENTICACIÓN, AUTORIZACIÓN Y REGISTRO*. URL: https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-glosario_abreviaturas/index.html?n=5.html (visitado 27-05-2021).

[4] Universidad de Aberdeen. *Address Resolution Protocol (ARP)*. Jun. de 1997. URL: https://erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html (visitado 29-05-2021).

[5] *About Python*. URL: https://www.python.org/about/ (visitado 23-05-2021).

[6] *Acerca del Control de Versiones*. URL: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones (visitado 22-05-2021).

[7] *Astah UML*. URL: https://astah.net/products/astah-uml/ (visitado 22-05-2021).

[8] Carlos Ávila. *TACACS+ y Latch: "Buena combinación para administrar tus dispositivos de Red*. Jun. de 2018. URL: https://empresas.blogthinkbig.com/tacacs-latch-combinacion-ciberseguridad/ (visitado 27-05-2021).

[9] *Balsamiq Wireframes*. URL: https://balsamiq.com/wireframes/ (visitado 22-05-2021).

[10] Bravent. *¿Qué es MVVM – Model View ViewModel?* URL: https://www.bravent.net/xamarin-forms-y-mvvm/ (visitado 10-06-2021).

[11] Cabsa. *Cuáles son los 3 roles de Scrum y sus características*. Nov. de 2020. URL: https://www.cabsa.es/blog/3-roles-de-scrum-y-sus-caractersticas (visitado 18-05-2021).

[12] Marina Camacho. *El coste de un trabajador para la empresa*. Abr. de 2021. URL: https://factorialhr.es/blog/coste-empresa-trabajador/ (visitado 11-06-2021).

[13] *Client-Side Storage*. URL: https://vuejs.org/v2/cookbook/client-side-storage.html (visitado 30-05-2021).

[14] *CORS policy*. URL: https://es.wikipedia.org/wiki/Intercambio_de_recursos_de_origen_cruzado (visitado 03-03-2021).

[15] *Creating a CLI configlet.* URL: https://www.juniper.net/documentation/en_US/junos-space18.2/platform/topics/task/configuration/cli-configet-creating.html (visitado 29-05-2021).

[16] *Cron crontab, explicados.* URL: https://blog.desdelinux.net/cron-crontab-explicados/ (visitado 10-06-2021).

[17] *Cypress.* URL: https://www.cypress.io/ (visitado 27-05-2021).

[18] *Developer Survey Results.* Stack Overflow, 2016. URL: https://insights.stackoverflow.com/survey/2016 (visitado 26-04-2021).

[19] José Esterkin. *¿Qué es el riesgo en un proyecto?* Abr. de 2007. URL: https://iaap.wordpress.com/2007/04/29/%C2%BFque-es-el-riesgo-en-un-proyecto/ (visitado 10-06-2021).

[20] *Facade.* URL: https://refactoring.guru/es/design-patterns/facade (visitado 20-06-2021).

[21] *Flask User's Guide.* URL: https://flask.palletsprojects.com/en/2.0.x/ (visitado 27-05-2021).

[22] Brad Frost. *Atomic Design.* Oct. de 2013. URL: https://bradfrost.com/blog/post/atomic-web-design/ (visitado 10-06-2021).

[23] *Git Branching - Branches in a Nutshell.* URL: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell (visitado 22-05-2021).

[24] Miguel A. Gómez. *TDD (Test Driven Development). Desarrollo dirigido por pruebas.* 2011. URL: https://softwarecrafters.io/javascript/tdd-test-driven-development (visitado 08-06-2021).

[25] *Guía de Estilo.* URL: https://es.vuejs.org/v2/style-guide/ (visitado 27-05-2021).

[26] Helpsystems. *Automation Advantages: 5 Benefits of Automation.* URL: https://www.helpsystems.com/resources/guides/automated-operations-5-benefits-your-organization (visitado 16-05-2021).

[27] Jose Ignacio Herranz. *TDD como metodología de diseño de software.* 2011. URL: https://www.paradigmadigital.com/dev/tdd-como-metodologia-de-diseno-de-software/ (visitado 08-06-2021).

[28] *HTML: Lenguaje de etiquetas de hipertexto.* URL: https://developer.mozilla.org/es/docs/Web/HTML (visitado 23-05-2021).

[29] *i18n.* URL: https://www.npmjs.com/package/i18n (visitado 27-05-2021).

[30] IBM. *Interfaces de red TCP/IP.* URL: https://www.ibm.com/docs/es/aix/7.2?topic=protocol-tcpip-network-interfaces (visitado 29-05-2021).

[31] *Introduction to Model/View/ViewModel pattern for building WPF apps.* Oct. de 2005. URL: https://docs.microsoft.com/es-es/archive/blogs/johngossman/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps (visitado 10-06-2021).

[32] *Introduction to Vuetify.* URL: https://vuetifyjs.com/en/introduction/why-vuetify/ (visitado 31-05-2021).

[33] Martin James. *Managing the Data Base Environment.* Prentice Hall PTR, 1983. ISBN: 9780135505823.

[34] *Javascript*. URL: https://developer.mozilla.org/es/docs/Web/JavaScript (visitado 23-05-2021).

[35] *Jira Software*. URL: https://www.atlassian.com/es/software/jira (visitado 22-05-2021).

[36] *Junos PyEZ Developer Guide*. URL: https://www.juniper.net/documentation/en_US/junos-pyez/information-products/pathway-pages/junos-pyez-developer-guide.html (visitado 27-05-2021).

[37] *Junos Space Network Management*. URL: https://www.juniper.net/us/en/products-services/network-management/junos-space-platform/ (visitado 17-04-2021).

[38] *Junos Space Platform API Reference*. URL: https://www.juniper.net/documentation/en_US/junos-space-sdk/16.1/apiref/index.html (visitado 27-05-2021).

[39] *La instancia Vue*. URL: https://es.vuejs.org/v2/guide/instance.html (visitado 27-05-2021).

[40] Zoraida Ceballos de Mariño. *Hirotaka Takeuchi e Ikujiro Nonaka: los creadores del SCRUM*. Sep. de 2019. URL: https://xn--zoraidaceballosdemario-4ec.info/scrum/hirotaka-takeuchi-e-ikujiro-nonaka-los-creadores-del-scrum/ (visitado 18-05-2021).

[41] *Material Design Icons*. URL: https://materialdesignicons.com/ (visitado 27-05-2021).

[42] Thomas Menze. *The State of Industrial Cybersecurity 2019*. Kaspersky, 2019. URL: https://ics.kaspersky.com/media/2019_Kaspersky_ARC_ICS_report.pdf (visitado 26-04-2021).

[43] Gobierno de España Ministerio de Empleo y Seguridad Social. *XVII Convenio colectivo estatal de empresas de consultoría, y estudios de mercado y de la opinión pública*. Mar. de 2018. URL: https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf (visitado 11-06-2021).

[44] Bradley Mitchell. *MAC Addresses With Formatting Examples*. LifeWire, mayo de 2021. URL: https://www.lifewire.com/introduction-to-mac-addresses-817937 (visitado 26-04-2021).

[45] Yanina Muradas. *Qué es NPM y para qué sirve*. Sep. de 2019. URL: https://openwebinars.net/blog/que-es-node-package-manager/ (visitado 27-05-2021).

[46] *MySQL*. URL: https://www.artima.com/articles/the-making-of-python (visitado 23-05-2021).

[47] *Netmiko*. URL: https://pypi.org/project/netmiko/ (visitado 27-05-2021).

[48] *Pandas User guide*. URL: https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html (visitado 27-05-2021).

[49] Juan Peláez. *Arquitectura basada en componentes*. URL: https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/ (visitado 10-06-2021).

[50] *Principales Usos de Python*. Stack Overflow, oct. de 2018. URL: https://www.akademus.es/blog/programacion/principales-usos-python/ (visitado 23-05-2021).

[51] *Qué es Javascript*. URL: https://developer.mozilla.org/es/docs/Web/JavaScript (visitado 23-05-2021).

[52] *Qué es Sass?* URL: https://openwebinars.net/blog/que-es-sass-ventajas-desventajas-y-ejemplos-de-desarrollo/ (visitado 23-05-2021).

[53] *Qué es Vue.js?* URL: https://es.vuejs.org/v2/guide/#Renderizacion-Declarativa (visitado 23-05-2021).

[54] Ivonne Chaves Ríos. *¿Modelo – Vista – Vista Modelo? ¿Eso es un patrón?* URL: https://shirivo.wordpress.com/2015/06/30/modelo-vista-vista-modelo-eso-es-un-patron/ (visitado 10-06-2021).

[55] *Sobre el Control de Versiones - Fundamentos de Git.* URL: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git (visitado 22-05-2021).

[56] *SQLAlchemy.* URL: https://www.sqlalchemy.org/ (visitado 27-05-2021).

[57] Ken Schwaber Jeff Sutherland. *The Scrum Guide.* Nov. de 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (visitado 18-05-2021).

[58] *tacacs_plus module.* URL: https://pypi.org/project/tacacs_plus/ (visitado 27-05-2021).

[59] *The Making of Python.* URL: https://www.artima.com/articles/the-making-of-python (visitado 23-05-2021).

[60] Universidad de Valladolid. *Proyecto docente del trabajo de fin de grado 2019-2020 (Mención Ingeniería de Software).* https://alojamientos.uva.es/guia_docente/uploads/2019/545/46976/1/Documento.pdf.

[61] *VanDyke SecureCRT.* URL: https://www.vandyke.com/products/securecrt/ (visitado 17-04-2021).

[62] Adrián Alonso Vega. *Atomic Web Design o Diseño Guiado por Componentes.* URL: https://adrianalonso.es/arquitectura-del-software/atomic-web-design-o-diseno-guiado-por-componentes/ (visitado 10-06-2021).

[63] *Version Control Systems Popularity in 2016.* RhodeCode, 2016. URL: https://rhodecode.com/insights/version-control-systems-2016 (visitado 26-04-2021).

[64] *Vue Router.* URL: https://router.vuejs.org/ (visitado 27-05-2021).