Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

GNS3 como herramienta para el diseño y desarrollo de prácticas de laboratorio de redes

---

Alumno: Gustavo González Ramírez

Tutor: Jesús María Vegas Hernández

A mi padre y a mi madre, gracias por todo.

Agradecimientos

A mis padres y hermanos, por ser siempre el pilar más importante de mi vida, desde el principio hasta el final.

A mis amigos de la TFS, porque, a casi cumplir 10 años de habernos conocido, mantenemos la amistad como el primer día o incluso mejor, por nunca juzgarme, por el apoyo que nos brindamos los unos a otros incluso en los peores momentos de nuestras vidas, por absolutamente todos los momentos vividos y porque gracias a vosotros sé que jamás estaré solo, ya que sé que estaréis ahí siempre, aunque todo salga mal. Soy completamente feliz de teneros a mi lado, os agradezco absolutamente todo.

A mis amigos de Pystacho, por el gran compañerismo que tenemos, por haberme permitido ser yo mismo con vosotros desde el primer momento sabiendo lo mucho que me cuesta y porque, aunque nos hayamos juntado en los últimos momentos de la etapa universitaria, habéis logrado ser importantísimos para mí, sacándome una sonrisa cada vez que nos juntamos, espero no perderos nunca.

A mis amigos de Vivoreo, por todas las experiencias vividas juntos durante todos estos años que jamás olvidaré y por el apoyo incondicional que me habéis mostrado siempre, sé que os llevaré conmigo toda la vida.

A todos mis amigos de Maristas, porque a pesar de que la etapa en la que convivíamos a diario se acabó, habéis estado a mi lado como si nada hubiese cambiado, en los momentos más duros y en los mejores.

A Marco y Diego Corral, por todos los cafés, charlas, bromas y apoyos vividos y por hacer que pase de vivir con temor mi etapa universitaria a hacerla una de las mejores de mi vida. También a Diego Fraile y Nico, por todos los trabajos en grupo, las cervezas después de clase y las tardes de sufrimiento y risas en Discord trabajando. Os estaré eternamente agradecido.

A mi tutor, por ser la persona que inició mi interés por las redes de computadoras en primer curso, por toda la ayuda y paciencia y por la correcta guía que ha supuesto para mi en este proyecto. Y al profesor Diego García, por toda la ayuda en la fase de diseño y análisis.

A los profesores de la escuela, por brindarme la mejor educación posible y por formarme durante todos estos años para convertirme en ingeniero informático.

A todos, gracias.

Resumen

Este Trabajo de Fin de Grado consiste en la demostración de la potencia, utilidad y eficacia de GNS3 como herramienta para realizar prácticas de laboratorio de redes en el ámbito educativo, enfocado para los alumnos de asignaturas introductorias de redes de computadoras.

Para demostrar esto, se realizarán dos scripts a modo de talleres de laboratorio, uno para generar una topología de red en base a unos parámetros dados por el usuario y configurar la red hasta un punto u otro en función de la dificultad seleccionada por el usuario y otro script para corregir la configuración de la red que el usuario ha generado previamente con el anterior script.

Está pensado para que el usuario utilice el primer script para generar la topología, configure la red, y compruebe el resultado su tutor o él mismo si la configuración que ha aplicado es correcta o es erronea con el segundo script, haciendo más fácil las dinámicas de prácticas de laboratorio de una forma solamente posible con GNS3.

Abstract

This capstone project is about the GNS3 power's display, utility and effectiveness as a tool to accomplish net lab practices in the educational sector, focused on the introduction of computer networks subjects students.

To demonstrate this, two scripts by way of lab practices will be made, one to generate a network topology based on parameters given by the user and configure the network to a certain point according to the level of difficulty chosen by the user, and another script to check the network configuration that the user previously made with the first script.

It is intended for the user to use the first script to generate the topology, configure the net, and check that if the configuration is correct or not by their tutor or themselves with the second script, making the lab practices dynamics easier with a pattern attainable only with GNS3.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción y Objetivos

1.1. Contexto

Este Trabajo de Fin de Grado ha sido realizado a partir de una idea conjunta de mi tutor Jesús M. Vegas y yo, dirigido en forma de caso de uso a los alumnos que estudien alguna asignatura fundamental relacionada con las redes de computadoras, como en el caso de la Universidad de Valladolid es la asignatura de Fundamentos de Redes de Computadoras, con el objetivo de mejorar su aprendizaje y comprensión de la temática de la asignatura lo máximo posible.

Se optó por este tema debido al especial interés que tengo por las redes de computadoras, siendo mi intención desde el principio enfocar mi TFG en este campo y aprovechar al máximo todos los conocimientos de esta temática aprendidos en la carrera y en las prácticas curriculares que estoy realizando para mejorar el aprendizaje de los alumnos, además de demostrar con esto la potencia y utilidad que presenta la herramienta GNS3 en el ámbito de la enseñanza.

En él, y para demostrar la potencia que esta herramienta posee enfocada en el ámbito educativo, se realizarán unos talleres de laboratorio elaborados de una forma distinta a la habitual, permitiendo a los alumnos poner en práctica los diferentes protocolos, dispositivos y conceptos aprendidos en entornos mucho más cercanos a la realidad para comprobar el verdadero funcionamiento de estos con la ayuda del software libre GNS3.

1.2. Motivación

La motivación principal de este trabajo es generar un interés y un conocimiento mucho mayor del actual en los alumnos por las redes de computadoras enfocándonos más en la parte práctica de estas, esto será posible elaborando unos talleres de laboratorio de una forma

novedosa gracias a la potente herramienta GNS3 que les permite, además de crear, configurar y poner en funcionamiento sus propias redes como haría Packer Tracer, implementar en estas sus propias máquinas virtuales y conectarlas entre sí, ampliando en gran medida las posibilidades prácticas que se pueden extraer de la teoría de la asignatura, ya que a raíz de esto se pueden generar un gran número de talleres de laboratorio en los que poner en práctica los distintos protocolos vistos en la teoría, además de las distintas tecnologías vistas. Añadida a esta, existe también la motivación de poder demostrar la utilidad, gran potencia, facilidad de uso y funcionalidades únicas que posee esta herramienta, haciéndola muy válida para ser utilizada en el ámbito educativo.

Además, GNS3 permite también utilizar tecnologías de fabricantes distintos a Cisco, ya que el usuario tiene que instalar las imágenes de los dispositivos a utilizar manualmente, por lo que, además de poder enseñar tecnología Cisco a los alumnos está la motivación de contar con la posibilidad de poder introducirles en las tecnologías de distintos fabricantes como HP o Huawei entre muchos otros, esto será muy útil de cara a su futuro profesional ya que hoy en día muchas empresas buscan que sus ingenieros tengan conocimientos de tecnologías de distintos fabricantes.

Por último, en resumidas cuentas, la motivación principal es que los alumnos aprendan todo lo que pueden ya pueden hacer actualmente con Packet Tracer como configurar y crear la red desde cero e implementar algunos host virtuales, y ampliarlo con la ayuda de GNS3, ya que los alumnos podrían, además de configurar la red, configurar las propias máquinas virtuales para implementar diversos protocolos y servicios que se ejecuten bajo esa misma red que previamente han configurado, además de aprovechar las funcionalidades únicas de GNS3 para ofrecer a los alumnos unos talleres de laboratorios distintos y novedosos respecto a lo que se presenta típicamente con algunas herramientas como Packet Tracer.

1.3. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es desarrollar un script que permita al usuario crear una topología de red en GNS3 y proporcione, además, su evaluación, suponiendo una forma novedosa y más fácil de realizar los talleres de laboratorio de redes de computadoras para el aprendizaje de los alumnos en este campo. Relacionados a este objetivo principal podemos identificar un pequeño desglose de subobjetivos del proyecto:

- Ampliar de forma práctica las nociones básicas de diseño y configuración de redes aprendidas en Packet Tracer aplicadas a GNS3.
- Dar a conocer GNS3, además de las cualidades, funciones y utilidad de uso que posee esta herramienta en el ámbito educativo.
- Enseñar a los alumnos el uso de algunos de los diversos protocolos vistos en la teoría de la asignatura, como HTTP o DNS entre muchos otros en unas máquinas virtuales para comprobar su funcionamiento utilizando como base las redes configuradas por los propios alumnos.

- Diseñar el reto de los talleres de laboratorio de una forma novedosa y útil, como podría ser mediante la automatización con scripts.
- Facilitar la corrección de los talleres de laboratorio mediante la realización de scripts que comprueben que los retos que el alumno debe superar para completar el taller de laboratorio sean superados con éxito.
- Permitir a los alumnos comprobar que las redes que han diseñado y configurado funcionan perfectamente con equipos virtuales, es decir, otorgar un grado mayor de realismo a la parte práctica de la asignatura.
- Realizar unos talleres optimizados en lo que a recursos se refiere, permitiendo que cualquier alumno pueda completarlos desde cualquier máquina.

1.4. Estructura de la memoria

Se ha decidido que la estructura de este TFG sea dividida en capítulos, dentro de los cuales pueden existir subsecciones que dividen el contenido del capítulo en apartados para una mayor claridad y un mejor orden. Los capítulos principales son los siguientes:

Capítulo 1. Introducción Es el capítulo al que pertenece este apartado, en el cual se habla de manera general y superficial del Trabajo de Fin de Grado. Existen 4 subsecciones dentro de este capítulo:

1.1. Contexto Aquí se describe el origen de este TFG junto a una breve descripción del mismo.

1.2. Motivación En esta subsección se exponen las principales motivaciones de este trabajo, es decir, los distintos aspectos que me han motivado y decidido a realizarlo.

1.3. Objetivos Aquí se exponen una serie de objetivos o metas que se quieren lograr con la realización final de este trabajo.

1.4. Estructura de la memoria Es justo este apartado que se está desarrollando, en el que expongo la manera en la que he estructurado este TFG, exponiendo los capítulos y subsecciones en los que está dividido.

Capítulo 2: Estado del Arte En este capítulo se expondrán numerosas referencias bibliográficas con las cuales se demuestra y justifica que mi proyecto es útil debido a que en otras universidades y centros de enseñanza existen algunos similares al mío y que el resultado del mismo también es utilizado en diversas universidad y centros de enseñanza del mundo. El capítulo se divide en subsecciones, cada una se corresponde a una entrada del estado del arte y, por tanto, a un trabajo distinto que se desarrollarán en el propio capítulo, las subsecciones son las siguientes:

2.1 Evaluación de herramientas de simulación para una asignatura de diseño y configuración de redes

2.2 Propuesta de manual de prácticas de laboratorio de redes utilizando el emulador GNS3

2.3 Integración de la materia Laboratorio de Telemática para la facultad técnica usando el simulador gráfico de redes GNS3

2.4 Guía de Laboratorio 10: Simulador GNS3

Capítulo 3: Requisitos y planificación Este capítulo consta de dos subsecciones que tratan requisitos de los scripts de laboratorio y del proyecto y de la planificación del mismo.

3.1 Requisitos Aquí se tratarán los distintos requisitos identificados a lo largo del proyecto sobre los scripts, tanto los funcionales como los no funcionales, y los relacionados con el proyecto en sí.

3.2 Planificación del proyecto En este subcapítulo se abordará la metodología elegida para la planificación del proyecto y se hablará acerca de cómo ha sido, las actividades planeadas y los riesgos y el presupuesto identificado.

Capítulo 4: Tecnologías utilizadas Esta sección habla acerca de las distintas tecnologías utilizadas a lo largo del proyecto y la razón de la selección de estas.

4.1. GNS3

4.2. Shell de Unix (Dash)

4.3. JSON

4.4. VirtualBox

4.5. Overleaf

4.6. Microsoft Project

4.7. WSL

4.8. Git y GitHub

4.9. Notepad++

4.10. Microsoft Teams

4.11. draw.io

4.12. Astah Professional

4.13. Visual Paradigm

Capítulo 5: Análisis Este capítulo trata la fase de análisis del proyecto, hablando sobre aspectos como los casos de uso, el modelo de dominio, BCE y los diagramas de secuencia

5.1. Diagrama y descripción de casos de uso Se hablará acerca de los distintos actores y casos de uso identificados en el proyecto y una descripción de los mismos para una mayor comprensión

5.2. Modelo de dominio En esta subsección se trata el modelo de dominio identificado para nuestro sistema y las clases que lo componen

5.3. BCE Aquí se trata y explica BCE, como método de identificación de clases de análisis

5.4. Diagrama de secuencia En esta última subsección se desarrollan los diagramas de secuencia por cada caso de uso.

Capítulo 6: Diseño Aquí se trata la fase de diseño de la topología de red seleccionada para el proyecto y de los scripts de laboratorio en sí.

6.1. Diseño de la topología de red Este subcapítulo habla acerca de la topología de red seleccionada como base para el proyecto, el motivo de esta, su configuración y de dónde procede.

6.2. Diseño de los scripts de talleres de laboratorio Aquí se trata la fase de diseño de los scripts complementando a la de análisis anterior, se hablará de los distintos patrones seleccionados y de los diagramas realizados.

Capítulo 7: Implementación y pruebas Este capítulo trata sobre la forma en la que se han implementado los scripts, explicando sus algoritmos, formas de ejecutarlo y cómo está pensado, también se habla sobre las pruebas realizadas, una descripción de estas y si han sido superadas o no

Capítulo 8: Seguimiento del proyecto Aquí se habla sobre el transcurso del proyecto en sí, una descripción de cómo ha sido su desarrollo a la vez que se hablan acerca de los problemas encontrados y los cambios realizados en él.

Capítulo 9: Conclusiones Trata acerca de las conclusiones finales del proyecto, explicando lo aprendido y lo logrado con su desarrollo. También cuenta con un subcapítulo que trata acerca de las posibles lineas de trabajo futuras que se podrían hacer usando este proyecto como base

9.1 Lineas de trabajo futuras Este subcapítulo expone ideas para futuros proyectos que se podrían realizar utilizando este proyecto y sus resultados como base.

Bibliografía Esta sección trata de las referencias bibliográficas utilizadas a lo largo del proyecto

Anexo A. Manuales Esta sección del anexo incluye los distintos manuales para el usuario y el administrador del sistema en el que se instalarán los scripts resultados del proyecto.

A.1. Manual de despliegue e instalación Este manual enseña a instalar todo lo necesario para poner en marcha y utilizar los scripts resultados del proyecto.

A.2. Manual de mantenimiento En este manual se aclaran los distintos puntos a tener en cuenta para evitar errores en la instalación y ejecución de los scripts, está pensado para el administrador del sistema

A.3. Manual de usuario Está enfocado para que el usuario aprenda a ejecutar los scripts usando la red base propuesta como ejemplo. También cuenta con un tutorial-solución de la configuración de la red propuesta.

Anexo B. Resumen de enlaces adicionales Está ultima sección del proyecto describe los enlaces adicionales de interés, como el repositorio de GitHub con el código del proyecto

Capítulo 2

Estado del arte

En esta sección se procederá a explicar de forma más detallada todos los aspectos relacionados con el estado del arte del proyecto. Este capítulo se irá desarrollando en subapartados en los cuales se irá explicando una a una cada entrada bibliográfica perteneciente al estado del arte.

El estado del arte consiste en la presentación de las diferentes investigaciones y publicaciones relacionadas con el proyecto que desarrollo en este informe, también se pueden incluir aquí referencias a otros proyectos que resulten similares al mio con el fin de demostrar y justificar la realización del mismo.

Aunque es cierto que principalmente se compone de proyectos similares al mío, ya que considero que para este tipo de proyectos lo más convincente es ver cómo se aplica su resultado en entornos similares, en mi caso, demostrar y enseñar cómo y por qué se utiliza GNS3 en otras universidades con objetivos académicos.

2.1. Evaluación de herramientas de simulación para una asignatura de diseño y configuración de redes

En esta primera entrada encontramos una búsqueda de la herramienta adecuada para una asignatura relacionada con las redes de ordenadores, para ello la autora desarrolla una comparación de herramientas de simulación de red. Esta entrada es un Trabajo de Fin de Grado escrito por la alumna Gemma María García Mínguez, estudiante del Grado en Ingeniería de Tecnología Específicas de Telecomunicación de la Universidad de Valladolid [45].

En él, Gemma desarrolla una comparación entre diferentes herramientas de simulación de red con el fin de encontrar uno que más se adapte a la asignatura Laboratorio de Diseño y Configuración de Redes perteneciente a su grado, para ello compara una multitud de simuladores entre los que se encuentran: NS2[46], CISCO VIRL[55], CLOONIX[27], GNS3[49]

y Cisco Packet Tracer[53], siendo este último el simulador utilizado actualmente en nuestra escuela.

La autora compara estas herramientas utilizando diferentes métricas: primero define una serie de requisitos que estos simuladores deben cumplir, como por ejemplo que sea de código abierto y que sea compatible con Windows y Linux; después compara las prestaciones que cada herramienta proporciona y que resulten útiles para los estándares de la asignatura relacionada con el trabajo, algunos de estas prestaciones es el soporte de VLAN, el soporte del protocolo de enrutamiento OSPF[12] y RIP[15] además de soportar funcionalidades básicas como permitir un direccionamiento IP y la creación de redes de área local.

También compara otros aspectos interesantes para el proyecto, como el soporte de funcionalidades estadísticas que permita analizar el resultado de las simulaciones de las prácticas que realicen los alumnos en la asignatura y algunas funcionalidades más avanzadas y específicas, como la capacidad de simulación de servicios de Quality of Service (QoS)[3] y otras características destacables como la compatibilidad con máquinas virtuales, si está orientado a titulación o estudios y no para uso académico, si implementa otros protocolos de diferentes capas tales como HTTP[13], Telnet[18] o Ethernet entre muchos otros y si incorpora la capacidad de implementar distintos tipos de red como redes inalámbricas o redes móviles ad-hoc[21].

Una vez realizadas las comparaciones de las funcionalidades y de las prestaciones de los diferentes simuladores de red procede a desarrollar algunos últimos aspectos en tener en cuenta como por ejemplo la curva de aprendizaje de la herramienta para los alumnos y la cantidad de actualizaciones y el mantenimiento que el software recibe para, finalmente, tomar una decisión de cual simulador será el elegido y el más adecuado para la asignatura, que ha resultado ser GSN3, debido a que cumple la gran mayoría de requisitos que la alumna ha descrito a lo largo de las diferentes comparaciones, por lo que esa será la herramienta que se utilizará en la asignatura.

Por último, la autora desarrolla un pequeño tutorial de instalación y configuración de GNS3 y unos talleres de laboratorio básicos para la asignatura. Este Trabajo de Fin de Grado nos sirve para comprender cómo GNS3 es muy adecuado para el entorno educativo y las razones de esto, además de contar con unos pequeños ejemplos de talleres de laboratorio que pueden ser útiles para las propuestas que se desarrollarán en siguientes capítulos, demostrando con esto también que GNS3 es útil para este tipo de metodología de enseñanza enfocada más en la parte práctica con el uso de talleres de laboratorio.

2.2. Propuesta de manual de prácticas de laboratorio de redes utilizando el emulador GNS3

En esta segunda entrada el autor desea realizar un manual de laboratorio de redes con la ayuda de GNS3, exponiendo además por qué esta herramienta es superior al resto y es la que ha decidido finalmente utilizar para su proyecto. En esta ocasión es un Trabajo de Diploma escrito por el alumno Joaquín Gómez Carmona, estudiante de la especialidad Ingeniería en

Telecomunicaciones y Electrónica en la Universidad Central "Marta Abreu" de Las Villas en Cuba[26].

En primer lugar, el alumno en este trabajo habla, de forma general, de la simulación y emulación de redes, describiendo sus ventajas y limitaciones y las principales características que poseen las herramientas software que se dedican a esto. Después, hace una comparativa directa entre GNS3 y Packet Tracer, describiendo de forma bastante general las características principales, funcionalidades y la finalidad principal para la que fueron diseñados ambos simuladores de red, saliendo con un mejor resultado Packet Tracer en lo que a consumos de recursos se refiere pero GNS3 en cuanto a la cantidad de funcionalidades que se ofrece. El autor también nos ofrece una descripción de una serie de investigaciones que descubrió previamente para la utilización de GNS3 con finalidad de enseñanza de redes, con artículos que miden desde la interfaz gráfica hasta su rendimiento, también presenta algunos artículos en los que alumnos y profesores trabajan con esta herramienta ara llevarlo al aprendizaje de redes en estudios de ingeniería, con ayuda de talleres de laboratorio y habiéndolo comparado con algunas otras herramientas software similares, destacando que GNS3 permite acercarse más a un entorno real debido a la posibilidad de implementación de máquinas virtuales. A continuación, el autor procede a explicar las características principales de este simulador y de la emulación de dispositivos como routers, switches y PC bajo esta herramienta.

Por último, el alumno expone el diseño de una serie de talleres de laboratorio con el objetivo de reforzar los conocimientos de redes de los alumnos en un ambiente práctico, presentando unos talleres de laboratorio compuesto por una guía que el alumno debe seguir para luego responder una serie de reflexiones finales. En estos talleres también pone empeño en la superioridad de funcionalidades que tiene GNS3 sobre Packet Tracer, ya que emplea, en algunos de ellos, máquinas virtuales que amplían sobremedida las posibilidades de enseñanza de redes a los alumnos, por ejemplo, en uno de los talleres el alumno enseña a realizar un ataque utilizando SLAAC[63] desde cero gracias a la capacidad de GNS3 de permitir el uso de máquinas virtuales, enseñando a configurar los dispositivos y la red para, a posteriori, invitar al alumno a realizar una reflexión sobre los resultados de la práctica y sobre los ataques mediante IPv6[9].

En definitiva, gracias a este Trabajo de Diploma podemos conocer más aspectos en los que GNS3 supera a Packet Tracer y nos demuestra que sí es una herramienta realmente útil para enseñar redes a los alumnos en talleres de laboratorios, ya que nos demuestra una y otra vez las superiores funcionalidades que nos ofrece y como nos amplía las posibilidades a la hora de poder enseñar diferentes aspectos de las redes y de los protocolos a los alumnos, aunque cuente con la desventaja del mayor uso de recursos respecto a Packet Tracer, el autor de este trabajo eligió para los talleres de laboratorio unas topologías enfocadas a un menor consumo de recursos de la máquina local, por lo que a la hora de realizar los talleres de laboratorio de este Trabajo de Fin de Grado habrá que seguir un enfoque similar para que cualquier alumno pueda realizarlos en cualquier máquina.

2.3. Integración de la materia laboratorio de telemática para la facultad técnica usando el simulador gráfico de redes GNS3

En esta tercera entrada del estado del arte podemos encontrarnos con una tesis en la que los autores desean crear una asignatura relacionada con la parte práctica de las redes de computadoras y para ello utilizarán GNS3. Está escrita por Mario Javier Gil Cevallos y Verónica Paola Berruz Silva, estudiantes del Grado en Ingeniería de Telecomunicaciones en la Universidad Católica de Santiago de Guayaquil[66]

Esta Tesis sigue una senda muy parecida a la de los dos trabajos presentados en los dos subcapitulos anteriores, teniendo como objetivo principal la creación de una asignatura enfocada a la formación de estudiantes en el area del networking y tecnologías de la información de cara a su futuro laboral, teniendo un enfoque más práctico con unos talleres de laboratorios, para realizarlos ha optado por utilizar como herramienta de simulación GNS3.

La Tesis se divide en 6 capítulos:

1. El primero de ellos consiste en una introducción, en la que expone el planteamiento del problema, una justificación, unos objetivos y la metodología de investigación.

2. Después, en el segundo capítulo, constata unos fundamentos teóricos, introduce de manera breve el entorno de laboratorio de la asignatura resultado del proyecto y un poco de teoría de redes como los conceptos de Routing y Switching. A continuación, en el mismo capítulo, los autores hablan, de manera bastante detallada, acerca de GNS3, sus características principales y las tecnologías en las que se basa para su funcionamiento.

3. En el tercer capítulo se detallan algunas de las diferentes tecnologías que es capaz de emular GNS3 con el fin de demostrar la potencia de esta herramienta, entre estas tecnologías se encuentran VPN[14], Frame Relay[7], MPLS[11], VLAN Y VPLS[20].

4. En el capítulo 4 los autores ya proceden a hablar del diseño de la asignatura, para ello se describe su plan de estudios, el calendario académico de esta y las ocho propuestas de talleres de laboratorio que la componen. Estos talleres son bastante básicos, enseñando los conceptos y la herramienta desde el principio para asimilar bien todos los conceptos desde un nivel más bajo como podría interconectar simplemente 3 router e ir aumentando dicha dificultad progresivamente añadiendo nuevos conceptos hasta llegar a que los alumnos formen su propia VPN con la herramienta o simulen su propio acceso remoto. Los autores exponen también su intención de que, al final de curso, los alumnos realicen un proyecto final relacionado con la temática de la asignatura utilizando GNS3.

5. En relación con este cuarto capítulo tenemos el capítulo 5, que trata de los costes aproximados que supondría integrar la asignatura en el grado,en él los autores exponen el coste que supondría comprar los equipos físicos y el software necesario para desarrollar la asignatura para que al universidad a la hora de aprobar el proyecto tenga en cuenta estos costes.

6. Para finalizar, el capítulo 6 supone una conclusión general de toda la Tesis y los futuros trabajos relacionados con este.

Por último, con la adición de esta Tesis al estado del arte de este trabajo quiero, además de recalcar lo dicho en las entradas anteriores sobre que GNS3 es una herramienta muy útil para la enseñanza, remarcar que es una herramienta de simulación muy intuitiva y fácil de usar, además de potente, lo que nos permite enseñar tanto conceptos básicos de redes de computadoras como conceptos cada vez más avanzados e ir ampliando en gran medida los protocolos, dispositivos y conceptos relacionados con el mundo de las redes de computadoras que se pueden enseñar a los alumnos.

2.4. Guía de Laboratorio 10: Simulador GNS3

Para finalizar este capítulo, la última entrada del estado del arte no trata de un Trabajo de Fin de Grado o un Trabajo de Diplomatura como los anteriores, es un ejemplo de taller de laboratorio de la asignatura Redes de Computación de la Facultad de Ingeniería de la Universidad Don Bosco en El Salvador[61].

En este taller se enseña al alumno los fundamentos básicos de GNS3, empezando por enseñar a instalarlo mientras se le proporciona una explicación técnica del simulador y sus funcionalidades, se diferencia de las propuestas de talleres de subcapítulos anteriores en que este está mas detallado paso por paso debido a que enseña al alumno a usar la herramienta desde cero, señalándole en todo momento qué es lo que debe hacer y también debido a que en los trabajos anteriores se hablaba en todo momento de propuestas, en cambio ahora estamos hablando de un guión definitivo y listo para presentar al alumno.

Tras esta introducción, comienza el verdadero taller, en el que, paso a paso de manera detallada y sin posibilidad de perderse, se va guiando al alumno para, primero, enseñar de una forma general la interfaz del simulador y luego ir guiándolo para que coloque y conecte los dispositivos con el fin de crear la topología de red propuesta. Una vez ha creado la topología se explica cómo configurarla, es decir, configurar la IP de los equipos virtuales, la imagen IOS del router y la IP de los mismos, todo también de manera muy detallada.

En esta ocasión no se invita al alumno a hacer una reflexión final sobre lo que ha aprendido, como si hemos visto en las propuestas de talleres de laboratorio de subapartados anteriores, pero sí es cierto que la forma de calificar al alumno o comprobar lo que ha aprendido consiste en la propia realización del taller de laboratorio mediante algunas pruebas de funcionamiento en la parte final de este.

Finalmente, con la presentación de este taller de laboratorio en este subcapítulo pretendo acompañar a los otros en demostrar que GNS3 es una herramienta de simulación útil y potente para la enseñanza de redes de computadoras y demostrar que se usa en distintos lugares de enseñanza con el mismo propósito que el mio en este proyecto.

Capítulo 3

Requisitos y Planificación

Este capítulo se divirá en dos subapartados, en el primero se procederá a explicar los requisitos del proyecto, es decir, los requisitos que se deberán cumplir en el proyecto en sí y en el objeto resultado del mismo, en el segundo subapartado se explicará la planificación general del proyecto, así como sus riesgos, plazos, presupuesto y actividades del mismo.

3.1. Requisitos

En las tablas siguientes, desde la 3.1 a la 3.5 se muestran los diferentes requisitos identificados para los dos scripts y para el proyecto en general, para los scripts son requisitos funcionales y no funcionales pero para el proyecto solo funcionales. Por cada uno de los requisitos se muestra un código que lo identifica, el nombre del requisito y una descripción del mismo que ayuda al lector a su comprensión.

Los talleres de laboratorio se explicarán más adelante con detenimiento, pero a grandes rasgos consistirá en la creación de un script que genere una red base en GNS3 y, dependiendo de la dificultad que el usuario indique, configure unos aspectos u otros, esperando por parte del usuario que configure las partes restantes para completar la actividad. Al terminar dicha actividad se ejecutará otro script que compruebe que todo funciona correctamente y, por ende, indique si la actividad ha sido completada con éxito o no.

El proyecto, como se ha indicado en otros capítulos anteriores, consiste en demostrar la utilidad de GNS3 en entornos educativos explicando sus posibilidades, funcionalidades y potencia además de demostrarlo con los talleres de laboratorio que se generarán como resultado del proyecto, ya que el objetivo final del mismo es demostrar que es una herramienta útil para desarrollar prácticas de laboratorio de redes en entornos educativos. También consiste en, apoyando este objetivo principal, demostrar la potencia de esta herramienta en el entorno educativo con la automatización de los talleres de laboratorio, siendo esto una forma novedosa de enseñanza para los alumnos únicamente posible con GNS3.

Código	Nombre	Descripción
RF1	El script de los talleres de laboratorio debe seguir una curva de nivel de dificultad adecuada al aprendizaje de los usuarios	La curva de dificultad de los talleres de laboratorio debe ser moderada y no extrema, deben generarse unos niveles de dificultad que escalen de manera progresiva sin diferencias exageradas entre un nivel y otro.
RF2	El script de los talleres de laboratorio debe estar diseñados de una forma que se asegure que los usuarios adquieran los conocimientos que se intentan inculcar	El objetivo final del taller de laboratorio es enseñar algo a los alumnos, por ende, el taller debe ser lo más completo posible para poder inculcar los máximos conocimientos en los usuarios, además de asegurarse que se han adquirido.
RF3	El script de los talleres de laboratorio debe proporcionar todo el material necesario para su realización al usuario	A la hora de realizar los talleres de laboratorio debe proporcionarse al usuario todos los materiales necesarios o enseñarle a conseguirlos, ya sea dar materiales como podría ser las máquinas virtuales relacionadas con la red o enseñar a conseguir el software como podría ser en caso de que no tuviera instalado GNS3 o VirtualBox[47].
RF4	El script de los talleres de laboratorio debe permitir corregir su resultado de manera automática	Va relacionado con el script de corrección mencionado en la introducción del capítulo, se expone que el resultado de realizar el taller, es decir, una red completamente funcional y configurada, debe permitir ser puesta a prueba mediante el mismo script u otro que se obtenga como resultado en este proyecto, dependerá del diseño del mismo.
RF5	El script de los talleres de laboratorio debe permitir a los usuarios elegir la dificultad	Va relacionado con el requisito RF1, el script debe permitir al usuario seleccionar una dificultad para el taller nada mas ejecutarlo.
RF6	El script de los talleres de laboratorio debe actuar distinto dependiendo de la dificultad elegida por el usuario	Continuando el requisitor RF5, el script actuará de manera distinta dependiendo de la dificultad elegida por el usuario nada más ejecutarlo, ya sea configurando la red de una forma o configurándola hasta cierto punto, dependerá de la fase de diseño del mismo.
RF7	El script de los talleres de laboratorio debe comprobar que los comandos que va a utilizar están disponibles	Antes de su ejecución, el script deberá comprobar que todos los comandos que va a utilizar están disponibles en la máquina e informar en caso de que no sea así.

Código	Nombre	Descripción
RF8	El script de los talleres de laboratorio debe generar un proyecto GNS3 de manera automática	Al finalizar la ejecución del script, habiendo seleccionado previamente el nivel de dificultad y cumplir todos los requisitos de ejecución del mismo debe generar como resultado un proyecto GNS3 con un fichero .gns3 asociado el cual el usuario debe ser capaz de abrir en la aplicación gráfica sin problema.
RF9	El script de los talleres de laboratorio debe leer la configuración de la red desde un fichero JSON	El script requerirá al usuario que se le indique un fichero JSON con la configuración y topología de la red sobre la cual se quiere trabajar, indicando, por ejemplo, el número de routers, switches y host y la configuración de estos.
RF10	El script de los talleres de laboratorio incluirá un manual de uso	En este mismo proyecto existirá un apartado en el que se explicará de principio a fin cómo utilizar este script de manera clara y concisa, para que cualquiera pueda entenderlo y utilizarlo
RF11	El script de los talleres de laboratorio estará documentado	El código del script estará documentado por si es necesaria su revisión, haciendo que el código sea mucho más legible y fácil de comprender por si otra persona o yo mismo en un futuro necesita revisar o cambiar alguna parte del código ya sea para hacerlo más eficiente o para un futuro trabajo que use el mío como base
RF12	El script de los talleres de laboratorio debe comprobar que el fichero JSON de entrada existe	Antes de la ejecución del script se deberá comprobar que existe dicho fichero JSON del que se extraerá la configuración de la red.
RF13	El script de los talleres de laboratorio deberá informar del estado de ejecución del mismo	Durante su ejecución, el script deberá informar al usuario del estado del mismo, es decir, deberá informar en caso de que haya algún error, de que finalice correctamente o de que se necesita instalar algún comando que el usuario no tiene instalado.

Código	Nombre	Descripción
RF14	El script de los talleres de laboratorio debe limpiar los ficheros temporales que haya creado	El script deberá hacer una limpieza de todos los ficheros temporales utilizados para almacenar información, ya se al finalizar o si se cancela la ejecución del script.
RF15	El script debe eliminar el proyecto creado en caso de que se cancele la ejecución del mismo	Para una mayor limpieza en caso de que el usuario desee cancelar la ejecución del script se deberá eliminar el proyecto GNS3 creado en caso de que el usuario cancele la ejecución.
RF16	El script de los talleres de laboratorio debe operar mediante peticiones web a un servidor que ejecuta GNS3 en la máquina local	GNS3 funciona mediante un servidor web, al cual se le hacen peticiones para utilizar la plataforma, dependiendo de la petición se pueden crear proyectos, crear nodos... por lo que el script aprovechará esta utilidad para su funcionamiento.
RF17	El script de los talleres de laboratorio debe comprobar que todos los parametros de entrada sean válidos	Antes de su ejecución el script comprobará que los parámetros que el usuario ha introducido sean válidos.
RF18	El script de los talleres de laboratorio debe comprobar que la conexión con el servidor GNS3 se establece correctamente	El script, nada mas comenzar su ejecución, deberá comprobar que la conexión con el servidor local de GNS3 se establece correctamente e informar al usuario en caso de que no sea así.

Tabla 3.1: Tabla de requisitos funcionales del script de los talleres de laboratorio

Código	Nombre	Descripción
RNF1	El script de los talleres de laboratorio debe ser fácil de usar	El script debe resultar fácil de utilizar para que los alumnos puedan usarlos sin problema, para ello su funcionamiento debe ser simple, útil y rápido para el usuario
RNF2	El script de los talleres de laboratorio debe poder ejecutarse en cualquier máquina Unix compatible con GNS3	Como se explicará más adelante, el script de laboratorio será desarrollado mediante un Shell Script[17], el cual solo será posible ejecutarse en máquinas que estén basadas en Unix[19] como GNU/Linux[31], por ende, debe ser posible de ejecutar en cualquier máquina que sea capaz de ejecutar el intérprete de comandos Dash[65].
RNF3	El script de los talleres de laboratorio debe consumir el mínimo de recursos posibles	Uno de los objetivos de este proyecto consiste en que el usuario pueda completar los talleres de laboratorio utilizando cualquier máquina. Es cierto que GNS3 consume bastante más recursos que alguna otra plataforma similar como podría ser Packet Tracer debido a las tecnologías que utiliza por debajo, por lo que un requisito fundamental es que, tanto el propio script, como la red resultado de realizar el taller completo, deben consumir el mínimo de recursos posible para aumentar la accesibilidad del mismo.
RNF4	El script de los talleres de laboratorio debe tener como base los conocimientos que los alumnos ya han adquirido anteriormente en redes.	Es decir, este script solicitará conocimientos básicos de administración de redes y protocolos simples que se estudian en asignaturas introductorias a la materia, por lo que no habrá problema por parte de los usuarios en realizarlos debido a que no se requieren o piden conocimientos avanzados en la materia ni se estudiarán protocolos complejos.
RNF5	El script de los talleres de laboratorio debe utilizar comandos básicos de Unix	El script, al ser un Shell Script deberá utilizar solamente comandos básicos de Unix, en caso de que no esté instalado alguno de ellos se informará al usuario de que lo instale.
RNF6	El script de los talleres de laboratorio debe ejecutarse y completar la tarea en el mínimo tiempo posible	Es cierto que la ejecución del script dependerá de los recursos de la máquina en la que se lanza, pero el diseño del mismo deberá hacer que se ejecute en el mínimo tiempo posible para evitar incomodidad al usuario con el uso del mismo y generar una experiencia más satisfactoria.

Código	Nombre	Descripción
RNF7	El script de los talleres de laboratorio debe generar una red bien organizada, fácil de entender y bien nombrada	La red generada como resultado del script debe ser entendible de un solo vistazo, además de que los nombres de los nodos de red deben ser claros y seguir un orden. También, en caso de que haya que configurarlo, la asignación de direcciones, nombres de VLAN y elementos relacionados con la configuración deben ser claros y seguir un orden coherente, fácil de recordar y de comprobar
RNF8	El script de los talleres de laboratorio debe poder ejecutarse tantas veces como se desee	Es decir, el resultado del script solo debe variar si se cambia el fichero de configuración o el nivel de dificultad, sino deberá generar siempre el mismo resultado al ejecutarse, por tanto, el script debe ser repetible.
RNF9	El script de los talleres de laboratorio debe ser un Shell Script	Como se mencionó en el requisito RNF2, el script consistirá en un conjunto de instrucciones Unix que serán interpretadas por un interprete de comandos de Unix.
RNF10	El script de los talleres de laboratorio debe ser fácil de trasladar de un sistema a otro	Mientras el sistema en el que se encuentre tenga GNS3 instalado, sea una máquina Unix y cuente con los comandos necesarios, el script deberá poder ejecutarse sin problemas y su comportamiento debe ser el mismo que en la máquina original, por lo que existe una portabilidad fácil.

Tabla 3.2: Tabla de requisitos no funcionales del script de los talleres de laboratorio

Código	Nombre	Descripción
RF1	El script de corrección de los talleres de laboratorio debe funcionar en base a un proyecto GNS3 ya previamente creado	El script corregirá el taller de laboratorio que hemos creado con el anterior script, por lo que deberá funcionar en base a un proyecto ya creado previamente por el script previo.
RF2	El script de corrección de los talleres de laboratorio funcionará mediante peticiones web al servidor que ejecuta GNS3	Al igual que para crear los talleres de laboratorio hace falta realizar peticiones web al servidor web que abre GNS3, para recoger información del proyecto también hace falta esta funcionalidad
RF3	El script de corrección de los talleres de laboratorio debe informar al usuario del estado de ejecución del mismo	Si ha ocurrido un error durante la ejecución o ha terminado de realizarse con éxito el script debe informar al usuario en todo momento.
RF4	El script de corrección de los talleres de laboratorio debe informar al usuario del resultado de la corrección	La finalidad del script es corregir un taller de laboratorio que se ha realizado previamente, por lo que es imprescindible de que, al finalizar, el script informe al usuario de si ha completado el taller con éxito o no, y si no es así informar qué ha fallado.
RF5	El script de corrección de los talleres de laboratorio debe solicitar el ID del proyecto que se ha generado previamente para corregirlo	El script deberá recibir por parte del usuario un ID de proyecto que ha sido facilitado previamente por el script de los talleres de laboratorio además de poder comprobarlo en la propia plataforma para saber qué proyecto es el que debe corregir.
RF6	El script de corrección de los talleres de laboratorio debe solicitar el fichero JSON de configuración de la red	El script recibirá por parametro un fichero JSON desde el cual deberá basarse para la corrección del taller de laboratorio
RF7	El script de corrección de los talleres de laboratorio debe limpiar los ficheros temporales que haya creado	Al igual que el script de los talleres de laboratorio, en caso de que haya sido necesaria la creación de ficheros temporales, el script deberá eliminarlos una vez finalice su ejecución o se cancele la ejecución del mismo

Código	Nombre	Descripción
RF8	El script de corrección de los talleres de laboratorio debe comprobar que existe un proyecto con el ID que se le ha ordenado corregir	El script debe asegurarse que realmente existe un proyecto GNS3 con el ID de proyecto que se le ha indicado, en caso de no ser así informará al usuario con un error.
RF9	El script de corrección de los talleres de laboratorio debe comprobar que todos los comandos a utilizar están instalados y disponibles en la máquina	Antes de comenzar la corrección, el script debe comprobar que los comandos que necesita están instalados y están disponibles para usarse. En caso contrario deberá informar al usuario de ello.
RF10	El script de corrección de los talleres de laboratorio debe estar documentado	El código del script debe estar documentado con comentarios, así si otra persona utilizando mi trabajo como base o yo mismo pasado un tiempo necesitamos revisar el código será mucho más fácil entenderlo y realizar las modificaciones oportunas.
RF11	El script de corrección de los talleres de laboratorio debe tener un manual de uso	Al igual que el script de los talleres de laboratorio, este deberá contar con un manual de uso en un apartado de este proyecto que explicará cómo utilizar el script de forma que cualquiera pueda entenderlo sin dificultad.
RF12	El script de corrección de los talleres de laboratorio debe comprobar que la conexión con el servidor web local de GNS3 se establece correctamente	Antes de empezar la corrección del proyecto, el script realizará una prueba de conexión para comprobar que el servidor web de GNS3 es accesible, en caso contrario avisará al usuario.

Tabla 3.3: Tabla de requisitos funcionales del script de corrección de los talleres de laboratorio

Código	Nombre	Descripción
RNF1	El script de corrección de los talleres de laboratorio debe tener una corrección fácil de entender para el usuario	El objetivo final de este script consiste en ofrecer una corrección de los talleres de laboratorio que el usuario ha realizado previamente, por ello, la corrección que se ofrece al usuario debe ser concisa y fácil de comprender, para no dar lugar a dudas y permitir su aprendizaje de ello.
RNF2	El script de corrección de los talleres de laboratorio debe utilizar comandos Unix básicos	El script deberá basarse en comandos Unix básicos, esto ayudará a que haya una mayor accesibilidad y compatibilidad al mismo, haciéndolo compatible con más maquinas que si lo hacemos dependiente de librerias externas u otros programas o script.
RNF3	El script de corrección de los talleres de laboratorio debe poder ejecutarse en cualquier máquina Unix	Con el mismo objetivo de ofrecer una mayor accesibilidad y compatibilidad, el script deberá ser posible de ejecutar en cualquier máquina que corra bajo un sistema Unix, independientemente de la distribución que presente.
RNF4	El script de corrección de los talleres de laboratorio debe consumir el mínimo número de recursos posibles	Con el fin de hacer el script más compatible con máquinas con menos recursos y no sobrecargarlas, el script deberá consumir los mínimos recursos posibles.
RNF5	El script de corrección de los talleres de laboratorio debe ejecutar y completar la tarea en el mínimo tiempo posible	Esto ayuda a la usabilidad del mismo, ya que el script será mucho más cómodo de utilizar para el usuario si no requiere de mucho tiempo para realizar sus funciones y ofrece un resultado de manera rápida.
RNF6	El script de corrección de los talleres de laboratorio deberá ser un Shell Script	Al consistir en automatizar comandos básicos de Unix, lo más apropiado es desarrollar un Shell Script.
RNF7	El script de corrección de los talleres de laboratorio debe ser fácil de usar	Para una mayor accesibilidad para el usuario, el script deberá ser lo más sencillo posible de utilizar asi como simple, de esta forma una mayor cantidad de usuarios podrán utilizarlo al no resultarles complejo su uso.
RNF8	El script de corrección de los talleres de laboratorio debe ser fácil de trasladarse de un sistema a otro	Al tratarse de un Shell Script, la portabilidad del mismo es muy sencilla y, mientras la máquina desde la que se ejecuta tenga GNS3 instalado, sea una máquina Unix y tenga los comandos necesarios instalados debería poder ejecutarse sin problemas.

Tabla 3.4: Requisitos no funcionales del script de corrección de los talleres de laboratorio

Código	Nombre	Descripción
R1	El proyecto debe explicar con claridad qué es y en qué consiste la herramienta GNS3	A lo largo del proyecto debe dejarse claro al lector qué es la herramienta GNS3 y para qué sirve, ya que es el contexto principal de el proyecto y un elemento clave que debe conocer bien
R2	El proyecto debe explicar con claridad la importancia y utilidad de usar GNS3 en el entorno educativo	Un aspecto fundamental del proyecto es el ámbito al que va dirigido, en este caso es el ámbito educativo, por lo que se debe explicar muy claramente la importancia de la herramienta en dicho ámbito y lo que es capaz de aportar, además de lo que la diferencia de muchas otras herramientas similares ya existentes y adoptadas en dicho ámbito.
R3	El proyecto debe explicar las funcionalidades, utilidades, potencia y posibilidades de la herramienta GNS3	Va ligado a la explicación de qué es GNS3 que se detalla en el requisito R1, consiste en, además de explicar qué es, explicar todas sus cualidades de manera detallada y de qué es capaz la herramienta, además de demostrar la utilidad, potencia y posibilidades que ofrece.
R4	El proyecto debe explicar con claridad en qué consiste los scripts de laboratorio y de corrección que se quieren presentar	Como se ha explicado anteriormente, el resultado del proyecto, además de la memoria, son dos script, uno que genera un taller de laboratorio para que el usuario lo complete y otro que lo corrija, ambos han de ser explicados con claridad, tanto su finalidad, como su uso y cómo han sido desarrollados
R5	El proyecto debe demostrar la utilidad real de la plataforma para realizar prácticas de laboratorio	Se quiere demostrar que GNS3 es una herramienta muy útil y potente para el ámbito educativo, en este caso el proyecto se enfocará sobre todo en demostrar su utilidad para la parte práctica de este que consiste en realizar talleres de laboratorio para que los usuarios aprendan a configurar redes desde cero, por lo que se demostrará que esta herramienta es ideal para ello.
R6	El proyecto debe cumplir con los objetivos del mismo	Es importante que el proyecto cumpla los objetivos que se marcan en la memoria ya que son imprescindibles junto a estos requisitos para que el proyecto finalice satisfactoriamente.

Tabla 3.5: Tabla de requisitos del proyecto

3.2. Planificación del proyecto

Una vez se han expuesto los requisitos del proyecto y los script resultado es momento de conocer cómo será la planificación del proyecto.

La planificación consiste en aplicar una metodología de trabajo para poder estimar la duración del proyecto, además de cómo se gestionará el tiempo en las actividades que lo componen, además de conocer y prever los riesgos que podrían ocurrir, que afectarían al proyecto retrasándolo o modificando las actividades o el proyecto entero, por último se deberán conocer los costes estimados que agenciará el proyecto.

Se expondrán primeramente las actividades del proyecto, conociendo una lista de actividades con su duración y las dependencias que haya entre ellas.

Después, se expondrá un análisis de riesgos, el cual se compone de una lista con los posibles riesgos identificados que pueden ocurrir a lo largo del proyecto, además de con una descripción de los mismos, la probabilidad de que ocurra y el nivel de impacto que supondría en caso de que ocurriese.

Por último, se expondrá una lista con los costes estimados del proyecto, para tener una idea aproximada de cuanto costaría llevarlo a cabo.

La metodología elegida para la planificación del proyecto es la metodología de cascada debido a la simplicidad y sencillez que supone y, como no tengo que desarrollar un sistema software excesivamente complejo, lo veo ideal como enfoque para este proyecto. La metodología de desarrollo en cascada[5] expone que el desarrollo software surja en etapas y que una nueva etapa no puede comenzar hasta que la anterior termine, así hasta la finalización del proyecto. En la figura 3.1. se puede observar un esquema de ejemplo de cómo es el desarrollo en cascada

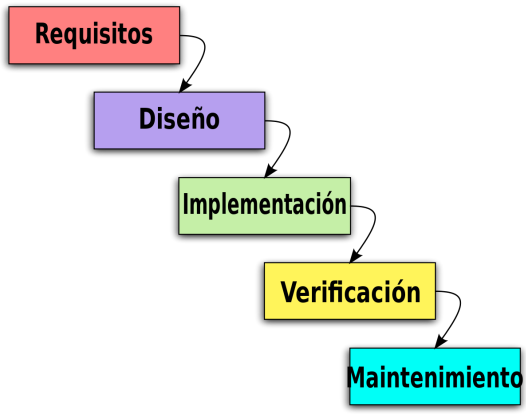

Figura 3.1: Ejemplo de desarrollo en cascada.

Una vez conocida la metodología a utilizar podemos desarrollar la lista de actividades, su duración y sus dependencias. El proyecto está pensado para empezar el 2 de marzo y terminar el 10 de junio para tener tiempo suficiente a revisiones y cambios por parte del tutor y para ser presentado en convocatoria ordinaria, en la tabla 3.6 se muestra esta lista de actividades que puede sufrir variaciones debido a los distintos riesgos o incidentes que puedan surgir, pero esta es la planificación del proyecto ideada. Cabe mencionar que la redacción de la memoria se realiza en cada parte, es decir, en la parte de diseño de los scripts está incluido también la inclusión de dicha parte en la memoria del proyecto, la actividad final con código 8 corresponde a la redacción de las partes que no hayan sido redactadas antes y la mejora que se pueda hacer a la totalidad de la memoria.

Código	**Nombre**	**Fecha Comienzo**	**Fecha final**	**Dependencias**
1	Puesta en marcha	02/03/2022	11/04/2022	-
1A	Investigación de la plataforma	02/03/2022	10/03/2022	-
1B	Aprendizaje de Shell Script	11/03/2022	16/03/2022	1A
1C	Aprendizaje de JSON	17/03/2022	21/03/2022	1B
1D	Aprendizaje de conexión entre Shell y la nueva plataforma	22/03/2022	28/03/2022	1C
1E	Redacción de primeras partes de la memoria	29/03/2022	11/04/2022	1D
2	Diseño de la topología de red	12/04/2022	15/04/2022	1E
3	Fase de analisis del shell script	18/04/2022	21/04/2022	2
4	Fase de diseño del shell script	22/04/2022	28/04/2022	3
5	Implementación del shell script	29/04/2022	19/05/2022	-
5A	Implentacion del script de creación de los talleres de laboratorio	29/04/2022	10/05/2022	4
5B	Implementación del script de corrección de los talleres de laboratorio	11/05/2022	19/05/2022	5A
6	Fase de pruebas	20/05/2022	23/05/2022	6
7	Desarrollo de los manuales de usuario	24/05/2022	26/05/2022	5B
8	Redacción restante de la memoria y mejora de esta	27/05/2022	10/06/2022	7

Tabla 3.6: Lista de actividades del proyecto

A continuación se expone la lista de riesgos de alto nivel identificados para el proyecto en la tabla 3.7, además de su código, una descripción, el nivel de probabilidad de que ocurra y el impacto que supondría para el proyecto en caso de que ocurriese.

Código	Nombre	Descripción	Probab.	Impacto
R1	Retraso en el diseño de los scripts	Puede que mientras se piensa y se realiza el diseño de los scripts ocurra un retraso por motivos de complicaciones en el diseño del mismo o que ocurra algún evento que impida durante un tiempo que se siga trabajando en el diseño del mismo, además que, a la hora de implementar los scripts aparezcan factores que en la fase de diseño no se tuvieron en cuenta e implique volver a esta fase a hacer cambios.	6	7
R2	Retraso en el diseño de la topología de red	Es posible que en el diseño de la topología de red en la que se basará este proyecto para la realización del taller de laboratorio no cumpla con los requisitos y especificaciones necesarias para la misma y retrase esta actividad para buscar una que sí los cumpla, también puede ocurrir que a la hora de hacer pruebas con esta no sea tan eficiente como en la fase de diseño se pensaba, e implique volver a la fase de diseño a realizar cambios.	4	6
R3	Retraso en la implementación de los scripts	Durante la fase de implementación de los scripts pueden ocurrir retrasos debido a errores durante la fase de implementación que deben solventarse o por cambios que haya que realizar en la fase de diseño	8	9

Código	Nombre	Descripción	Probab.	Impacto
R4	Aumento del tiempo dedicado a la redacción de la memoria	Cabe la posibilidad de que en la redacción de la memoria del proyecto se produzca un retraso debido a cambios que se tenga que realizar en esta o capítulos o partes de estos que deban descartarse o reescribirse enteros.	7	6
R5	Alargamiento del periodo de aprendizaje de las herramientas a utilizar	Puede ocurrir que el tiempo que se planeaba dedicar al periodo de aprendizaje de las herramientas y tecnologías que se van a utilizar en el proyecto se alargue más de la cuenta porque aparezcan tecnologías de última hora o porque se necesite aprender algún aspecto nuevo de una tecnología ya vista, o repasar lo aprendido.	4	2
R6	Errores en el testeo de los scripts	A la hora de probar los scripts en la fase de testeo pueden ocurrir errores de última hora que haya que solventar en la fase de implementación o incluso de diseño	6	8
R7	Pérdida de los archivos relacionados con el proyecto	Existe la posibilidad de que los archivos relacionados con el proyecto como la memoria, los scripts o el JSON de configuración se pierdan	3	10
R8	Errores en la portabilidad de los talleres	A la hora de trasladar todos los archivos del proyecto a otro ordenador para comprobar que de verdad es portable puede que ocurra errores que haya que solventar	4	5
R9	Enfermedad	Puede ocurrir que me ponga enfermo y haya que modificar los plazos debido a la incapacidad de continuar con el proyecto	2	7

Tabla 3.7: Catalogo de riesgos identificados del proyecto

Para finalizar con la planificación del proyecto en la tabla 3.8 se muestra el presupuesto del proyecto, en el cual se muestran los costes estimados del proyecto, en él se incluyen costes como los equipos necesarios para la realización del mismo, las licencias software necesarias y el coste de los recursos humanos del mismo. La planificación del proyecto narra un total de 584 horas según Microsoft Project, para el sueldo del ingeniero informático junior me he referenciado en el TFG de Juan Herruzo Herrero [35] en el que comenta que el sueldo es de 11,53€/hora, siendo el total del coste del trabajador 6733,52€. El coste del portátil hace referencia a una factura privada de Amazon, ya que el coste del portátil no está ya disponible en la web, el resto de costes están referenciados.

Nombre	**Descripción**	**Coste**
Portatil de trabajo	Portatil ASUS TUF Gaming A15 FA506IU FX506IU utilizado para realizar el proyecto	1111,76€
Ingeniero informático junior	Coste de un ingeniero informático para llevar el proyecto a cabo incluyendo Seguridad Social	6733,52€
Licencia Microsft Project[25]	Licencia de Microsoft Project para realizar la planificación del proyecto	30€/mes = 90€
Licencia Astah Professional[51]	Licencia de Astah Professional para realizar las fases de análisis y diseño del proyecto	10€/mes = 30€
Licencia Windows 11[48]	Licencia de Windows 11 para realizar el proyecto	170€
	Total	8135,28€

Tabla 3.8: Tabla de costes del proyecto.

Capítulo 4

Tecnologías utilizadas

En este capítulo se presentan las distintas tecnologías que he necesitado para la realización del proyecto, se presentan tecnologías que me han ayudado en todas las fases del proyecto, desde la redacción de memoria hasta la programación de los scripts y la planificación del proyecto.

Se detallarán cada una de las tecnologías utilizadas, describiendo qué es y para qué ha sido usada para una mayor comprensión del lector.

4.1. GNS3

Figura 4.1: Logo de GNS3

GNS3[49] es la herramienta clave de este proyecto. Consiste en un simulador de redes de computadoras que permite diseñar topologías completas y hacer una simulación de estas completamente funcionales.

Es un software libre desarrollado por toda una comunidad y programado en Pyhton, pero para la simulación de los nodos de la red utiliza varias tecnologías por debajo, es capaz de soportar desde topologías de red básicas y pequeñas hasta las topologías más grandes y complejas. Es compatible con todos los sistemas operativos.

Al principio estaba más enfocado en el uso de tecnologías Cisco, pero más adelante se ha ido mejorando para soportar diferentes tecnologías de diferentes fabricantes. Es utilizado por ingenieros, investigadores, estudiantes, profesionales de la informática... además, es utilizado para diferentes certificaciones en el ámbito de las redes de computadoras, como puede ser el CCNA[52] o los distintos certificados CCNP[54], dos tipos de certificaciones muy famosas en este campo, por lo que es ideal para la enseñanza.

Los dispositivos principales que permite simular y que se pueden añadir a las topologías son: routers, switches (GNS3 incorpora una versión reducida de un switch que incorpora una función básica de VLAN y reenvio de paquetes), EthernetSwitch (Router con un módulo para actuar como un switch, más potente ya que admite más protocolos), VPCS (versión reducida de un PC, solo permite asignar una IP, DNS y hacer ping, consume pocos recursos, unos 2MB de RAM por nodo, pensado para realizar test), máquinas virtuales, distintos tipos de switch y hub.

Esta herramienta permite al usuario generar una topología de red utilizando las tecnologías de distintos fabricantes como Cisco, Aruba, Extreme... y ponerla en funcionamiento debido a las diferentes tecnologías de simulación con las que están relacionadas. Otorga un gran nivel de realismo, debido a que las simulaciones se asemejan mucho a como sería tratar con estos dispositivos en la realidad, además, tiene funcionalidades muy importantes, como la capacidad de conectar esta topología de red a una real y a Internet y la capacidad de conectarse con softwares de virtualización como VirtualBox o VMware[59].

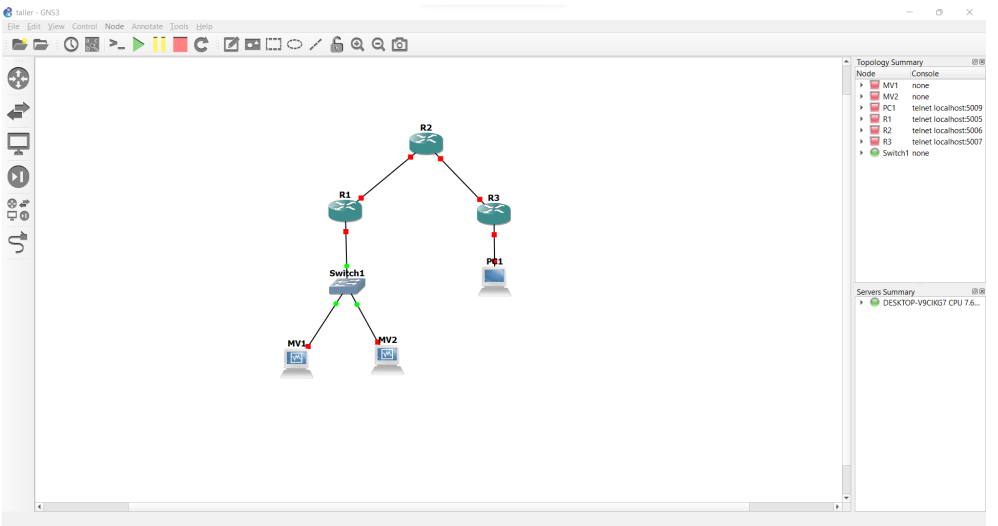

Figura 4.2: Interfaz principal de un proyecto GNS3.

Como todas las herramientas, presentan unas ventajas y unos inconvenientes que se reflejan en las siguientes listas:

Ventajas:

+ Permite simular diferentes tecnologías
+ Es un software libre de código abierto, por ende, es gratuito y en constante mejora por la comunidad
+ Ideal para la enseñanza
+ Permite simular sistemas de forma realista, con las imágenes reales de los dispositivos
+ Permite conexión con software de virtualización como VirtualBox o VMware
+ La topología de red que se configura puede conectarse a una red real o a Internet.
+ Soporta una gran cantidad de tecnologías de diferentes capas.
+ Compatible con todos los sistemas operativos.
+ Permite automatización
+ Permite trabajar y utilizarse con un servidor remoto.
+ Comunidad en constante crecimiento, con foros en los que se responden dudas

Inconvenientes:

– Consume una gran cantidad de recursos
– Es necesario conseguir las imágenes de los dispositivos que se quieren simular.
– Dependiendo del sistema, como por ejemplo en sistemas Windows, pueden crearse conflictos con otras aplicaciones o con el firewall.
– Al depender de otras aplicaciones para la simulación y no hacerla en la misma aplicación puede estar más expuesto a fallos.

Las diferentes tecnologías con las que colabora son las siguientes:

- **GNS3 Server:** Servidor HTTP frente al que funciona GNS3, puede ser interno o externo, en él se alojan los proyectos y todos los archivos. Cuenta con una API REST que permite el uso de diferentes métodos para el tratamiento de los proyectos y sus nodos, es en la que se basarán los scripts de los talleres de laboratorio. Necesita de otra tecnología llamada uBridge, para ofrecer conectividad entre los nodos.
- **Dynamips:** Es un programa creado para emular routers Cisco, necesita para ello la imágen IOS original de un router Cisco. Su propósito inicial era de entrenamiento para aprender a configurar los routers, pero también sirve para probar funcionalidades nuevas y para probar configuraciones. Utiliza una gran cantidad de recursos.

- **Qemu:** Software de virtualización de código abierto, permite virtualizar más sistemas aparte de routers, ya que Qemu permite virtualizar también sistemas operativos enteros como Ubuntu, por lo que gracias a esta tecnología podemos virtualizar todo tipo de dispositivos. Podemos saltarnos la limitación de Dynamips y poder emular routers de distintos fabricantes a Cisco.
- **IOU:** Consiste en una versión de una IOS de un router Cisco compilada específicamente para sistemas Unix, de ahí el nombre, IOS on UNIX.
- **VPCS:** Virtual PC Simulation, programa que simula una versión muy reducida de un PC, pensado para realizar test en la topología ya que sus funcionalidades son muy reducidas, consume muy pocos recursos, sobre unos 2MB de RAM.
- **VirtualBox:** Software de virtualización gratuito de Oracle, permite que una máquina aloje diferentes sistemas operativos por medio de la virtualización. GNS3 permite la conexión directa con este programa, permitiendo que se utilicen máquinas virtuales alojadas en esta plataforma en las topologías de red.
- **VMware:** Software de virtualización de EMC, tiene las mismas funcionalidades que VirtualBox, pero cuenta con diferentes licencias, entre ellas una de pago. GNS3 permite también la conexión directa con esta plataforma, permitiendo que las máquinas virtuales alojadas en esta puedan formar parte de las topologías de red.
- **GNS3 MV:** Es un sistema aislado, en vez de correr Qemu, Dynamips o IOU por separado se ejecutan todos dentro de una máquina virtual, así se ahorra problemas de compatibilidad ya que algunos sistemas operativos no son compatibles con algunos software de virtualización o simulación. Puede ejecutarse bajo cualquier software de virtualización compatible con GNS3, como VirtualBox o VMware.

Como se acaba de ver, GNS3 colabora con diferentes tecnologías, pero en el proyecto no he utilizado todas para implementar la topología de red propuesta, precisamente he utilizado: VirtualBox, GNS3 Server, Dynamips, VPCS y VirtualBox. Sobre Dynamips, he utilizado una imágen IOS de un Router Cisco 2691, debido a que este router es compatible con todos los protocolos y funcionalidades necesarias y consume menos recursos que uno más moderno, cada router está configurado para consumir unos 192MB de RAM.

4.2. Shell de Unix (Dash)

Dash[65] es un intérprete de comandos de Unix (un Shell[16]). Un Shell permite la intermediación entre el usuario y la máquina, permitiendo al usuario ejecutar comandos que hacen referencia a distintas herramientas para que la máquina las ejecute de la forma que el usuario desea.

Esta herramienta es muy potente, ya que permite la automatización de estos comandos a partir de un Shell Script, es decir, un fichero ejecutable con un guión de todos los comandos a usar que posteriormente el intérprete de comandos reconocerá linea a linea y ejecutará los

comandos escritos en él. Permite también el uso de bucles y elementos condicionales en él, por lo que aumenta aun más la potencia ya que permite la programación completa.

Hay una gran cantidad de intérpretes de comandos diferentes como Bash[30], cada Shell tiene sus particularidades y diferencias, pero se basan todos en un estándar, el Bourne Shell[2], que fue el Shell que utilizaban las primeras versiones de Unix, convirtiéndose en un estándar para todos los diferentes Shell.

Lo que interesa para este proyecto es un intérprete de comandos que nos pueda ejecutar de manera eficaz los Shell Scripts de los talleres de laboratorio, además de poder ejecutar los comandos que necesitamos para ellos.

Dash (Debian Almquist shell) es el interprete de comandos por defecto de las máquinas Debian[56] para sesiones no interactivas, es un derivado del Shell Ash. He optado por este intérprete en concreto debido a que, para empezar, GNS3 recomienda su instalación en Windows, MacOS y distribuciones Debian o basadas en Debian de Linux como Ubuntu, es por ello que, al usar estas últimas Dash como intérprete de comandos para sesiones no interactivas, el uso del script se ejecutará bajo ese intérprete de forma predeterminada, además, en caso de querer usarlo en Windows, existe la posibilidad de usarlo bajo una máquina Ubuntu bajo WSL, por lo que es el intérprete que más compatibilidad va a ofrecer con diferentes sistemas. La otra alternativa era utilizar Bash, pero Bash está pensado para sesiones interactivas con el usuario, además de ser más grande y lento que Dash y depende de más librerías, por lo que la opción más adecuada es utilizar Dash.

Una vez conocido el intérprete de comandos que se va a utilizar es momento de conocer todos los comandos que necesitaremos en los scripts:

- **curl:** Herramienta enfocada en la transferencia de archivos bajo distintos protocolos como HTTP, FTP... Está pensado para automatizar la transferencia de archivos, aunque también sirve para realizar acciones web. En este caso lo utilizo para realizar peticiones HTTP a la API REST del servidor HTTP GNS3, también para enviar y recibir ficheros de configuraciones de los nodos. No viene por defecto en todas las máquinas Unix, por lo que es probable que sea necesaria su instalación desde su repositorio.

- **jq:** [23]Herramienta especializada en el gestión de flujo de datos JSON. Es utilizada para el tratamiento de datos de ficheros en formato JSON, ya que permite, enter otras cosas, el uso de tuberías y filtros. Como en este proyecto es muy importante el tratamiento de datos JSON, esta herramienta, junto a curl, han sido las más utilizadas debido a que ha sido necesario un tratamiento de manera constante de un fichero JSON, sobre todo la búsqueda de contenido en dentro del fichero. Al igual que curl, es muy probable que no sea una herramienta instalada por defecto en un sistema Unix, por lo que muy probablemente sea necesaria su instalación desde el repositorio

- **grep:** Es uno de los comándos básicos de Unix, permite al usuario buscar contenido dentro de un fichero. Ha sido utilizado únicamente en el script de corrección para comprobar que la configuración extraida del fichero JSON está contenida en la configuración total del nodo extraida del servidor GNS3.

- **diff:** Herramienta dedicada a la comparación de ficheros, genera las diferencias existentes entre los dos ficheros linea a linea. Ha sido utilizado en este proyecto únicamente para el script de corrección, para comparar el fichero resultante de ejecutar el comando grep con el fichero de configuración extraido del fichero JSON, para comprobar que la información que extrajo el comando grep es exactamente idéntica a la extraida del fichero JSON de configuración y, por ende, la configuración es válida.

- **trap:** Herramienta que sirve para especificar al sistema acciones que debe realizar cuando reciba señales por parte del usuario. Ha sido utilizado en esta ocasión para que, en caso de que el usuario cancele la ejecución del script con Ctrl+C el sistema capture dicha acción y ejecute una rutina de limpieza de los archivos intermedios del script y, ya entonces, interrumpa la ejecución.

- **echo:** Es uno de los comandos básicos de Unix, se encarga de la impresión de texto en pantalla. Como este comando imprime en pantalla el texto que le indique el usuario, en esta ocasión ha sido utilizado para informar al usuario del estado de la ejecución de los scripts, informar del estado de la corrección y para la escritura automática de ficheros de configuración de los nodos, que cuentan con un formato predefinido.

- **seq:** Comando básico de Unix, su función principal consiste en la impresión de una secuencia de números, recibiendo por parte del usuario el que le indica el principio y el final de la secuencia. Ha sido utilizado para marcar el índice de los bucles for, ya que con la ayuda del comando jq y seq se puede marcar el inicio y final del bucle.

- **rm:** Comando básico de Unix, sirve para eliminar ficheros o directorios. En esta ocasión se utiliza para eliminar los ficheros intermedios utilizados durante la ejecución de los scripts.

4.3. JSON

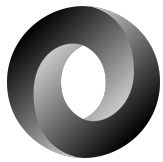

Figura 4.3: Logo de JSON

JavaScript Object Notation[8], es un formato de texto enfocado al intercambio de datos proveniente de la notación de objetos de JavaScript[10], es considerado a día de hoy como alternativa a XML[64], pero es común ver a ambos juntos en las aplicaciones.

La utilidad más común a día de hoy es la de intercambio de datos entre dos máquinas en el entorno web, ya que es muy sencillo de utilizar, ligero y rápido. Un ejemplo de su formato es el siguiente:

```
{
    "nombreObjeto1":"contenido",
    "nombreObjeto2":"contenido"
}
```

Su entendimiento es muy parecido a los mapas clave-valor, en los que hay aparece primero una clave o entidad (que sería nombreObjeto1) y luego un valor asociado (que sería contenido), por lo que al buscar el "nombreObjeto1"se retornaría el contenido de dicho objeto, es decir, se devolvería çontenido". Este contenido no tiene por qué ser solamente una cadena de caracteres, puede ser también un entero, un array o incluso otras entidades.

Un ejemplo de fichero JSON sería el siguiente, en el que podemos comprobar los diferentes campos que nos podemos encontrar en un JSON, este en concreto hace referencia a los datos de un alumno, en el que podemos ver varias entidades como su nombre o edad, también podemos ver un array de asignaturas que el alumno está cursando actualmente y un conjunto de entidades en el historial de cursos pasados en el que nos encontramos diferentes cursos en los que el alumno puede haber promocionado o no.

```
{
    "nombre":"Pedro",
    "edad":12,
    "curso":"Primero",
    "añoNacimiento":2000,
    "asignaturas": [
        "matemáticas",
        "lengua"
    ],
    "historialCursosPasados": {
        "2005-2006":"Promociona",
        "2006-2007":"Promociona"
    }
}
```

JSON es muy utilizado a lo largo del proyecto, principalmente se usa como formato para el archivo de configuración que ambos scripts leerán para conocer la información de la red a crear y corregir, también es el formato seguido por la API REST de GNS3 para transmitir información, por lo que a la hora de intercambiar información con el servidor HTTP de GNS3 es necesario utilizar este formato ya que, por ejemplo, si queremos crear un nodo o un enlace en una topología de red la API REST de GNS3 solicita la información en dicho formato ya que así es como la guarda en el servidor, al igual que si queremos consultar información sobre los distintos nodos el servidor nos devolverá dicha información en formato JSON. La configuración de alguno de los nodos, como los switches, también están escritas en formato JSON, ya que el servidor la guarda dentro del mismo nodo, que es un fichero JSON.

4.4. VirtualBox

Figura 4.4: Logo de VirtualBox.

VirtualBox[47] es una herramienta de virtualización de Oracle, permite al usuario crear máquinas virtuales, en las que se puede instalar un sistema operativo invitado dentro de un sistema operativo anfitrión que es la máquina local en la que está instalada VirtualBox, esta máquina anfitriona tiene que ser obligatoriamente una máquina de 64 bits.

Ofrece una gran cantidad de funcionalidades además de la principal descrita en el párrafo anterior, como una facilidad muy grande en la portabilidad de las máquinas virtuales, clonado de máquinas, utilización de archivos ISO como unidad de DVD, escritorio remoto para controlar las máquinas virtuales desde otra máquina remota y la creación de máquinas virtuales con sistemas de 32 bits en máquinas de 64 bits entre muchas otras.

GNS3 ofrece como funcionalidad la conexión con VirtualBox para agregar máquinas virtuales a la topología de red y esta ha sido su función principal para este proyecto. En esta ocasión para la topología de red diseñada para este proyecto se utilizan dos máquinas virtuales Ubuntu 10.04 de 32 bits, se ha seleccionado este sistema en concreto debido a los pocos recursos que requiere y porque ofrece todas las funcionalidades que se necesitan para esta topología. Una de las máquinas virtuales contiene un servidor web y esa es su función principal, servir para que las máquinas de la red accedan a dicho servidor web, para eso necesitarán utilizar el protocolo DNS y HTTP, la otra máquina virtual sirve para utilizar el protocolo DCHP y para servir de acceso al servidor web de la otra máquina.

Ubuntu es una distribución de Linux basada en Debian, enfocado a la facilidad de uso por parte del usuario promedio, ya que Linux, por lo general, no resulta muy sencillo de usar por parte del usuario promedio acostumbrado a otros tipos de sistemas como Windows o MacOS. Se ha seleccionado este sistema operativo para las máquinas virtuales por este mismo motivo, por la facilidad de uso, así podemos asegurar que más personas puedan usar las máquinas virtuales sin problemas aunque no tengan conocimientos de sistemas operativos. También se ha seleccionado una versión bastante anterior y obsoleta del mismo, ya que no supone un problema de seguridad debido a que la red no se conectará a internet, por lo que no expone a ataques ajenos y además ofrece todas las funcionalidades que necesitamos y ahorramos muchos recursos que las versiones más modernas necesitan.

Se ha optado por usar esta herramienta y no VMware, que también es compatible y cumple la misma función, debido a que VirtualBox tiene un uso más estandarizado, es fácil de configurar y es gratuito. Cabe mencionar además que las máquinas están completamente

configuradas y listas para su uso para que no sea necesario que los usuarios tengan conocimientos de sistemas operativos para poder realizar el taller, ya que el objetivo del mismo es mejorar el conocimiento de redes a los usuarios, no de sistemas operativos.

4.5. Overleaf

Figura 4.5: Logo de Overleaf.

Overleaf[38] es un editor de LaTeX basado en la nube, está pensado para la redacción de documentos técnicos y tesis en LaTeX. Permite y facilita la creación de proyectos para facilitar al máximo la redacción de documentos al usuario. Además, como se ha mencionado, está basado en la nube, por lo que se accede mediante el navegador y todo se almacena en la nube, por lo que permite el fácil acceso a los proyectos desde cualquier lugar.

LaTeX es un sistema de composición de textos de código abierto, su uso es la de redacción de documentos mediante comandos y existe con la premisa de que el usuario se centre en la redacción de los contenidos del documento sin preocuparse excesivamente por el formato de este. Su uso se compone de dos etapas: primero la redacción de un fichero .tex con el texto que se quiere mostrar en el documento más los comandos necesarios para el formato del mismo y una segunda etapa que consiste en la compilación de dicho fichero .tex, teniendo como resultado final un documento final formateado.

Primero, para justificar el uso de Overleaf hay que justificar el uso de LaTeX, se ha optado por el uso de LaTeX debido al tiempo que ahorraba en la creación del formato de la memoria en otros editores de texto como Word, además de la facilidad que ofrece a la hora de formatear el texto. El uso de Overleaf se debe a que es un editor de texto de LaTeX muy completo y en la nube, asegurándome en todo momento que, si pasa cualquier incidente en mi máquina local, no voy a perder todo lo escrito de la memoria, ya que es la parte que más tiempo conlleva de todo el proyecto, también ha sido elegido debido a que es gratuito y sus funcionalidades son muy completas.

La función de Overleaf en este proyecto es la de la redacción completa de la memoria del proyecto, toda la memoria de este proyecto ha sido redactada en Overleaf y, por ende, en LaTeX.

4.6. Microsoft Project

Figura 4.6: Logo de Microsoft Project

Microsoft Project[42] es un software de administación y gestión de proyectos desarrollado por Microsoft y siendo parte de la suite Office. Es utilizado como ayuda para la planificación y gestión de proyectos para las empresas. El inconveniente principal de este software es que no es libre, sino que forma parte de la suite de Microsoft Office que es de pago y requiere de licencia.

Project ofrece una gran cantidad de herramientas para el gestor o director de proyecto de la empresa, contando con distintas funcionalidades como la planificación y definición de las tareas y sus plazos, gestión de los recursos del proyecto (incluidos los horarios del personal), gestión de costes y presupuesto del proyecto, gestión de riesgos entre muchas otras. También sirve para gestionar proyectos que se basan en distintas metodologías como puede ser Scrum y para generar diversos tipos de esquemas.

He optado por este software de gestión de proyectos debido a que ofrece las funcionalidades necesarias para la planificación de este proyecto y debido a que ya previamente manejaba el uso de este software por asignaturas de la universidad en la que se ha utilizado, por lo que no necesitaría utilizar tiempo del plazo del proyecto para aprender a utilizar esta herramienta.

En este proyecto Microsoft Project ha sido utilizado para la gestión de las actividades, es decir, las dependencias entre estas y sus plazos y para expresar de un vistazo cómo se ha desarrollado y gestionado el proyecto y sus plazos en todo este tiempo.

4.7. WSL

WSL[44] es Windows Subsystem For Linux, consiste en una capa de compatibilidad de Microsoft Windows con los sistemas Linux, permite ejecutar el kernel de una distribución de Linux nativamente en el sistema, llevando eso consigo la posibilidad de ejecutar comandos propios de Unix y aplicaciones de la propia distribución sin necesidad de máquinas virtuales intermediarias o la instalación del sistema en una partición separada de Windows, ya que este subsistema aprovecha todos los recursos de la máquina al ejecutarse de manera nativa a diferencia de si se ejecutara en una máquina virtual. Apareció por primera vez en Windows 10 y se mejoró mucho en Windows 11, permitiendo el uso de aplicaciones de Linux gráficas y con sonido y facilitando mucho el acceso, instalación y uso de este subsistema.

Esto ofrece una interfaz de comandos propia de un kernel de Linux que, además de permitir ejecutar los propios comandos del núcleo como cd, ls, rm... permite una función muy interesante, la posibilidad de ejecutar un espacio de usuario GNU[33], lo que conlleva la instalación de una distribución de Linux en este espacio, como podrían ser Ubuntu, Debian o Kali Linux. Este subsistema cuenta con su propio espacio de almacenamiento dentro del disco duro de la máquina Windows, permitiendo además el intercambio de ficheros entre Windows y este subsistema. Cuenta con Bash por defecto como Shell principal de interacción con el usuario.

El sistema operativo que yo he utilizado para este proyecto es Windows 11 versión 21H2, GNS3 es compatible perfectamente con este sistema, por lo que para desarrollar el Shell Script no necesitaba la instalación completa de una distribución Linux o una máquina virtual, es por ello que opté por desarrollar y ejecutar los scripts desde WSL, precisamente con una distribución de Ubuntu instalada, ya que esto además me permitía el acceso al servidor local GSN3 debido a que ambos están alojados en la misma máquina y el acceso a todos los repositorios que necesitase sin instalar una distribución completa.

Figura 4.7: Versión utilizada en el proyecto de Microsoft Windows

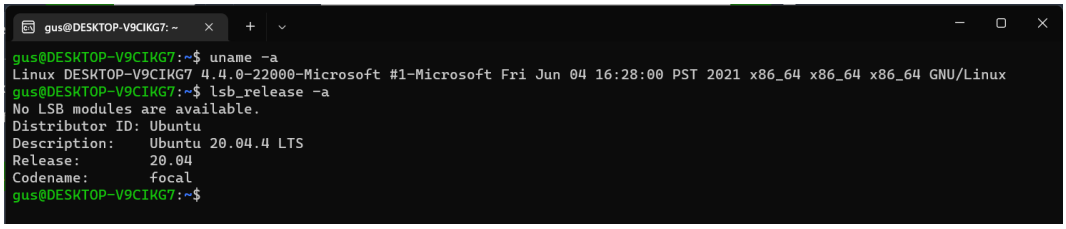

Figura 4.8: Versión de núcleo de Linux utilizada en el WSL y la versión de la distribución Ubuntu instalada.

4.8. Git y GitHub

Figura 4.9: Logo de Git

Figura 4.10: Logo de GitHub

Git[62] es un software de control de versiones desarrollado por Linus Torvalds. Un sistema de control de versiones es un software que permite controlar los distintos cambios que se hacen en un proyecto software permitiendo, en caso de errores en una nueva versión, volver al código de versiones anteriores, crear versiones alternativas para probar funcionalidades sin afectar a la versión principal, entre otras cosas.

Se basa en el concepto de repositorio, un repositorio es donde se alojan todos los ficheros del proyecto y en el que se mantiene el control de versiones, este repositorio puede encontrarse en la máquina local o en una máquina remota bajo otra aplicación.

Una de las aplicaciones que permite albergar repositorios de Git de manera remota en la nube es GitHub[34]. Es una plataforma que permite alojar repositorios Git, además de permitir un desarrollo software cooperativo con otros usuarios de la plataforma. Es la plataforma más importante para el desarrollo de programas de código abierto, ya que cuenta con diferentes características que ayudan a los desarrolladores a crear una comunidad en torno al programa, permitiendo a otros usuarios debatir, reportar errores en el código o incluso subir sus propias modificaciones del código para que sean aceptadas por los desarrolladores originales de este, permitiendo que todo el mundo aporte su granito de arena al correcto desempeño del programa. Cabe mencionar que también los repositorios alojados en esta plataforma no tienen por qué ser todos de código abierto o de varios desarrolladores, también puede usarse para proyectos personales y privados.

En este proyecto el uso de Git y GitHub ha cumplido la función de tener un backup privado del proyecto personal, sin compartirlo con nadie ni ser público, se ha usado en la parte del desarrollo de los Shell Scripts. Ha sido usado por si, en caso de que ocurra algún incidente con mi máquina local, tener un backup en la nube de todo el código de los scripts y

del fichero JSON, además de que por si ocurre algún error en alguna nueva implementación del código tener una versión anterior de respaldo.

4.9. Notepad++

Figura 4.11: Logo de Notepad++

Notepad++[36] es un editor de texto gratuito de código libre, permite escribir tanto texto plano como programar en diferentes lenguajes de programación. Además es compatible con una gran cantidad de lenguajes de programación.

Es muy similar al Bloc de Notas de Microsoft Windows, pero con muchas mejoras ya que, como se ha mencionado, permite además programar en diferentes lenguajes de programación como si de un IDE se tratase, con colores en la sintaxis para ayudar al usuario en la lectura del código o el resaltado de las llaves, corchetes y paréntesis para ayudar al usuario a saber dónde empieza y dónde acaba. También cuenta con soporte para numerosas extensiones que ofrecen funcionalidades nuevas a los usuarios y pestañas para organizar diferentes documentos en una sola ventana, permitiendo incluso tener dos pestañas abiertas a la vez con pantalla dividida.

Ha sido usado en este proyecto para redactar los Shell Scripts y el fichero JSON de configuración. He optado por utilizar Notepad++ debido a que prefería no usar Vim, ya que me ralentizaría demasiado porque no controlo su uso de manera eficaz y, como desde Windows puedo acceder a los ficheros del espacio de almacenamiento del WSL de manera sencilla gracias a la mejora que implementó Windows 11, me parecía una buena herramienta para desarrollar los Shell Scripts. Además, este programa lo he usado en asignaturas anteriores, por lo que controlo bien su uso y cumple para lo que necesito.

4.10. Microsoft Teams

Figura 4.12: Logo de Microsoft Teams

Pertenece, al igual que Microsoft Project, al paquete de Microsoft Office, es una herramienta de comunicación y colaboración con empresas, aunque también se usa en entornos educativos para la comunicación del profesor con los alumnos. Su uso se ha extendido durante los últimos años de manera exponencial debido a la pandemia del COVID-19

Microsoft Teams[43] cuenta con numerosas funcionalidades para la colaboración entre equipos, permite crear un equipo de colaboración en el que los miembros pueden compartir contenido en forma de publicaciones, permite reuniones entre los miembros del equipo, chats entre ellos, trabajo simultaneo y colaborativo en documentos de Office, configurar un calendario de empresa para apuntar reuniones y eventos... También en el entorno educativo permite a los profesores mandar tareas a los alumnos y luego corregirlas, además de permitir alojarles los contenidos de la asignatura en la plataforma para que puedan consultarlos y grabar las clases por si un alumno no puede asistir.

En este proyecto se ha usado Microsoft Teams como método de comunicación con el tutor del TFG, el tutor ha creado un equipo con todos los alumnos cuyos TFG también tutela, también ha creado un canal privado con cada uno de los alumnos, así si tiene un mensaje para todos nosotros lo envía en el canal compartido y para las comunicaciones privadas utiliza el canal privado, en dicho canal he compartido mis avances del TFG y las dudas que me surgían. También he utilizado Teams para las videoconferencias con el tutor que realizábamos frecuentemente, para poder hablar con él de manera más directa y poder enseñarle avances compartiendo la pantalla.

4.11. draw.io

Figura 4.13: Logo de draw.io

También conocido como diagrams.net[1], es una herramienta web libre de código abierto de dibujo gráfico. Es utilizada para realizar distintos diagramas, sobre todo en el ámbito informático como diagramas de flujo, de redes y de UML[22]

Ofrece distintas funcionalidades como el almacenamiento en la nube de los diagramas, ya que cuenta con conexión con distintas plataformas como Google Drive o GitHub, también permite la exportación de dichos diagramas en muchos formatos como PDF, PNG o JPEG.

Esta herramienta ha sido usada para realizar el esquema del funcionamiento de los scripts de la figura 7.1 y para representar el diseño arquitectónico en las figuras 6.4, 6.5 y 6.6. He optado por esta herramienta debido a que ofrece una gran facilidad de uso y de acceso bastante elevada debido a que es tan fácil como entrar a la web y ponerse a diseñar, también ha influido en su elección la gran galería de iconos e imágenes que posee para los esquemas y la gran velocidad a la que permite diseñar.

4.12. Astah Professional

Figura 4.14: Logo de Astah Professional

Astah[50] es una herramienta profesional de modelado UML utilizado para el proceso de modelado de sistemas software, normalmente para el diseño del mismo.

Permite realizar distintos diagramas, entre los que se encuentra el diagrama de clases utilizado para el modelo de dominio, diagramas de secuencia, diagramas de despliegue y

diagramas de casos de uso, entre muchos otros. Se organiza en base a proyectos y paquetes, los diagramas están dentro de paquetes junto con todos sus elementos como las clases, nodos, actores...

He usado esta herramienta para crear todos los diagramas y modelos de las fases de análisis y diseño salvo para el diagrama de despliegue, ya que en esta herramienta no funciona muy bien, no permitiendo usar artefactos. El motivo de su uso es que es mucho más sencilla de usar que otras herramientas que hemos usado a lo largo de la carrera como Visual Paradigm.

4.13. Visual Paradigm

Figura 4.15: Logo de Visual Paradigm

Visual Paradigm[58] es otra herramienta de modelación UML orientada a realizar diagramas y modelos para el desarrollo software, normalmente su uso es el mismo que el de Astah Professionalm realizar modelados y diagramas para las fases de análsis y diseño del desarrollo de un sistema software.

Visual Paradigm permite numerosas funcionalidades, pero las más comunes son las de realizar diagramas y modelos, entre los que permite realizar están los diagramas de clases, diagramas de secuencia, diagramas de despliegue , diagrama de paquetes, diagramas de actividad y diagramas de casos de uso, a la vez que definir sus especificaciones. Tiene una funcionalidad muy interesante y es que permite realizar ingeniería inversa para realizar diagramas y crear un proyecto a partir de un código de un software ya hecho.

Esta herramienta ha sido utilizada exclusivamente para realizar el diagrama de despliegue del sistema en la fase de diseño, debido a que Astah no era muy adecuado para esta tarea, no permitiendo el uso de artefactos en sus diagramas de despliegue, solo nodos y componentes. No utilicé esta herramienta para realizar el resto de diagramas debido a que siempre me ha parecido mucho menos manejable y usable que Astah, además de que en asignaturas anteriores más recientes hemos utilizado Astah, lo que me ahorra tiempos de aprendizaje de acordarme de cómo se utiliza Visual Paradigm, ahorrando tiempo en las fases de análisis y diseño.

Capítulo 5

Análisis

En este capítulo comienza ya el proceso de desarrollo de los dos scripts resultado del proyecto: el script de creación de los talleres de laboratorio y el script de corrección de los talleres de laboratorio.

La fase de análisis de un desarrollo software presenta el problema que se quiere hacer frente en el proyecto mediante diferentes diagramas y recursos como los requisitos que se deben cumplir, el diagrama de casos de uso, el modelo de dominio y el diagrama de secuencia de los casos de uso. A continuación, se expresa la fase de análisis realizada para este proyecto.

5.1. Diagrama y descripción de casos de uso

Además de los requisitos, en esta fase de análisis se presenta el diagrama de casos de uso, en él se exponen las diferentes formas que existe de utilizar este sistema software que, en este caso, son los dos scripts.

En la figura 5.1 se expone dicho diagrama de casos de uso y, a continuación de este, se detalla una explicación de dichos casos de uso.

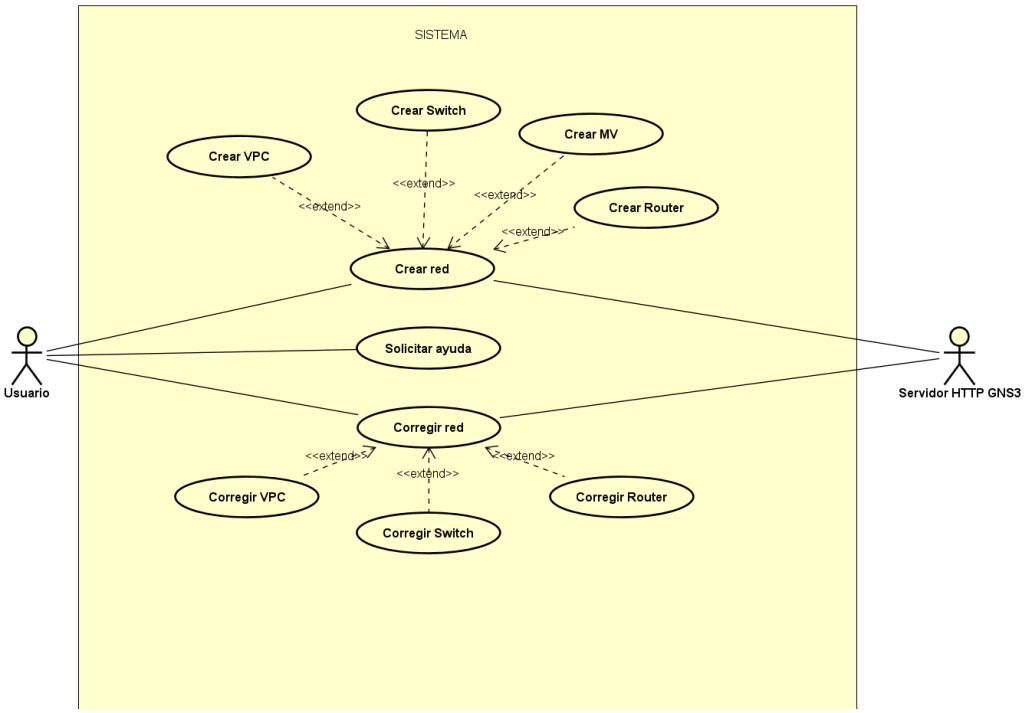

Figura 5.1: Diagrama de casos de uso

Este diagrama de casos de uso cuenta con un sistema central y dos actores, uno primario y uno secundario. El sistema central agrupa ambos scripts, el de creación de los talleres de laboratorio y el de corrección.

Los dos actores que nos encontramos son los siguientes:

- **Usuario:** Es el actor primario y principal de este diagrama, hace referencia al usuario que quiere realizar los talleres de laboratorio, ya sea crear la red o corregir una red que ha creado previamente con el script de creación, por lo que es el único actor primario que tiene cabida en este sistema y en este diagrama.
- **Servidor HTTP GNS3:** Es el único actor secundario del diagrama y no es un actor humano, corresponde al servidor HTTP de GNS3 que se aloja en la máquina local y gestiona todos los proyectos GNS3 y, con ello, los nodos y la configuración de estos. Interviene en los casos de uso de creación y de corrección de la red, ya que para crear el proyecto, los nodos y configurarlos es necesario realizar peticiones a dicho servidor y, para corregir la red creada, es necesario también realizar distintas peticiones al servidor.

Y los distintos casos de uso que nos encontramos en el diagrama son los siguientes:

- **Crear red:** Hace referencia directa al script de creación, este caso de uso es el que se utiliza cuando el usuario quiere ejecutar este script, y por ende, crear un taller de

laboratorio. Tiene como actor secundario al servidor HTTP GNS3 debido a que para crear la red tiene que mandar peticiones constantemente a este servidor. Se divide en otros cuatro casos de uso para una mayor comprensión, cada caso de uso corresponde a la creación de los distintos tipos de nodos de la red.

- **Solicitar ayuda:** Este caso de uso hace referencia a cuando el usuario ejecuta la opción -h o –help en ambos scripts, esto provoca que el script muestre un mensaje de ayuda al usuario con información de qué es el script, para qué sirve y los argumentos que necesita para su uso. No tiene como actor secundario el servidor HTTP GNS3, ya que nunca llega a ejecutarse el script realmente, solo muestra la información al usuario, por lo que no accede al servidor en ningún momento.
- **Corregir red:** Hace referencia directa al script de corrección de los talleres de laboratorio, este caso de uso corresponde a cuando el usuario ejecuta este script queriendo corregir un taller de laboratorio creado previamente con el script de creación. También cuenta con el servidor HTTP GNS3 como actor secundario debido a que hace peticiones constantemente a este servidor para llevar a cabo su tarea. Se divide en otros tres casos de uso para una mayor comprensión, cada caso de uso corresponde a la corrección de los distintos tipos de nodos de la red.

A continuación, en las tablas de la 5.1 a la 5.10, se presenta una descripción de los casos de uso más detallada, en la que se muestran los siguientes aspectos:

- **Caso de uso:** Nombre del caso de uso
- **Actor:** Actores involucrados en el caso de uso
- **Precondicion:** Condiciones que se deben cumplir antes de realizar el caso de uso
- **Escenario:** Escenario principal del caso de uso
- **Flujo alternativo:** Posibles caminos alternativos que puede tomar el escenario principal del caso de uso.
- **Postcondición:** Condición que cumplirá el caso de uso al finalizar el escenario.

Caso de uso	Crear red
Actor	Usuario y Servidor HTTP GNS3
Precondición	Ninguna
Escenario	1- El usuario indica al sistema que desea crear una red con un nombre punto de extensión: Crear Switch (Si existen switches) punto de extensión: Crear Router (Si existen routers) punto de extensión: Crear MV (Si existen MV) punto de extensión: Crear VPC (Si existen VPC)
Flujo alternativo	1-No existe algún elemento de la topología, como switches, routers, alguna parte de la configuración de los routers o VPCS. El sistema ignorará dicha parte del escenario y pasará a la siguiente.
Postcondición	El usuario puede acceder a la red que ha creado desde la aplicación GNS3 y dicha red estará configurada de una forma u otra dependiendo de lo que quiera el usuario

Tabla 5.1: Descripción del caso de uso Crear Red

Caso de uso	Crear Switch
Actor	Usuario y Servidor HTTP GNS3
Precondición	Ninguna
Escenario	1. El usuario indica al sistema que quiere crear un switch con un nombre y unas coordenadas 2. El sistema crea el switch con el nombre y las coordenadas indicadas por el usuario 3. El sistema envía al servidor el switch creado. 4. El usuario indica al sistema que quiere conectar el switch a unos nodos 5. El sistema conecta el switch a cada nodo 6. El sistema envía la información actualizada del switch al servidor
Flujo alternativo	1- El usuario desea configurar el switch 6. El sistema envía la información actualizada del switch al servidor 7. El usuario indica al sistema que quiere configurar el switch con unas VLAN 8. El sistema configura las VLAN en el switch 9. El sistema envía la información actualizada del switch al servidor.
Postcondición	El usuario ha creado un switch en la red, si lo desea, estará configurado.

Tabla 5.2: Descripción del caso de uso Crear Switch

Caso de uso	Crear Router
Actor	Usuario y Servidor HTTP GNS3
Precondición	Ninguna
Escenario	1. El usuario indica al sistema que quiere crear un router con un nombre y unas coordenadas 2. El sistema crea el router con el nombre y las coordenadas indicadas por el usuario 3. El sistema envía al servidor el router creado. 4. El usuario indica al sistema que quiere conectar el router a unos nodos 5. El sistema conecta el router a cada nodo 6. El sistema envía la información actualizada del router al servidor
Flujo alternativo	1- El usuario desea configurar el router 6. El sistema envía la información actualizada del router al servidor 7. El usuario indica al sistema que quiere configurar el router con una configuración DHCP, DNS, IP de interfaces y RIP. 8. El sistema configura el DHCP, DNS, IP de las interfaces y el RIP en el router. 9. El sistema envía la información actualizada del router al servidor.
Postcondición	El usuario ha creado un router en la red, si lo desea, estará configurado.

Tabla 5.3: Descripción del caso de uso Crear Router

Caso de uso	Crear MV
Actor	Usuario y Servidor HTTP GNS3
Precondición	Ninguna
Escenario	1. El usuario indica al sistema que quiere crear una máquina virtual con un nombre y unas coordenadas 2. El sistema crea la máquina virtual con el nombre y las coordenadas indicadas por el usuario 3. El sistema envía al servidor la máquina virtual creada.
Postcondición	El usuario ha creado una máquina virtual en la red.

Tabla 5.4: Descripción del caso de uso Crear MV

Caso de uso	Crear VPC
Actor	Usuario y Servidor HTTP GNS3
Precondición	Ninguna
Escenario	1. El usuario indica al sistema que quiere crear un VPC con un nombre y unas coordenadas 2. El sistema crea el VPC con el nombre y las coordenadas indicadas por el usuario 3. El sistema envía al servidor el VPC creado.
Flujo alternativo	1- El usuario desea configurar el VPC 6. El sistema envía la información actualizada del VPC al servidor 7. El usuario indica al sistema que quiere configurar el VPC con una IP 8. El sistema configura la IP en el VPC 9. El sistema envía la información actualizada del VPC al servidor.
Postcondición	El usuario ha creado un VPC en la red, si lo desea, estará configurado.

Tabla 5.5: Descripción del caso de uso Crear VPC

Caso de uso	Solicitar ayuda
Actor	Usuario
Precondición	Ninguna
Escenario	1. El usuario solicita ayuda sobre el funcionamiento del taller 2. El sistema muestra al usuario una breve descripción de ayuda sobre el taller.
Flujo alternativo	Ninguno.
Postcondición	El usuario puede leer una breve ayuda sobre el taller.

Tabla 5.6: Descripción del caso de uso Solicitar Ayuda.

Caso de uso	Corregir red.
Actor	Usuario y Servidor HTTP GNS3
Precondición	El usuario debe haber creado previamente la red que quiere corregir y conocer su ID
Escenario	1. El usuario solicita al sistema corregir una red punto de extensión: Corregir Switch (Si existen switches) punto de extensión: Corregir Router (Si existen routers) punto de extensión: Corregir VPC (Si existen VPC)
Flujo alternativo	1-No existe algún elemento de la topología, como switches, routers, alguna parte de la configuración de los routers o VPCS. El sistema ignorará esa parte de la ejecución y pasará a la siguiente.
Postcondición	El sistema informará al usuario de que partes de la red están bien configuradas y que partes no lo están.

Tabla 5.7: Descripción del caso de uso Corregir red.

Caso de uso	Corregir Switch.
Actor	Usuario y Servidor HTTP GNS3
Precondición	El usuario debe haber creado previamente la red que quiere corregir y conocer su ID
Escenario	1. El usuario indica al sistema que quiere corregir un switch. 2. El sistema extrae la configuración de dicho switch (VLAN). 3. El sistema extrae la configuración del switch del servidor. 4. El sistema compara ambas configuraciones 5. El sistema informa al usuario del resultado.
Flujo alternativo	Ninguno.
Postcondición	El usuario conocerá si la configuración del switch es la correcta.

Tabla 5.8: Descripción del caso de uso Corregir Switch

Caso de uso	Corregir Router.
Actor	Usuario y Servidor HTTP GNS3
Precondición	El usuario debe haber creado previamente la red que quiere corregir y conocer su ID
Escenario	1. El usuario indica al sistema que quiere corregir un router. 2. El sistema extrae la configuración de dicho router (DHCP, DNS, IP de las interfaces y RIP). 3. El sistema extrae la configuración del router del servidor. 4. El sistema compara ambas configuraciones 6. El sistema informa al usuario del resultado.
Flujo alternativo	Ninguno.
Postcondición	El usuario conocerá si la configuración del router es la correcta.

Tabla 5.9: Descripción del caso de uso Corregir Router

Caso de uso	Corregir VPC.
Actor	Usuario y Servidor HTTP GNS3
Precondición	El usuario debe haber creado previamente la red que quiere corregir y conocer su ID
Escenario	1. El usuario indica al sistema que quiere corregir un VPC. 2. El sistema extrae la configuración de dicho VPC (IP). 3. El sistema extrae la configuración del VPC del servidor. 4. El sistema compara ambas configuraciones 6. El sistema informa al usuario del resultado.
Flujo alternativo	Ninguno.
Postcondición	El usuario conocerá si la configuración del VPC es la correcta.

Tabla 5.10: Descripción del caso de uso Corregir VPC.

5.2. Modelo de dominio

El modelo de dominio es un diagrama de clases en la fase de análisis que sirve para definir los conceptos clave del dominio del problema, es decir, un diagrama conceptual con todos los temas del problema para comprender de un vistazo todos los elementos que interactúan en él y cómo se relacionan entre sí.

En nuestro problema los elementos que nos conciernen en el dominio solo son los posibles nodos que puede tener nuestra red, ya que en todo momentos estos son los elementos sobre los que trabajaremos. En la figura 5.2 se muestra el modelo de dominio final de la fase de análisis del proyecto.

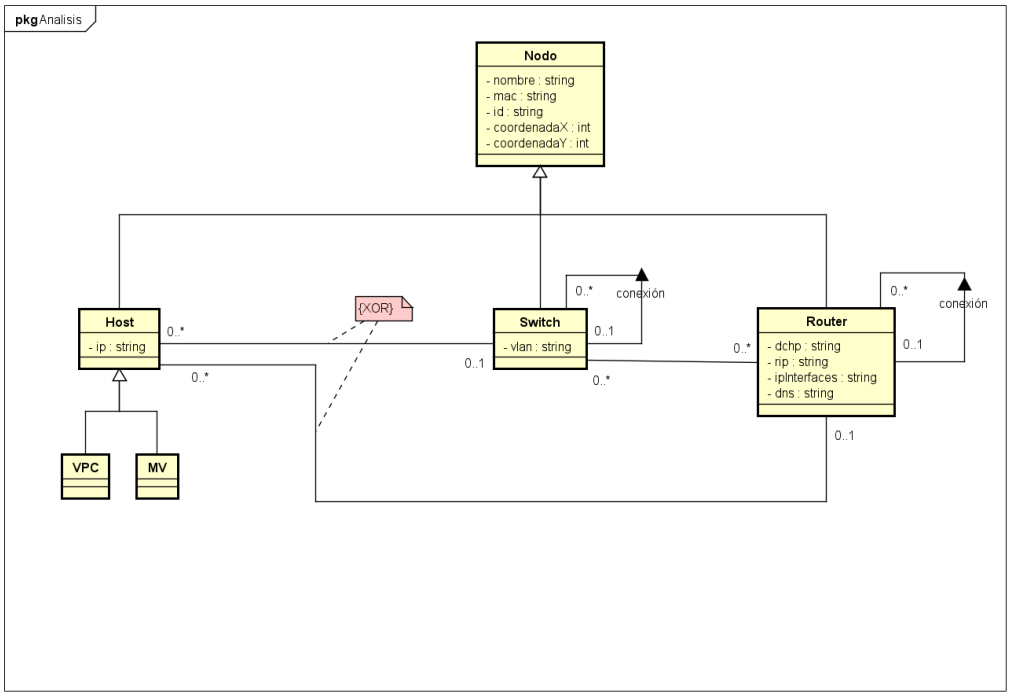

Figura 5.2: Modelo de dominio

Como hemos dicho antes, el modelo de dominio agrupa los diferentes nodos que podemos encontrarnos en las topologías de red, todos los elementos de la topología son nodos y, como nodos, comparten diferentes propiedades como la posesión de un nombre, una dirección MAC, un id, una coordenada X y una coordenada Y que los situará en un lugar u otro en la topología.

Un nodo es la mayor generalización posible del dominio del problema, pero una vez vamos especializando más nos encontramos con tres grandes clases:

- **Host:** Un host es un punto de inicio o de final en una red. Como el modelo de red de la actualidad está pensado para seguir un modelo cliente-servidor[4], las redes de este taller de laboratorio seguirán este modelo, por lo que el comienzo de la red siempre será un ordenador y el final de la red también será siempre un ordenador. Los host que admite GNS3 y por tanto admite nuestro dominio son VPC y máquinas virtuales y todos comparten una propiedad en común que es la asignación de una ip. Un host nunca puede estar conectado a otro host, solamente puede estar conectado a un switch o a un router, nunca puede ser de otra manera, es por ello que las relaciones de esta clase forman una XOR[6], porque solo puede relacionarse o con un switch o con un router o con ninguno.
- **Switch:** Un switch es un nodo intermedio en la red, por lo cual no entra en la categoría de host. Un switch es un elemento de capa dos por lo cual no tiene asignada una IP, que es un elemento de capa tres, es por ello que la clase nodo no tiene IP en sus atributos. Cuenta con muchas interfaces, por lo que puede relacionarse con múltiples nodos de distinto tipo, puede conectarse a otro switch, a uno o más router o host. Además, cuenta con la particularidad de que puede configurarse VLAN
- **Router:** Un router es otro tipo de nodo intermedio en la red, en este caso de capa tres por lo que si que cuenta con una IP. Cuenta con diferentes particularidades como el soporte de distintos protocolos como DHCP, RIP o DNS, por lo que estará presente en sus atributos. Un router, al igual que un switch, cuenta con diferentes interfaces en las que pueden conectarse host, switches u otros routers.

5.3. BCE

BCE (Boundary Controller Entity)[22] es un método para encontrar clases de análisis y relacionado con los casos de uso, ya que define el comportamiento de estos dividiendo las responsabilidades en tres tipos de clases:

- **Frontera (Boundary):** Es la clase encargada de la interacción con el actor, transmite las acciones del actor al controlador.
- **Controlador:** Es la clase más importante ya que es la que realiza, organiza y gestiona el caso de uso, recibe información de la frontera y lee y almacena información en la clases entidad. Suele haber uno por caso de uso
- **Entidad:** Almacena información persistente, su función es únicamente esa, la de almacenar información. Las otras clases leerán y escribirán en esta y suelen hacer referencia a las clases identificadas en el modelo de dominio.

He optado por utilizar este patrón debido a que es muy útil para, en los diagramas de secuencia de la fase de análisis, mostrar la secuencia del caso de uso de una forma simple, al conocer de qué se encarga cada tipo de clase. Además, este es el patrón que mejor controlo ya que es el que he utilizado en asignaturas de cursos anteriores.

El diagrama de clases quedaría de la forma que se muestra en la figura 5.3, siendo todas las clases del modelo de dominio entidades que almacenan datos las cuales sus relaciones y sus herencias se mantienen de la misma forma, se añaden controladores, uno por cada caso de uso, aunque los caso de uso extendidos comparten controlador con el caso de uso al que extienden para mayor claridad y simpleza, y se añaden dos fronteras, una para cada actor. Los diagramas de secuencia del subcapítulo siguiente se expresarán utilizando este mismo patrón.

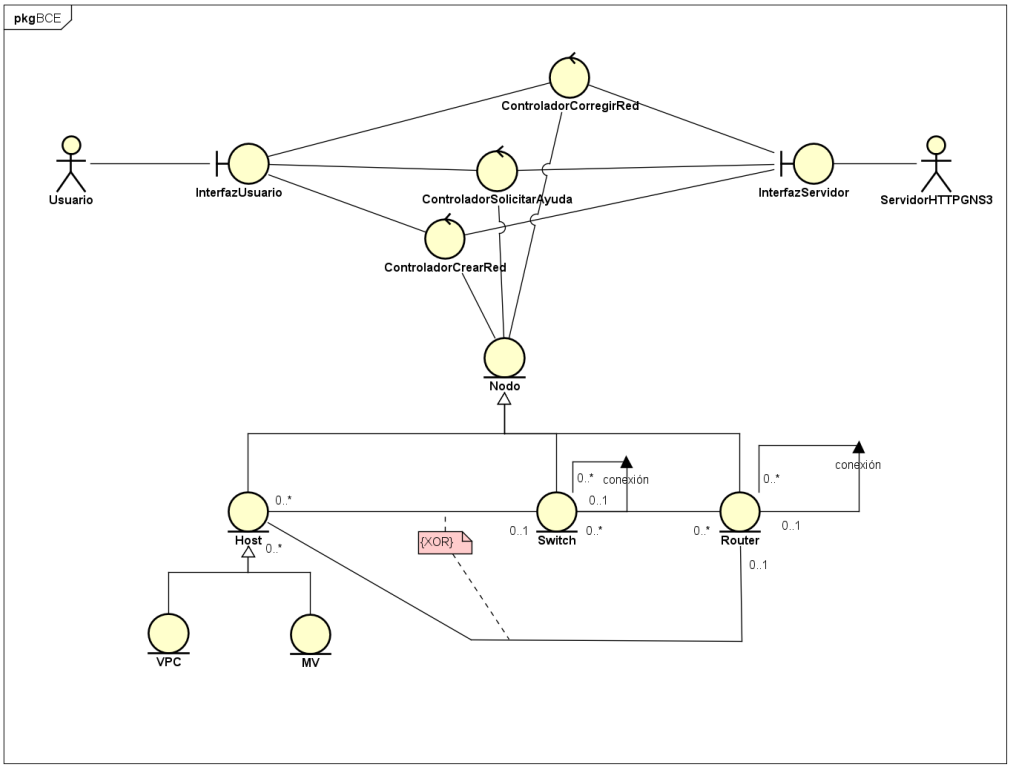

Figura 5.3: Diagrama de clases aplicando BCE.

5.4. Diagrama de secuencia

Un diagrama de secuencia muestra la interacción entre clases, es decir, cómo se relacionan entre sí. Aplico aquí también el patrón arquitectónico BCE, se mostrará como los distintos tipos de clases se relacionan entre sí y se desarrollan los casos de uso.

El diagrama de secuencia de la fase de análisis es mucho más general que el de la fase de diseño, ya que explica el problema en un nivel muy alto, sin entrar en tecnologías a utilizar e interacciones precisas entre clases como si se detallan en la fase de diseño, por lo que los diagramas de secuencia que se exponen a continuación expresan el desarrollo del caso de uso

de forma simple para que se comprenda más fácil, en la fase de diseño se refinará más.

Se presenta a continuación un diagrama de secuencia por cada caso de uso, pero como hemos podido ver en la figura 5.1 en algunos casos de uso hay otros casos de uso que extienden a estos, por lo que se presentará un diagrama de caso de uso general, en el cual se hará referencia de en que parte encaja cada diagrama de caso de uso que extiende a este primero, es decir, si en el primer caso de uso, en el de Crear Red, tenemos un caso de uso llamado Crear Switches que extiende a este primero, por lo que en el diagrama de secuencia del caso de uso Crear Red habrá una referencia al diagrama de secuencia del caso de uso Crear Switches, entendiéndose que ahí es donde encaja dicho diagrama pero para una mayor claridad se presentan por separado.

A continuación se presentan desde la figura 5.4 hasta la figura 5.14 los diferentes diagramas de secuencia realizados en la fase de análisis para todos los casos de uso y sus extensiones. Si se necesita ayuda para comprender el diagrama de secuencia se recomienda mirar de nuevo las tablas desde la 5.5 hasta la 5.14, ya que ahí se describen los escenarios del caso de uso y sus flujos alternativos, en los cuales están basados los diagramas de secuencia.

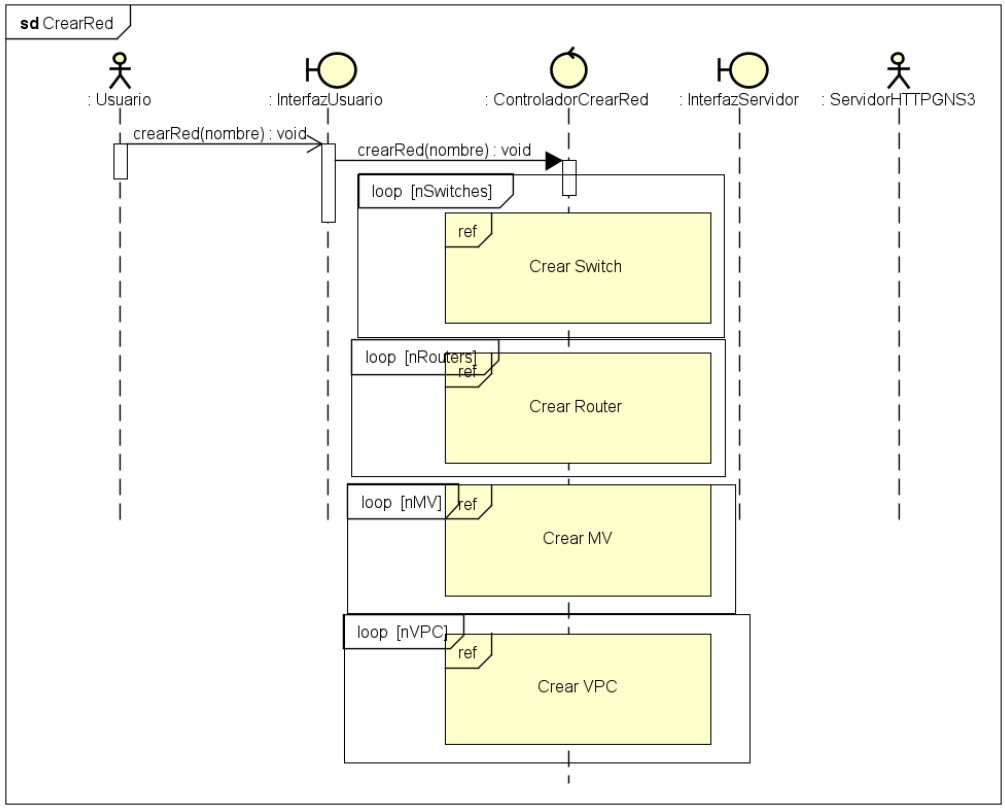

Figura 5.4: Diagrama de secuencia de análisis del caso de uso Crear Red

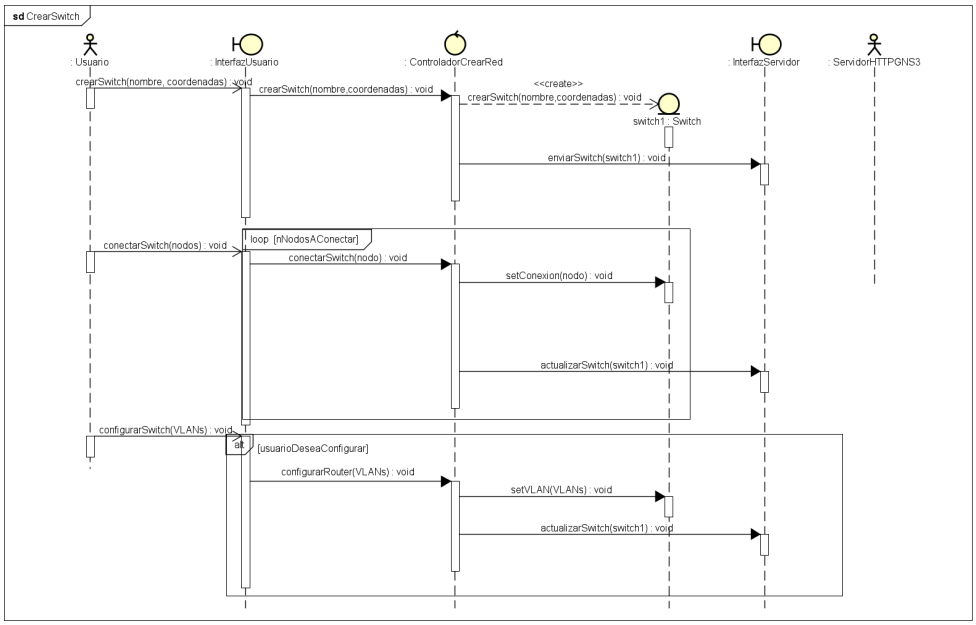

Figura 5.5: Diagrama de secuencia de análisis del caso de uso Crear Switch

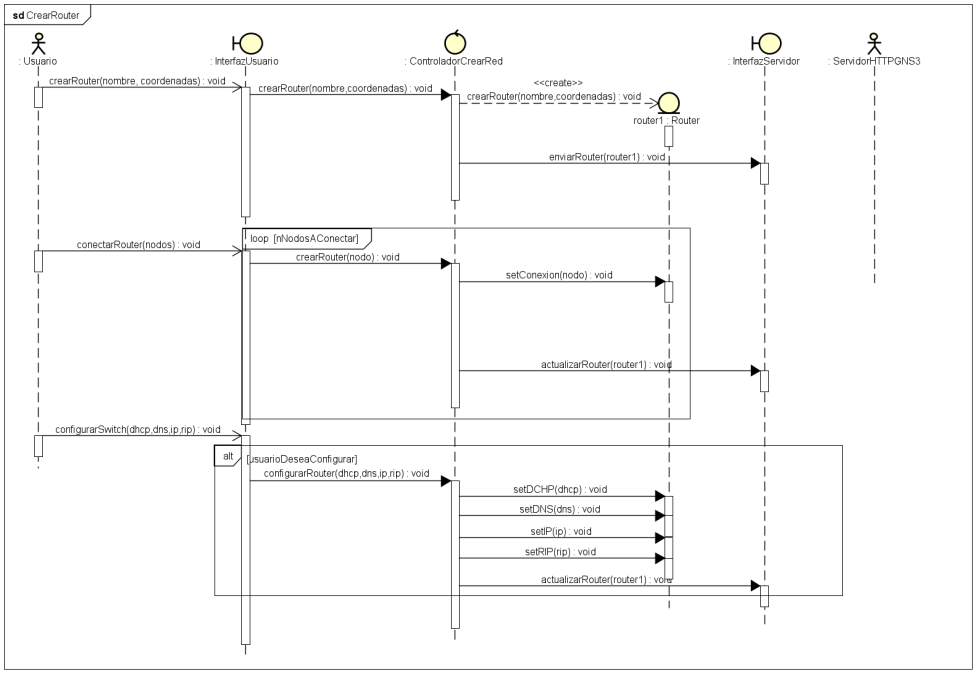

Figura 5.6: Diagrama de secuencia de análisis del caso de uso Crear Router

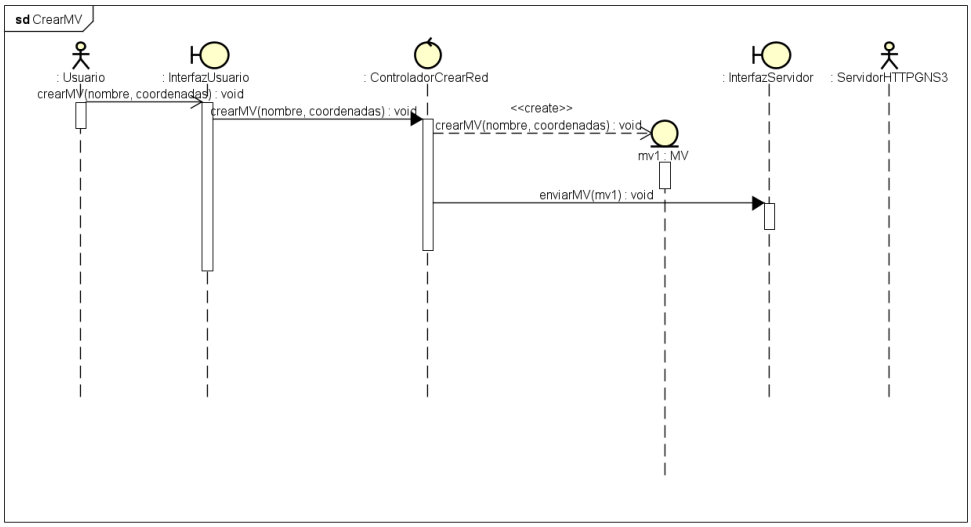

Figura 5.7: Diagrama de secuencia de análisis del caso de uso Crear MV

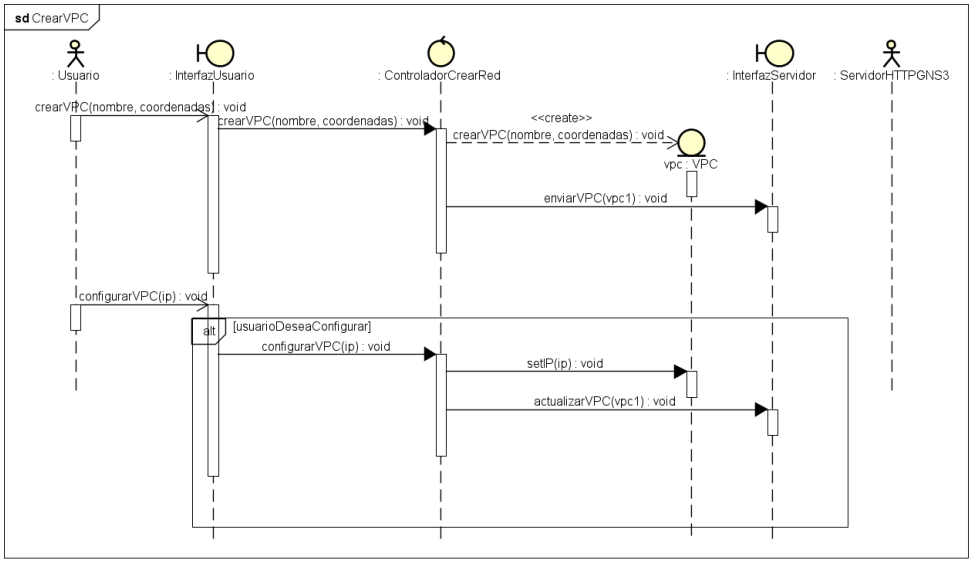

Figura 5.8: Diagrama de secuencia de análisis del caso de uso Crear VPC

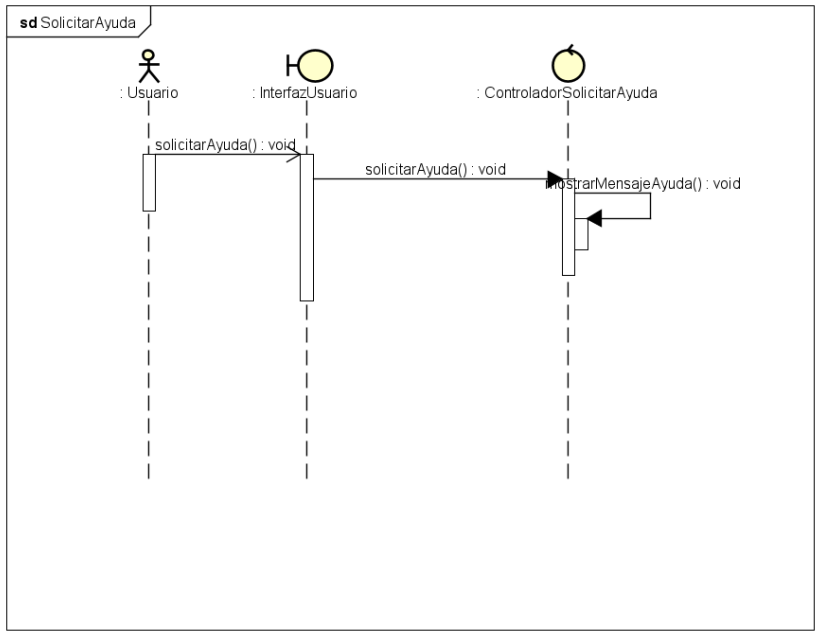

Figura 5.9: Diagrama de secuencia de análisis del caso de uso Solicitar Ayuda

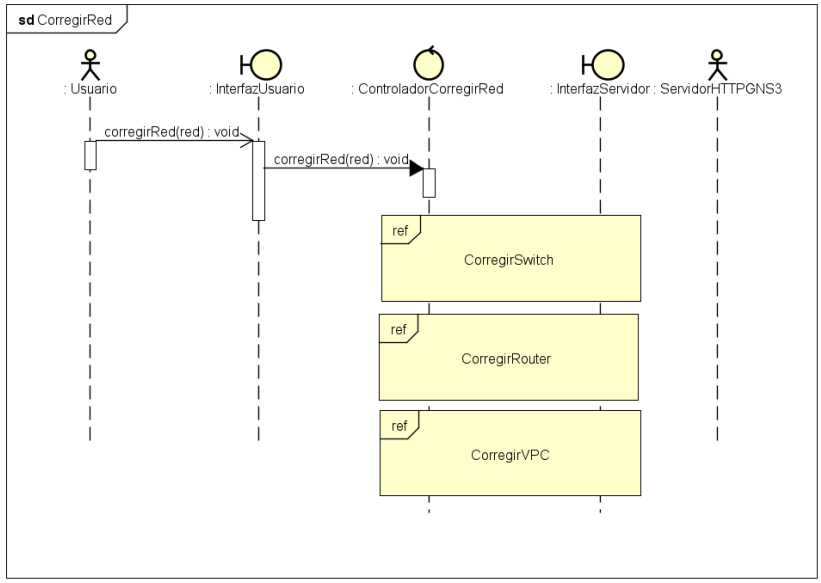

Figura 5.10: Diagrama de secuencia de análisis del caso de uso Corregir Red

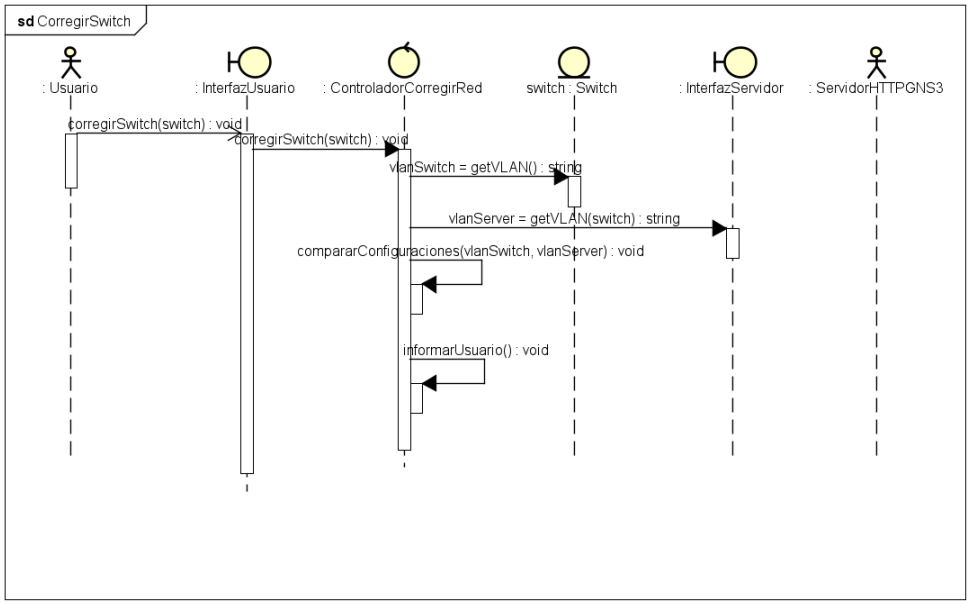

Figura 5.11: Diagrama de secuencia de análisis del caso de uso Corregir Switch

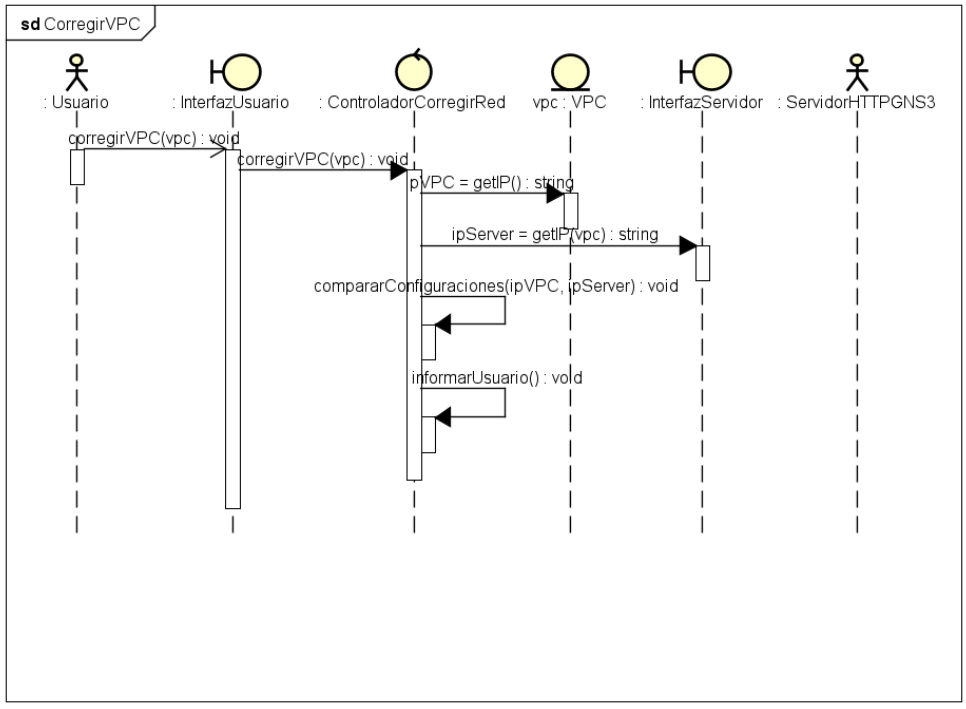

Figura 5.12: Diagrama de secuencia de análisis del caso de uso Corregir VPC

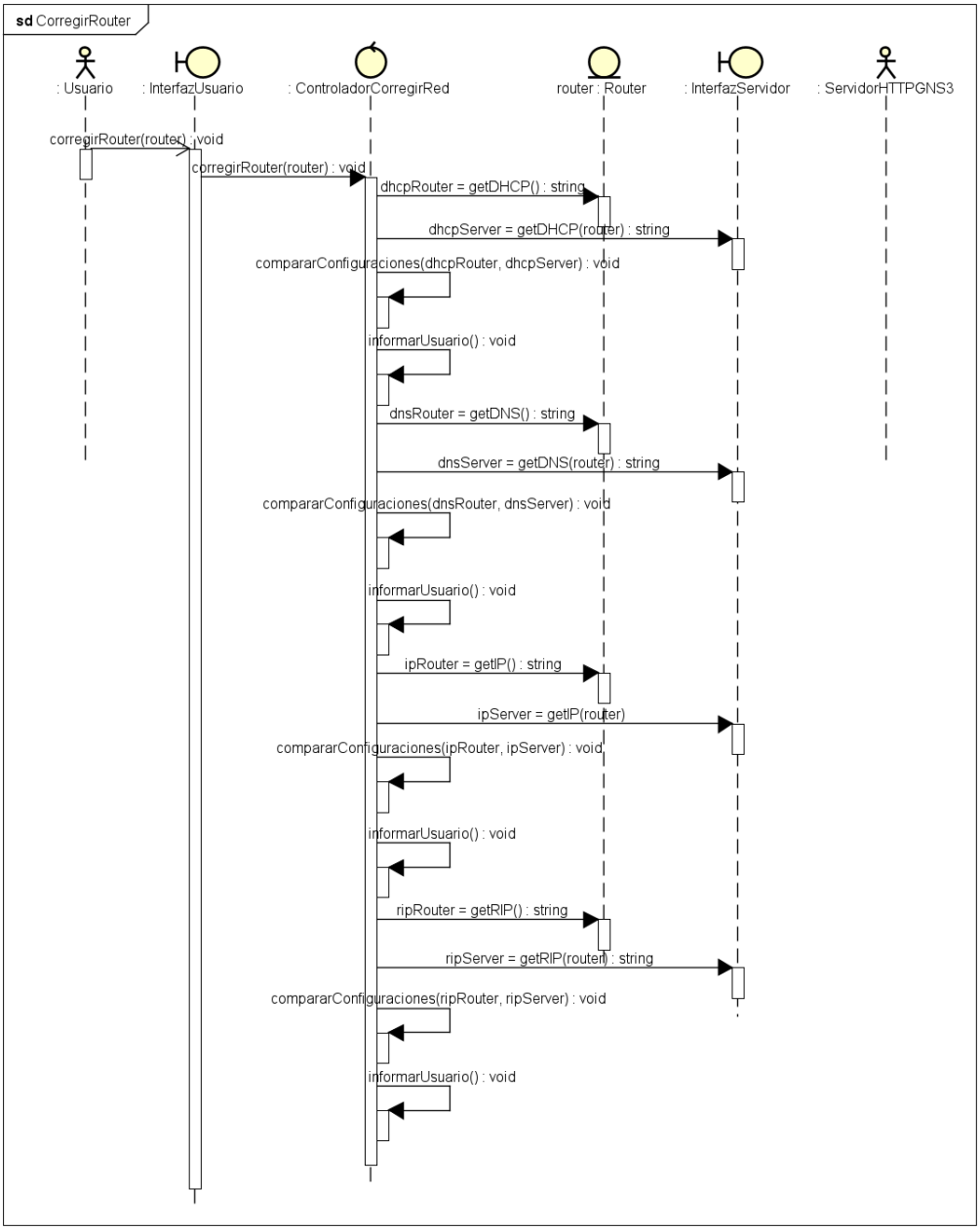

Figura 5.13: Diagrama de secuencia de análisis del caso de uso Corregir Router

Capítulo 6

Diseño

En este capítulo se explicará detalladamente el proceso de diseño realizado para los resultados del proyecto, en este caso son: el script de los talleres de laboratorio, el script de corrección de estos y de la topología de red que se usará para dicho taller.

Los resultados y entregables del proyecto, además de esta memoria y la planificación del mismo, son dos Shell Scripts, uno de ellos es el generador de los talleres de laboratorio que, a partir de un fichero JSON y un nivel de dificultad a elegir por el usuario, genera un proyecto GNS3 con una topología de red definida, el otro Shell Script trabajará en la corrección de dicho taller de laboratorio.

Se procederá a la explicación detallada del diseño de la topología de red seleccionada para este proyecto, además del proceso de diseño de dichos Shell Script mencionados anteriormente.

6.1. Diseño de la topología de red

En este caso, para este proyecto, el taller de laboratorio se realizará en torno a una topología de red previamente definida, pero debemos tener en cuenta las numerosas dificultades que pueden ocurrir en el diseño de la misma.

Como se ha mencionado en capítulos anteriores, GNS3 consume bastantes recursos del sistema debido a las tecnologías sobre las que se basa y en las formas de trabajar que utiliza, por lo que a la hora de seleccionar la red adecuada se ha primado por el menor uso de recursos posible, descartando las redes más complejas con una numerosa cantidad de nodos que haría que las máquinas con menos recursos no pudieran realizar el taller de laboratorio. A esto se le suma la dificultad de elegir una red que, con los menos recursos posibles, pueda ofrecer la posibilidad de configuración de todos los protocolos pensados para el aprendizaje del usuario como son la posibilidad de configurar enrutamiento entre VLAN, protocolos de

la capa de aplicación como HTTP o DNS, protocolos de enrutamiento... así que el objetivo y dificultad principal del diseño consiste en seleccionar una red que permita configurar numerosos protocolos al usuario de diversas capas pero sin consumir demasiados requisitos al sistema.

Tras investigar y ver numerosas topologías de red que podrían aplicarse o adaptarse a lo requerido y que tengan el mínimo número de nodos posibles para evitar saturar el sistema y, por ende, sean lo más adecuadas a lo pedido por el proyecto la red seleccionada es la siguiente:

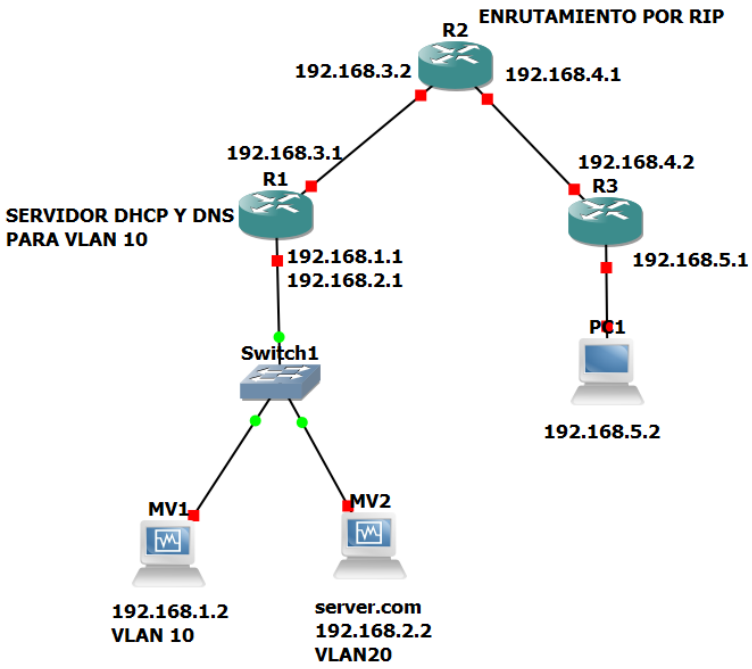

Figura 6.1: Representación gráfica de la topología de red seleccionada

La red propuesta en la figura 6.1 se compone, primero, por 2 host que son realmente dos máquinas virtuales cuyas especificaciones se detallarán después, estas dos máquinas virtuales están conectadas a un Switch, este a su vez está conectado a un Router que encaminará los paquetes al exterior; en esta primera parte existen dos redes, la red 1 y la red 2, separadas entre sí mediante la existencia de VLAN, la VLAN 10 hace referencia a la red 1 y la VLAN 20 hace referencia a la red 2, el router al que están conectadas hará de puente entre ellas. Este router está conectado a su vez a otro y este a otro más, del cual cuelga un host, este

no es una máquina virtual sino que para ahorrar recursos se ha decidido que sea un VPCS que consiste en una máquina muy simple con muy pocos comandos y una interfaz de red, solo servirá para realizar ping y comprobar que existe un encaminamiento de paquetes, todos estos routers estarán configurados para contar con un protocolo de enrutamiento, en nuestro caso RIPv2.

La red seleccionada es una versión modificada de una propuesta de topología para un laboratorio de redes que se muestra en una de las entradas del estado del arte de este proyecto, en concreto es la segunda entrada del estado del arte a la que estoy haciendo referencia. En esta entrada el autor escribe un apartado en el que diseña unos talleres de laboratorio para los alumnos, en dicho apartado muestra las distintas topologías en las que se basará para la realización de los talleres en la que hay justo una topología que me parecía muy adecuada si se le incluían unas pocas modificaciones ya que era simple al no necesitar muchos nodos y podía ofrecer todos los protocolos y funcionalidades necesarias para desarrollar un buen taller de laboratorio completo.

Me he interesado especialmente en esta entrada del estado del arte para buscar una topología de red acorde a lo necesitado para el proyecto debido a que el autor hace en todo momento especial hincapié en la gran cantidad de recursos que consume GNS3 y la necesidad de buscar el máximo ahorro de estos, por lo que es el que más se adapta a las necesidades que surgen en este proyecto. En concreto la red en la que me baso es la figura 2.20 de dicho documento, que hace referencia al laboratorio 22, se muestra aquí a continuación en la figura 6.2, la modificación que se le ha realizado es simplemente mover un host y conectarlo directamente al switch, para ofrecer la posibilidad de crear VLAN en la red, el resto es idéntico al original.

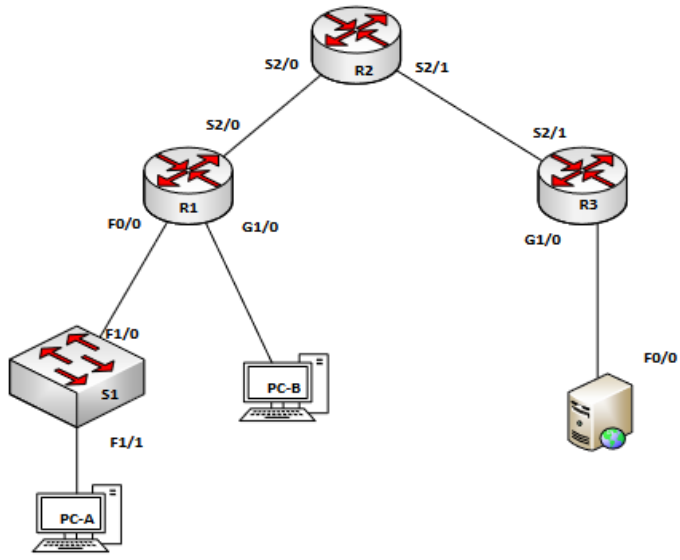

Figura 6.2: Topología de red base para el diseño de la topología de red utilizada en el proyecto

A continuación se expone una lista explicando uno a uno los nodos de la red seleccionada que se muestra en la figura 5.1, para una mayor comprensión del diseño de la misma:

1. **MV1:** Consiste en una máquina virtual Ubuntu 10.04, la cual corre una versión un poco antigua para ahorrarnos el uso de recursos extra innecesarios, está situada en la zona de abajo a la izquierda de la red y está conectada al switch, pertenece a la red 1 y a la VLAN 10, obtendrá su dirección IP mediante el protocolo DHCP, solicitando dicha IP al router, que en este caso hará de servidor DHCP y DNS para esta red. La contraseña de acceso de esta máquina virtual es:ubuntuv, igual que el nombre de usuario

2. **MV2:** Situado justo al lado del nodo MV1 pero perteneciente a una red distinta, en este caso a la red 2 y a la VLAN 20. Es otra máquina virtual Ubuntu 10.04 para también ahorrar recursos, la cual ejecuta, a su vez, un servidor web Apache[29] para comprobar que el protocolo HTTP y el protocolo DNS funcionan correctamente incluso para redes exteriores, cuenta con el nombre de dominio server.com y su IP será estática para poder hacer de fácil acceso para el exterior su servidor web. La contraseña de acceso de esta máquina virtual es:server.com, igual que el nombre de usuario

3. **Switch1:** Es el único switch de toda la red, consiste en un switch ethernet básico que solamente ofrece capacidad de retransmisión y de configurar VLAN. Será el encargado de retransmitir los paquetes del router a los host y de los host al router, tendrá en su bases de datos dos VLAN: la 10 y la 20, y asignará a cada puerto la VLAN correspondiente, la 10 en acceso para el puerto al que está conectado el nodo MV1, la 20 en acceso para el puerto al que está conectado la máquina MV2 y en trunk para el puerto al que está conectado el router, permitiendo que cualquier paquete de cualquier VLAN circule por él.

4. **R1:** Es el router que da acceso al exterior a las redes 1 y 2 y encamina los paquetes entre estas, consiste en un router Cisco 2691, cuenta a su vez con un servidor DHCP y DNS configurado en este únicamente para la red 1, esto se ha hecho para ahorrarnos muchos recursos en conectar otra máquina virtual a esta red para cumplir unas funciones que realmente podemos encapsular en el mismo router sin necesidad de máquinas extra consumiento recursos. Para dar soporte a las redes 1 y 2 utiliza un único puerto al cual se le encapsulan dos interfaces virtuales, una para cada VLAN, este puerto va directamente conectado al switch; el otro puerto está conectado a otro router y es el que permite el acceso al exterior, estos dos conforman la red 3. Además, cuenta con la configuración del protocolo de enrutamiento dinámico sin clase RIPv2 para permitir el encaminamiento de paquetes al exterior con la ayuda del resto de routers de la red.

5. **R2:** Router intermedio entre R1 y R3, cuenta también con la configuración de RIPv2 como protocolo de enrutamiento, tiene conectado en un puerto el router R1, formando estos dos conjuntamente la red 3, y el router R3, formando estos dos conjuntamente la red 4. El modelo es el mismo, un router Cisco 2691 y su papel sirve únicamente como router intermedio entre el router R1 y R3, permitiendo generar así una idea de que la

red 5 con el VPCS está más lejos que si simplemente conectamos dos routers entre sí, además de mostrar mejor cómo funciona un protocolo de enrutamiento dinámico.

6. **R3:** Router de la derecha del todo de la red del que cuelga el nodo PC1, al igual que los otros dos router se trata de un Cisco 2691 que cuenta con la configuración del protocolo de encaminamiento dinámico RIPv2. Su papel sirve para dar conectividad al exterior al host que cuelga de él, tiene conectado en un puerto el router R2 con el que conjuntamente crean la red 4 y en otro puerto un host VPCS, con el cual crean conjuntamente la red 5.

7. **PC1:** Host que cuelga del router R3, en esta ocasión no es una máquina virtual sino un VPCS, que es un simulador de un ordenador muy limitado en funcionalidades, ya que estas se limitan a aplicar el protocolo DHCP y a enviar y recibir ping, además gasta muy pocos recursos. Su única funcionalidad va a ser la de enviar y recibir ping, por lo que no nos hace falta una máquina virtual para eso y podemos ahorrarnos los recursos que conllevaría. Forma la red 5 junto al router R3.

Una vez comprendida la topología y todos los nodos uno a uno, podemos finalizar este subapartado con una tabla resumen de los protocolos y funcionalidades que soporta cada nodo (suponemos que absolutamente todos soportan ping salvo el switch), además de la dirección IP que tienen asignados en cada puerto para que sea mucho más fácil de entender de un solo vistazo. Esto se presenta en la tabla 6.1 a continuación.

Nodo	**Puerto**	**IP**	**Protocolos/Funcionalidades**
MV1	Ethernet0	Proporcionada por DHCP	DHCP, DNS
MV2	Ethernet0	192.168.2.2	HTTP
Switch1	Ethernet0	-	VLAN 10 Acceso
	Ethernet1	-	VLAN 20 Acceso
	Ethernet2	-	VLAN Trunk
R1	FastEthernet0/0.10	192.168.1.1	DHCP, DNS, RIPv2
	FastEthernet0/0.20	192.168.2.1	RIPv2
	FastEthernet0/1	192.168.3.1	RIPv2
R2	FastEthernet0/0	192.168.3.2	RIPv2
	FastEthernet0/1	192.168.4.1	RIPv2
R3	FastEthernet0/0	192.168.4.2	RIPv2
	FastEthernet0/1	192.168.5.1	RIPv2
PC1	Ethernet0	192.168.5.2	-

Tabla 6.1: Tabla de direccionamiento y funcionalidades de la topología de red seleccionada

6.2. Diseño de los scripts de talleres de laboratorio

Una vez vista la fase de análisis de los scripts y vista la fase de diseño de la topología de red seleccionada para este proyecto, llega la hora de ver la fase de diseño elegida para los scripts de talleres de laboratorio.

La fase de diseño es la que se encarga de planear y brindar una solución al problema expuesto en la fase de análisis, buscando cumplir una serie de objetivos y requisitos, además de estar expuesto a una serie de restricciones, también está a un nivel mucho más bajo que el análisis, otorgando modelos y diagramas mucho más refinados y cercanos a la realidad. Permite planificar la futura implementación del sistema software que otorgue la solución al problema que se ha propuesto.

Como se ha estado mencionando a lo largo de todo el proyecto, la automatización de las tareas de creación y corrección de las redes se realizará por medio de scripts, pero a diferencia de como se mostraba en la fase de análisis que mostraba una secuencia mediante una interacción entre usuario y sistema aquí, al haber bajado más el nivel, se leerá toda la información desde un fichero JSON, por lo que la única interacción entre el usuario y el sistema es la ejecución del script, después el sistema hará todo el trabajo.

Lo común es que en la fase de diseño se utilicen soluciones con patrones que se adapten al sistema software que estamos diseñando. Al igual que en la fase de análisis para una mayor claridad se utilizó el patrón arquitectónico BCE, en la fase de diseño he seleccionado el patrón arquitectónico Filtro Tubería, ya que, tras una investigación de diferentes patrones, es el que más se adaptaba a la solución que quería diseñar.

El patrón Filtro Tubería consiste en la consecución de dos elementos, filtros y tuberías para representar el flujo de datos del sistema en la ejecución de una tarea que ha sido dividida en subtareas. Cada filtro representa una de las subtareas, y las tuberías la transmisión de datos entre ellas. Suele utilizarse para representar el flujo de los datos durante una tarea de conversión de datos, como puede ser el caso de los compiladores, ya que es el patrón que utilizan estos por excelencia, en la figura 6.3. se puede comprender esto de manera más clara.

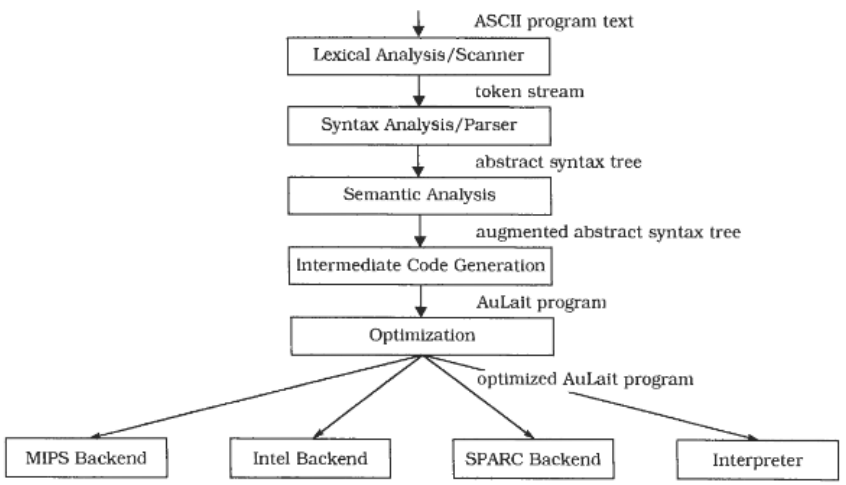

Figura 6.3: Patrón Filtro Tubería con un compilador, extraído del libro de Buschmann[24]

La figura 6.3 representa el flujo de datos en un compilador, la entrada es un programa

escrito en formato ASCII escrito por el usuario que entra al primer filtro, este filtro transforma los datos y mediante tuberías va pasando por distintos filtros que lo seguirán transformando hasta llegar a una salida final para cada tipo de procesador. La gran tarea sería compilar el programa, esta tarea se divide en diferentes subtareas como podrían ser optimización y análisis semántico, cada una de estas subtareas es un filtro que se comunican entre sí por las tuberías transmitiéndose los datos hasta llegar a un resultado final.

La mayoría de patrones que conozco y he estudiado están pensados para sistemas que utilizan una programación orientada a objetos, en este caso, al tratarse de scripts, es más difícil encontrar un patrón que se adapte bien ya que los scripts no están orientados a objetos, es por eso que el patrón Filtro Tubería es ideal, porque está pensado para este tipo de casos en los que cuesta mucho elegir un patrón que se adapte correctamente y especialmente está pensado para sistemas que realizan una tarea secuencial, con un inicio y un final, como es el caso de los scripts.

Para implementarlo he tenido que dividir ambos scripts en subtareas secuenciales. Para ambos scripts han surgido un total de 3 subtareas: Creación, conexión y configuración de nodos para el script de creación y corrección de switches, routers y VPC para el script de corrección. Esto provoca que para cada caso, tenga un total de 3 filtros y 4 tuberías salvo para el caso de solicitar ayuda, que tengo un solo filtro y 2 tuberías ya que es imposible de dividir su tarea principal en subtareas.

El patrón arquitectónico dicta la arquitectura lógica del sistema, es decir, muestra los componentes lógicos del mismo (subsistemas, objetos...) y las relaciones entre estos, sin tener en cuenta el soporte físico que los sostendrá (bases de datos, clientes, servidores...). Sobre el patrón arquitectónico se basa el patrón de diseño, que dictará el comportamiento y el diseño de los distintos subsistemas, refinándolos.

El patrón de diseño seleccionado es el patrón Transaction Script[32], este organiza cada una de las tareas o subtareas en transacciones, por ejemplo, en una biblioteca podrían darse dos grandes casos de uso prestar o devolver libros; si aplicamos el patrón Transaction Script cada uno de estos casos de uso podría encapsularse en un Transaction Script que se encargue de llevad dicha tarea a cabo. Si dividimos una tarea en subtareas se generarían más Transaction Script, uno para cada subtarea. También es importante mencionar que es un patrón de diseño no orientado a objetos, por lo que es ideal para nuestro sistema.

En nuestro caso he decidido que cada filtro, es decir, cada subtarea o subsistema, se represente en el patrón de diseño con un Transaction Script que sea el encargado de llevar sus responsabilidades a cabo.

En la figura 6.4 que se muestra a continuación podemos encontrarnos la arquitectura lógica que representa el script de creación. La tarea principal que es crear una red se ha dividido en tres subtareas representadas por tres filtros conectadas por tuberías entre sí: creación de los nodos, conexión de los nodos y configuración de los nodos. Además, todos los filtros están conectados al servidor GNS3, ya que hay un flujo de información constante con este. En el nivel de patrón de diseño cada filtro se representa con un Transaction Script que realizará la tarea encomendada al filtro y, al terminar, llamará al siguiente Transaction Script transmitiéndole la información para que realice su tarea. Los datos de entrada son un fichero JSON por parte del usuario con la configuración a seguir y un nivel de dificultad para

configurar la red de una forma u otra y el resultado final, es decir, los datos de salida, es un proyecto GNS3 perfectamente utilizable por el usuario.

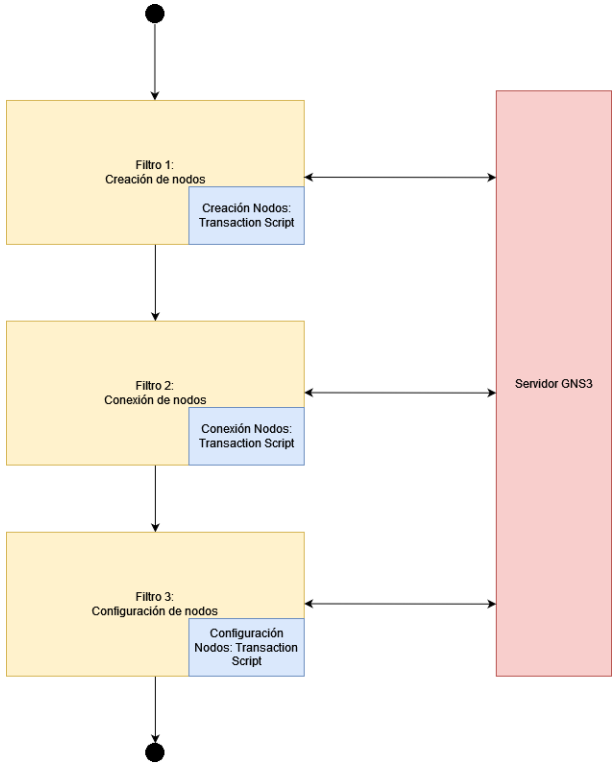

Figura 6.4: Diseño arquitectonico para el script de creación

En la figura 6.5 se muestra la arquitectura lógica del segundo script, el script de corrección. En este caso la tarea principal es la de corregir una red que se ha creado con el script de creación anteriormente, esta tarea se ha dividido a su vez en tres subtareas: corregir switches, corregir routers y corregir VPC. Cada subtarea se representa con un filtro que transmiten la información entre sí a través de tuberías, también, como en el script de creación, están conectados todos los filtros al servidor GNS3, ya que hay un intercambio de información constante con este. En el nivel de patrón de diseño, cada filtro se representa como un Transaction Script al igual que en el script de creación, este realizará toda la tarea que corresponde al filtro y transmitirá la información al siguiente al finalizar. La entrada será por parte del usuario un fichero JSON de configuración de la red y un ID del proyecto que se quiere corregir, el resultado será información para el usuario indicándole si la configuración de cada nodo es correcta o es errónea.

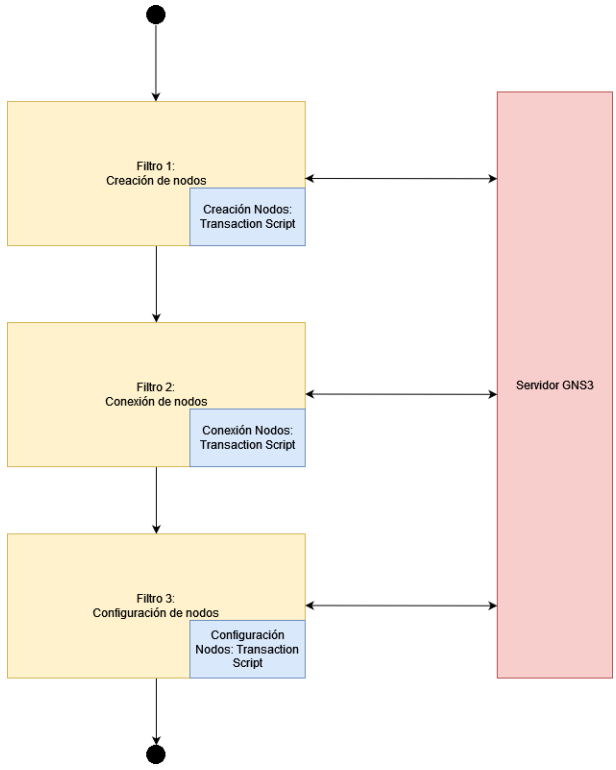

Figura 6.5: Diseño arquitectonico para el script de corrección

Por último, en la figura 6.6. se muestra la arquitectura lógica que es común para ambos scripts, tiene relación con el caso de uso Solicitar Ayuda, ya que representa la solución en caso de que el usuario requiera que cualquiera de los dos scripts le muestre una ayuda acerca de su funcionamiento y los argumentos que requiere. Esta tarea no es posible dividirla en subtareas por lo que se representa con un único filtro y dos tuberías, la de entrada del usuario y la de salida del sistema. Internamente, como patrón de diseño, también se ha usado Transaction Script, pero solamente existe uno, al existir solamente un filtro. La entrada es el argumento -h o –help por parte del usuario que indicará al sistema que el usuario necesita que se muestre la ayuda y el sistema tendrá como salida dicha ayuda para el usuario.

A continuación, una vez entendida la arquitectura lógica, se pasa a explicar la arquitectura física del sistema. La arquitectura física corresponde a la representación de todos los elementos físicos que intervienen en el sistema para su implementación y funcionamiento, como podrían ser los servidores, las bases de datos o los clientes y los diferentes elementos tanto hardware como software que necesitan para su uso.

La arquitectura física se suele representar con un diagrama de despliegue, que consiste en un diagrama que modela y sitúa los elementos hardware y los elementos software del sistema en nodos y la relación entre ellos con el objetivo de tener una visión más clara de cómo será la instalación del sistema una vez finalizado su diseño.

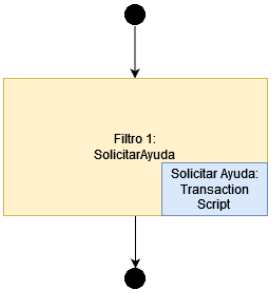

Figura 6.6: Diseño arquitectonico para solicitar ayuda

En nuestro caso el diagrama de despliegue es bastante sencillo, ya que todo ocurre en una misma máquina local, sin tener la necesidad de conectarnos al ningún servidor exterior. Se compone de una gran máquina local que agrupa todo el hardware necesario que, dentro de ella, existen una serie de componente software que interactúan entre sí: los scripts, el fichero JSON de configuración y el servidor local de GNS3, que dentro de él tiene la API REST que se necesita para poder comunicarse con este servidor. El fichero JSON de configuración es el fichero que leerán ambos scripts para poder trabajar ya que contiene toda la información necesaria para la ejecución de estos, estos scripts a su vez se comunicarán con el servidor GNS3 mediante la API REST que contiene todos los métodos necesarios. En la figura 6.7 se muestra el diagrama de despliegue del sistema.

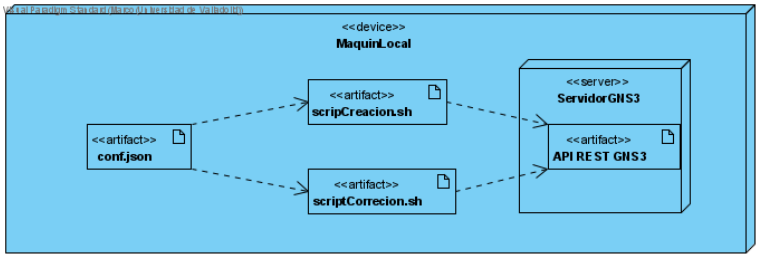

Figura 6.7: Diagrama de despliegue del sistema

Sobre los datos de entrada de la aplicación creo necesario explicar la estructura del fichero JSON de configuración de la red, ya que es un eje fundamental para el correcto funcionamiento de ambos scripts. Este fichero contiene toda la información de la topología de red, desde el número de nodos hasta la configuración de los mismos. Tiene diferentes campos basados en la topología de red seleccionada para el proyecto por lo que es a partir de este fichero por el que se generará una red u otra, radicando aquí la gran potencia de la universalidad de los script, ya que dependiendo de cómo estén los campos del JSON la topología de red cambia completamente y la configuración de los mismos también, pero siendo compatibles con ambos scripts siempre y cuando se respete la estructura, ya que el script de creación leerá la información de este JSON para crear la topología de red y configurarla y el script de corrección también lo leerá para comprobar la configuración hecha por el usuario y realizar la corrección. A continuación se procede a explicar estos campos en base a la jerarquía de

estos, ya que un campo puede contener más de un campo:

- **nombreTaller:** Nombre que tendrá el taller a la hora de su creación.
- **nRouters:** Número de routers totales de la topología.
- **nSwitches:** Número de switches totales de la topología.
- **nMV:** Número de máquinas virtuales totales de la topología.
- **nVPCS:** Número de VPCS totales de la topología.
- **routers:** Información sobre los routers de la topología, cada router se compone de los siguientes campos:
  - **nombre:** Nombre del router.
  - **puertosEnUso:** Puertos que están ocupados en el router conectados a otros nodos
  - **maquinasConectadas:** Nombre de las máquinas que están ocupando los puertos en orden. En caso de ser un switch es recomendable dejar dicho puesto en null, para que, en la fase de conexión de switches se realice la conexión y no en la de conexión de routers para evitar problemas con la limitación de la API REST de GNS3 con la configuración de los switches.
  - **vlan:** Número de la vlan asignada a cada puerto en uso en orden.
  - **ip:** Dirección IP y máscara de red asignada a cada puerto en uso en orden, si hay más de una dirección IP en un puerto significa que están encapsuladas en direcciones virtuales.
  - **dhcpPool:** Su existencia indica si el router actuará como servidor DHCP. Contiene información de cada pool DHCP.
    - **nombre:** Nombre del pool DHCP
    - **red:** Rango de red asignado al pool
    - **routerPorDefecto:** Dirección IP del router por defecto del pool
    - **dns:** Dirección IP del servidor DNS asociado al pool.
  - **dnsHost:** Su existencia indica si el router actuará también como servidor DNS, su contenido se compone de una lista con el nombre del host que se quiere asociar a una IP y dicha dirección IP.
  - **rip:** Su existencia indica si el router pooserá enrutamiento dinámico mediante el protocolo RIPv2, su contenido se compone de las redes que anunciará al resto de routers.
  - **x:** Coordenada X del router en la topología de red
  - **y:** Coordenada Y del router en la topología de red
- **switches:** Contiene información sobre cada uno de los switches de la topología de red.
  - **nombre:** Nombre del switch.
  - **puertosEnUso:** Número de puertos ocupados en el switch por otros equipos

- **máquinasConectadas:**Nombre de las máquinas que están ocupando los puertos en orden.
- **vlan:** Número de la VLAN asociada a cada puerto en orden, en caso de ser un puerto en modo trunk no aparecerá un número, sino dicha palabra.
- **x:** Coordenada X del switch en la topología de red.
- **y:** Coordenada Y del switch en la topología de red.

- **MV:** Contiene información sobre cada una de las máquinas virtuales de la topología de red.
  - **nombre:** Nombre con el que la máquina virtual se representará en la topología de red.
  - **nombreMV:** Nombre real de la máquina virtual dentro de VirtualBox.
  - **puertosEnUso:** Número de puertos ocupados en la máquian virtual por otros equipos.
  - **maquinasConectadas:** Nombre de las máquinas que están ocupando los puertos en orden.
  - **x:** Coordenada X de la máquina virtual en la topología de red.
  - **y:** Coordenada Y de la máquina virtual en la topología de red.
- **VPCS:** Contiene información sobre cada uno de los VPCS de la topología de red.
  - **nombre:** Nombre del VPC.
  - **puertosEnUso:** Número de puertos ocupados en el VPC por otros equipos.
  - **máquinasConectadas:** Nombre de las máquinas que están ocupando los puertos en orden.
  - **x:** Coordenada X del VPC en la topología de red.
  - **y:** Coordenada Y del VPC en la topología de red.

Por ejemplo, para crear la red propuesta en el diseño en la figura 6.1, el fichero JSON de configuración debería ser el siguiente:

```
{
        "nombreTaller": "taller",
        "nRouters" : 3,
        "nSwitches": 1,
        "nMV": 2,
        "nVPCS":1,
        "routers": [
                {
                        "nombre": "R1",
                        "puertosEnUso": 2,
                        "maquinasConectadas": [
                                        "Switch1",
```

```
                                        "R2"
                        ],
                "vlan": [
                                [
                                        10,
                                        20
                                ]
                ],
                "ip":[
                                [
                                "192.168.1.1 255.255.255.0",
                                "192.168.2.1 255.255.255.0"
                                ],
                                "192.168.3.1 255.255.255.0"
                ],
                "dhcpPool":[
                        {
                                "nombre": "RED10",
                                "red": "192.168.1.0 255.255.255.0",
                                "routerPorDefecto": "192.168.1.1",
                                "dns": "192.168.1.1"
                        }
                ],
                "dnsHost": [
                        "server.com 192.168.2.2"
                ],
                "rip": [
                                "192.168.1.0",
                                "192.168.2.0",
                                "192.168.3.0"
                        ],
                "x":-215,
                "y":-156
        },
        {
                "nombre": "R2",
                "puertosEnUso": 2,
                "maquinasConectadas": [
                                "R1",
                                "R3"
                        ],
                "ip": [
                        "192.168.3.2 255.255.255.0",
                        "192.168.4.1 255.255.255.0"
                        ],
                "rip": [
                        "192.168.3.0",
                        "192.168.4.0"
                ],
```

```
                        "x":-57,
                        "y":-287
                },
                {
                        "nombre": "R3",
                        "puertosEnUso": 2,
                        "maquinasConectadas": [
                                        "R2",
                                        "PC1"
                                ],
                        "ip": [
                                "192.168.4.2 255.255.255.0",
                                "192.168.5.1 255.255.255.0"
                                ],
                        "rip": [
                                "192.168.4.0",
                                "192.168.5.0"
                        ],
                        "x":75,
                        "y":-150
                }
        ],
        "switches": [
                {
                        "nombre": "Switch1",
                        "puertosEnUso": 3,
                        "maquinasConectadas": [
                                        "MV1",
                                        "MV2",
                                        "R1"
                                ],
                        "vlan": [
                                        10,
                                        20,
                                        "trunk"
                        ],
                        "x":-214,
                        "y":-3
                }
        ],
        "MV": [
                {
                "nombre": "MV1",
                "nombreMV": "UbuntuV",
                "puertosEnUso": 1,
                "maquinasConectadas":"ESW1",
                "x":-309,
                "y":134
                },
```

```
                {
                "nombre": "MV2",
                "nombreMV": "UbuntuWeb",
                "puertosEnUso": 1,
                "maquinasConectadas":"ESW1",
                "x":-132,
                "y":127
                }
        ],
        "VPCS": [
                {
                        "nombre": "PC1",
                        "puertosEnUso": 1,
                        "maquinasConectadas":"R3",
                        "ip": "192.168.5.2 192.168.5.1 24",
                        "x":77,
                        "y":-26
                }
        ]
}
```

Fragmento de código 6.1: JSON de configuración de la red propuesta

En este caso para la red propuesta queremos crear tres routers, un switch, dos máquinas virtuales y un VPC, por lo que los campos «nRouters», «nSwitches», «nMV» y «nVPCS» reflejarán esta información. Después, en el campo «routers» aparecerán los datos de cada router, como los nodos que tiene conectados, el nombre, la configuración RIPv2, DHCP o DNS y las coordenadas, lo mismo con el resto de nodos, los switches están reflejados en el campo «switches» con sus datos, configuraciones (como las VLAN y los nodos que tiene conectados) y sus coordenadas, las MV con sus datos y coordenadas dentro del campo «MV» y los VPCS con sus configuraciones (IP y Gateway) y coordenadas dentro del campo «VPCS». Si queremos añadir un nodo o modificar algún aspecto de este simplemente debemos añadirlo o modificarlo dentro del JSON en el campo correspondiente, asegurándonos de que los campos están rellenos correctamente para su correcta implementación por ambos scripts.

Finalizando con la parte de diseño, se muestran los diagramas de secuencia de más bajo nivel pertenecientes a esta parte, representan cómo realizan las tareas los Transaction Script y cómo será la interacción de estos con el servidor y con el usuario. Hay que aclarar un par de aspectos antes, a lo largo de los diagramas de secuencias se hacen referencia a unos niveles de dificultad que modifican el comportamiento de algunos Transaction Script, estos niveles de dificultad se corresponden al nivel que quiere el usuario que el script configure la red, siendo 0 la totalidad de la red y 3 no configurar nada, esto será explicado con mayor detenimiento en el capítulo de implementación y pruebas. Por limitaciones de GNS3, en la parte de conexión del switch a los nodos si el usuario quiere configurar los switches estos no se conectarán a los nodos en ese momento, sino en la parte de configuración de estos debido a que GNS3 no permite la modificación de la configuración de las interfaces mediante el uso de

la API REST si tienen nodos conectados a estas. En las figuras de la 6.8 y 6.17 se presentan estos diagramas de secuencia.

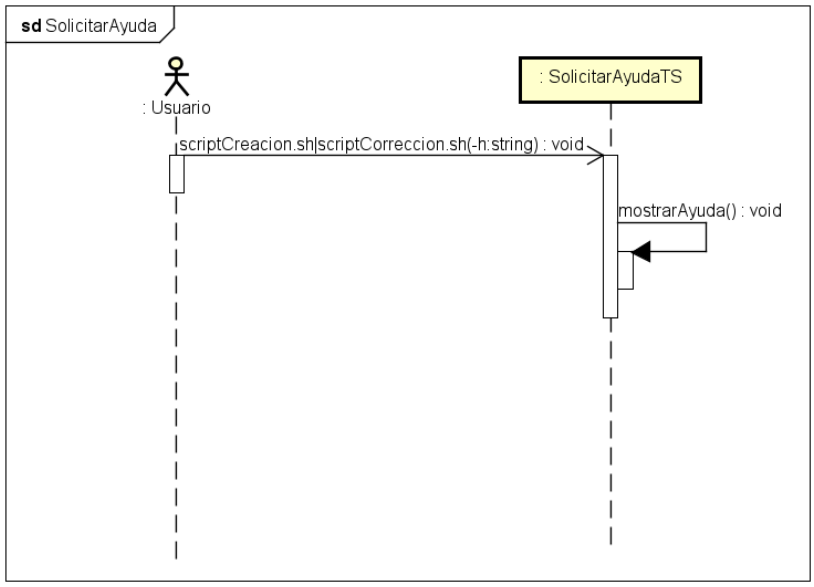

Figura 6.8: Diagrama de secuencia de diseño de solicitar ayuda

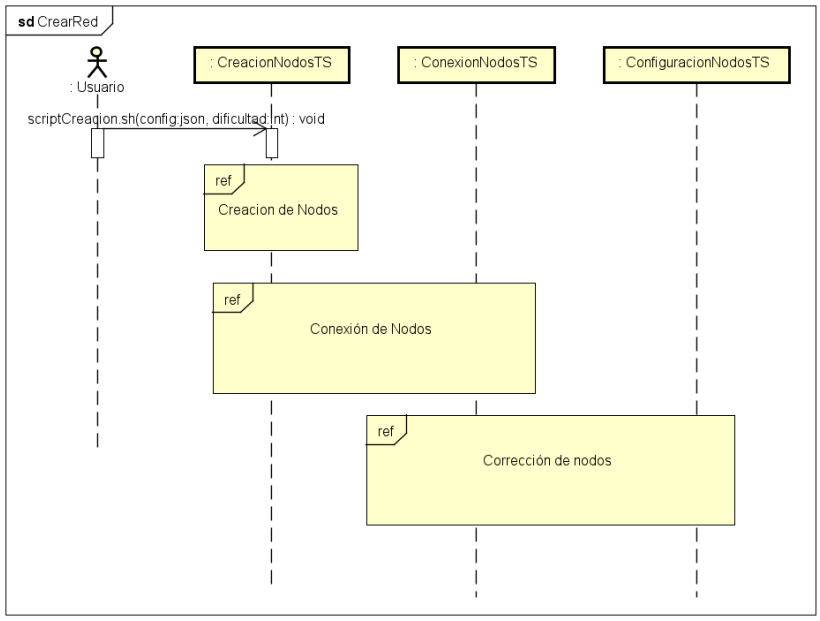

Figura 6.9: Diagrama de secuencia de diseño del script de creación

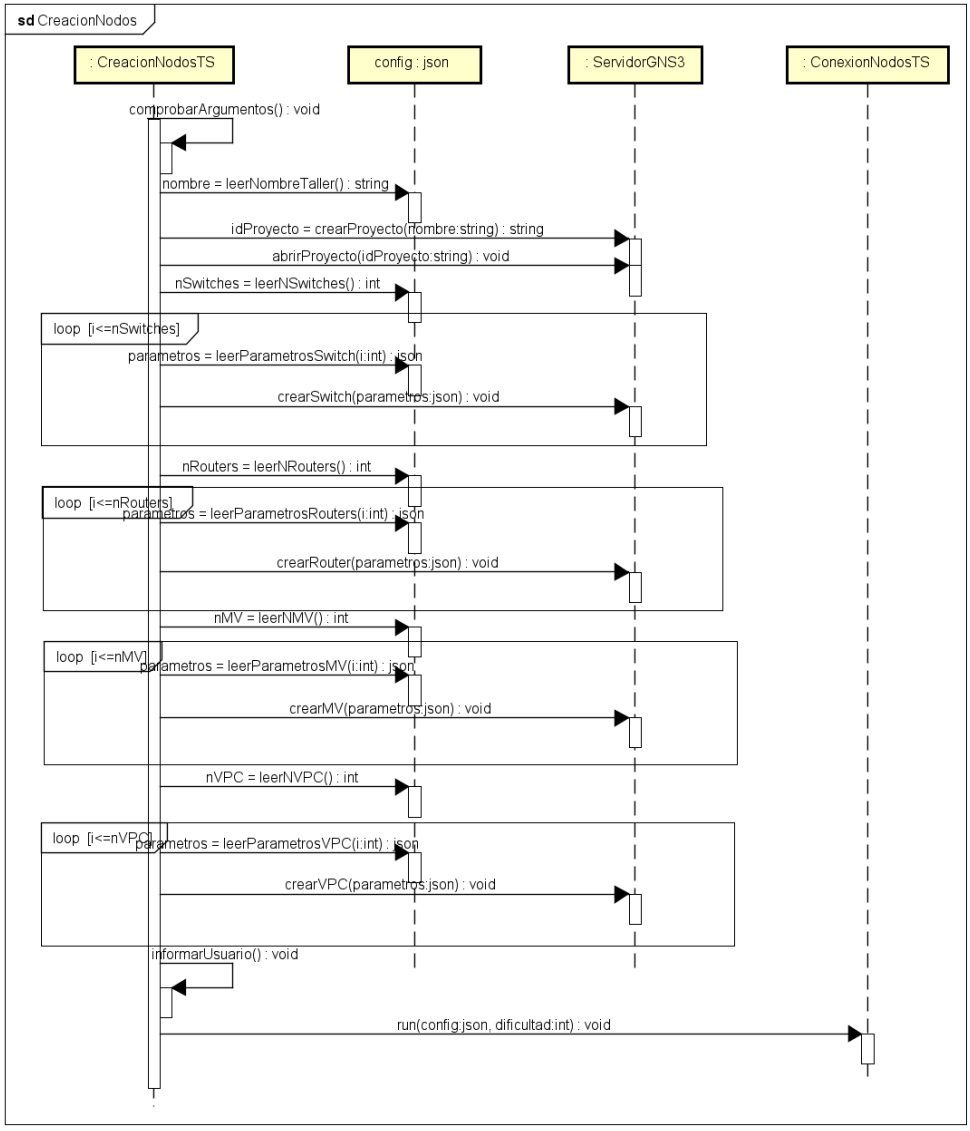

Figura 6.10: Diagrama de secuencia de diseño de la creación de nodos

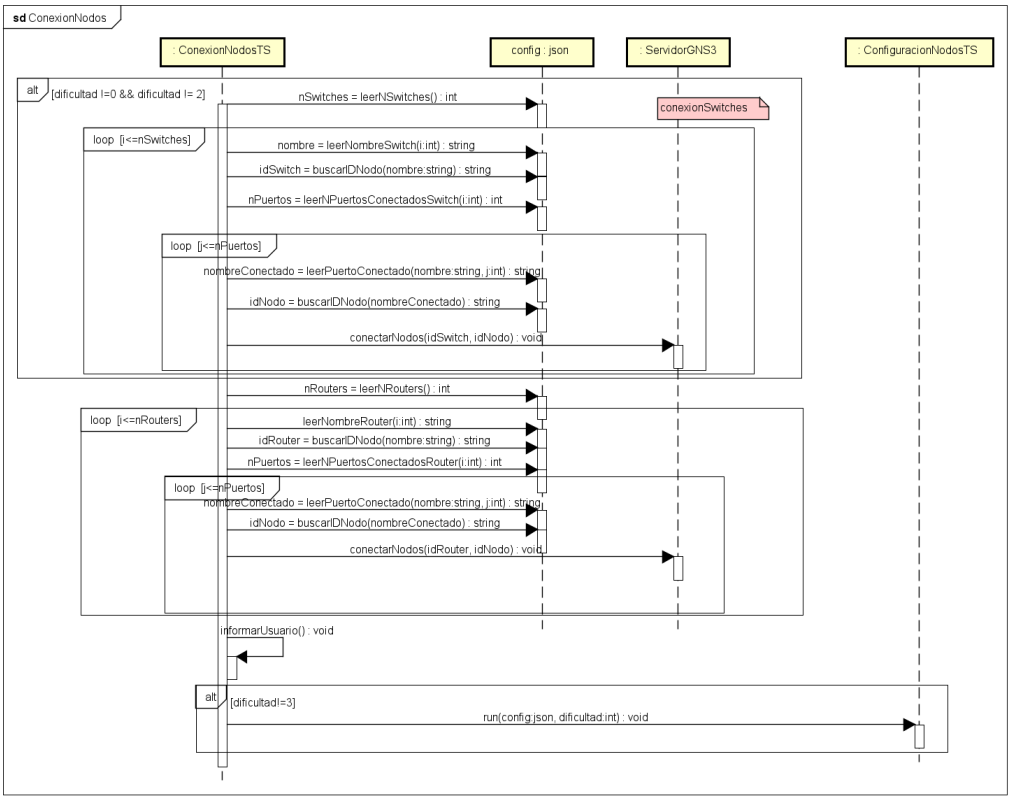

Figura 6.11: Diagrama de secuencia de diseño de la conexión de nodos

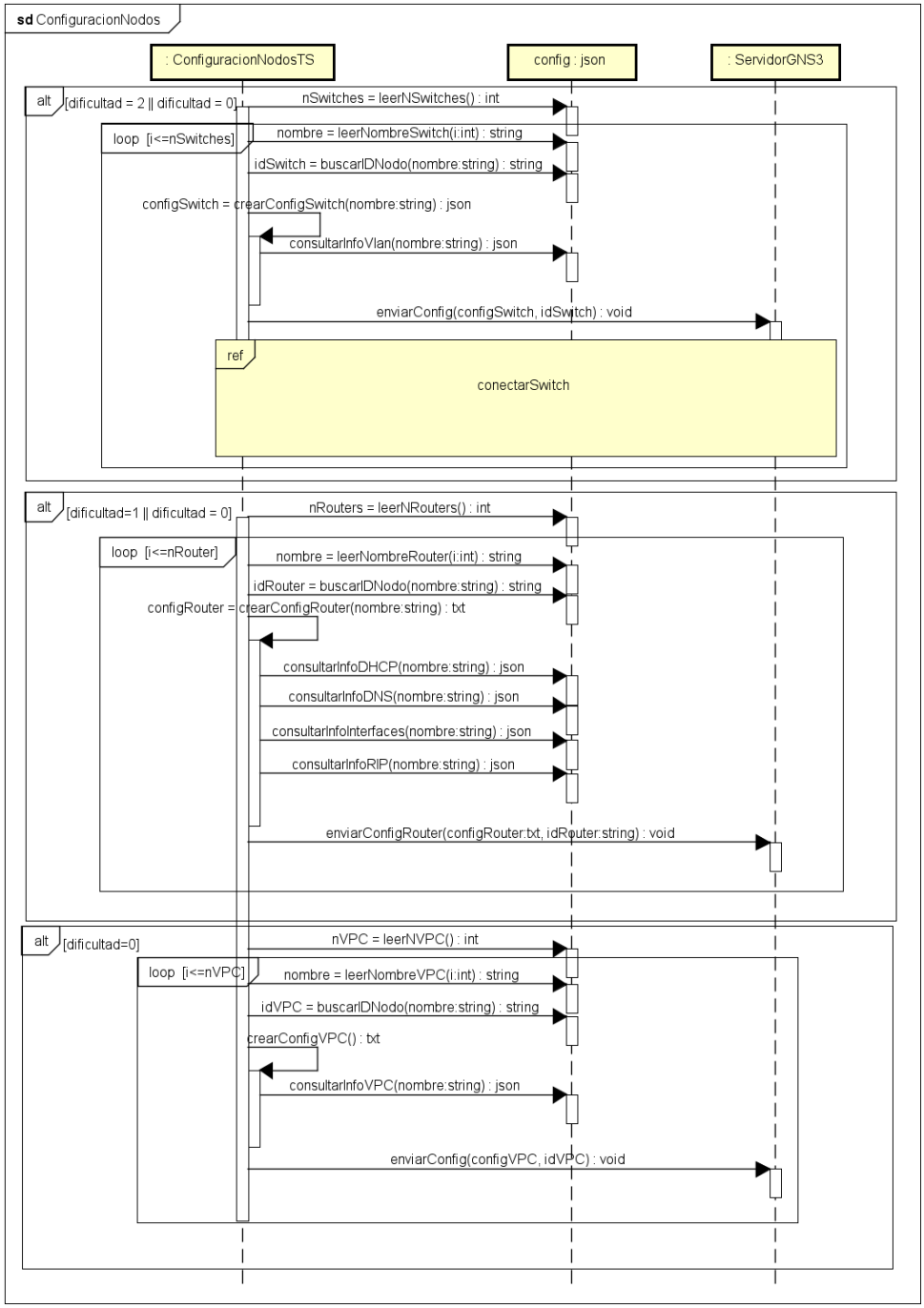

Figura 6.12: Diagrama de secuencia de diseño de la creación de nodos

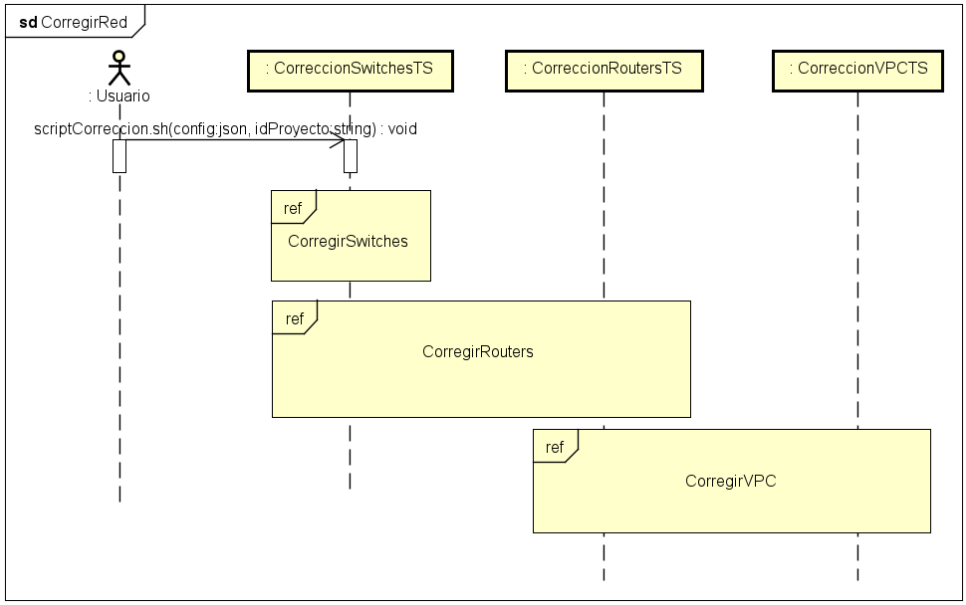

Figura 6.13: Diagrama de secuencia de diseño de la corrección de nodos

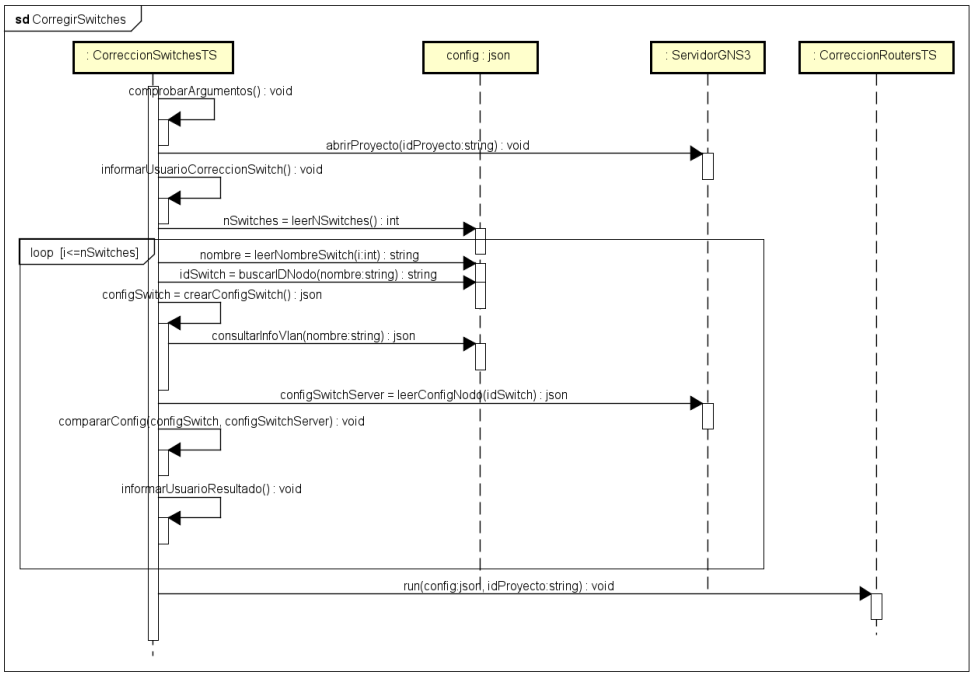

Figura 6.14: Diagrama de secuencia de diseño de la corrección de switches

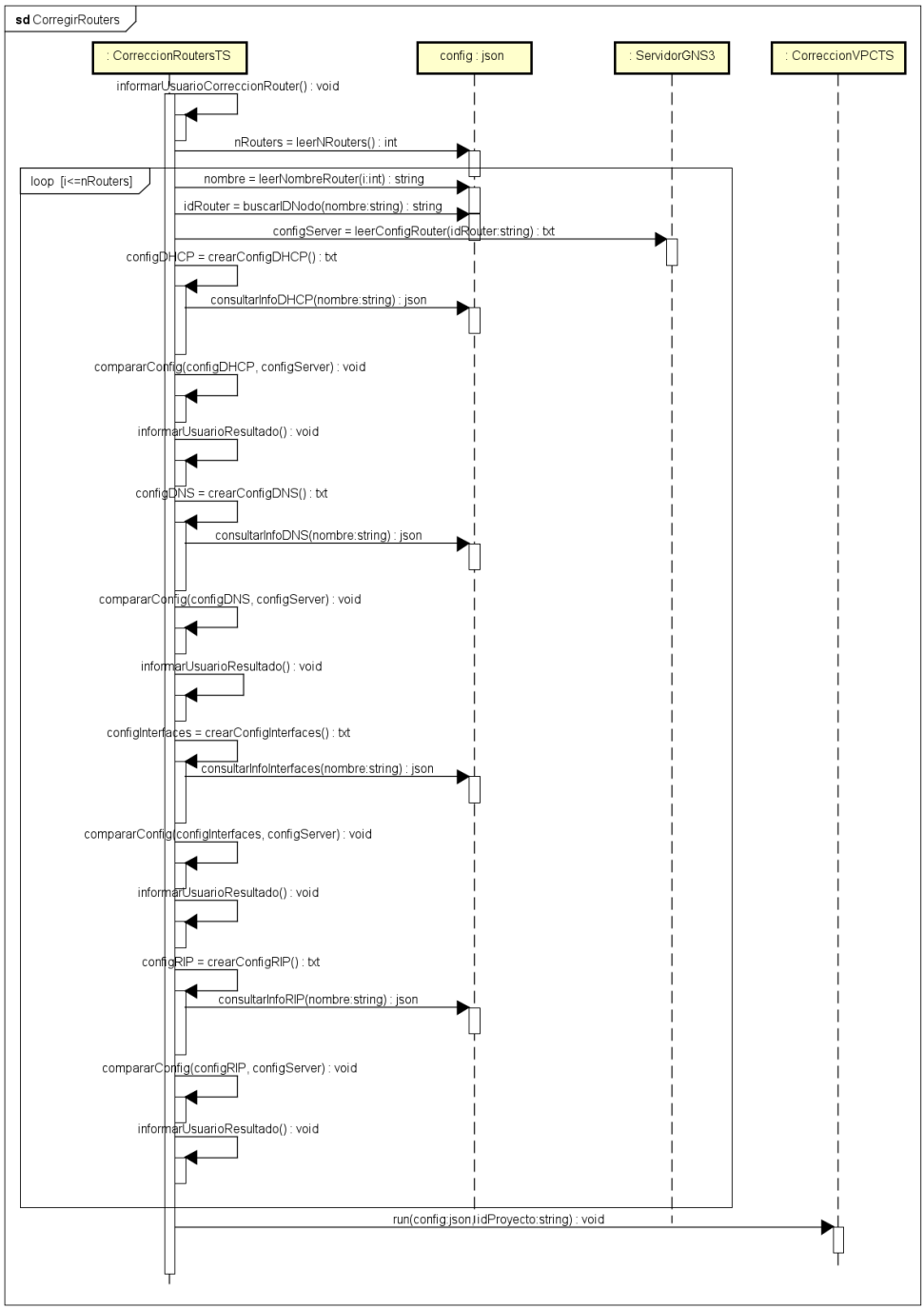

Figura 6.15: Diagrama de secuencia de diseño de la corrección de routers

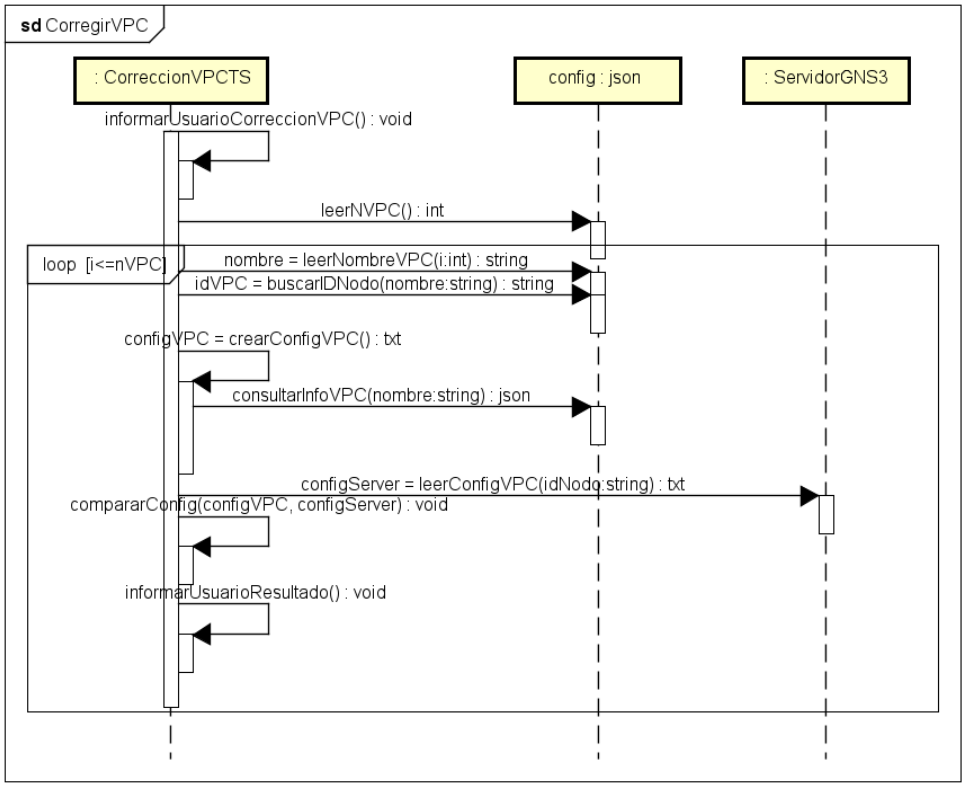

Figura 6.16: Diagrama de secuencia de diseño de la corrección de VPC

Capítulo 7

Implementación y pruebas

Una vez hemos visto el diseño final de la topología de red planificada para el taller de laboratorio y la fase de diseño de los scripts, es hora de ver el proceso de implementación y pruebas de los scripts que se encargarán de generarlo y comprobar que su configuración es correcta. Esta es la fase final del desarrollo del software.

Estos scripts se crearon con la idea de ofrecer a los usuarios una forma distinta de aprender más y de forma más práctica sobre las redes de computadoras que solo sería posible con la ayuda de GNS3, queriendo demostrar con ello el potente uso que puede darse a esta herramienta en ámbitos educativos y potencias las características que no poseen otras herramientas muy estandarizadas en el entorno educativo como Packer Tracer.

Para hacer esto posible he optado por desarrollar dos Shell Script, uno enfocado en la creación del taller y otro enfocado en la corrección del mismo. El primero necesita de un fichero JSON de configuración para leer los nodos de la topología de red a crear, sus conexiones y sus configuraciones en caso de que el usuario desea que el script las aplique, además de un nivel de dificultad que se explicará más adelante; en el caso del segundo script, el de corrección, también necesita de ese mismo fichero JSON para leer la configuración que debería haberse aplicado para conocer si es correcta o no, además de necesitar también el ID del proyecto que se quiere corregir.

El objetivo de estos dos scripts consiste en crear una topología de red para que el usuario la configure y la haga completamente funcional, una vez el usuario ha terminado de configurar la red acorde al fichero JSON de configuración, ya que la configuración de la red ha sido planeada anteriormente para que el usuario la replique, se ejecuta el script de corrección del taller, que comprobará que la red ha sido configurada correctamente.

Estos scripts y este proyecto están pensados de forma que el alumno, en caso de no querer que el script de creación le configure la red, sea él mismo el que replique la configuración de los nodos escrita en el fichero JSON, es decir, que toda la configuración de la red ha sido prevista en el fichero JSON de configuración. Esto provoca que sea recomendable para máxima fiabilidad que se aplique el script de corrección sobre proyectos GNS3 que hayan

sido creados con el script de creación, ya que al compartir estos dos el fichero JSON de configuración que contiene la información de todos los nodos de la red será mucho más preciso de corregir posteriormente con el script de corrección que leerá del mismo fichero de configuración.

Para una mayor comprensión, en la figura 7.1 se expone un diagrama que esquematiza el funcionamiento de ambos scripts, después se da una explicación del diagrama con algunos fragmentos de código, solo he incluido algunos debido a que, por la longitud del código, no caben en este documento.

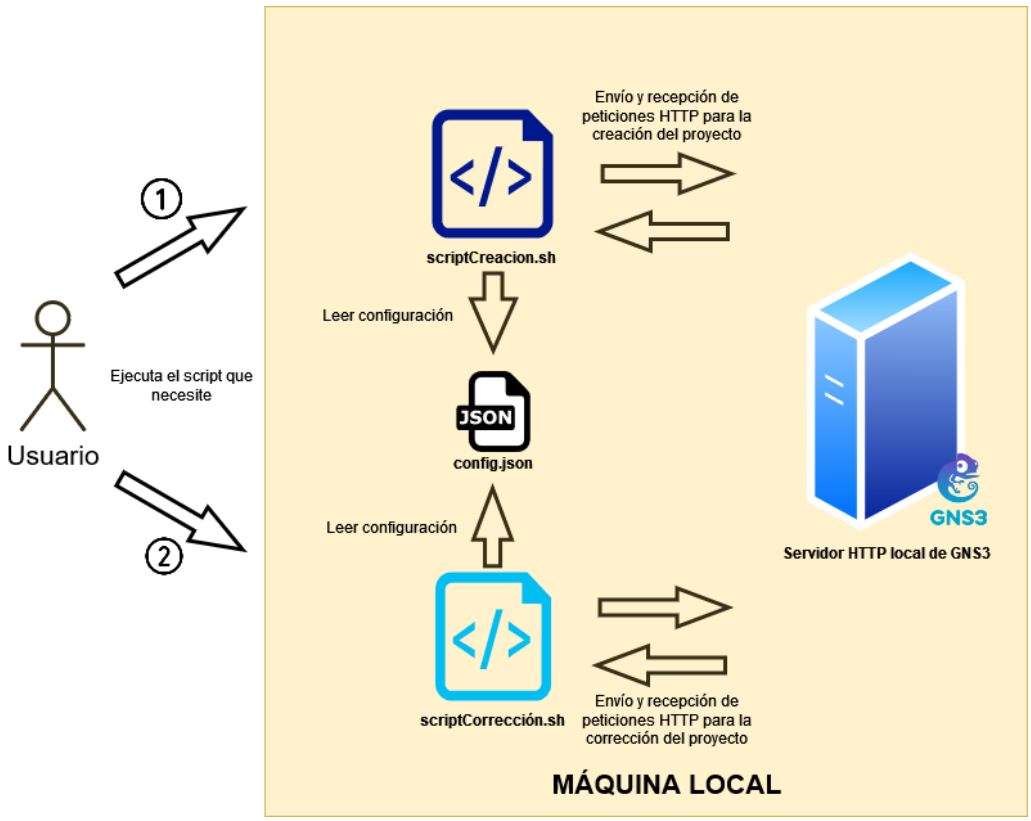

Figura 7.1: Esquema del funcionamiento de los scripts

El esquema se puede dividir en dos partes, una es la relacionada con el script de creación de los talleres de laboratorio y otra relacionada con el script de corrección de estos.

En esta primera parte el usuario ejecutará el script de creación de los talleres de laboratorio scriptCreacion.sh introduciendo dos parámetros: el primero es la ruta al fichero JSON de configuración cuya estructura se explicará más adelante y el segundo un nivel de dificultad que puede oscilar entre los siguientes valores:

- **Nivel 0:** Este es el nivel de dificultad más bajo. Si el usuario selecciona este nivel de

dificultad el script configurará automáticamente todos los nodos de la red y creará una red completamente funcional sin necesidad de intervención del usuario. Este nivel de dificultad está pensado para que el usuario, en caso de tener complicaciones y no ser capaz de configurar la red por si mismo, pueda comprobar la solución final y aprender en el proceso qué es lo que le faltaba para lograr que la red funcionara completamente.

- **Nivel 1:** En este nivel de dificultad el script configurará únicamente los routers de la red, dejando sin configurar switches y VPCS. Se ha decidido que el 1 corresponda a la configuración exclusiva de los routers debido a que la comparado con los switches los routers tienen mucha más tarea de configuración que los switches y, por ende, considero más fácil que te ofrezcan un router configurado que un switch configurado. Al igual que el nivel 0, está pensado por si al usuario se le complica la configuración de los routers.
- **Nivel 2:** Si el usuario selecciona este nivel de dificultad el script configurará únicamente los switches, dejando sin configurar routers y VPCS. Como los anteriores, está pensado para que el usuario compruebe la configuración de los switches en caso de que tuviera complicaciones a la hora de realizarla.
- **Nivel 3:** Este es el nivel de dificultad superior de todos. En caso de que el usuario seleccione este nivel de dificultad el script no configurará absolutamente ningún nodo de la red, reduciendo su ejecución a la creación y conexión de los nodos de la red, dejando al usuario la parte de configuración completa de la misma.

Una vez comienza la ejecución del script este realiza una primera fase de comprobación de errores, es decir, esta fase consiste en comprobar primero que los argumentos que el usuario ha introducido son correctos, después comprobará que los comandos necesarios para la ejecución del script están instaladas en la máquina local y después, por último, comprobará que se puede establecer una conexión con el servidor local de GNS3. En caso de que alguna de estas comprobaciones falle se informará al usuario con un mensaje de error. En el fragmento de código 7.1 se puede observar un ejemplo de comprobación de argumentos, en el código original hay muchos más. También en el fragmento de código 7.2. se puede observar cómo se comprueba la conexión con el servidor GNS3 y la comprobación de uno de los comandos.

```
1 if [ $# != 2 ]
2 then
3     echo "Número de argumentos inválido.
4         Usa la opción -h (--help) para consultar la ayuda." > /dev/stderr
5     exit 2
6 fi
```

Fragmento de código 7.1. Ejemplo de comprobación de argumentos

```
1 if [ "$(which curl)" = "" ]
2 then
3     echo "Es necesario instalar el comando curl para utilizar el script."
```

```
    > /dev/stderr
    exit 5
fi

if [ "$(curl -s -X GET http://localhost:3080/v2/version 2> /dev/null)" = "" ]
then
    echo "No se ha podido establecer la comunicación con el servidor local GNS3.
    Revisa que esté activo y vuelve a intentarlo." > /dev/stderr
    exit 6
fi
```

Fragmento de código 7.2. Ejemplo de comprobación de comando y de conexión al servidor

Además, si el usuario ejecuta el script con el argumento -h o –help el script mostrará un breve texto de ayuda para el usuario en el que se le indica el orden y el tipo de argumentos a introducir y una breve descripción de la utilidad del script.

Una vez terminada la fase de comprobación de errores comienza la ejecución real del script. Como se puede apreciar en el esquema de la figura 5.3, el funcionamiento del script se basa en peticiones HTTP constantes al servidor GNS3 aprovechándonos de su API REST, por lo que la ejecución del mismo se basa en el uso constante del comando curl y el comando jq para leer los datos del fichero JSON de configuración. Esta fase central de ejecución del script, para una mayor comprensión, se divide en 3 grandes bloques:

1. **Fase de creación de nodos:** En esta primera fase el script, en base a la configuración que ha leido del fichero JSON, crea el proyecto y los nodos uno a uno y los coloca en la posición que corresponde. En el fragmento de código 7.3. se puede ver cómo se crea un VPC, los datos han sido sustituidos por tres puntos suspensivos debido a que el JSON que se envía al servidor es muy largo y no cabe en el documento.

```
#Creacion de VPCS
if [ $(jq -r ".nVPCS" $1) -gt 0 ]
then
        for i in $(seq 1 $(jq -r ".nVPCS" $1) )
        do
                curl -s -X POST http://localhost:3080/v2/projects/$id/nodes -d '{...}'
        done
fi
```

Fragmento de código 7.3. Creación de VPCS

2. **Fase de conexión de nodos:** Esta segunda fase consiste en conectar los nodos entre sí mediante una conexión Ethernet. El algoritmo de conexión pensado parte de que todos los nodos están conectados o a un switch o a un router, por lo que se recorre cada switch y cada router en el fichero JSON conociendo a qué nodos está conectado y

conectando uno a uno mediante una petición HTTP al servidor con la función de la API REST correspondiente. El orden de los puertos sigue el mismo orden de aparición en el fichero JSON, es decir, si según el fichero de configuración el Switch1 está conectado a la máquina MV1, MV2 y al router R1, el Switch irá ocupando sus puertos en función a dicho orden, el puerto 0 para la máquina MV1, el puerto 1 para la máquina MV2 y el puerto 2 para el router R1. Importante mencionar que, en caso de que el nivel de dificultad seleccionado por el usuario sea el 2 o el 0 no se conectarán los switches en esta fase, esto es debido a una limitación de la API REST de GNS3 que no permite configurar las VLAN si tiene nodos conectados previamente, por lo que si el usuario selecciona dichos niveles de dificultad se traspasará la conexión de los nodos de los switches a la siguiente fase.

3. **Fase de configuración de nodos:** Esta tercera fase dependerá del nivel de dificultad seleccionado por el usuario: Si el usuario ha seleccionado el nivel de dificultad 0 se procederá a la configuración de todos los nodos, es decir, se configurarán routers, switches y VPCS, si selecciona el modo 1 solamente se configurarán los routers, si selecciona el modo 2 solamente se configurarán los switches y si selecciona el modo 3 esta fase no se ejecuta. La configuración de los nodos se basa en crear un fichero replicando la estructura de los ficheros de configuración de estos nodos, por ejemplo, la de los routers se basará en crear un fichero startup_config con la configuración que se precise en base al fichero de configuración. Una vez se ha creado el fichero de configuración este se envía mediante una petición HTTP al servidor, para que el nodo, al encenderse lea esa configuración. En caso de que se seleccione el modo 0 o 2 es aquí donde, tras la configuración del switch, es en esta fase en la que se procede a su conexión con el resto de nodos. Al terminar la configuración de los nodos el script elimina todos los ficheros temporales que haya necesitado durante la configuración de estos. Acerca de la configuración de las máquinas virtuales, estas no se configuran en ninguno de los modos de dificultad, se supone que están ya previamente configuradas y listas para su implementación.

El resultado de la ejecución de este script es la creación de un proyecto GNS3 que el usuario puede abrir y probar desde la propia aplicación normal, en caso de que no haya elegido el modo de dificultad 0, deberá terminar de configurar la red para que funcione en su totalidad. En la figura 7.2 se puede ver un ejemplo de un taller creado con el script.

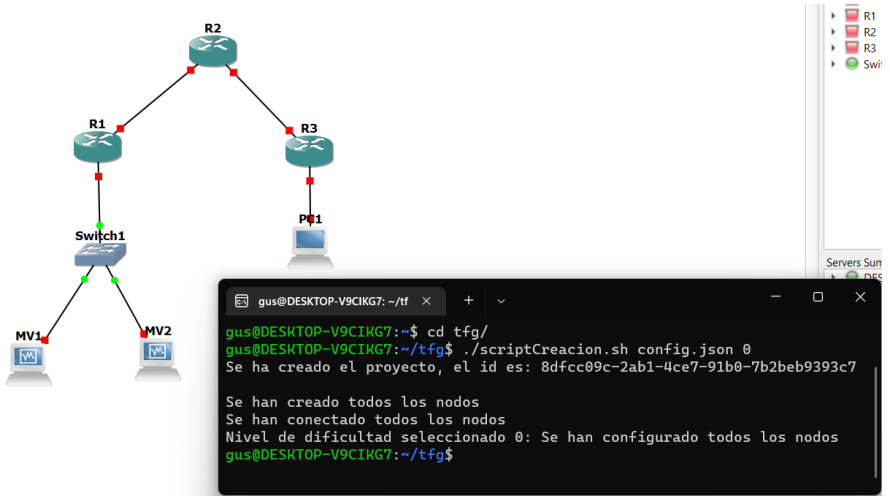

Figura 7.2: Ejemplo de uso del script de creación

En la segunda parte, una vez la red creada por el script de creación ya ha sido configurada completamente, el usuario puede ejecutar el script de corrección scriptCorreccion.sh, el cual recibe dos argumentos: El fichero JSON de configuración utilizado para crear la red con el anterior script y el ID del proyecto que ha sido generado por el script anterior.

Al igual que el script de creación, este empieza su ejecución con una comprobación de errores muy similar a la anterior: primero comprueba que todos los argumentos son correctos, después comprueba que los comandos que necesita el script están instalados en la máquina local, en esta ocasión necesitamos de dos comandos extra además de curl y jq, necesitamos la ayuda de grep para buscar texto en ficheros y de diff para conocer la existencia de diferencias entre ficheros, por lo que se comprobarán esos dos comandos además de los anteriores; por último comprueba que se puede establecer una comunicación con el servidor HTTP de GNS3, ya que necesitamos utilizar nuevamente su API REST.

Una vez ha terminado la fase de comprobación de errores comienza la ejecución del script. A la hora de la implementación del script se ha tenido en cuenta la topología inicial descrita en el apartado 5.1 como base, se ha planteado de forma que se corrijan los nodos pensando en los protocolos que soporta esa red, es decir, si al usuario le apetece prescindir de RIPv2 y configurar OSPF, el script no tendrá en cuenta este último protocolo ya que no aparece en dicha topología, además tampoco corregirá configuraciones de máquinas virtuales, ya que estas están configuradas de base antes de la creación del script por lo que se entiende que su configuración es correcta.

Nuevamente, la corrección del script se puede dividir en tres grandes fases:

1. **Corrección de switches:** La primera fase consiste en la corrección de los switches, para realizar esta tarea primero se genera un fichero con la configuración que debería tener cada switch en base a lo leido en el fichero JSON de configuración, después

se extrae la configuración real de cada switch desde el servidor HTTP de GNS3, si ambas configuraciones coinciden entonces se da por válida la configuración y se pasa al siguiente switch. La configuración que se valida para cada switch es la configuración de las VLAN, es decir, se comprueba que las VLAN estén bien configuradas y asignadas al puerto que corresponde.

2. **Corrección de routers:** Esta segunda fase consiste en la corrección de los routers, se sigue casi exactamente el mismo procedimiento que para los switches. Primero se extrae del servidor local de GNS3 la configuración real de cada router, después se comprueba en el fichero JSON la configuración que debería tener cada router y se va comprobando cada aspecto por separado en caso de que ese router deba tenerlo, DHCP, DNS, Interfaces, VLAN y RIPv2; si coincide la configuración extraída del JSON con la extraída del router entonces se da por válida la configuración y se pasa al siguiente router.

3. **Corrección de VPCS:** Esta es la última fase, el procedimiento es el mismo que para las demás fases. Primero se genera un fichero con la configuración que debería tener cada VPC en función de lo leído en el JSON, después se extrae la configuración de cada VPC desde el servidor de GNS3, se comparan ambas configuraciones y si coinciden se da por válida y se pasa a comprobar el siguiente VPC.

Es importante destacar que cuando finaliza una fase de corrección los ficheros intermedios generados para ella se eliminan, además en cada aspecto del proyecto del cual se esté validando la configuración, se informa al usuario con un mensaje indicándole si ese nodo está bien configurado o no, y en caso de que haya varios aspectos a comprobar de cada nodo en qué aspecto la configuración no es válida, siendo esta información el resultado final de la ejecución del script. En la figura 7.3 se puede observar un ejemplo de uso de este script de corrección.

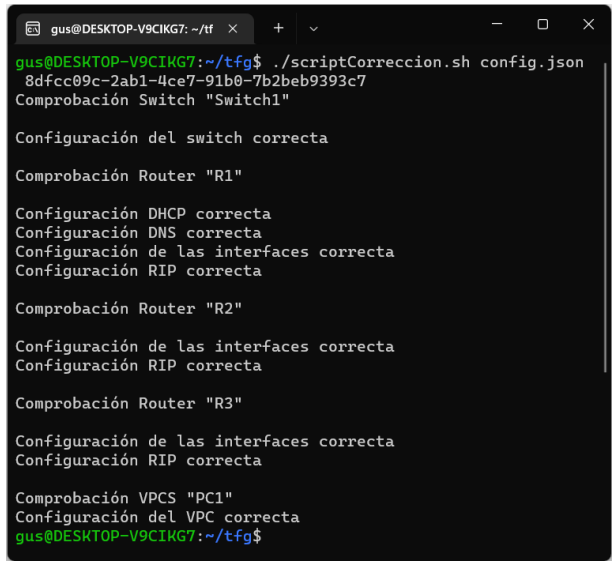

Figura 7.3: Ejemplo de uso del script de corrección

Para finalizar con este capítulo, he desarrollado una pequeña tabla de pruebas de caja negra con diversos casos para probar y poner a prueba ambos scripts en situaciones distintas partiendo de la red propuesta, para cada prueba se ha generado un fichero JSON distinto con las especificaciones deseadas para la prueba. Esta tabla se puede encontrar a continuación en la tabla 7.1, mostrando el número de prueba, el nombre de la misma, una descripción y si cada script ha superado la prueba.

Nº prueba	**Nombre**	**Descripción**	**Superado por script-Creacion**	**Superado por script-Correccion**
1	Creación de la red propuesta original	Crear la red propuesta tal y como se describe en el diseño	SI	SI
2	Creación de la red propuesta sin R2	Crear la red propuesta pero sin el router R2, creando una conexión entre R1 y R3 directamente	SI	SI
3	Creación de la red propuesta con un router extra R4	Crear la red propuesta añadiendo un router extra R4 como paso intermedio entre R3 y PC1	SI	SI
4	Creación de la red propuesta con un switch extra Switch2 y un VPC extra PC2	Crear la red propuesta pero añadiendo un switch extra Switch2 entre PC1 y R3 y añadiendo un VPC PC2 extra conectado a dicho switch pero en la misma red, sin VLAN	SI	SI
5	Creación de la red propuesta con una MV extra y una VLAN extra en el switch Switch1	Crear la red propuesta pero añadiendo una VLAN extra en Switch1 en la que se conectará la máquina virtual MV3	SI	SI
6	Creación de la red propuesta sin PC1	Crear la red propuesta pero sin PC1	SI	SI

Tabla 7.1: Resultado de las pruebas realizadas

Para cada una de las pruebas se ha tenido que generar un fichero JSON diferente con los cambios deseados para la prueba, con motivo de mostrar un fichero JSON distinto al mostrado en el capítulo de diseño y para demostrar la potencia que tiene este fichero y los scripts para crear distintas topologías de red y permitir su configuración se muestra a continuación en el fragmento de código 7.4 el fichero JSON de configuración generado para la prueba cuatro, ya que es una de las que más varía respecto a la propuesta original, para ahorrar espacio solo se muestran los campos que son diferentes al fichero original.

```
{
        "nombreTaller": "prueba4",
        "nSwitches": 2,
        "nVPCS":2,
        "switches": [
                {
                        "nombre": "Switch1",
                        "puertosEnUso": 3,
                        "maquinasConectadas": [
                                        "MV1",
                                         "MV2",
                                         "R1"
                                ],
                        "vlan": [
                                        10,
                                        20,
                                        "trunk"
                        ],
                        "x":-214,
                        "y":-3
                },
                {
                        "nombre": "Switch2",
                        "puertosEnUso": 3,
                        "maquinasConectadas": [
                                        "PC1",
                                        "PC2",
                                         "R3"
                                ],
                        "x":77,
                        "y":0
                }
        ],
        "VPCS": [
                {
                        "nombre": "PC1",
                        "puertosEnUso": 1,
                        "maquinasConectadas":"Switch2",
                        "ip": "192.168.5.2 192.168.5.1 24",
                        "x":22,
                        "y":70
                },
                                {
                        "nombre": "PC2",
                        "puertosEnUso": 1,
                        "maquinasConectadas":"Switch2",
                        "ip": "192.168.5.3 192.168.5.1 24",
                        "x":140,
                        "y":70
```

```
50                  }
51          ]
52  }
```

Fragmento de código 7.4. JSON de configuración de la prueba número cuatro

Esta prueba consiste en añadir un switch Switch2 y un VPC entre PC1 y R3, para ello no hay que hacer excesivas variaciones respecto a la red original, pero nos sirve para comprobar como habría que configurar el JSON en esta ocasión, además de demostrar con esto la universalidad de los scripts. Los cambios realizados son: añadir un switch en el campo «switches» con sus respectivos datos, que en este caso serían el nombre, los nodos conectados y sus coordenadas, ya que no va a haber VLAN configuradas, también hay que añadir un VPC nuevo, PC2, para ello lo añadimos en el campo «VPCS» con sus respectivos datos y en la configuración con una IP de la misma red que PC1, ya que no existen VLAN como en el Switch1. Importante también modificar los campos «nSwitches» y «nVPCS».

El resto de pruebas son similares, modificar el JSON añadiendo o eliminando nodos o modificando configuraciones, el script leerá ese JSON para crear la topología de red y configurar los nodos si es que el usuario lo desea, después el script de corrección leerá también el fichero JSON para comprobar que el usuario ha configurado la red correctamente. Si el usuario desea realizar una topología de red completamente distinta a la propuesta solamente tiene que modificar los campos del JSON y los scripts se encargarán del resto, es por eso que es importane destacar la importancia del JSON, ya que es una pieza fundamental de este proyecto, ya que en él radica la característica de universalidad de los scripts, permitiendo crear y corregir prácticamente cualquier topología con ellos, siendo muy útil para plantear talleres de laboratorio distintos para los alumnos con esta herramienta. En las figuras 7.4 y 7.5 se puede ver el resultado de la ejecución de los scripts con el fichero JSON de configuración de la prueba cuatro.

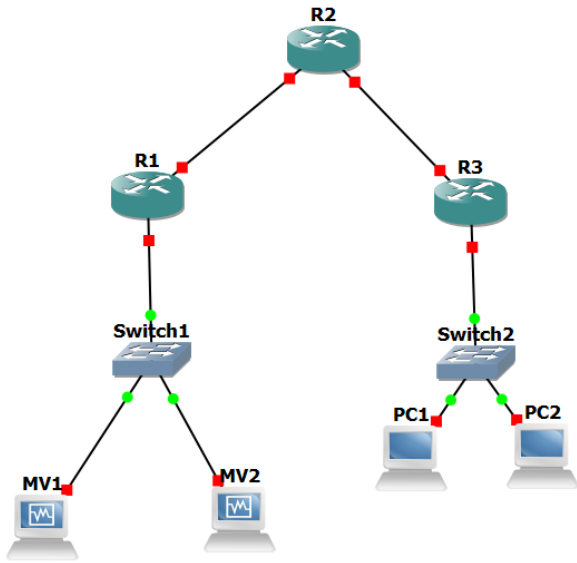

Figura 7.4: Topología de red generada por el script de creación en la prueba cuatro

```
gus@DESKTOP-V9CIKG7:~/tfg$ ./scriptCorreccion.sh ./JSONPruebas/prueba4.json 471da57f-4c84-47f8-a
d8b-e4e3ce2e1233

Comprobación Switch "Switch1"

Configuración del switch correcta

Comprobación Switch "Switch2"

Configuración del switch correcta

Comprobación Router "R1"

Configuración DHCP correcta
Configuración DNS correcta
Configuración de las interfaces correcta
Configuración RIP correcta

Comprobación Router "R2"

Configuración de las interfaces correcta
Configuración RIP correcta

Comprobación Router "R3"

Configuración de las interfaces correcta
Configuración RIP correcta

Comprobación VPCS "PC1"
Configuración del VPC correcta

Comprobación VPCS "PC2"
Configuración del VPC correcta
gus@DESKTOP-V9CIKG7:~/tfg$
```

Figura 7.5: Resultado de la corrección de la prueba cuatro

Capítulo 8

Seguimiento del proyecto

En este capítulo relataré los hechos más relevantes en cada etapa del proyecto, además de comentar las diferentes complicaciones, dificultades y cambios que hayan surgido a lo largo de este.

Empezando por la puesta en marcha para conocer bien el proyecto y los materiales que necesito para llevarlo a cabo entró el problema que retrasaría las cosas y me obligaría a hacer numerosos cambios y es el refinamiento del enfoque de la temática del mismo, es decir, el tema del TFG ha tenido siempre la misma base, pero se ha ido refinando y cambiando los enfoques hasta llegar a este resultado final, lo que ha hecho que la redacción de la introducción sea modificada unas cuantas veces para adaptar dichos cambios. También entra en juego aprender sobre GNS3 y sobre el Shell de Linux, que ha llevado bastante tiempo debido a que antes de realizar el proyecto yo no conocía GNS3, no había casi programado en Shell y nunca había tratado con un fichero JSON. En esta fase ha habido reuniones con el tutor para refinar el enfoque y estar de acuerdo entre los dos acerca de qué debía ser el resultado final del proyecto, es por ello también que esta parte ha llevado bastante tiempo.

Antes de realizar el diseño de la red tuve una reunión con el tutor para aclarar un par de puntos sobre cómo enfocar los scripts, decidimos que, para evitar problemas y complicaciones excesivas con el proyecto, los realizaríamos en torno a una red propuesta que contuviera todo lo necesario como la implementación del protocolo DHCP, DNS, RIPv2 y la utilización de nodos de distintos tipos para la familiarización del usuario con estos, todo ello utilizando el mínimo de recursos posibles para una mayor compatibilidad. Nacería aquí una nueva fase, la selección y el diseño de la red propuesta sobre la que nos basaríamos y haríamos óptima los scripts, no fue una tarea muy larga ni compleja ya que la limitación de los recursos se trataba en una de las entradas del estado el arte y en dicha entrada había numerosas propuestas de topología para afrontar dicho problema, decidí utilizar una de ahí y modificarla un poco para que encajara con nuestro proyecto, no hubo complicaciones en este aspecto.

Las fases de análisis y diseño de los script no han llevado un tiempo excesivo gracias a una reunión que tuve con el profesor Diego García Álvarez, que me ayudó mucho a comprender qué debía hacer en dichas etapas y cómo abordarlas, además de recomendarme libros y

capítulos de los mismos para leer y entender mejor lo que debo hacer en estas fases y poder plasmar una mejor explicación en la memoria. Gracias a esto he podido actuar con mayor fluidez en la realización del análisis y el diseño de los scripts y en la redacción de dichas partes de la memoria. El único problema encontrado es mi confusión a la hora de realizar dichas partes sin un sistema software orientado a objetos, por lo que en un principio veía imposible realizar un análisis y diseño, ya que no conocía patrones no orientados a objetos, pero gracias a la reunión con el profesor pude comprender cómo hacerlo y llevar dichas fases a cabo.

Comenzaría la fase más complicada de todo el proyecto, la implementación y codificación de los scripts. Como ya había hecho un periodo de puesta en marcha anteriormente para familiarizarme un poco con Shell y GNS3 y conocer su API REST esta fase no fue excesivamente complicada en ese aspecto, pero sí lo fue a la hora de pensar los algoritmos tanto de conexión de los nodos como de configuración de los mismos. La idea principal era que la configuración de los nodos fuera a través del procolo Telnet, ya que GNS3 permite conectarse a los nodos de su red desde el exterior a través de Telnet, pero no pudo ser posible debido a los numerosos errores que provocaba y la gran lentitud del mismo, tuvo que sustituirse dicha idea por crear los startup_config de los router desde el script y enviárselos al servidor a partir de una función de la API REST, lo mismo ocurre con el resto de nodos como los VPC y los switches. La idea original era que el switch de la topología fuese un router Cisco pero con un módulo para actuar como switch, lo que permitiría a los alumnos practicar dichos comandos, pero debido al mal funcionamiento de estos tuvo que ser sustituído por un switch básico de GNS3 que no permite su configuración por comandos, pero seguía ofreciendo funcionalidades de VLAN, que era lo que necesitábamos, esto provocó un cambio en el diseño de la red al tener que modificar el tipo de switch y provocó cambios en el algoritmo de conexión de los nodos debido a la limitación en su configuración que generaba. Finalmente se logró completar la implementación en un tiempo decente y se pudo redactar en la memoria y se pudo realizar unos script que, aunque al principio se planteasen para estar enfocados en una topología de red específica, se pudo hacer que fueran universales gracias a la facilidad del fichero JSON de configuración, que también fue desarrollado en esta etapa y modificado varias veces añadiendo información que se consideraba necesaria.

La fase de pruebas se realizó tras terminar la implementación de los scripts, en esta fase no hubo grandes complicaciones, pero sí hubo cambios importantes en el código para corregir problemas que ocurrían con las pruebas, como podría ser el algoritmo de conexión de los nodos que tuvo un cambio importante ya que antes estaba mal planteada, pero funcionaba para la red propuesta, después del cambio el algoritmo es más universal.

La redacción de los manuales no fue difícil ya que tenía claro el procedimiento y solo era documentarlo. El problema vino a la hora de realizar el procedimiento para máquinas Ubuntu, ya que en algunas máquinas Dynamips generaba problemas con los routers Cisco pero en otras máquinas no, esto ha quedado documentado en el manual para que el usuario sea consciente de este fallo, que no tiene que ver con los scripts sino con Dynamips.

Para finalizar, la fase de redacción de las partes finales de la memoria no tuvo ninguna complicación ya que era solo redactar, y las ideas estaban más que claras tras todo el proceso vivido, solo era plasmarlas adecuadamente y corregir fallos de redacción en las fases anteriores para hacer una memoria más clara y consistente.

Capítulo 9

Conclusiones

Como fin de este proyecto se procede a hablar de las conclusiones finales del mismo. Lo primero que se puede decir del proyecto es que se han cumplido con éxito todos los objetivos propuestos en el primer capítulo satisfactoriamente, además de cumplir con todos los requisitos que se identificaron en la fase de análisis, tanto los funcionales como los no funcionales y los requisitos del proyecto.

Las reuniones con el tutor han servido para aclarar el enfoque y el rumbo con el que abordar el proyecto y resolver numerosas dudas que me han surgido a lo largo del mismo, siempre con paciencia y comprensión, además de mantener un estrecho contacto por Microsoft Teams para las dudas rápidas o compartir avances.

Con este trabajo de fin de grado se ha logrado demostrar la utilidad y potencia de GNS3 en el ámbito educativo, demostrándolo además con los scripts de taller de laboratorio, el gran reto del proyecto. Estos scripts han servido para ofrecer una forma nueva de realizar estas prácticas de laboratorio típicas de las asignaturas fundamentales de redes de computadoras solamente posible con GNS3, permitiendo que, con un script, sea posible generar una topología de red y que esta incluso esté configurada siendo completamente funcional si el usuario lo desea y además permitir una corrección automática con otro script para que el usuario o el profesor sea consciente de qué partes están bien configuradas o que partes están mal configuradas.

De forma personal, el resultado de los scripts ha salido incluso mejor de lo esperado originalmente, ya que la idea original era que para evitar complicaciones excesivas no fuese un script pensado para distintas topologías de red, sino uno que crease una red específica y otro script que la corrigiese, dejando para un trabajo futuro la posibilidad de crear cualquier topología, pero finalmente debido a conocer a fondo la API REST de GNS3 y gracias a la potencia de la tecnología JSON y de Shell ha sido posible universalizar ambos scripts y que opere de la forma más rápida y eficiente posible.

Por último, también de forma personal, este proyecto ha sido una gran puesta a prueba a mis conocimientos y a todo lo aprendido a lo largo de la carrera, juntando conocimientos

de todos los campos de las asignaturas y enfocados al campo de las redes y a las tecnologías de la información, que era mi objetivo principal cuando se me apareció el reto de realizar un trabajo de fin de grado, hacer algo relacionado con las redes y las tecnologías de información. También ha sido un gran ejercicio de planificación y constancia, para ser consciente en todo momento de los plazos que tengo y qué trabajo diario debo hacer para cumplirlos. También ha servido para ser consciente de mi propio potencial, aprendiendo que soy capaz dominar aspectos nuevos de la informática y refinar los ya conocidos si me esfuerzo lo suficiente. Como broche final a esta evaluación personal quiero aclarar que termino quedando completamente satisfecho con el resultado del proyecto.

9.1. Líneas de trabajo futuras

Esta sección está pensada para que las futuras personas que quieran realizar un TFG o un proyecto relacionado con el mío sepan futuras mejoras que pueden realizar al proyecto. El código ha sido subido a un repositorio en GitHub, para que cualquier persona interesada pueda echar un vistazo al código para futuras modificaciones o mejoras.

Mientras terminaba y quedaban claros cuales iban a ser los resultados del proyecto se me han ido ocurriendo diversas mejoras que podrían refinar más el propósito del proyecto.

Entre estas mejoras se encuentran una que llevaría el objetivo a un nivel superior, se podría llevar el objetivo de los scripts al nivel de una aplicación, que se encargue de generar las redes y corregirlas o simplemente que genere el JSON de forma automática, sin necesidad de generar un JSON a mano y ejecutar dos scripts que no tienen ningún tipo de interacción con el usuario, todo con una interfaz gráfica y que agrupara la funcionalidad de ambos scripts.

Por último, también se podría investigar una forma de automatizar tecnologías alternativas a IOS de Cisco con Dynamips para implementar diversas tecnologías de otros fabricantes, como podría ser la GNS3 MV o Qemu.

Bibliografía

[1] Gaudenz Alder. Diagrams.net. draw.io [Avaiable] https://www.diagrams.net/.

[2] Anónimo. Bourne shell. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Bourne_Shell.

[3] Anónimo. Calidad de servicio. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Calidad_de_servicio.

[4] Anónimo. Cliente-servidor. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Cliente-servidor.

[5] Anónimo. Desarrollo en cascada. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Desarrollo_en_cascada.

[6] Anónimo. Disyunción exclusiva. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Disyunci%C3%B3n_exclusiva.

[7] Anónimo. Frame relay. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Frame_Relay.

[8] Anónimo. Introducción a json. JSON [Avaiable] https://www.json.org/json-es.html.

[9] Anónimo. Ipv6. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/IPv6.

[10] Anónimo. Javascript. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/JavaScript.

[11] Anónimo. Multiprotocol label switching. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Multiprotocol_Label_Switching.

[12] Anónimo. Open shortest path first. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Open_Shortest_Path_First.

[13] Anónimo. Protocolo de transferencia de hipertexto. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto.

[14] Anónimo. Red privada virtual. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Red_privada_virtual.

[15] Anónimo. Routing information protocol. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Routing_Information_Protocol.

[16] Anónimo. Shell de unix. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Shell_de_Unix.

[17] Anónimo. Shell script. Wikipedia [Avaiable] https://en.wikipedia.org/wiki/Shell_script.

[18] Anónimo. Telnet. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Telnet.

[19] Anónimo. Unix. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Unix.

[20] Anónimo. Virtual private lan service. Wikipedia [Avaiable] https://es.wikipedia.org/wiki/Virtual_Private_LAN_Service.

[21] Anónimo. Wireless ad hoc network. Wikipedia [Avaiable] https://en.wikipedia.org/wiki/Wireless_ad_hoc_network.

[22] Arlow, Jim, and Ila Neustad. Uml 2. Anaya Multimedia [Avaiable] https://almena.uva.es/permalink/34BUC_UVA/eseo99/alma991004944729705774.

[23] atareao. Tratar archivos json en bash. Atareao.es [Avaiable] https://atareao.es/como/tratar-archivos-json-en-bash//.

[24] F. Bushmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-oriented software architecture: A system of patterns. volume 1. Wiley, 1996.

[25] Caltico. ¿qué precio tiene microsoft project? Caltico [Avaiable] https://www.caltico.es/microsoft-project-precio/.

[26] Joaquín Gómez Carmona. Propuesta de manual de prácticas de laboratorio de redes utilizando el emulador gns3. Universidad Central "Marta Abreu" de Las Villas [Avaiable] https://dspace.uclv.edu.cu/bitstream/handle/123456789/7888/Joaqu%C3%ADn%20G%C3%B3mez.pdf?sequence=1&isAllowed=n.

[27] Clownix. Cloonix: Virtual network creator. Github [Avaiable] https://github.com/clownix/cloonix.

[28] Gerald Combs. Wireshark. go deep. Wireshark [Avaiable] https://www.wireshark.org/.

[29] Apache Software Foundation. Welcome! - the apache http server project. Apache. [Avaiable] https://httpd.apache.org/.

[30] The Free Software Foundation. Gnu bash. GNU [Avaiable] https://www.gnu.org/software/bash/.

[31] The Free Software Foundation. Gnu/linux. GNU [Avaiable] http://www.gnu.org/.

[32] M. Fowler. Patterns of enterprise application architecture. Addison Wesley, 2002. [Avaiable] https://almena.uva.es/permalink/34BUC_UVA/12tq2h1/alma991005750009705774.

[33] Free Software Fundation. El sistema operativo gnu y el movimiento del software libre. GNU [Avaiable] https://www.gnu.org/home.es.html.

[34] Inc. GitHub. Github: Where the world builds software · github. GitHub [Avaiable] https://github.com/.

[35] Juan Herruzo Herrero. Reglas y control de calidad automatizado para el lenguaje vensim. Universidad de Valladolid [Avaiable] https://uvadoc.uva.es/bitstream/handle/10324/50393/TFG-G5252.pdf?sequence=1&isAllowed=y.

[36] GNU General Public License. Notepad++. Notepad++ [Avaiable] https://notepad-plus-plus.org/.

[37] Canonical Group Limited. Ubuntu on windows. Microsoft Store [Avaiable] https://www.microsoft.com/store/productId/9NBLGGH4MSV6.

[38] WriteLatex Limited. Overleaf, editor de latex online. Overleaf [Avaiable] https://es.overleaf.com.

[39] Gordon Lyon. Npcap. Npcap [Avaiable] https://npcap.com/.

[40] Microsoft. Download .net framework. Microsoft [Avaiable] https://dotnet.microsoft.com/en-us/download/dotnet-framework.

[41] Microsoft. May 10, 2022—kb5013943 (os build 22000.675). Soporte de Microsoft [Avaiable] https://support.microsoft.com/en-us/topic/may-10-2022-kb5013943-os-build-22000-675-14aa767a-aa87-414e-8491-b6e845541755.

[42] Microsoft. Microsoft project. Microsoft 365 [Avaiable] https://www.microsoft.com/es-es/microsoft-365/project/project-management-software.

[43] Microsoft. Microsoft teams. Microsoft [Avaiable] https://www.microsoft.com/es-es/microsoft-teams/group-chat-software.

[44] Microsoft. ¿qué es el subsistema de windows para linux? Microsoft Docs [Avaiable] https://docs.microsoft.com/es-es/windows/wsl/about.

[45] Gemma María García Mínguez. Evaluación de herramientas de simulación para una asignatura de diseño y configuración de redes. Universidad de Valladolid [Avaiable] https://uvadoc.uva.es/bitstream/handle/10324/42482/TFG-G4150.pdf?sequence=1&isAllowed=y.

[46] nsnam. ns3 — a discrete-event network simulator for internet systems. ns3 [Avaiable] https://www.nsnam.org/.

[47] Oracle. Oracle vm virtualbox. Oracle [Avaiable] https://www.virtualbox.org/.

[48] PcComponentes. Microsoft windows 11 pro 64bits oem. PcComponentes [Avaiable] https://www.pccomponentes.com/microsoft-windows-11-pro-64bits-oem.

[49] LLC. SolarWinds Worldwide. Gns3 — the software that empowers network professionals. GNS3 [Avaiable] https://www.gns3.com/.

[50] Astah Team. Astah: Premier diagramming, modeling software & tools. Astah [Avaiable] https://astah.net/.

[51] Astah Team. Princing for individual licenses of astah software. Astah [Avaiable] https://astah.net/pricing/individual/.

[52] Cisco Team. Ccna: Introduction to networks. Cisco Networking Academy [Avaiable] https://www.netacad.com/es/courses/networking/ccna-introduction-networks.

[53] Cisco Team. Cisco packet tracer - networking simulation tool. Cisco Networking Academy [Avaiable] https://www.netacad.com/es/courses/packet-tracer.

[54] Cisco Team. Professional certifications. Cisco [Avaiable] https://www.cisco.com/c/en/us/training-events/training-certifications/certifications/professional.html.

[55] Cisco Team. What is virl (virtual internet routing lab). The Cisco Learning Network [Avaiable] https://learningnetwork.cisco.com/s/article/what-is-virl-virtual-internet-routing-lab-x.

[56] Debian Team. Debian - el sistema operativo universal. Debian [Avaiable] https://www.debian.org/index.es.html.

[57] GNS3 Team. ubridge, bridge por udp tunnels, ethernet, taip and vmnet interfaces. Github [Avaiable] https://github.com/GNS3/ubridge.

[58] Visual Paradigm Team. Visual paradigm. Visual Paradigm [Avaiable] https://www.visual-paradigm.com/.

[59] VMware Team. Vmware. VMware [Avaiable] https://www.vmware.com/es.html.

[60] Riverbed Technology. Winpcap. WinPcap [Avaiable] https://www.winpcap.org/.

[61] Profesor titular de la asignatura Redes de Computación de la Facultad de Ingeniería de la Universidad Don Bosco en El Salvador. Guía de laboratorio 10: Simulador gns3. Universidad Don Bosco [Avaiable] https://www.udb.edu.sv/udb_files/recursos_guias/informatica-tecnologico/redes-de-comunicacion/2019/i/guia-10.pdf.

[62] Linus Torvalds. Git. Git [Avaiable] https://git-scm.com/.

[63] Instituto técnico de Roque. Configuración automática de dirección sin estado (slaac). Instituto técnico de Roque [Avaiable] http://itroque.edu.mx/cisco/cisco1/course/module8/8.2.4.3/8.2.4.3.html.

[64] W3C. Extensible markup language (xml). XML [Avaiable] https://www.w3.org/XML/.

[65] Herbert Xu. Dash. apana.org [Avaiable] http://gondor.apana.org.au/~herbert/dash/.

[66] Mario Javier Gil Cevallos y Verónica Paola Berruz Silva. Integración de la materia laboratorio de telemática para la facultad técnica usando el simulador gráfico de redes gns3. Universidad Católica de Santiago de Guayaquil [Avaiable] http://repositorio.ucsg.edu.ec/bitstream/3317/1354/1/T-UCSG-PRE-TEC-ITEL-6.pdf.

Apéndice A

Manuales

En este primer apéndice se proceden a detallar los manuales necesarios para la instalación, uso y mantenimiento de los scripts resultado de este TFG. El manual de instalación incluirá todos los pasos necesarios para la instalación del scripts en la máquina del usuario, en el manual de usuario se enseñará el uso de los scripts y un tutorial de cómo configurar correctamente la red propuesta para este TFG y en el manual de mantenimiento restricciones y problemas obtenidos en el desarrollo que el usuario deberá tener en cuenta.

El manual está pensado para enseñar al usuario a instalar y ejecutar el taller de laboratorio y su corrección para la red propuesta para el taller, si este desea hacer alguna modificación deberá correr por su cuenta.

A.1. Manual de despliegue e instalación

En este manual se enseñará a instalar todo lo necesario para los scripts, tanto para su uso en Windows 10 y 11 como para su uso en sistemas Ubuntu.

Para empezar será necesaria la instalación de los siguientes programas: VirtualBox y GNS3 en sistemas Ubuntu y VirtualBox, GNS3, WSL y Ubuntu en WSL para sistemas Windows.

El primer paso será obtener todos los archivos necesarios para la ejecución del script y para la creación de la red, estos ficheros son:

1. Un fichero .ova que contiene las dos máquinas virtuales listas para importar y completamente configuradas
2. Un fichero .bin con la imágen ISO del Router Cisco 2691
3. Un fichero .json con la configuración predefinida de la red propuesta

4. Un fichero .sh con el script de creación del taller de laboratorio
5. Un fichero .sh con el script de corrección del taller de laboratorio

En la figura A.1 se puede encontrar una imagen con los archivos que se deben tener.

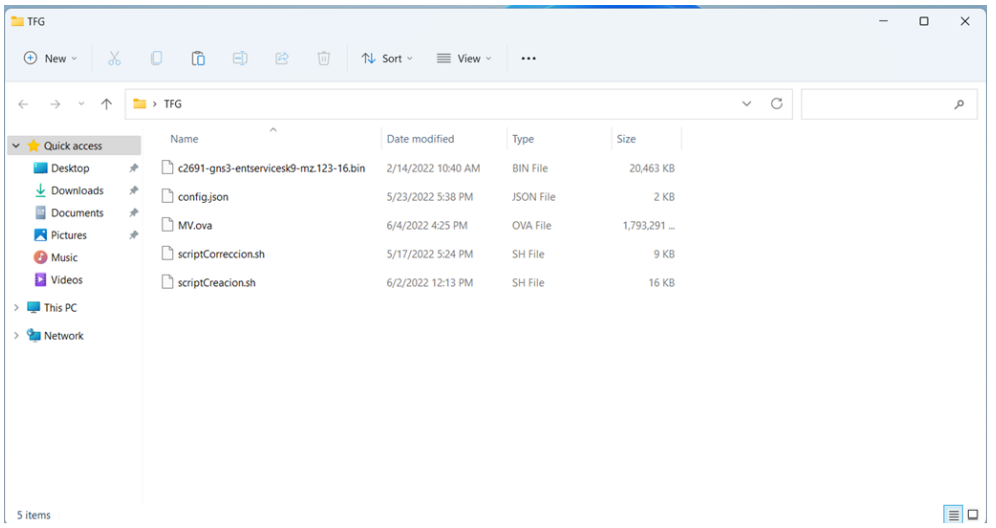

Figura A.1: Archivos necesarios para la ejecución del script.

Una vez tenemos listos todos los ficheros se procederá a instalar VirtualBox, el procedimiento es distinto para los sistemas Ubuntu y los sistemas Windows.

Para los sistemas Windows será necesario acceder a la página oficial de VirtualBox[47], una vez allí se deberá descargar el fichero de instalación para los host Windows, una vez descargado lo ejecutamos y seguimos los pasos del asistente de instalación hasta que finalice la instalación. En las figuras A.2 y A.3 se muestran imágenes que enseñan este proceso.

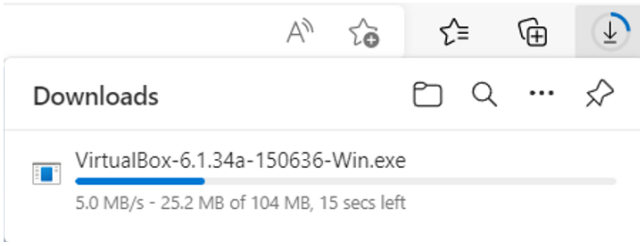

Figura A.2: Descarga de VirtualBox en Windows

Figura A.3: Asistente de instalación de VirtualBox en Windows

En el caso de los sistemas Ubuntu, para su instalación será necesario utiliza el comando **sudo apt-get install virtualbox** como se muestra en la figura A.4. y hay dos formas, desde la terminal con el comando **virtualbox** o desde el lanzador de aplicaciones de Ubuntu, en la figura A.5. se muestra la ejecución desde terminal.

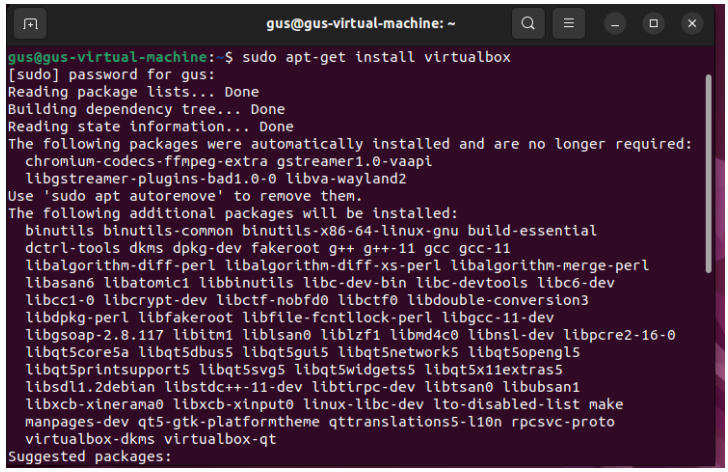

Figura A.4: Instalación de VirtualBox desde Ubuntu.

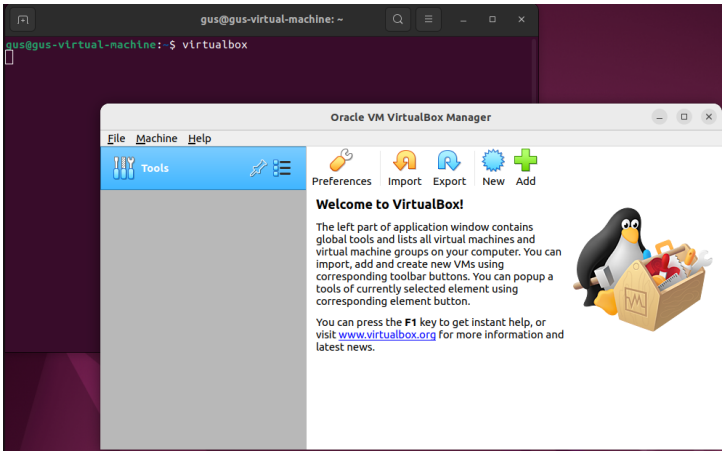

Figura A.5: Ejecución de VirtualBox desde terminal.

A partir de ahora el proceso es el mismo para los dos tipos de sistemas. Una vez hemos instalado VirtualBox lo ejecutamos, accedemos al menú de archivo y seleccionamos la opción «Importar servicio virtualizado», una vez hecho esto se nos desplegará un menú, en el cual examinaremos y seleccionaremos el archivo MV.ova que habíamos obtenido anteriormente, seguimos el asistente y se nos habrán importado las dos máquinas virtuales, UbuntuV y UbuntuWeb, que serán las máquinas virtuales de la topología de red del taller, estas máquinas recordemos que están completamente configuradas. Las figuras A.6, A.7 y A.8 ilustran el proceso de importación de estas.

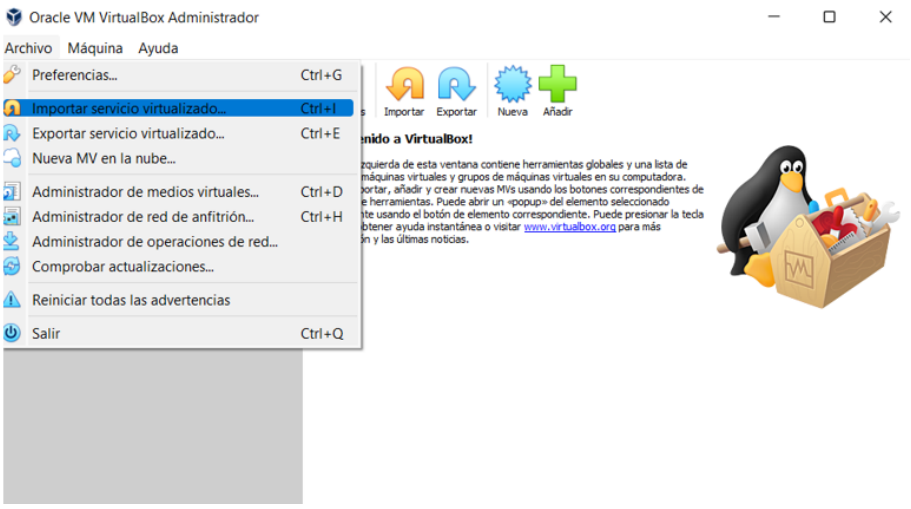

Figura A.6: Selección de la opción de «Importar Servicio Virtualizado» en VirtualBox

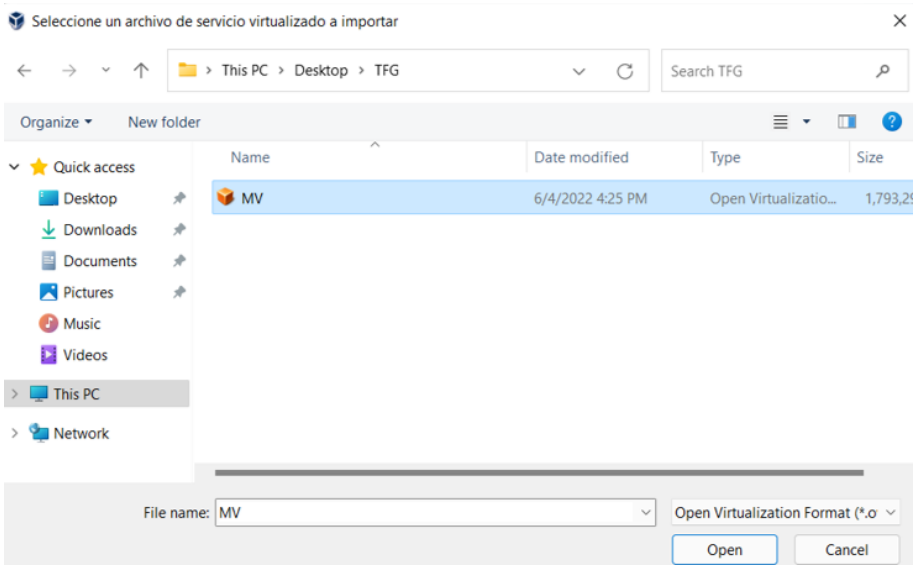

Figura A.7: Selección del fichero MV.ova

Figura A.8: Resultado de la importación en VirtualBox

Con esto las máquinas virtuales están completamente configuradas y listas para su uso en la topología de red. Por defecto, las máquinas virtuales deberían venir con el adaptador de red en el modo No Conectado, en caso de que no sea así es importante cambiarlo a dicho estado, ya que sino GNS3 no será capaz de utilizar la máquina virtual para la topología. La configuración correcta se ve reflejada en la figura A.9.

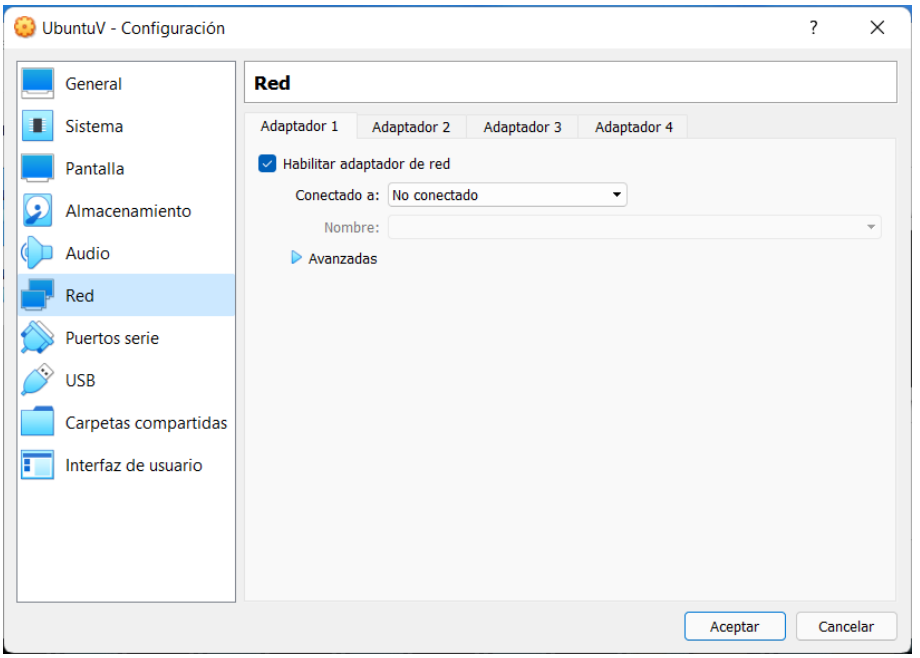

Figura A.9: Modo correcto del adaptador de red de las máquinas virtuales para la topología.

Una vez hemos instalado VirtualBox, se puede proceder a instalar GNS3, el proceso una vez más difiere dependiendo del sistema que estamos usando.

Para sistemas Windows será necesario bajar el instalador desde la página oficial, una vez descargado lo ejecutamos y seguimos los pasos del asistente, requerirá que instalemos WinPcap[60], Npcap[39] y Wireshark[28], todo lo hará el propio asistente de instalación de GNS3, el usuario solo debe limitarse a seguir los pasos. En las figuras A.10 y A.11 se muestra la descarga y la ejecución del asistente de instalación de GNS3.

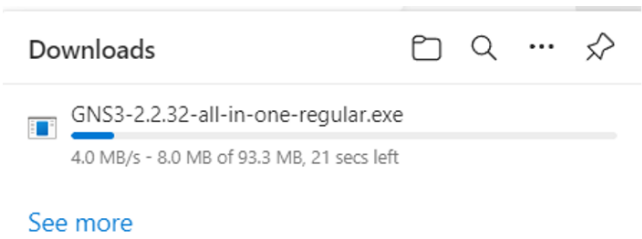

Figura A.10: Descarga del asistente de instalación de GNS3 en Windows

Figura A.11: Asistente de instalación de GNS3 en Windows.

En Ubuntu el proceso varía, para empezar será necesario añadir el repositorio de GNS3 con el comando **sudo add-apt-repository ppa:gns3/ppa**. Una vez añadido el repositorio hará falta actualizarlos con el comando **sudo apt update** y proceder a la instalación de GNS3 y del servidor con el comando **sudo apt install gns3-gui gns3-server**, durante la instalación aparecerá un mensaje sobre la instalación de uBridge[57] y sobre Wireshark, sera muy importante decir que sí a ambos mensajes. En las figuras A.12, A.13, A.14 y A.15 se puede ver el proceso de instalación descrito.

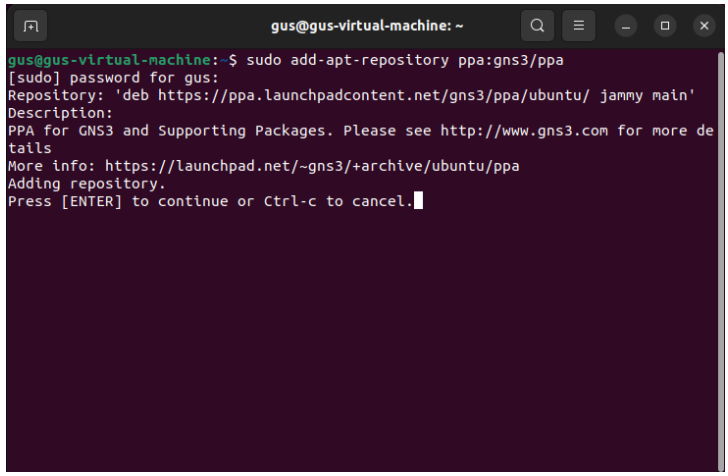

Figura A.12: Adición de los repositorios de GNS3 en Ubuntu

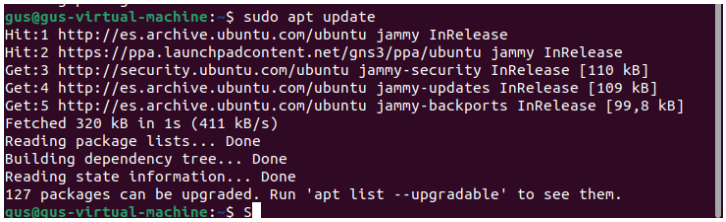

Figura A.13: Actualización de los repositorios de Ubuntu

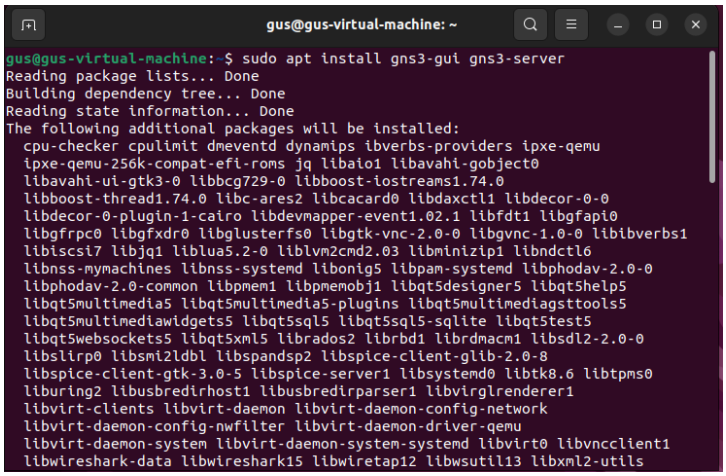

Figura A.14: Instalación de GNS3 en Ubuntu

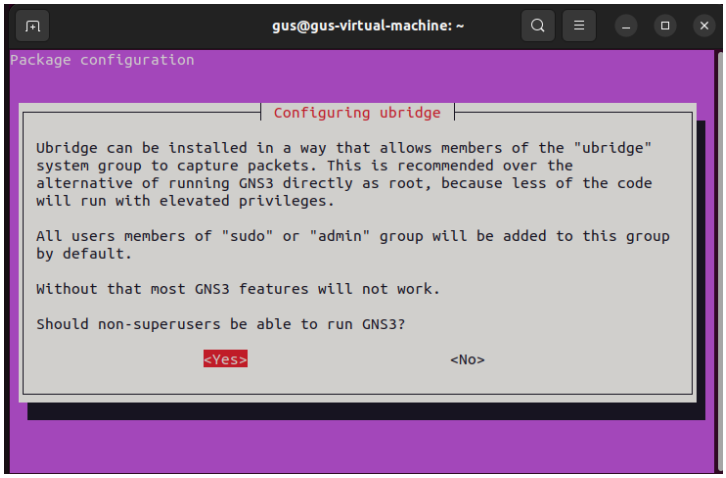

Figura A.15: Aviso acerca de la instalación de uBridge en Ubuntu

Con esto ya tenemos GNS3 perfectamente instalado en ambos tipos de sistemas. Será necesario hacer una configuración en la aplicación para que los scripts funcionen bien. Nada más instalar la aplicación se nos solicitará configurar el servidor GNS3, la configuración que debemos aplicar es que el servidor sea ejecutado en la máquina local y en el puerto TCP 3080. En las figuras A.16 y A.17 se enseña la configuración que se debe seguir.

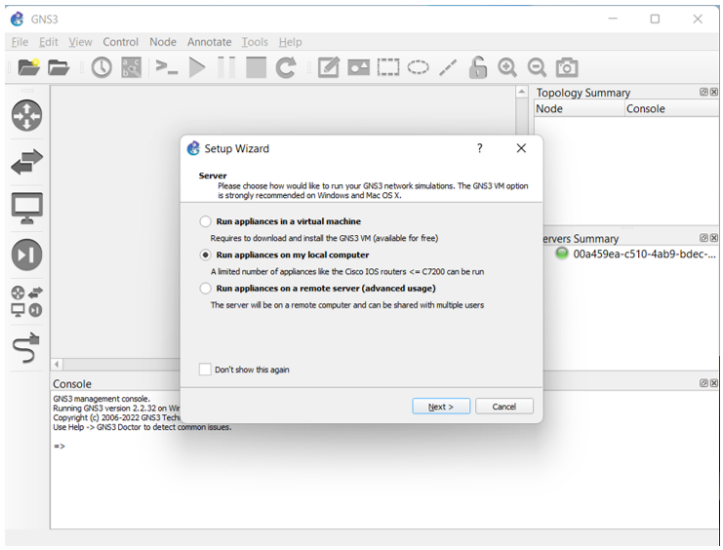

Figura A.16: Selección de localización del servidor en la configuración incial.

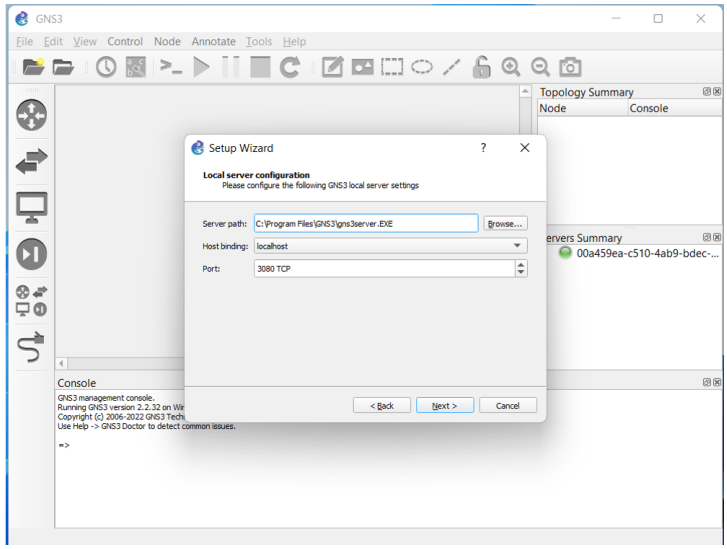

Figura A.17: Configuración del servidor GNS3.

Después, en la aplicación, vamos al menú «Edit» y vamos a la opción «Preferencias», una vez allí vamos al apartado «Server» y deberemos desmarcar la opción que dice «Protect server with password», esto es debido a que el script no utilizará autenticación con contraseña para autenticarse frente al servidor GNS3, por lo que si este la requiere el script no funcionará. En la figura A.18 se muestra la opción desmarcada.

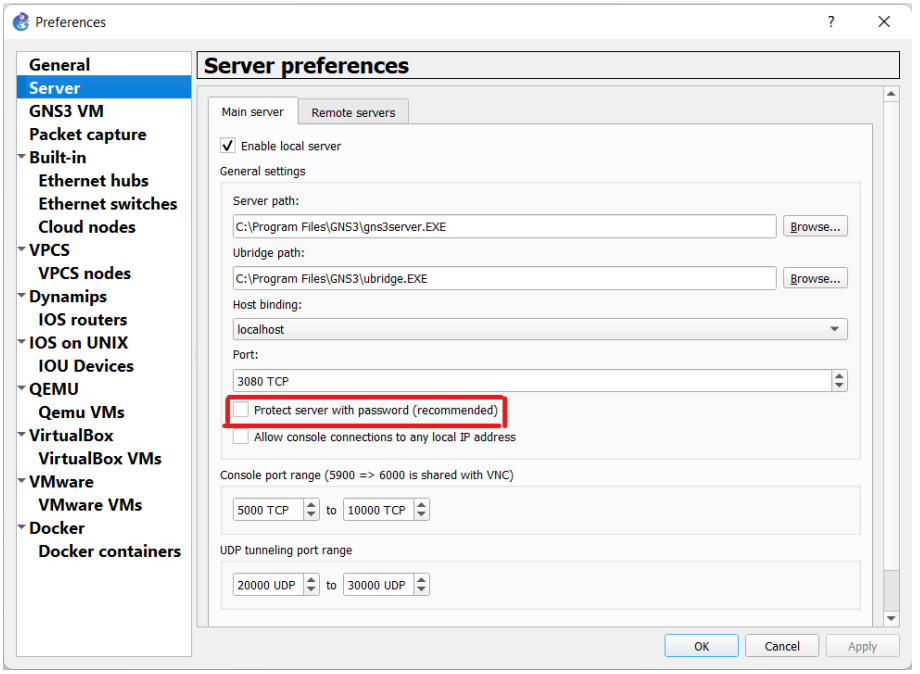

Figura A.18: Opción «Protect server with password» desmarcada

Con esto ya tenemos GNS3 listo para su uso y el servidor listo para que el script se comunique con él.

En el caso de las máquinas Windows todavía quedaría pendiente la instalación de WSL y de Ubuntu para WSL, ya que es la mejor forma de ejecutar los scripts en máquinas Windows.

Para instalar WSL será necesario acceder a la aplicación «Activar o desactivar las características de Windows», consiste en una aplicación interna en sistemas Windows 10 y 11 que permite activar o desactivar funcionalidades adicionales en los sistemas. Una vez abrimos la aplicación deberemos activar la opción «Subsistema de Windows para Linux» y darle al botón de Aceptar, esto hará que la aplicación instale tal característica, al finalizar solicitará un reinicio que debe realizarse antes de continuar. En la figura A.19 se muestra la aplicación con la opción a activar en ella.

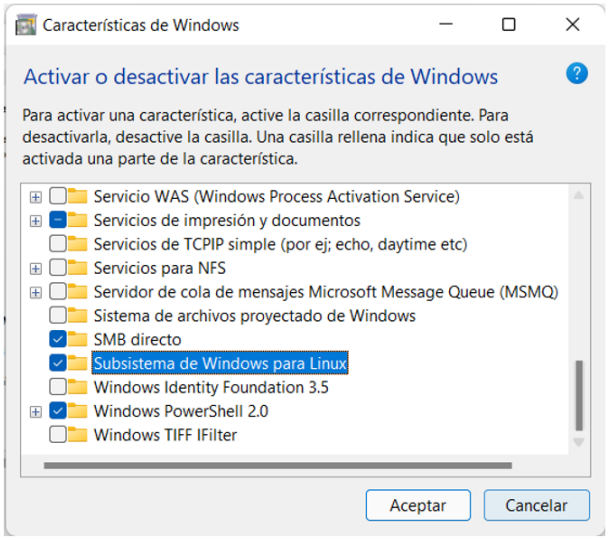

Figura A.19: Instalación de WSL en Windows.

Una vez instalado WSL y reiniciado el sistema, se debe instalar Ubuntu para WSL, ya que por defecto WSL viene vacío sin un sistema. Para esto, se debe acceder a la Microsoft Store e instalar la aplicación «Ubuntu on Windows»[37] como se muestra en la figura A.20. una vez instalado se puede abrir perfectamente como cualquier otra aplicación del sistema.

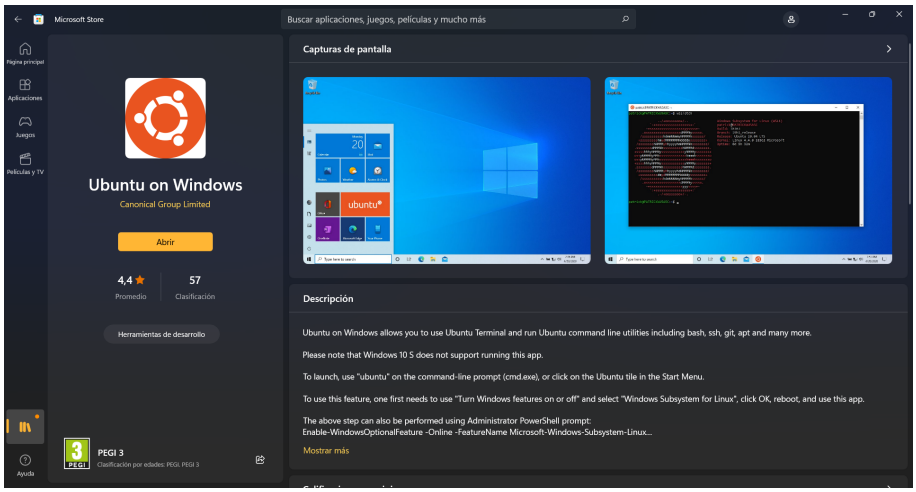

Figura A.20: Ubuntu on Windows en la Microsoft Store

En común para las máquinas Ubuntu y Windows, será necesario instalar los comandos jq y curl en caso de que no estén instalados por defecto, para ello utilizamos los comandos **sudo apt-get install jq** y **sudo apt-get install curl**, en caso de que no encuentre alguno

de los dos comandos en los repositorios podemos actualizarlos con **sudo apt-get update**. En las figuras A.21 y A.22 podemos ver el procedimiento de instalación de ambos comandos.

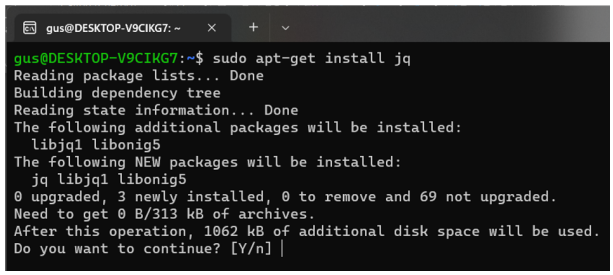

Figura A.21: Instalación del comando jq

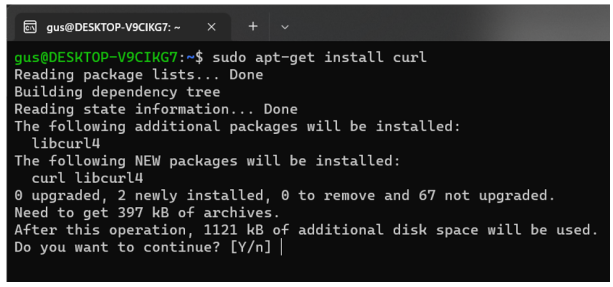

Figura A.22: Instalación del comando curl

Para finalizar, hay que colocar los ficheros restantes en ubicaciones correctas, en las máquinas Windows los scripts y el JSON deben estar en el dominio del WSL o, desde el WSL acceder a la carpeta del sistema Windows, el sistema Windows está montado en la ubicación **/mnt/c**. En cambio, el fichero .bin que corresponde a la imágen del router Cisco 2690 debe estar colocado en una ubicación específica, en concreto en la ruta **/GNS3/images/IOS/** como se muestra en la figura A.22.

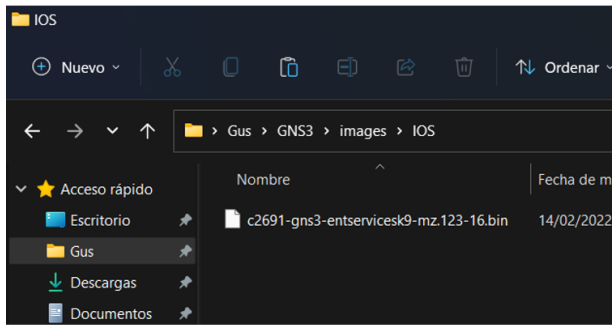

Figura A.23: Ubicación correcta de la imagen IOS del router

A.2. Manual de mantenimiento

En este manual se relatará aspectos a tener en cuenta para prevenir errores en el usuario final, es importante tener todos estos detalles en cuenta antes de ejecutar los scripts. Todos los detalles mencionados están basados en la experiencia propia, por lo que puede haber detalles a tener en cuenta que no estén reflejados debido al desconocimiento de los mismos.

Es recomendable el uso de GNS3 y de los scripts en máquinas Windows 10 o Windows 11, no se recomienda su uso en máquinas Windows con versiones anteriores a Windows 10 versión 2004, entre las que también se incluyen versiones como Windows 8.1, 8, 7, Vista o XP. Esto es debido a que WSL se incluyó en Windows como característica a partir de dicha versión de Windows 10, siendo necesario para la ejecución de los scripts, Windows 11 trae esta característica de serie, por lo que no hay problema con su uso en dicha versión. De hecho, todo el desarrollo y las pruebas del script ha sido desarrollado utilizando el WSL de Windows 11, como se ha podido ver en el capítulo cuatro.

El uso de los scripts y de GNS3 en máquinas Ubuntu es posible y es funcional, pero es posible que de lugar a más errores que si se ejecuta en máquinas Windows. Me he encontrado en diversas máquinas Ubuntu un error con Dynamips, aunque es cierto que en otras máquinas ha funcionado perfectamente. El error con Dynamips es un problema ajeno a los scripts y perteneciente al funcionamiento propio de la aplicación con los nodos de este tipo como los routers Cisco, los scripts se pueden ejecutar sin problemas y cumplen su función a la perfección ignornando este error, pero si este error con Dynamips ocurre puede ser que los routers no puedan encenderse. Para asegurarse de estar libre de errores con Dynamips recomiendo el uso de los scripts en máquinas Windows 10 y 11.

A fecha 05/06/2022 en Windows 11 se ha podido encontrar un error relacionado con VirtualBox, en el cual si se instalaba la actualización acumulativa KB5013943[41] podrían fallar las aplicaciones que utilizan .NET Framework[40] como es el caso de VirtualBox, por lo que, hasta que Microsoft u Oracle solucionen este problema, es recomendable no instalar dicha actualización para que no afecte al funcionamiento de VirtualBox y se puedan ejecutar las máquinas virtuales de la topología de red. Este error tampoco está relacionado con los scripts pero sí con el funcionamiento de la aplicación en sí.

También, para el correcto funcionamiento de los scripts y como se ha mencionado en el manual de instalación, cabe recordar tres aspectos en la configuración a tener en cuenta:

- Las máquinas virtuales han de tener el adaptador de red en modo «No conectado» como se puede ver en la figura A.9 del manual de instalación
- El servidor GNS3 debe ser local y estar ubicado en el puerto TCP 3080 como se muestra en la figura A.17 y debe estar sin protección por contraseña como se muestra en la figura A.18.
- Para que el servidor encuentre bien la imagen del router Cisco 2691 debe estar ubicada en la ruta **GNS3/images/IOS/** y tener el nombre concreto **c2691-gns3-entservicesk9-mz.123-16.bin** como se muestra en la figura A.22

A.3. Manual de usuario

En el manual de usuario se enseñará al usuario a ejecutar los scripts y la solución al taller de laboratorio de la red propuesta por si el usuario desea comprobar la solución y cómo ejecutar el script de corrección.

Para comenzar el taller de laboratorio, si hemos seguido el manual de instalación al pie de la letra, se debe ejecutar el script de creación con los siguientes argumentos:

1. El fichero JSON de configuración

2. El nivel de dificultad, del 0 al 3 siendo 0 el nivel más bajo donde se da la red totalmente configurada, el 1 el siguiente nivel en el que solamente se dan los routers configurados, el 2 donde solo se dan los switches configurados y el 3 que sería el nivel más alto en el que no se da configurado ningún nodo de la red.

En la figura A.24. se puede ver un ejemplo de ejecución del script en el modo 3 y con el fichero json de configuración de la red propuesta. Este script devuelve un ID de proyecto que recomiendo apuntar porque será necesario para el script de corrección.

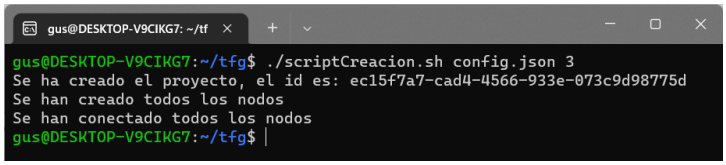

Figura A.24: Ejecución del script en el modo 3.

En este caso el usuario ha decidido ejecutar el script con el modo 3, por lo que el script no configurará ningún nodo para que el usuario sea el que complete la configuración entera de todos los nodos. La configuración de referencia es la que está escrita en el capítulo 6, en el que se habla del diseño de la topología de red.

A partir de ahora se van a mostrar las **SOLUCIONES** de la configuración del taller, por lo que si el usuario no quiere conocerlas y quiere descubrir por su cuenta la solución se le sugiere ir hasta el final del manual en el que se enseña a utilizar el script de corrección.

Al abrir GNS3 nos encontramos con que se ha creado un proyecto llamado «taller» que es el nombre que en el JSON se le ha dado al taller, si se abre muestra la topología, los nodos creados y conectados pero no configurados, por lo que es el turno de configurarlos. En la figura A.25. nos encontramos la topología como debe aparecer en GNS3 si todo ha sido configurado correctamente en la instalación.

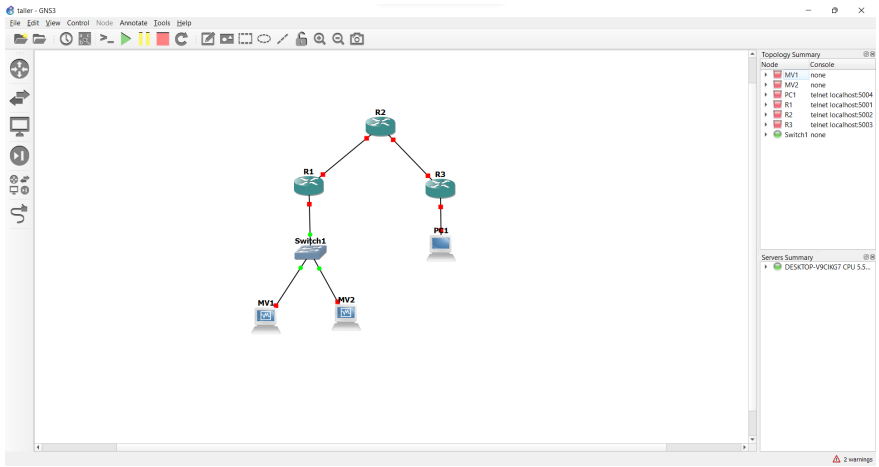

Figura A.25: Ejecución de la topología creada por el script en GNS3.

Empezaremos por configurar los switches, en este caso solo tenemos uno, el Switch1, la configuración es muy simple ya que GNS3 solo permite la configuración de switches mediante interfaz gráfica dando click derecho al nodo y seleccionando «Properties».

Según el diseño de la red existen 2 VLAN: la 10 y la 20. La 10 pertenece a la red 192.168.1.0/24 y la 20 a la red 192.168.2.0/24, el nodo que pertenece a la VLAN 10 es el host MV1 que está conectado en la interfaz 0 y el nodo que pertenece a la VLAN 20 es el host MV2 que está conectado en la interfaz 1, por lo que configuraremos las interfaces del switch para que las interfaces a las que están conectados esos nodos tengan el modo acceso con la VLAN correspondiente y la interfaz en la que está conectada el router tenga el modo dot1q (equivalente en GNS3 al modo trunk) y la VLAN 1 siendo esta la interfaz 2, quedando la configuración como se muestra en la figura A.26.

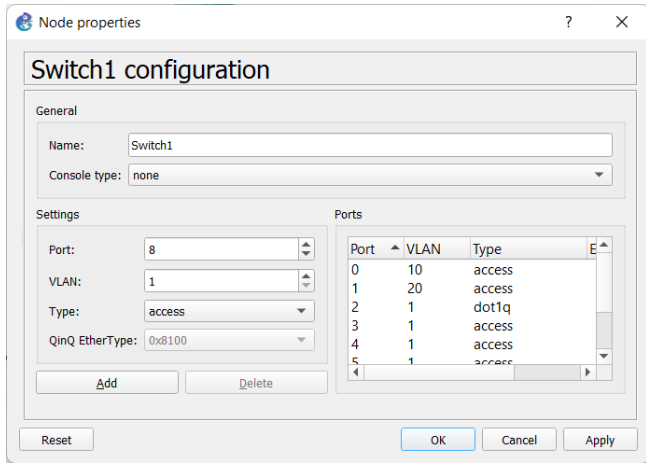

Figura A.26: Resultado correcto de la configuración del switch.

Una vez configurado el switch, empezaremos la configuración de los routers configurando el router R1, este router es el más complejo de los tres ya que es el encargado de proporcionar servicios DHCP y DNS a la VLAN 10 y además de tener que encapsular las dos VLAN en interfaces virtuales. Empezaremos configurando los servicios DHCP y DNS, para ello será necesario encender el router con click derecho y la opción «Start» y conectarnos a la consola dando click derecho sobre él y seleccionando la opción «Console», después esperamos a que se encienda y ejecutaremos el comando «enable» para acceder al router. Después, ejecutaremos los siguientes comandos para habilitar el DHCP de la VLAN 10:

```
Router#configure terminal
Router(config)#ip dhcp pool RED10
Router(dhcp-config)#network 192.168.1.0 255.255.255.0
Router(dhcp-config)#default-router 192.168.1.1
Router(dhcp-config)#dns-server 192.168.1.1
Router(dhcp-config)#exit
```

Con esto tenemos el DHCP activo para la VLAN 10, procederemos a configurar el servidor DNS y a registrar el dominio server.com para que haga referencia a la máquina MV2 con IP 192.168.2.2 con los siguientes comandos:

```
Router(config)#ip dns server
Router(config)#ip host server.com 192.168.2.2
```

El siguiente paso sería configurar las interfaces, empezaremos con la interfaz FastEthernet0/0, que es en la que debemos crear interfaces virtuales para encapsular las dos VLAN, los comandos a realizar son los siguientes:

```
Router(config)#interface fastEthernet 0/0.10
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#ip address 192.168.1.1 255.255.255.0
Router(config-subif)#exit
Router(config)#interface fastEthernet 0/0.20
Router(config-subif)#encapsulation dot1Q 20
Router(config-subif)#ip address 192.168.2.1 255.255.255.0
Router(config-subif)#exit
```

Para terminar con esta interfaz debemos encender la interfaz física para poder encender con ella las virtuales, con los siguientes comandos podemos hacerlo:

```
Router(config)#interface fastEthernet 0/0
Router(config-if)#no shutdown
Router(config-if)#exit
```

Una vez tenemos la interfaz FastEthernet0/0 correspondiente a las VLAN configuradas podemos configurar la otra interfaz, la FastEthernet0/1, que está conectada al router R2, con los siguientes comandos:

```
Router(config)#interface fastEthernet 0/1
Router(config-if)#no shutdown
Router(config-if)#ip address 192.168.3.1 255.255.255.0
Router(config-if)#exit
```

Ahora ya tenemos todas las interfaces correctamente configuradas, solamente quedaría configurar el protocolo RIPv2 para que anuncie al resto de routers las redes a las que tiene acceso, para ello utilizaremos los siguientes comandos:

```
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 192.168.1.0
Router(config-router)#network 192.168.2.0
Router(config-router)#network 192.168.3.0
Router(config-router)#exit
```

Con esto ya tenemos la configuración del router R1 completa, para finalizar debemos hacer permanente la configuración escribiéndola en el startup-config, para eso utilizamos el siguiente comando:

```
Router#copy running-config startup-config
```

A continuación procedemos a configurar el router R2, en este caso solamente hay que configurar las interfaces y el protocolo RIPv2 para configurarlo hacemos lo mismo que con el router R1, click derecho «Start» y click derecho «Console» para acceder a la consola. Empezaremos configurando las interfaces FastEthernet0/0, que es la que se comunica con el router R1 y FastEthernet0/1 que es la que se comunica con el router R3. Empezaremos configurando las interfaces, para ello primero deberemos encender el router de la misma forma que el router R1 y acceder a la consola, ejecutamos el comando «enable» para acceder al CLI del router y ejecutaremos los siguientes comandos:

```
Router#configure terminal
Router(config)#interface fastEthernet 0/0
Router(config-if)#no shutdown
Router(config-if)#ip address 192.168.3.2 255.255.255.0
Router(config-if)#exit
Router(config)#interface fastEthernet 0/1
Router(config-if)#no shutdown
Router(config-if)#ip address 192.168.4.1 255.255.255.0
Router(config-if)#exit
```

Una vez configuradas las interfaces es hora de configurar el protocolo RIPv2 con los mismos comandos que en el router R1 pero con las redes que están conectadas a este router:

```
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 192.168.3.0
Router(config-router)#network 192.168.4.0
Router(config-router)#exit
```

Una vez configurado el protocolo RIPv2 quedaría concluida también la configuración del router R2, pero es muy importante guardar la configuración en el startup-config para que el script de corrección pueda leerla, para ello utilizamos el mismo comando que para R1:

```
Router#copy running-config startup-config
```

Una vez guardada la configuración, para terminar con la configuración de los routers quedaría únicamente la configuración del router R3, que es muy similar a la del router R2 ya que solamente hay que configurar el protocolo RIPv2 y las interfaces de red.

Las dos interfaces que nos encontramos en este router son la FastEthernet0/0 que está conectada al router R2 y la FastEthernet0/1 que está conectada al PC1. Para configurar las interfaces de red utilizamos los siguientes comandos:

```
Router#configure terminal
Router(config)#interface fastEthernet 0/0
Router(config-if)#no shutdown
Router(config-if)#ip address 192.168.4.2 255.255.255.0
Router(config-if)#exit
Router(config)#interface fastEthernet 0/1
Router(config-if)#no shutdown
Router(config-if)#ip address 192.168.5.1 255.255.255.0
Router(config-if)#exit
```

Y, para finalizar con la configuración del último router, procedemos a configurar el protocolo RIPv2 como en los demás routers para anunciarles las redes a las que tiene acceso directo, con los siguientes comandos:

```
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 192.168.4.0
Router(config-router)#network 192.168.5.0
Router(config-router)#exit
```

Importante, al igual que en los otros routers, hay que guardar la configuración en el startup-config con el siguiente comando:

```
Router#copy running-config startup-config
```

Solamente quedaría para finalizar el taller configurar el PC1, al cual solamente es necesario asignar una IP y un gateway para su funcionamiento, para ello primero lo encendemos con click derecho y «Start» y accedemos a la consola con click derecho y «Console», una vez dentro de la consola ejecutamos el siguiente comando para asignar una IP:

```
VPCS> ip 192.168.5.2 192.168.5.1 24
```

Y guardamos la configuración con el comando:

```
VPCS> save
```

FIN DE SOLUCIÓN

Con esto se da por finalizada la configuración de los nodos de la red, para su comprobación haremos tres pruebas, la ejecución del script de corrección, un ping desde PC1 a MV1 y un acceso al servidor web de MV2 desde MV1 estas dos últimas sirven para corroborar la veracidad del resultado del script de corrección.

Empezaremos con la ejecución del script de corrección, para ello ejecutaremos el script de corrección con los siguientes argumentos:

- El mismo fichero JSON utilizado en el script de creación con la configuración de la red
- El ID del proyecto GNS3 creado con el script de creación

El resultado de la ejecución nos indica que se ha configurado todo correctamente y que nuestra red es completamente funcional, mostrándonos un mensaje por cada nodo y cada parte de la configuración indicándonos si dicha configuración es correcta o no, en nuestro caso todo está configurado correctamente, como se puede ver en la figura A.27.

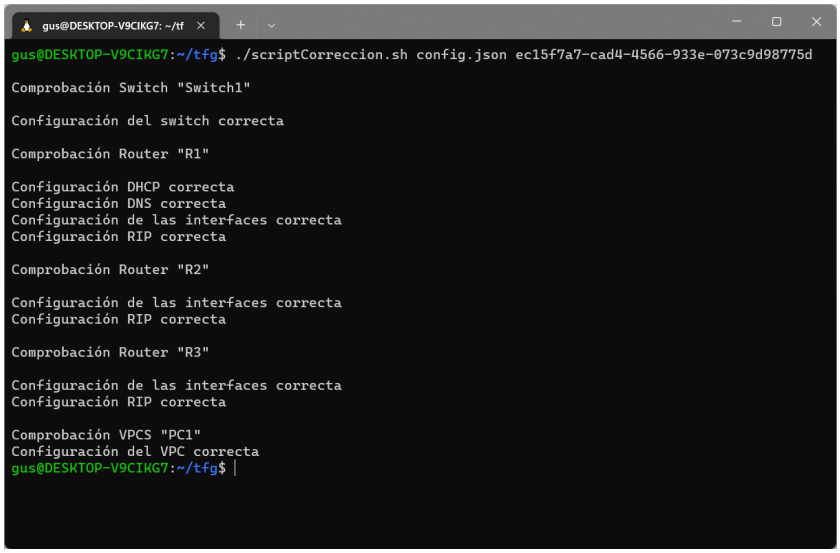

Figura A.27: Corrección del taller proporcionada por el script de corrección

Para corroborar esta corrección haremos primero un ping desde PC1 a MV1 y a MV2, para comprobar que el protocolo RIP, el protocolo DCHP y las interfaces de todos los nodos están bien configuradas, el resultado es satisfactorio, recibiendo PC1 respuesta de MV1 y de MV2, como podemos ver en las figuras A.28 Y A.29, esto también nos indica que R1 tiene bien configurado el DHCP ya que MV1 recibe IP de este. Para realizar esta prueba accedemos a las máquinas virtuales, siendo la contraseña de estas la misma que el nombre del usuario: ubuntuv para la máquina MV1 y server.com para la máquina MV2.

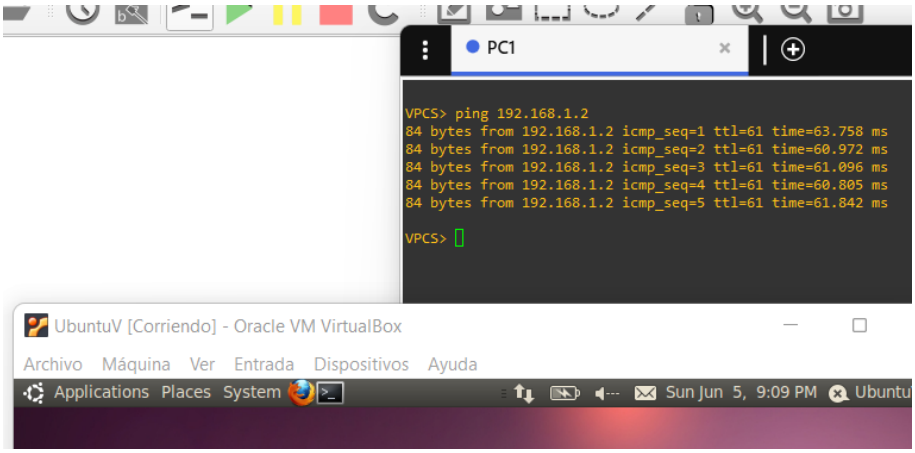

Figura A.28: Ping desde la máquina PC1 a MV1

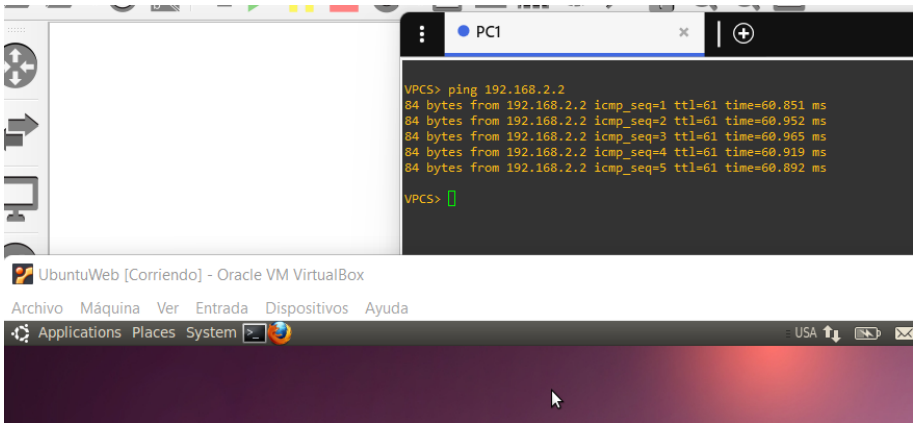

Figura A.29: Ping desde la máquina PC1 a MV2

Como última prueba y para finalizar el manual de usuario, y como prueba de la configuración del protocolo DNS y el enrutamiento entre VLAN, se procede a acceder al servidor web server.com de la máquina MV2 desde la máquina MV1. Para ello abrimos Firefox en la máquina MV1 y accedemos a la dirección server.com, vemos que tarda un poquito pero finalmente accede, dando la prueba por satisfactoria. El resultado puede verse en la figura A.30.

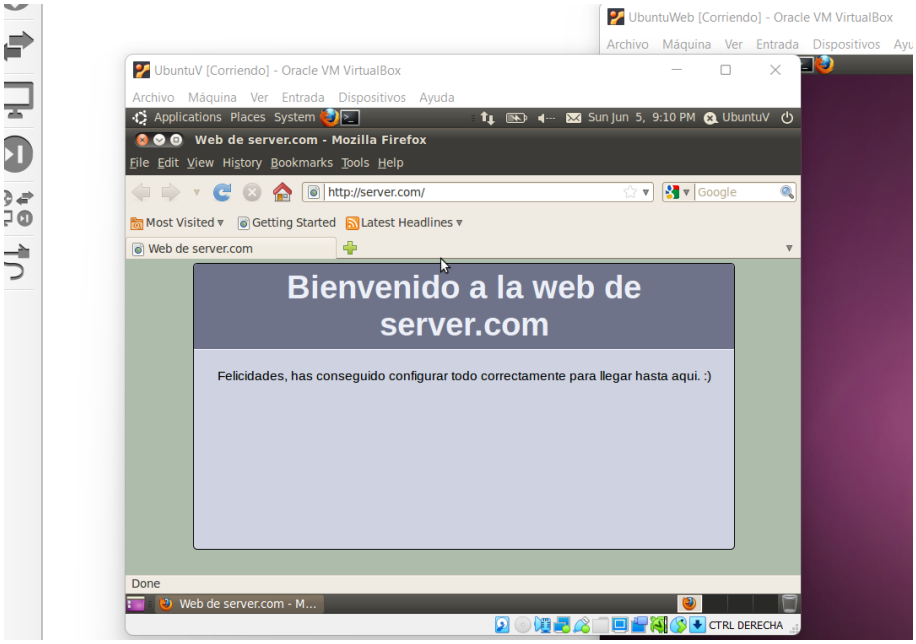

Figura A.30: Acceso web desde la máquina MV1 a MV2 (server.com)

Apéndice B

Resumen de enlaces adicionales

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

- Repositorio del código: `https://github.com/gackerman54/TFG`.