Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

CyberCrunch: Herramienta para despliegue y operación de laboratorios de ciberseguridad

Marco Cacho Sánchez

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

CyberCrunch: Herramienta para despliegue y operación de laboratorios de ciberseguridad

Autor: Marco Cacho Sánchez

Tutor: Valentín Cardeñoso Payo

Dedico este logro a todos aquellos que creyeron en mí y me inspiraron a superar los desafíos. Vuestra confianza y aliento han sido el combustible que me ha impulsado a explorar nuevos horizontes y alcanzar el éxito en este fascinante campo tecnológico.

El éxito no es la clave de la felicidad. La felicidad es la clave del éxito. Si amas lo que estás haciendo, tendrás éxito - Albert Schweitzer

Agradecimientos

Me gustaría agradecer el haber sido capaz de hacer este proyecto a mi familia, que gracias a su apoyo incondicional han conseguido que halla llegado hasta aquí y superar todas las adversidades vividas.

También quiero agradecer a mi tutor Valentín Cardeñoso por su disponibilidad, consejos y apoyo prestado durante toda la duración de este proyecto. Gracias a su visión, experiencia y conocimiento me ha ayudado a orientar los problemas con lo que me he ido enfrentando. Por supuesto, quiero agradecer al profesor Diego García, por la ayuda brindada en la fase de análisis y diseño.

Igualmente les quiero agradecer a mis compañeros del curro por enseñarme y darme consejos en los aspectos relacionados con el trabajo que desempeñamos y en las que cosas que han podido.

Obviamente no me olvido de los amigos que se han hecho durante la carrera y a las horas de trabajo en Discord. Gracias por vuestros consejos y experiencias adquirida de cuando pasasteis por este mismo momento. Cabe remarcar todos los cafés y cervezas que se han tomado juntos y han servido para seguir adelante.

A mi amigo de la infancia que me metió en este mundillo y me enseño en los inicios, simplemente gracias.

Por supuesto me quiero dar una mención a mi mismo por haber sido capaz de superar todas las adversidades vividas y por ser capaz de superar todas las que vengan.

En resumen, agradezco de todo corazón a mi familia, amigos, tutor, profesores y a todos aquellos que han contribuido de alguna manera a la realización de este Trabajo de Fin de Grado. Su apoyo incondicional, orientación y colaboración han sido fundamentales para el logro de este hito académico. ¡Sköl! ¡Gracias por su invalorable contribución y por formar parte de este importante logro en mi vida académica!

Resumen

El interés creciente por la formación en ciberseguridad a todos los niveles, tanto en programas de formación reglada como en de formación contínua, se hace necesario disponer de infraestructuras controladas para que los estudiantes puedan adquirir las competencias necesarias tanto en el desarrollo y aplicación de técnicas de defensa como en el conocimiento de las diferentes estrategias de ataque.

Dada la dificultad para crear y replicar este tipo de entornos, donde poder practicar de forma libre y gratuita, este Trabajo de Fin de Grado tiene como objetivo el desarrollo del software necesario para la automatización del proceso de creación de laboratorios de seguridad virtualizados, así como la configuración y monitorización de los servidores e infraestructura de la que consta.

Para realizar este trabajo se ha partido de software libre que nos ayude a conseguir los fines propuestos, desde herramientas para virtualizar maquinas y redes hasta aplicaciones de análisis de datos para la monitorización de dichas redes.

Una vez finalizado el trabajo se dispone de una aplicación para poder crear laboratorios virtuales y monitorizados, junto con una serie de servidores configurados para poder virtualizar y monitorizar los laboratorios que se creen.

Gracias a esto cualquier estudiante va a poder configurar sus equipos y montar sus propias redes para mejorar sus conocimientos.

Abstract

The growing interest in cybersecurity training at all levels, both in formal and continuing education programmes, makes it necessary to have controlled infrastructures so that students can acquire the necessary skills both in the development and application of defence techniques and in the knowledge of the different attack strategies.

Given the difficulty in creating and replicating this type of environment, where they can practice freely and free of charge, this Final Degree Project aims to develop the necessary software to automate the process of creating virtualised security laboratories, as well as the configuration and monitoring of the servers and infrastructure of which it consists.

To accomplish this project, open-source software has been utilized to achieve the proposed objectives, ranging from tools for virtualizing machines and networks to data analysis applications for network monitoring.

Upon completion of the project, an application will be available for creating virtual and monitored laboratories, along with a set of configured servers for virtualizing and monitoring the created laboratories.

Thanks to this tool, any student will be able to configure their own equipment and set up their own networks to enhance their knowledge in the field.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1 Introducción

En este Trabajo de Fin de Grado (TFG) se van a realizar todas las fases para la creación de una herramienta llamada **CyberCrunch**. Esta herramienta se va utilizar para poder desplegar redes virtuales, controlarlas y monitorizarlas, siendo capaz de crear distintos laboratorios de ciberseguridad, aunque pudiera tener muchos mas usos.

Las fases que se van a realizar son: la investigación, la realización del análisis y diseño completo de nuestra herramienta. Con el fin de que pueda ser usada de manera fácil y sencilla por todas personas que desee crear sus propios laboratorios y así mejorar sus habilidades ofensivas como sus herramientas para detectar los posibles ataques en un entorno controlado.

1.2 Motivación

Este tema que proponemos tendrá muchos usos dentro del campo de la ciberseguridad, creando tus propias honeynets autocontenidas, que son un tipo especial de Honeypots que actúan sobre una red entera, diseñada para ser atacada y recobrar mucha más información sobre posibles atacantes. Uno de estos usos es que a través de la creación de estas redes se puedan elaborar distintos laboratorios para que un estudiante en este campo pueda utilizarlas como herramienta para mejorar sus capacidades como hacker.

Teniendo ya un concepto de lo que queremos hacer, vamos a realizar un pequeño análisis de otras plataformas que tengan prestaciones similares. Empezamos preguntándonos que pasa si hacemos una pequeña búsqueda en Google sobre ”la construcción de laboratorios de seguridad”podemos encontrar 260 millones de resultados relacionados con montar de manera manual tu propio laboratorio, tanto

vídeos como blogs; por lo que podemos considerar que es un tema bastante llamativo para esta comunidad. También podemos observar que hay paginas para que los estudiantes en ciberseguridad puedan mejorar sus habilidades en este campo, paginas como ”HackTheBox”, ”TryHackMe”, etc. Pero se suelen centrar en un ámbito más ofensivo que defensivo y centrado en máquinas y no en entornos de red, cabe destacar que estas páginas suelen tener subscripciones para poder acceder a todo el contenido disponible.

Hemos detectado que la creación de esta herramienta facilitara la creación de laboratorios de seguridad, pudiendo mejorar las habilidades ofensivas, vulnerando la red creada, mejorar y optimizar las técnicas defensivas en un entorno de red controlado. También, en el caso de estar conectado a internet, la detección de ip’s relacionadas con botnets de ataque o grupos de hackers.

Capítulo 2

Objetivos y Alcance

2.1 Objetivos

Nuestro principal objetivo en este Trabajo de Fin de Grado es proporcionar una herramienta que facilite la creación de diversos laboratorios, permitiendo la automatización de tareas como la creación, el control y el monitoreo. A mayores se han planteado una serie de objetivos secundarios para el proyecto, que incluyen:

- Aumentar los conocimientos de diseño configuración de redes, aplicando los aprendidos en Packet Tracer en `GNS3`.
- Dar a conocer algunas de las posibilidades disponibles en la herramienta de simulación de redes `GNS3`.
- Ampliar los conocimientos y habilidades como desarrollador en Python (orientado a la creación de Scripts)
- Ampliar mis conocimientos de administración de sistemas operativos y ciberseguridad
- Ampliar mis conocimientos sobre routers y switches cisco.
- Aprender más sobre el proceso de ingeniería y el resto de habilidades adquiridas durante la realización del grado en ingeniería informática a las que de uso durante la realización del proyecto.
- Aprender a realizar laboratorios de ciberseguridad.

2.1.1 Tareas a realizar

Para poder conseguir todos los objetivos propuestos, tanto principales como secundarios, se va a definir una metodología:

1. Definir el trabajo y elaborar una planificación: En esta etapa inicial, se realizará una definición clara de los objetivos, alcance y requisitos del proyecto. Además, se elaborará una planificación detallada que establezca los hitos y plazos de trabajo, así como la asignación de recursos necesarios.

2. Estudiar el problema: En esta fase, se llevará a cabo un estudio exhaustivo del problema que se pretende abordar. Esto implica investigar y comprender en profundidad los conceptos y principios de la ciberseguridad, así como analizar las necesidades y desafíos actuales en el ámbito de los laboratorios virtuales junto con entender las herramientas que se quieren usar.

3. Localizar soluciones similares: En esta etapa, se realizará una investigación de soluciones existentes y similares a la herramienta que se pretende desarrollar. Esto nos permitirá conocer las mejores prácticas y enfoques utilizados en el campo de los laboratorios virtuales, y nos ayudará a identificar oportunidades de mejora e innovación.

4. Desarrollar nuestra solución: En esta fase, se procederá al diseño y desarrollo de la herramienta de creación de laboratorios virtuales. Se utilizarán las mejores prácticas de ingeniería de software y se aplicarán técnicas de programación para implementar las funcionalidades requeridas. Además, se realizarán pruebas y validaciones durante todo el proceso de desarrollo para asegurar la calidad del software.

5. Probarla y realizar experimentos con ella: Una vez que la herramienta ha sido desarrollada, se llevarán a cabo pruebas exhaustivas para verificar su funcionamiento y comprobar su rendimiento. Además, se realizarán experimentos prácticos utilizando la herramienta en escenarios reales, con el fin de evaluar su eficacia y utilidad en la creación de laboratorios virtuales.

Es importante destacar que esta metodología proporciona una estructura general para el desarrollo de la herramienta, pero puede modificarse y ajustarse según avancen las necesidades específicas del proyecto. El objetivo principal es seguir un enfoque sistemático y ordenado que nos permita alcanzar nuestros objetivos de manera eficiente y efectiva.

Capítulo 3

Metodología

Durante la realización de este proyecto se ha seguido una metodología ágil basada en **`Kanban`**. La palabra `Kanban` significa “tablero” o “tarjeta visual” en japonés. Éste se refiere a un sistema

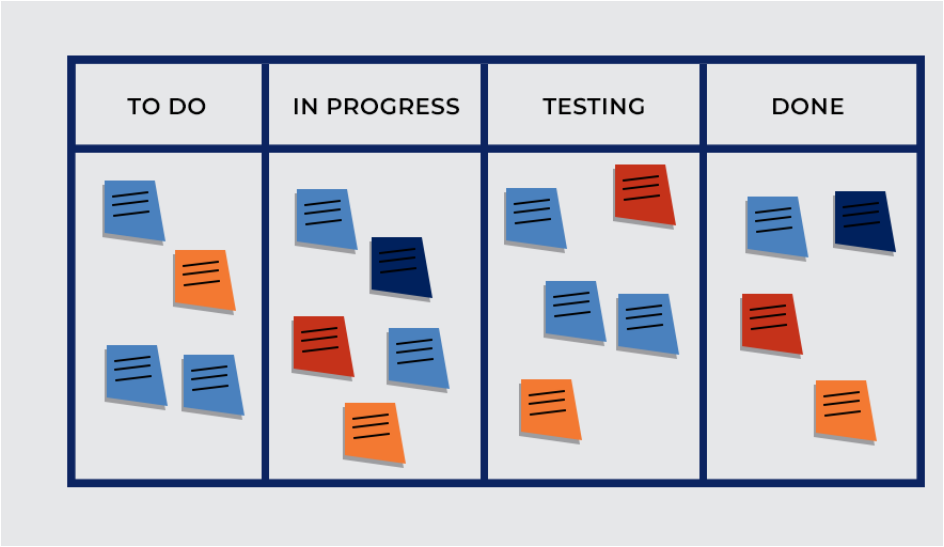


Figura 3.1: Ejemplo de desarrollo `Kanban`

de tarjetas que ayuda a visualizar el estado en el que está cada actividad o tarea. Para organizar el ”tablero”se han usado 5 posibles columnas, estos han sido:

- Meeting Topics
- To Do
- Pending
- Testing

- Done

Gracias a estos estados se puede seguir de forma simple y fácil como avanza el proyecto. Esta metodología también permite añadir relaciones entre las distintas ”tarjetas”, fechas inicio y finalización estimados, etc.

3.1 Fases y costes

Una vez conocida la metodología a utilizar podemos desarrollar la lista de actividades, su duración y sus dependencias 3.1. El proyecto está pensado para empezar el día 30 de enero y finalizar el 4 de junio, aunque para poder empezar el día previsto previamente se realizaran una serie de actividades como la preparación de una propuesta y la investigación para poder comenzar con el proyecto. Este preproyecto se comenzó el día 7 de noviembre y se finalizara el 25 de enero. Cabe destacar que gracias a esta metodología varias actividades pueden estar en un mismo estado, lo que permite avanzar en la redacción del documento a la vez que otras actividades.

Nombre de actividad	**Semanas**	**Dependencias**
Propuesta TFG	1	N/A
Investigación	8	N/A
Hito0: Redacción Documento	18	N/A
Hito1: Desplegar Máquinas	3	N/A
Hito2: Desplegar Redes	3	N/A
Hito3: Integración despliegue de host y redes	2	H1 y H2
Hito4: Implementación Control básico	1	H3
Hito5: Implementación Monitorización Básica	1	H3
Hito6: Implementación Control Avanzado	2	H4
Hito7: Implementación Monitorización Avanzada	3	H5
Hito9: Configuración Equipos	1	H5 y H6
Hito10: Migración	1	H9

Cuadro 3.1: Fases de desarrollo del proyecto previstas.

Una vez visto la lista de las actividades que se van a realizar se van a definir una serie de riesgos de alto nivel 3.2 con una breve descripción, el nivel de probabilidad de que ocurra y el impacto que supondría para el proyecto en caso de que ocurriese.

Nombre	**Descripción**	**Probab.**	**Impacto**
Desconocimiento de la tecnología base del proyecto	Falta de conocimiento y experiencia en la tecnología fundamental utilizada en el proyecto, lo que puede afectar negativamente la implementación y el rendimiento del sistema	Alta	Medio
Alto nivel de complejidad técnica	El proyecto implica una complejidad técnica significativa que puede dificultar la implementación y requerir más tiempo y recursos de lo esperado	Alta	Alto
Integraciones con sistemas externos desconocidos	La necesidad de integrarse con sistemas externos desconocidos puede presentar desafíos imprevistos y afectar el funcionamiento del sistema en general	Media	Alto
Estimación inadecuada del tiempo de ejecución	Una estimación incorrecta del tiempo necesario para completar las tareas puede llevar a retrasos y dificultades para cumplir con los plazos establecidos	Alta	Medio
Alta variación de los requerimientos	Cambios frecuentes y significativos en los requisitos del proyecto pueden generar dificultades en el seguimiento y la gestión del alcance, afectando la calidad y el cumplimiento de los objetivos establecidos	Alta	Alto
Falta de claridad en los requerimientos	Requisitos mal definidos o ambiguos pueden causar malentendidos y retrasos en el desarrollo, además de afectar la satisfacción del cliente al no cumplir sus expectativas	Media	Medio
Retraso en el diseño de los scripts	Demoras en el diseño de los scripts necesarios para la implementación del sistema pueden ralentizar el desarrollo y comprometer los plazos de entrega	Media	Medio
Aumento del tiempo dedicado a la redacción de la memoria	Un mayor tiempo dedicado a la redacción de la memoria del TFG puede afectar la dedicación y el progreso en otras actividades importantes del proyecto	Media	Medio
Pérdida de los archivos relacionados con el proyecto	La pérdida accidental de archivos o datos importantes del proyecto puede causar retrasos significativos y dificultades en la recuperación de la información perdida	Baja	Alto
Enfermedad	La enfermedad de los miembros del equipo puede afectar la disponibilidad y la capacidad de trabajo, lo que puede ocasionar retrasos en el progreso del proyecto	Baja	Medio

Cuadro 3.2: Catalogo de riesgos identificados del proyecto.

Por ultimo, en este apartado va realizarse un presupuesto 3.3 con los gastos estimados 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. Cabe destacar, como se describe en el apartado de requisitos, que se ha buscado crear una herramienta que no necesite gastos en licencias de software. Se ha estimado que para este proyecto se dediquen 20 horas semanales y una duración de 18 semanas, ha esto hay que sumarle la formación previa 40 horas y el desarrollo de la propuesta que fueron otras 20 horas, dejando un total de 420 horas a 12€/hora, basándome en mi salario actual, el trabajador tendrá un costo de 5.040€. A mayores se ha dispuesto de 3 servidores donde se ha desplegado las tecnologías utilizadas que han sido provistos por la universidad, para esto se estima un gasto de 2500€ servidor.

Nombre	**Descripción**	**C.U.**	**Unidades**	**C.T.**
Ingeniero informático junior	Costo del ingeniero informático junior para el proyecto	12 €/h	420 h	5.040€
3 Servidores	Costo de adquisición de 3 servidores	7.500€	3/60	375 €
Licencia VisualParadigm	Costo de la licencia de VisualParadigm (gratuita)	0€	0	0 €
Trello	Costo de la herramienta Trello (gratuita)	0€	0	0 €
`GNS3`	Costo de la herramienta `GNS3` (gratuita)	0€	0	0 €
OpenSearch	Costo de la herramienta OpenSearch (gratuita)	0€	0	0 €
Docker	Costo de la plataforma Docker (gratuita)	0€	0	0 €
SO servidores	Costo del sistema operativo de los servidores (gratuito)	0€	0	0 €
			Total:	**5.415€**

Cuadro 3.3: Presupuesto aproximado del proyecto.

Capítulo 4

Marco Conceptual

4.1 Contexto de la Ciberseguridad

La ciberseguridad [38] es un campo de estudio y práctica que se centra en proteger los sistemas informáticos y las redes contra amenazas, ataques y vulnerabilidades. Se ocupa de salvaguardar la confidencialidad, integridad y disponibilidad de la información, así como prevenir el acceso no autorizado y la alteración maliciosa de datos y recursos.

El estudio de la ciberseguridad abarca diversas facetas, que incluyen la defensa, el análisis forense, la gestión de riesgos, la respuesta a incidentes y la investigación de ataques. Los profesionales de la ciberseguridad pueden especializarse en una de estas áreas o desarrollar habilidades en múltiples disciplinas. También existe la figura del ”hacker ético”que busca identificar vulnerabilidades en sistemas para mejorar su seguridad.

En cuanto a la formación en ciberseguridad [4], se están llevando a cabo numerosas iniciativas para cubrir la creciente demanda de profesionales en este campo. Se ofrecen programas de capacitación y certificaciones en diferentes niveles, desde cursos introductorios hasta programas avanzados de posgrado. También se promueve la investigación y el desarrollo de nuevas tecnologías y metodologías para hacer frente a las amenazas cibernéticas. En estos últimos años, también ha habido un auge de plataformas online donde poder aprender de manera autónoma, fomentando la resolución de problemas de este campo.

Para implementar una infraestructura sólida de ciberseguridad, se requiere una combinación de hardware, software y políticas de seguridad. Esto incluye firewalls, sistemas de detección y prevención de intrusiones, herramientas de cifrado, autenticación multifactor, protección contra malware y actualizaciones regulares de software. Además, es esencial contar con un equipo de profesionales capacitados que puedan gestionar y mantener esta infraestructura.

4.2 Contexto de la Virtualización

La virtualización [2] es una tecnología que permite crear versiones virtuales de recursos informáticos, como servidores, redes, almacenamiento y sistemas operativos. Esta técnica ha revolucionado la forma en que se implementan y gestionan los recursos informáticos en entornos empresariales.

Por eso ha convertido en una alternativa popular en el campo de la ciberseguridad. Permite crear entornos aislados y controlados, donde se pueden realizar pruebas y experimentos sin poner en peligro la infraestructura real. Las tecnologías de virtualización, como las máquinas virtuales y los contenedores, ofrecen flexibilidad, escalabilidad y un alto grado de control sobre los recursos y las configuraciones.

Con todos los avances que se han conseguido en la virtualización se ha convertido en la opción mas popular para la creación de laboratorios de ciberseguridad. En sus inicios, consistían en entornos físicos reales para simular ataques y defensas. Sin embargo, el uso de laboratorios virtuales, donde se pueden crear y gestionar entornos de forma rápida y eficiente es la forma que se usa actualmente.

4.3 Contexto de la Aplicación

Como acabamos de explicar hoy en día, el campo de la ciberseguridad es un campo que se encuentra en pleno auge, y gracias a todos los avances en la virtualización cualquiera tiene en su mano alguna forma de poder crear sus propio laboratorio desde 0. También estamos viendo la popularización de paginas donde la gente puede resolver los retos que se encuentran disponibles, pero no dan la posibilidad al usuario de tener un control sobre que quiere aprender y mejorar, muchas de estas paginas incluyen servicios de pago para poder usar todas sus capacidades.

Como existe la falta de herramientas gratuitas y accesibles para crear laboratorios virtuales de ciberseguridad es una necesidad evidente en un campo en constante evolución. Los interesados en adquirir habilidades en ciberseguridad se enfrentan a limitaciones debido al alto costo y la falta de accesibilidad de las herramientas existentes. La creación de una aplicación que permita a los usuarios crear laboratorios virtuales de forma gratuita y fácil de usar abordaría esta necesidad, democratizando el acceso a entornos seguros y controlados para practicar y desarrollar habilidades en ciberseguridad. Esta aplicación proporciona una forma accesible, intuitiva y de bajo costo, fomentando el aprendizaje y la formación de profesionales más capacitados en el campo de la ciberseguridad.

El problema de no encontrar una herramienta gratuita o diseñada para poder facilitar el aprendizaje de estas habilidades, generando sorbe todo en el ámbito educativo que los profesores no puedan crear sus propios laboratorios en función de las necesidades del curso, clase, asignatura, etc. para que los docentes practiquen libremente las habilidades que se les van enseñado y explicando teóricamente o con demos.

Por otro lado, se ha apreciado un elevado número de blogs y tutoriales en internet que te enseñan a crear una red donde practicar una serie de habilidades especificas de la ciberseguridad pudiendo observar que un gran número de personas estén interesadas en poder crear y replicar de forma sencilla los laboratorios que se exponen en estas paginas de web.

Con todo esto, podemos concluir que hay un hueco en el mercado para nuestra aplicación que se ve justificado por el alto número de interesados. Aportando una herramienta totalmente gratuita facilitando así que cualquier usuario pueda utilizarla libremente para poder aprender y mejorar todos sus conocimientos en la ciberseguridad de una forma practica.

Capítulo 5

Soluciones Existentes

En este proyecto, nuestro objetivo es desarrollar una herramienta versátil que permita la creación de laboratorios de ciberseguridad. Estos laboratorios podrán ser utilizados con diversos propósitos, tal como hemos mencionado en secciones anteriores. La aplicación que estamos desarrollando ofrece una amplia gama de usos en el mercado, lo que la hace muy versátil y adaptable a diferentes necesidades. Sin embargo, debido a su gran alcance, es posible encontrar otras soluciones existentes que se centren únicamente en ciertas características específicas que nuestra herramienta ofrece. Estas soluciones especializadas se enfocan en brindar funcionalidades específicas y pueden ser útiles para usuarios que requieren un enfoque más preciso y especializado en su trabajo con laboratorios de ciberseguridad.

En este capitulo vamos a presentar algunas de las alternativas comerciales con uso o usos similares para los que podrían ser usados en nuestra herramienta, con este apartado podemos encontrar una justificación para la realización de este proyecto.

5.1 Laboratorios Ciberseguridad

La funcionalidad inicial de este trabajo fue la creación de una herramienta para poder desplegar laboratorios a los distintos estudiantes en este arte para poder trabajar, aprender y mejorar las capacidades de los usuarios dentro de un entorno controlado y monitorizado. Observando al mercado podemos encontrar muchas otras opciones que ofrecen características similares como podrían ser los de la tabla 5.1. Todas estas herramientas mayormente están diseñadas para resolver retos ofensivos, no permiten diseñar tus propios retos libremente, ni organizar laboratorios para un grupo de alumnos de forma gratuita. Si queremos hacer nuestros laboratorios personalizados nos tocaría hacerlo manualmente a través de máquinas virtuales, que veremos mas adelante, para ello hemos hecho una pequeña búsqueda en google sobre ”la construcción de laboratorios de seguridad”donde hemos podido observar un total de 260 millones de resultados relacionados con montar de manera manual tu propio laboratorio.

Herramienta	Definición
Hack The Box [15]	Hack The Box (HTB) es una plataforma en línea que ofrece una serie de desafíos de hacking y laboratorios prácticos para que los usuarios exploren.
bWAPP [16]	bWAPP (Buggy Web Application) es una aplicación web deliberadamente insegura diseñada para ser utilizada con fines educativos y de entrenamiento.
RootMe [27]	RootMe es una plataforma en línea que proporciona una variedad de desafíos y laboratorios de seguridad informática para que los usuarios practiquen y aprendan.
TryHackMe [34]	TryHackMe es una plataforma en línea que ofrece escenarios de aprendizaje y laboratorios prácticos para enseñar habilidades de hacking y seguridad informática.
Mutillidae II [23]	Mutillidae II es una aplicación web de código abierto vulnerable desarrollada en PHP/MySQL que se utiliza como entorno de entrenamiento para pruebas de penetración.

Cuadro 5.1: Soluciones para laboratorios de ciberseguridad.

Con todo esto, podemos concluir que en el mercado no existe un herramienta que ofrezca exactamente los requisitos que estamos buscando. También podemos inferir que en este campo que hay que estar en constante aprendizaje y cambio poder tener una herramienta para entrenar nuestras habilidades es algo bastante útil.

5.2 Redes Virtuales

Como hemos comentado en el apartado anterior, para realizar nuestros propios laboratorios se va a necesitar de una herramienta de virtualización, o montar una máquina en un equipo físico, pero esta opción necesitaría de un gran numero de máquinas para poder crear redes y solo se podría tener una activa al mismo tiempo. Para virtualizar máquinas podemos encontrar multitud de opciones que nos lo permiten y también crear redes con estos equipos, incluso algunas de estas herramientas nos permiten desplegar los equipos y conectarlos entre si. Algunos ejemplos serian los del siguiente cuadro 5.2.

Herramienta	Descripción
GNS3 [13]	GNS3 es un emulador de red gráfico que te permite diseñar redes complejas y simularlas virtualmente para propósitos de aprendizaje, prueba y desarrollo. Este emulador es de código abierto y gratuito.
VMware [37]	VMware es una empresa que ofrece una amplia gama de productos de virtualización, incluyendo VMware Workstation, que permite crear y ejecutar múltiples sistemas operativos en una sola máquina física.
Virtual Box [35]	VirtualBox es una herramienta de virtualización de código abierto que permite a los usuarios crear y ejecutar máquinas virtuales en su computadora. Es compatible con una variedad de sistemas operativos invitados.
Red Hat [26]	Red Hat Virtualization una solución de virtualización de servidores que utiliza el hipervisor KVM (Kernel-based Virtual Machine). Permite crear y gestionar múltiples máquinas virtuales en un entorno de servidor, lo que facilita la consolidación de recursos y la optimización del rendimiento.
Terraform [32]	Terraform es una herramienta de infraestructura como código (IaC) que permite definir y desplegar la infraestructura de forma declarativa. Es ampliamente utilizado para la gestión y automatización de la infraestructura en la nube.

Cuadro 5.2: Soluciones para redes virtuales.

La mayoría de estas herramientas ofrecen la opción de crear máquinas virtuales y/o redes virtuales pero para ello, en muchos casos, requiere de adquirir un producto o un servicio. Excepto la opción de GNS3 que ha sido seleccionada para soportar la virtualización de nuestras redes.

5.3 HoneyPot y Sandbox

También podríamos usar estas redes como honeypots [41] y sandboxes [40]. Una honeypot[41], o sistema trampa o señuelo, es una herramienta de la seguridad informática dispuesto en una red o sistema informático para ser el objetivo de un posible ataque informático, y así poder detectarlo y obtener información del mismo y del atacante. Algunos ejemplos de honeypots ya diseñadas la podemos encontrar en la tabla 5.3.

En este caso podemos encontrar aplicaciones y herramientas para poder crear los sistema trampa diseñados y sin poder modificarlos manualmente, en cambio, gracias a nuestra herramienta podríamos crear nuestras propias redes vulnerables, exportarlas, modificarlas, etc.

Por otro lado, un sandbox [40] en informática o un entorno de pruebas, es una máquina o máquinas virtuales aislada en la que se puede ejecutar código de software potencialmente inseguro sin afectar

Herramienta	Descripción
Cowrie [6]	Cowrie es una herramienta de honeypot para SSH de código abierto. Simula un servidor SSH vulnerable para atraer a posibles atacantes y registrar sus acciones, permitiendo el análisis y estudio de sus técnicas y comportamientos.
phpMyAdmin Honeypot [1]	phpMyAdmin Honeypot es un honeypot diseñado para simular una instalación vulnerable de php, una popular herramienta de administración de bases de datos MySQL. Se utiliza para atraer y detectar intentos de ataques dirigidos a paginas web cone estas caracteristicas.
StrutsHoneypot [8]	StrutsHoneypot es un honeypot diseñado para detectar y registrar intentos de explotación de vulnerabilidades en aplicaciones web basadas en Apache Struts, un framework popular utilizado para el desarrollo web en Java. Ayuda a identificar posibles ataques dirigidos a aplicaciones Struts.
UDPot Honeypot [28]	UDPot Honeypot es un honeypot de red diseñado para simular servicios de protocolo UDP (User Datagram Protocol) vulnerables. Registra el tráfico y los intentos de ataque dirigidos a servicios UDP, permitiendo el análisis y la detección de amenazas en la red.

Cuadro 5.3: Soluciones para HoneyPots

a los recursos de red o a las aplicaciones locales. Algunos ejemplo de sandbox podrian ser los de la tabla 5.4

En este caso en el mercado existen varios productos que servirían para analizar el comportamiento del malware, al igual que se puede hacer con nuestra herramienta.

A parte de aplicaciones en el ámbito de la ciberseguridad esta programa también se podría usar en otros ámbitos, como el diseño de redes SDN y NFV. Las redes definidas por software (SDN) son un conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación e implantación de servicios de red de una manera determinista, dinámica y escalable, evitando al administrador de red gestionar dichos servicios a bajo nivel; La virtualización de funciones de red, denominada NFV, es un marco de referencia general relacionado con la arquitectura de redes, orientado a virtualizar diferentes elementos dentro de las mismas.

Herramienta	Descripción
Cuckoo Sandbox [7]	Cuckoo Sandbox es una plataforma de análisis de malware de código abierto. Permite ejecutar y analizar archivos sospechosos en un entorno aislado para identificar y entender el comportamiento malicioso.
FireEye Sandbox [9]	FireEye Sandbox es una solución de análisis de malware que utiliza técnicas de sandboxing y análisis de comportamiento para detectar y analizar amenazas avanzadas. Proporciona visibilidad detallada sobre el comportamiento y las técnicas de evasión utilizadas por el malware.
Palo Alto Networks WildFire [46]	Palo Alto Networks WildFire es un servicio basado en la nube que ofrece análisis automatizado de archivos sospechosos y malware. Utiliza técnicas de sandboxing y análisis estático y dinámico para identificar y prevenir amenazas avanzadas.
Fortinet FortiSandbox [11]	Fortinet FortiSandbox es una solución de sandboxing que proporciona análisis de malware en tiempo real. Utiliza técnicas de virtualización y análisis de comportamiento para identificar y detener amenazas desconocidas antes de que puedan causar un impacto en las organizaciones.

Cuadro 5.4: Soluciones para Sandboxes

Capítulo 6

Análisis

El análisis es una etapa esencial en el desarrollo de aplicaciones de software. Durante este proceso, se exploran a fondo los requisitos y necesidades del negocio, así como las expectativas de los usuarios. El objetivo principal del análisis es comprender el alcance del proyecto y establecer una base sólida para el diseño y desarrollo de la aplicación.

En este apartado de análisis, nos sumergiremos en la identificación, recopilación y documentación de los requisitos funcionales y no funcionales de la aplicación. Se emplearán técnicas de investigación y entrevistas con los interesados para obtener una visión completa de las necesidades y deseos de los usuarios finales.

El análisis también involucra el examen detallado de los casos de uso, que describen cómo los usuarios interactuarán con la aplicación en diferentes escenarios. Además, se presta atención a la identificación de los requisitos no funcionales, como la usabilidad, el rendimiento, la seguridad y la escalabilidad. Estos requisitos son cruciales para garantizar que la aplicación cumpla con los estándares de calidad y satisfaga las demandas de los usuarios.

En resumen, el apartado de análisis es el punto de partida esencial para el desarrollo exitoso de aplicaciones de software. Mediante una investigación exhaustiva y la definición clara de los requisitos, se establece el marco necesario para el diseño y desarrollo de una aplicación que cumpla con las expectativas de los usuarios y los objetivos del negocio.

6.1 Requisitos Funcionales

Como se ha podido apreciar en apartados anteriores, las herramientas de despliegue, control y monitorización de redes ofrecen una gran facilidad de uso una vez que se cuenta con una leve noción sobre informática y redes. Estas herramientas han sido diseñadas para simplificar y agilizar los procesos relacionados con la creación y gestión de laboratorios virtuales.

Sin embargo, es importante tener en cuenta que el requisito de poseer conocimientos técnicos en informática y redes solo se aplica en el caso de aquellos usuarios que deseen diseñar su propio laboratorio personalizado. Para los usuarios que deseen replicar laboratorios existentes, no es necesario contar con un profundo conocimiento en estas áreas, ya que funcionalidades predefinidas están diseñadas con el fin de facilitar el proceso de replicación.

A medida que avanzamos en el desarrollo del preproyecto, se han llevado a cabo reuniones de trabajo en las cuales se han definido y establecido tres características principales para el proyecto: despliegue, control y monitorización. Estas características se han identificado como fundamentales para cumplir con los objetivos y requisitos del sistema. Los requisitos funcionales que se han definido a para cumplir con las características han sido:

Despliegue		
Código	**Nombre**	**Descripción**
RF01	Desplegar host	La aplicación debe ser capaz de desplegar un host de los disponibles. Esto implica configurar y poner en funcionamiento un servidor o máquina virtual que pueda ser utilizado como parte de la infraestructura del sistema.
RF02	Desplegar entorno de red	La aplicación debe ser capaz de desplegar un entorno de red. Esto implica establecer la configuración de red necesaria para que los distintos componentes de la aplicación puedan comunicarse entre sí de manera adecuada.
RF03	Desplegar entorno de red dinámico e incremental	La aplicación debe ser capaz de desplegar un entorno de red de forma dinámica e incremental. Esto implica que pueda añadir o eliminar componentes de red según las necesidades del sistema, permitiendo una adaptabilidad y escalabilidad eficiente.
Control		
Código	**Nombre**	**Descripción**
RF04	Despliegue de red automática	La aplicación debe ser capaz de desplegar automáticamente una red. Esto implica que el sistema pueda configurar y poner en funcionamiento una red de manera automática, sin intervención manual, siguiendo las especificaciones definidas.
RF05	Exportar red desplegada	La aplicación debe permitir la exportación de una red desplegada, de modo que pueda ser guardada y reutilizada posteriormente. Esto facilita la reproducción de configuraciones de red específicas y agiliza los procesos de despliegue en entornos similares.

RF06	Eliminar laboratorio virtualizado	La aplicación debe proporcionar la capacidad de eliminar un laboratorio virtualizado. Esto implica desmantelar y eliminar todos los recursos asociados a un laboratorio, incluyendo hosts, redes y otros componentes virtuales.
RF07	Modificar red desplegada	La aplicación debe permitir la modificación de una red desplegada existente. Esto implica la capacidad de agregar, eliminar o modificar componentes de red, como hosts o conexiones, de manera que se puedan adaptar las configuraciones según las necesidades del sistema.
RF08	Comprobar estado de un elemento de la red	La aplicación debe proporcionar la capacidad de comprobar el estado de un elemento concreto de la red. Esto implica obtener información sobre el estado de funcionamiento de un host, una conexión o cualquier otro componente de red, permitiendo verificar su disponibilidad y rendimiento.
Monitorización		
Código	**Nombre**	**Descripción**
RF09	Obtener información de un Router	La aplicación debe ser capaz de obtener información específica de un Router de la red. Esto implica recolectar y mostrar datos como la tabla de enrutamiento, la configuración de interfaces, el estado de las interfaces, entre otros parámetros relevantes.
RF10	Obtener tráfico recibido por un host desplegado	La aplicación debe proporcionar la capacidad de obtener información sobre el tráfico recibido por un host específico desplegado en la red. Esto implica recolectar y mostrar datos como el número de paquetes recibidos, la tasa de transferencia, las conexiones establecidas, entre otros indicadores de rendimiento.

Cuadro 6.1: Requisitos funcionales Principales

Además de los objetivos principales, existen una serie de objetivos secundarios que amplían la funcionalidad del sistema. Estos requisitos secundarios se centran en la capacidad de modificar características en tiempo real de los hosts desplegados, de los elementos relacionados con la red y de la capacidad de incluir nuevas máquinas disponibles en la red. A continuación, se detallan estos requisitos secundarios:

Código	Nombre	Descripción
RF11	Modificación de características en tiempo real de hosts desplegados	La aplicación debe permitir ajustar parámetros de configuración, recursos asignados y redes virtuales internas y externas.
RF12	Modificación de características en tiempo real de elementos relacionados con la red	La aplicación debe permitir ajustar la configuración de enrutamiento, asignación de ancho de banda, filtros de seguridad, políticas de acceso, entre otros, en dispositivos de red sin interrumpir la conectividad.
RF13	Capacidad de incluir nuevas máquinas disponibles en la red	La aplicación debe permitir agregar ordenadores, switches, routers, firewalls u otros dispositivos de red compatibles para ampliar la capacidad y funcionalidad del laboratorio virtual.

Cuadro 6.2: Requisitos funcionales Secundarios

6.2 Requisitos No Funcionales

Para lograr una implementación efectiva de los requisitos funcionales previamente mencionados, es fundamental considerar los requisitos no funcionales correspondientes. Estos requisitos no funcionales, como la eficiencia, la escalabilidad, la seguridad y la disponibilidad, son esenciales para garantizar un despliegue, control y monitorización exitosos en la aplicación sin interfaz gráfica.

Código	Nombre	Descripción
RNF1	Compatibilidad con Python 3	El programa debe ser capaz de ejecutarse en cualquier máquina que tenga instalado Python 3.
RNF2	Eficiencia en el consumo de recursos	El programa debe minimizar el uso de recursos, como CPU, memoria y almacenamiento.
RNF3	Compatibilidad con servidores `GNS3`	El programa debe poder ejecutarse en cualquier servidor `GNS3`, ya sea local o remoto.
RNF4	Exportación de red en formato reutilizable	La red exportada debe mantener el mismo formato de entrada para permitir su reutilización.
RNF5	Conexión disponible con el servidor `GNS3`	El script debe funcionar siempre que exista una conexión disponible con el servidor `GNS3`.
RNF6	Independencia de la ubicación del nodo para la red creada	La red creada debe ser independiente de la ubicación del nodo en la infraestructura de red.

Cuadro 6.3: Requisitos no funcionales

En resumen, los requisitos funcionales y no funcionales presentados anteriormente son de vital importancia para el desarrollo de la aplicación. Los requisitos funcionales, como el despliegue de

hosts, el control de elementos de red y la monitorización, definen las funcionalidades clave que la aplicación debe cumplir para cumplir con sus objetivos principales. Estos requisitos establecen las bases para que los usuarios puedan aprender de una forma fácil y sencilla.

Por otro lado, los requisitos no funcionales, como la compatibilidad con Python 3, la eficiencia en el consumo de recursos y la independencia de la ubicación del nodo, se centran en aspectos que complementan la funcionalidad de la aplicación. Estos requisitos garantizan que la aplicación pueda ejecutarse en diferentes entornos, optimice el uso de recursos y brinde una experiencia de usuario satisfactoria.

En conjunto, tanto los requisitos funcionales como los no funcionales definidos en el preproyecto han marcado un papel crucial en el diseño y desarrollo de la aplicación.

6.3 Diagramas de Casos de Uso

Una vez presentados los requisitos funcionales y no funcionales que se han definido para esta herramienta para el aprendizaje se van a presentar los diagramas de caso de uso. En estos se exporte las formas de usar este sistema software, que para nuestro caso esta basado en scripts.

En la siguiente figura 6.1 podemos observar el diagrama de casos de uso, para el usuario se pueden apreciar 3 grandes funcionalidades. Estas son: crear un red, exportar la red y información de los nodos.

Como se puede apreciar en el diagrama 6.1 de casos de uso, este cuenta con un sistema central y tres actores. El sistema central agrupa los scripts ya mencionados, junto con otros para poder completar las necesidades del usuario.

Los actores que nos encontramos son los siguientes:

- **Usuario:** El usuario es un actor primario en el sistema. Es aquel que utiliza la aplicación y realiza las diferentes acciones y operaciones disponibles. Su rol principal es interactuar con la aplicación para llevar a cabo tareas relacionadas con el despliegue y control de laboratorios virtuales de red. El usuario puede ser un administrador de red, un ingeniero de sistemas o cualquier persona con conocimientos en informática y redes.
- **Docker:** Docker es un actor secundario en el sistema. Representa una plataforma de virtualización basada en contenedores que permite el despliegue y ejecución de aplicaciones de forma aislada. En el contexto de la aplicación, Docker puede ser utilizado como una opción para el despliegue de los diferentes componentes y servicios necesarios para el funcionamiento de los laboratorios virtuales de red. Su función es proporcionar un entorno eficiente y seguro para ejecutar las distintas máquinas virtuales y elementos de red requeridos en el sistema.
- **`GNS3`:** `GNS3` es otro actor secundario en el sistema. Se trata de un simulador de red gráfico que permite el diseño, configuración y emulación de redes complejas. `GNS3` proporciona una plataforma para crear laboratorios virtuales de red y simular el comportamiento de diferentes elementos de red, como routers, switches y firewalls. En la aplicación, `GNS3` se utiliza como

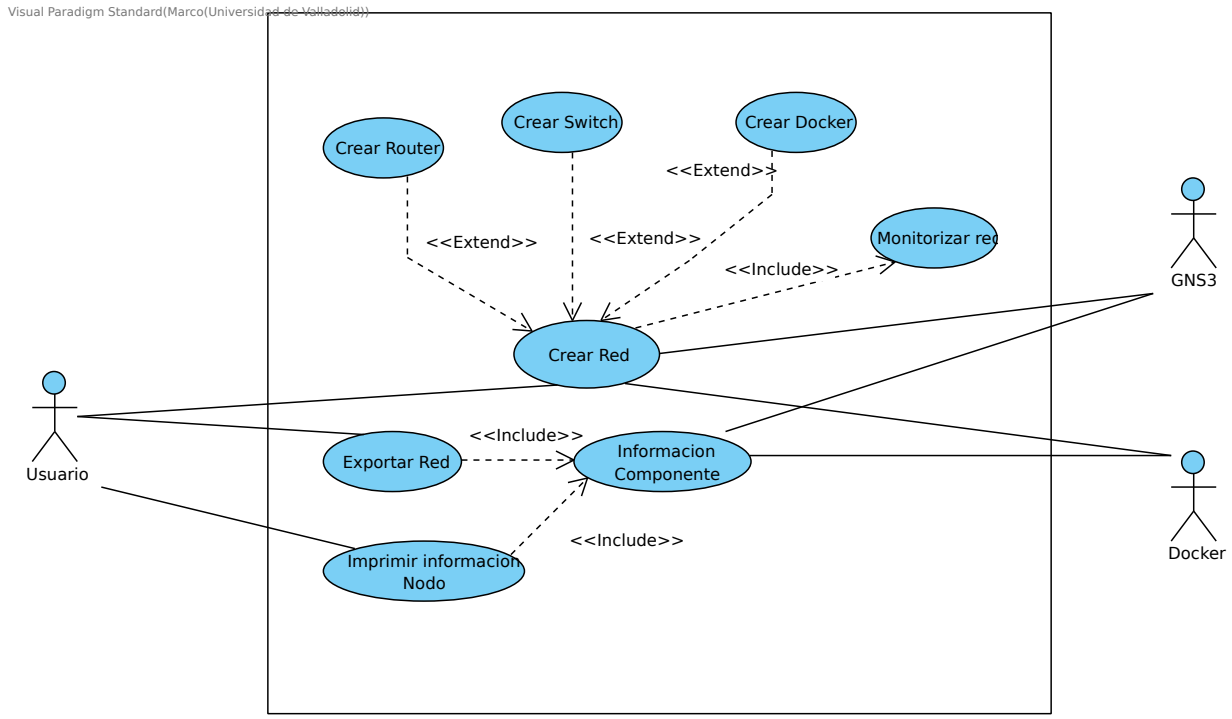


Figura 6.1: Diagrama de Casos de Uso

una herramienta para el control y gestión de los elementos de red desplegados en los laboratorios virtuales. Permite configurar y monitorear el estado de los dispositivos virtuales y supervisar el tráfico de red.

Una vez descritos los actores presentes en el sistema vamos a pasar a explicar los casos de uso:

Apartado	**Descripción**
Nombre	Crear red
Código	CU01
Actor principal	Usuario
Actor secundario	`GNS3` y Docker
Precondición	El usuario tiene acceso a la aplicación y un archivo JSON con la configuración de la red valido
Postcondición	Se ha creado y desplegado la red virtual según la configuración del archivo JSON
Escenario principal	1. El usuario selecciona la opción de crear red en la interfaz de la aplicación. 2. La aplicación lee el archivo JSON y obtiene la configuración de la red. 3. La aplicación llama al caso de uso Çrear Docker"(Si existen dockers). 4. La aplicación llama al caso de uso Çrear Router"(Si existen routers). 5. La aplicación llama al caso de uso Çrear Switch"(Si existen switches). 6. La aplicación llama al caso de uso "Monitorizar Red"para supervisar la red desplegada. 7. La aplicación finaliza el proceso de creación de la red y muestra un mensaje de éxito al usuario.
Flujo alternativo	▪ Si el archivo JSON es inválido o contiene errores de configuración, la aplicación muestra un mensaje de error y el proceso de creación se detiene. ▪ Si no se incluyen nodos de algún tipo el sistema ignara esa parte de la ejecución y pasara al siguiente caso.

Cuadro 6.4: Caso de uso para Crear Red

Apartado	**Descripción**
Nombre	Crear Router
Código	CU02
Precondición	▪ El template seleccionado para crear existe en el sistema `GNS3`. ▪ El usuario introduce características validas para su configuración.
Postcondición	Se ha creado y configurado el router existente.
Escenario principal	1. El sistema crear una router virtual a partir del template indicado. 2. El sistema configura el router en función de las características introducidas.
Flujo alternativo	- El template seleccionado existe en el servidor `GNS3`. - Si se produce un error durante la creación o configuración del router, se muestra un mensaje de error al usuario.

Cuadro 6.5: Caso de uso para Crear Router

Apartado	**Descripción**
Nombre	Crear Switch
Código	CU03
Precondición	▪ El template seleccionado para crear existe en el sistema `GNS3`. ▪ El usuario introduce características validas para su configuración.
Postcondición	Se ha creado y configurado el router existente.
Escenario principal	1. El sistema crear una switch virtual a partir del template indicado. 2. El sistema configura el switch en función de las características introducidas.
Flujo alternativo	- El template seleccionado existe en el servidor `GNS3`. - Si se produce un error durante la creación o configuración del switch, se muestra un mensaje de error al usuario.

Cuadro 6.6: Caso de uso para Crear Switch

Apartado	**Descripción**
Nombre	Crear Docker
Código	CU04
Precondición	▪ El template seleccionado para crear existe en el sistema `GNS3`. ▪ El usuario introduce características validas para su configuración.
Postcondición	Se ha creado y configurado el router existente.
Escenario principal	1. El sistema crear una docker virtual a partir del template indicado. 2. El sistema configura el dcoker en función de las características introducidas.
Flujo alternativo	- El template seleccionado existe en el servidor `GNS3`. - Si se produce un error durante la creación o configuración del docker, se muestra un mensaje de error al usuario.

Cuadro 6.7: Caso de uso para Crear Docker

Apartado	**Descripción**
Nombre	Monitorizar Red
Código	CU05
Precondición	▪ Los equipos permiten alguna forma de monitorización valida. ▪ El usuario introduce características validas para su monitorización.
Postcondición	Se ha creado y configurado el router existente.
Escenario principal	1. El sistema configura los equipos para monitorizarlos 2. Configura el envió de la monitorización a una máquina externa (Si selecciona la posibilidad)
Flujo alternativo	- El equipo puede ser monitorizado de alguna forma valida para el sistema, sino se saltara esta fase - Si la red no tiene conexión con la máquina externa el envió de logs no se producirá aunque este configurado.

Cuadro 6.8: Caso de uso para Monitorizar Red

Apartado	**Descripción**
Nombre	Exportar Red
Código	CU06
Actor principal	Usuario
Precondición	El usuario ha seleccionado la opción de exportar red para un servidor `GNS3`
Postcondición	La red virtual ha sido exportada en un formato compatible para su reutilización.
Escenario Principal	1. El usuario selecciona la opción de exportar red en la aplicación. 2. La aplicación llama al caso de uso Ïnformación Componente"para recopilar la información de todos los componentes de la red virtual. 3. La aplicación genera un archivo en un formato específico que representa la red virtual, incluyendo la información de cada componente obtenida en el paso anterior. 4. La aplicación guarda el archivo de exportación en una ubicación especificada por el usuario. 5. La aplicación muestra un mensaje de éxito al usuario, indicando que la red ha sido exportada correctamente.
Flujo Alternativo	- Si no existe la red virtual pedida, la aplicación muestra un mensaje de error al usuario. - Si ocurre algún error durante el proceso de exportación, la aplicación muestra un mensaje de error y el caso de uso se detiene.

Cuadro 6.9: Caso de uso para Exportar Red

Apartado	**Descripción**
Nombre	Imprimir Información Nodo
Código	CU07
Actor principal	Usuario
Precondición	El usuario ha seleccionado la opción de obtener información de un nodo para una red virtual.
Postcondición	El sistema muestra la infracción recopilada de dicho nodo.
Escenario Principal	1. El usuario selecciona la Imprimir información Nodo en la aplicación. 2. La aplicación llama al caso de uso Ïnformación Componente"para recopilar la información del componente de la red virtual. 3. La aplicación muestra la información recopilada del nodo.
Flujo Alternativo	- Si no existe la red virtual pedida, la aplicación muestra un mensaje de error al usuario. - Si no existe el nodo pedida, la aplicación muestra un mensaje de error al usuario. - Si ocurre algún error durante el proceso de exportación, la aplicación muestra un mensaje de error y el caso de uso se detiene.

Cuadro 6.10: Caso de uso para Imprimir Información Nodo

Apartado	Descripción
Nombre	Información Componente
Código	CU08
Actor secundario	`GNS3` y Docker
Precondición	El sistema ha seleccionado obtener información de un nodo.
Postcondición	El sistema devuelve los datos obtenidos del nodo.
Escenario Principal	1. El sistema se conecta al nodo seleccionado. 2. El sistema recopila información del nodo. 3. El sistema devuelve la información recopilada.
Flujo Alternativo	- Si no existe la red virtual pedida, la aplicación muestra un mensaje de error al usuario. - Si no existe el nodo pedida, la aplicación muestra un mensaje de error al usuario. - Si ocurre algún error durante el proceso de exportación, la aplicación muestra un mensaje de error y el caso de uso se detiene.

Cuadro 6.11: Caso de uso para Información Componente

6.4 Modelo de dominio

En la sección del Modelo de Dominio, se presenta una representación conceptual del entorno en el que se desarrollará el software. Este modelo proporciona una visión estructurada y detallada de las entidades principales, sus atributos y las relaciones entre ellas. El objetivo fundamental del Modelo de Dominio es comprender y visualizar de manera clara y precisa los elementos fundamentales del dominio de aplicación, lo cual facilita la comunicación y comprensión entre los diferentes actores involucrados en el proyecto. A través del análisis y diseño del Modelo de Dominio, se establecerán las bases para el desarrollo de un software que se ajuste de manera óptima a las necesidades y requerimientos del dominio, permitiendo así la creación de un sistema funcional y eficiente.

En la figura 6.2 se pueden apreciar los diferentes elementos que se van a usara para la creación de la red, siendo los mas importantes los diferentes tipos de nodos que puede tener nuestra red al ser estos los elementos sobre los que se va a trabar. Aunque también hay otros tres elementos que también son necesarios para que la herramienta pueda funcionar sin problemas.

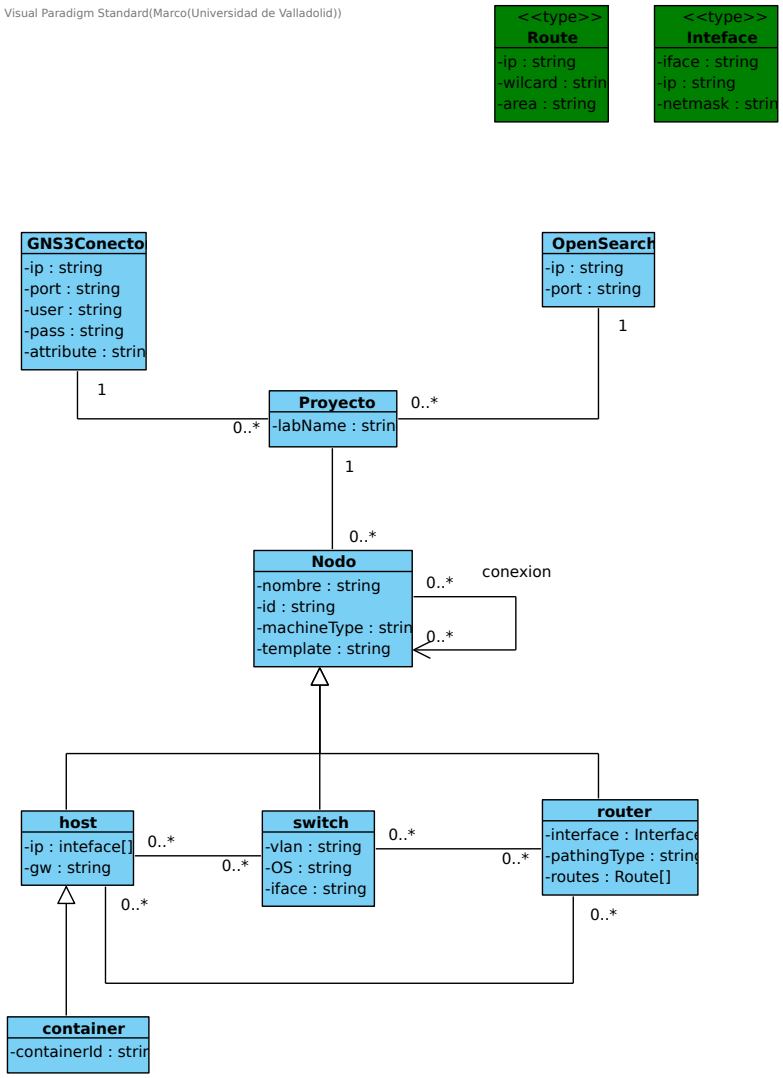


Figura 6.2: Modelo de dominio

Como ya hemos comentado, el modelo de dominio agrupa los diferentes nodos que podemos encontrarnos en las topologías de red y los elementos necesarios para crear y configurar los nodos. Todos los nodos comparten diferentes propiedades como el nombre, el id, el tipo de máquina y el template.

Un nodo es una entidad fundamental en el dominio del problema y representa la mayor generalización posible dentro de este contexto. A medida que nos adentramos en el análisis y la especialización, encontramos tres clases principales que surgen como resultado. Estas clases se distinguen por sus características particulares y su relevancia en el dominio del problema:

- **Router:** en el contexto del sistema en desarrollo, es un componente esencial que se encarga de dirigir el tráfico de red entre diferentes redes o subredes. En este caso específico, los routers son implementados como elementos virtuales dentro del entorno de ejecución. Estos routers virtuales son capaces de gestionar y enrutar los paquetes de datos, permitiendo la comunicación entre los diferentes hosts y subredes dentro del sistema. Los routers desempeñan un papel crucial en el establecimiento y mantenimiento de las conexiones de red, asegurando que los datos sean

enviados al destino correcto de acuerdo con las reglas y configuraciones establecidas. Estos routers virtuales ofrecen flexibilidad y escalabilidad, ya que pueden ser creados y configurados según las necesidades específicas del usuario. Además, brindan la capacidad de implementar y administrar distintas políticas de enrutamiento y seguridad para garantizar un flujo de datos eficiente y protegido en la red.

- **Switch:** en el contexto del sistema en desarrollo, es un componente esencial que se encarga de interconectar diferentes dispositivos de red dentro de una misma red local (LAN). En este caso específico, los switches son implementados como elementos virtuales dentro del entorno de ejecución. Estos switches virtuales actúan como puntos de conexión centralizados, permitiendo que los dispositivos de red se comuniquen entre sí de manera eficiente y segura. Los switches son responsables de examinar las direcciones MAC (Media Access Control) de los dispositivos conectados a ellos y de enviar los datos únicamente al destino adecuado, mejorando así el rendimiento y la eficiencia de la red. Estos switches virtuales ofrecen flexibilidad y escalabilidad, ya que pueden ser creados y configurados según las necesidades específicas del usuario.
- **Host:** En el contexto del sistema en desarrollo, se refiere a un componente fundamental que representa un entorno de ejecución independiente. En este caso específico, los hosts son implementados mediante el uso de contenedores Docker. Un host Docker es una instancia aislada y ligera que encapsula una aplicación junto con sus dependencias y configuraciones necesarias para su correcto funcionamiento. Estos hosts ofrecen un entorno virtualizado y autónomo, lo que permite la ejecución de múltiples instancias de aplicaciones en un mismo servidor físico. Los hosts Docker brindan flexibilidad, escalabilidad y portabilidad, ya que pueden ser fácilmente desplegados y gestionados en diferentes entornos de ejecución. Estos hosts son utilizados para alojar y ejecutar los diferentes componentes del sistema, permitiendo su interacción y comunicación de manera eficiente y segura.

Como hemos comentado y se puede observar en el diagrama 6.2 existen otros tres tipos de clases: la clase Proyecto que tiene un nombre y donde podemos encontrar los nodos pertenecientes a ese laboratorio, la clase GNS3Conector, en esta clase encontramos un conector para comunicarse con el servidor de `GNS3` y por ultimo encontramos la clase OpenSearch que se necesita para poder enviar los logs al servidor externo.

6.5 BCE

El enfoque BCE (Boundary Controller Entity) es un patrón arquitectónico que proporciona una estructura clara y modular para el desarrollo de software. Este enfoque divide el sistema en tres componentes principales: Boundary (Límite), Controller (Controlador) y Entity (Entidad). El componente Boundary define y gestiona la interacción entre el sistema y los actores externos, como los usuarios o sistemas externos. El Controller se encarga de la lógica del negocio y la coordinación de las operaciones del sistema, mientras que la Entity representa las entidades del dominio y almacena los datos relevantes. Esta separación de responsabilidades facilita la comprensión, mantenimiento y extensibilidad del sistema, al tiempo que promueve la reutilización de componentes y la modularidad.

Como se aprecia en la siguiente figura 6.3 el diagrama incluiría todas las clases que se han definido en el modelo de dominio, entidades que almacenarían los datos y relaciones. Podemos observar también un controlador por caso de uso, aunque los casos de uso extendidos comparten controlador para una mayor claridad. También se han añadido frontera, una por actor. En la siguiente sección se expresara usando este mismo patrón.

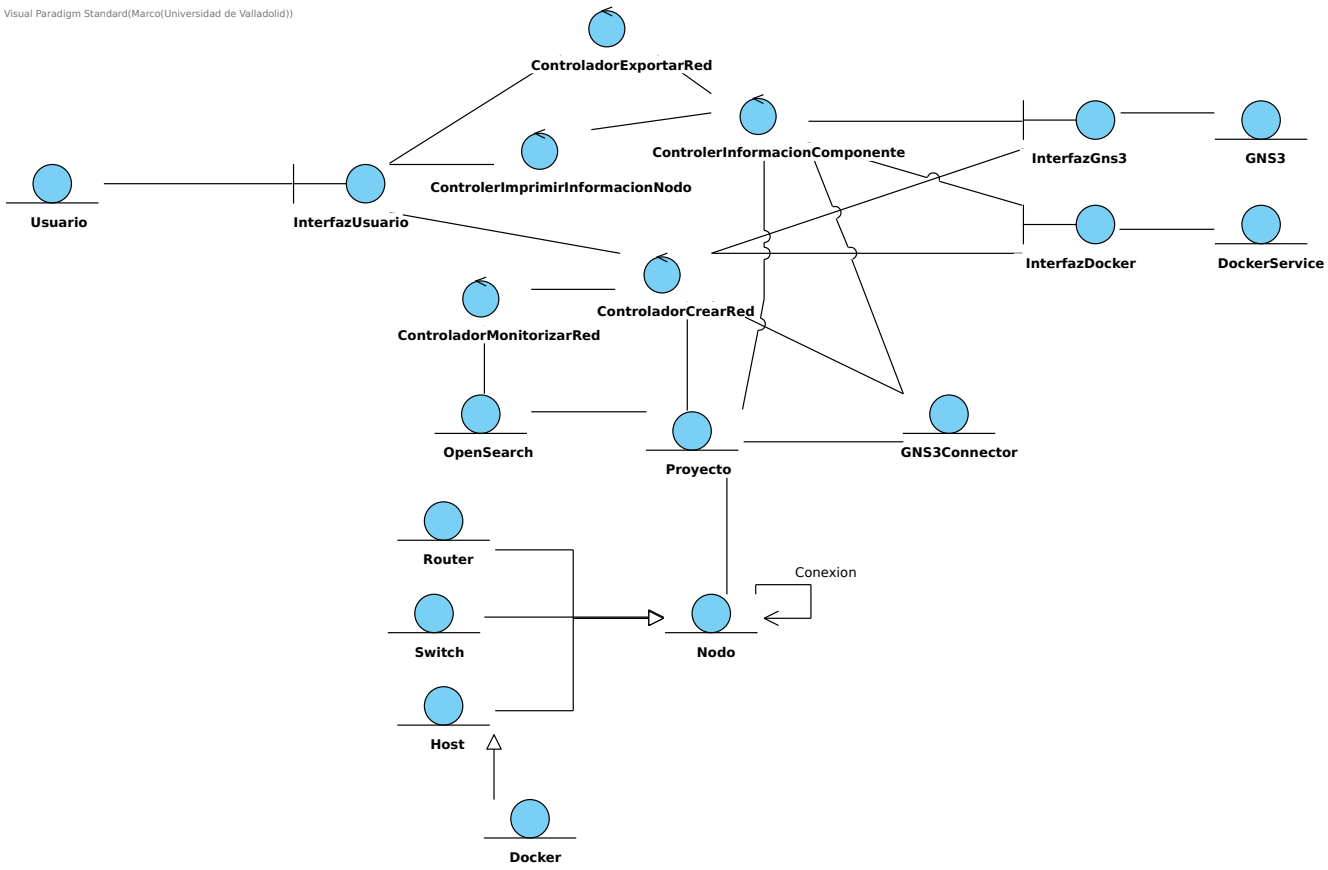


Figura 6.3: Diagrama de clases aplicando BCE

6.6 Diagrama de secuencia

El diagrama de secuencia es una herramienta fundamental en el diseño y análisis de sistemas de software, y se utiliza para representar la interacción entre los distintos componentes de un sistema en un escenario específico. En el contexto del patrón arquitectónico escogido, BCE (Boundary Controller Entity), el diagrama de secuencia nos permite visualizar de manera clara y detallada cómo los actores externos interactúan con los componentes Boundary, cómo estos a su vez se comunican con los Controllers para coordinar la lógica del negocio, y cómo finalmente los Controllers interactúan con las Entities para llevar a cabo las operaciones necesarias. Mediante el diagrama de secuencia, podemos comprender y analizar de manera efectiva el flujo de información y la secuencia de eventos en el sistema, lo que nos ayuda a identificar posibles mejoras, optimizaciones y posibles problemas en la comunicación entre los diferentes componentes.

A continuación, vamos a presentar los casos de uso para dos ejemplos crear red y exportar red, aunque como se ha visto en la figura 6.1 algunos casos de uso incluyen o extienden otros en este caso solo mostraremos una de estas opciones al tener estos entre si un comportamiento similar.

Vamos a empezar por el diagrama de secuencia para crear red (figura 6.4) y sus caso de uso extendido crear router (figura 6.5). Se puede apreciar como el sistema va creando los distintos nodos en función de un archivo introducido por el usuario y este se configurar en función de esa configuración. Ahora si nos fijamos en el ultimo diagrama (figura 6.6) podemos observar los otros dos casos de uso, para este caso únicamente se necesita concocer el laboratorio y servidor `GNS3` donde se encuentra dicho proyecto.

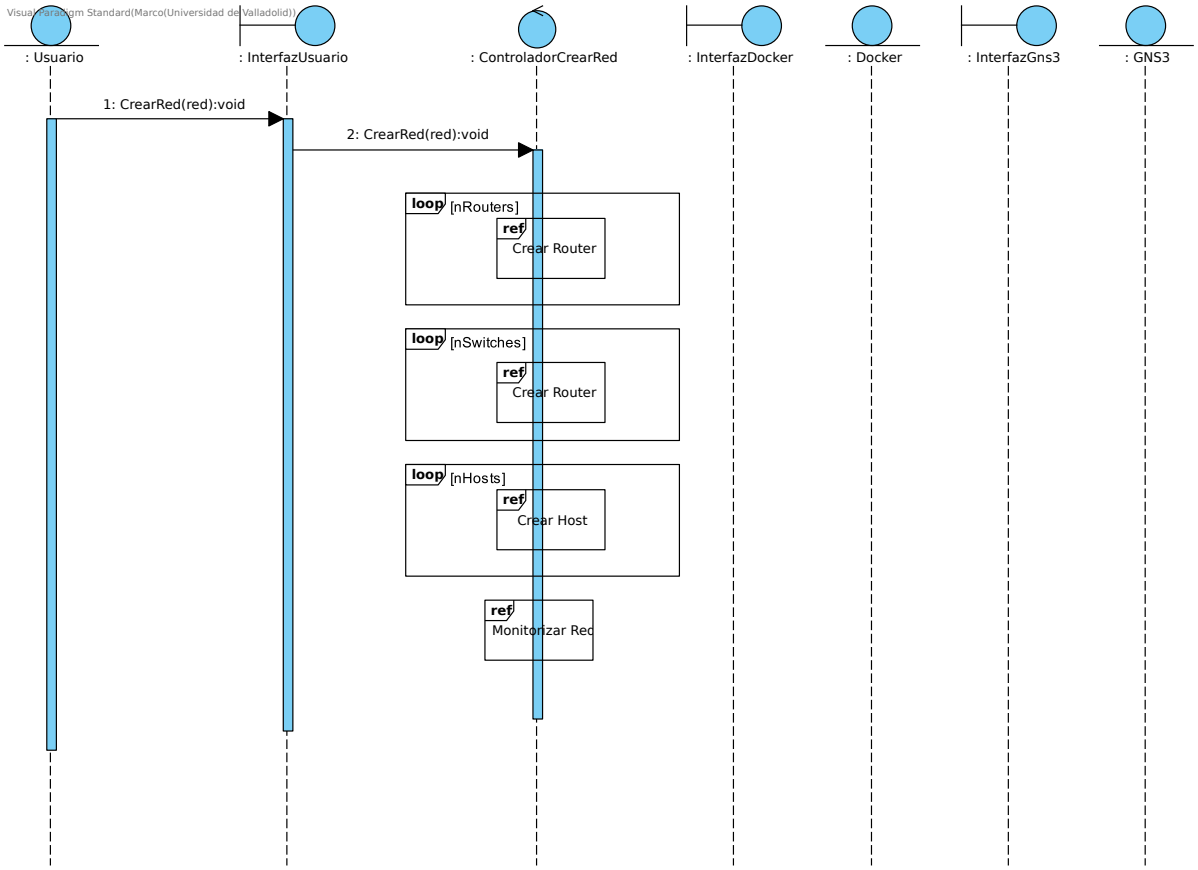


Figura 6.4: Diagrama de secuencia de análisis del caso de uso Crear Red

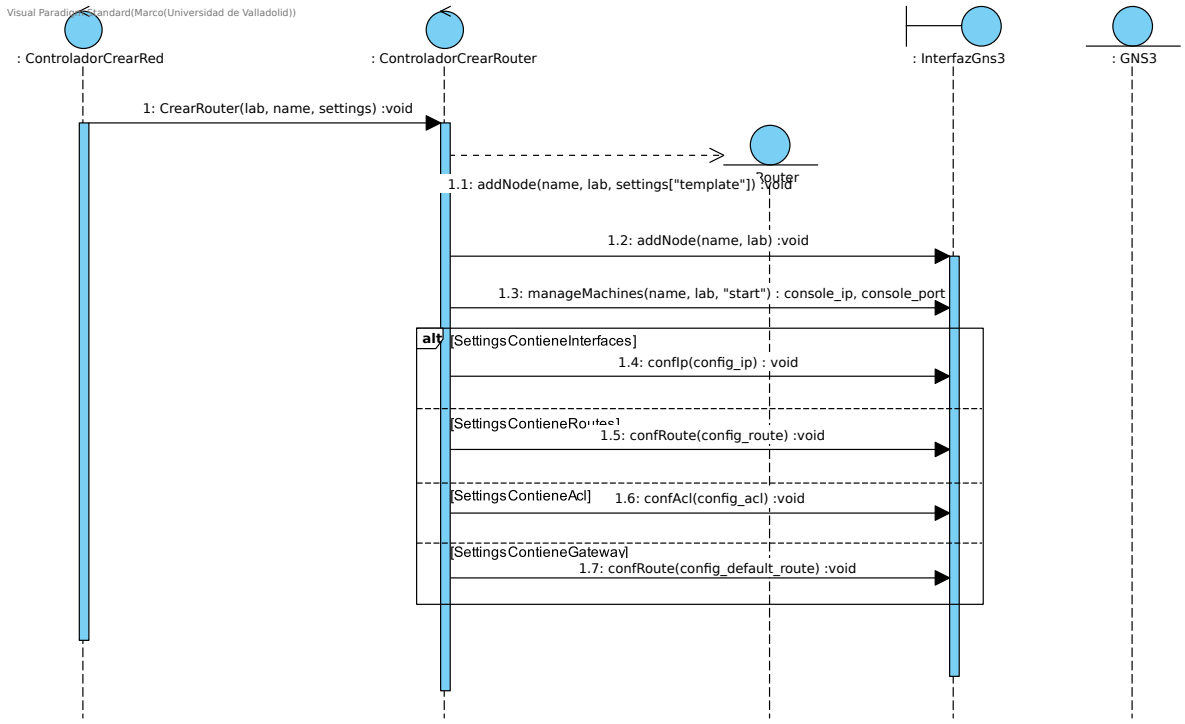


Figura 6.5: Diagrama de secuencia de análisis del caso de uso Crear Router

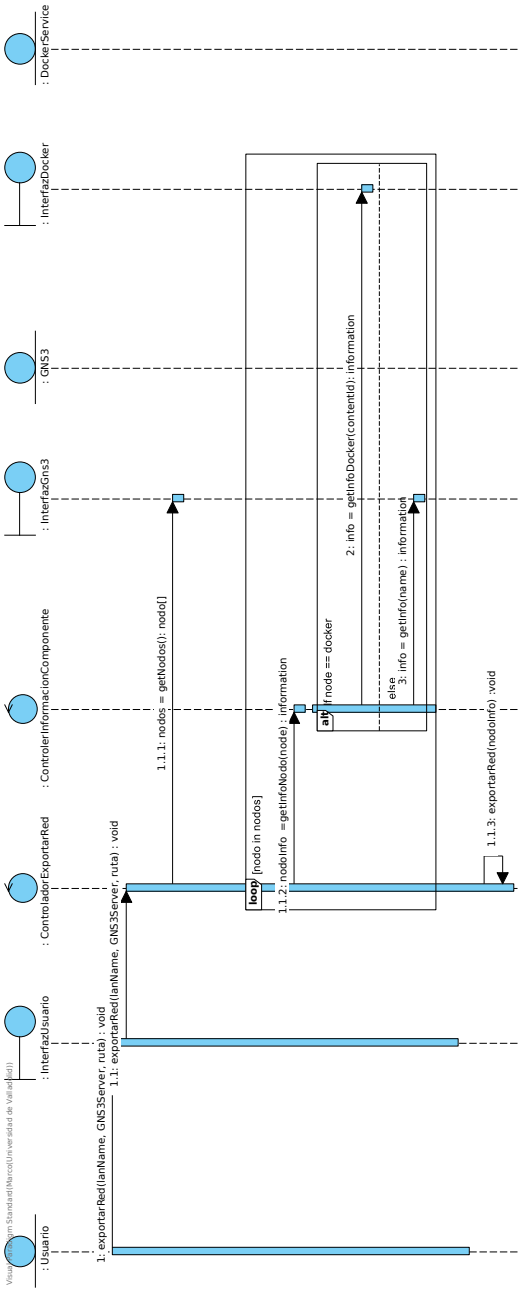


Figura 6.6: Diagrama de secuencia de análisis del caso de uso ExportarRed

Capítulo 7

Diseño

El diseño es una etapa crucial en el desarrollo de software, ya que establece la estructura y la organización del sistema. En este apartado, abordaremos el proceso de diseño de nuestro proyecto, donde se traducirán los requisitos y las especificaciones en un conjunto de componentes y su interacción.

El objetivo principal del diseño es crear una arquitectura robusta, eficiente y escalable que cumpla con los objetivos y requisitos del sistema. Durante esta etapa, se definirán los módulos, los scripts y las relaciones entre ellos, así como las estrategias de implementación y las consideraciones de rendimiento.

Además, en el diseño se busca maximizar la reutilización de componentes, fomentar la modularidad y facilitar el mantenimiento y la evolución del sistema a lo largo del tiempo. Se emplearán principios y patrones de diseño reconocidos para garantizar la calidad y la flexibilidad del software.

En resumen, en este apartado exploraremos el proceso de diseño del sistema, donde se establecerán las bases para la implementación exitosa del proyecto, teniendo en cuenta aspectos como la arquitectura, la modularidad, la reutilización de componentes y las consideraciones de rendimiento.

7.1 Arquitectura Lógica

Con todo lo expuesto en la fase de análisis se va a tratar de encontrar una solución a dicho problema, tratando de cumplir con todos los objetivos y requisitos marcados. En esta fase nos encontramos a un nivel mas bajo, otorgando modelos y diagramas mas concretos y próximos a la realidad.

Dentro de todas las arquitecturas estudiadas durante el grado universitario y un estudio para escoger el que se adapte mejor a nuestro problema se ha optado por escoger el patrón arquitectónico Filtro Tubería, al igual que en la fase de análisis se escogió el patrón BCE, esto se hace para poder modelar el problema con una mayor claridad.

El filtro tubería ha sido seleccionado debido a sus beneficios en términos de modularidad, flexibilidad y rendimiento. Este patrón arquitectónico nos permite dividir el proceso de procesamiento de datos en una secuencia de pasos interconectados, cada uno realizado por un componente independiente llamado filtro, favoreciendo el uso de scripts. Esto nos brinda la capacidad de escalar y reutilizar los filtros de manera eficiente, así como de aprovechar la concurrencia y el paralelismo para mejorar el rendimiento del sistema en el manejo de grandes volúmenes de datos. En la siguiente figura 7.1 podemos observar una imagen que nos permite entenderlo de forma mas sencilla.

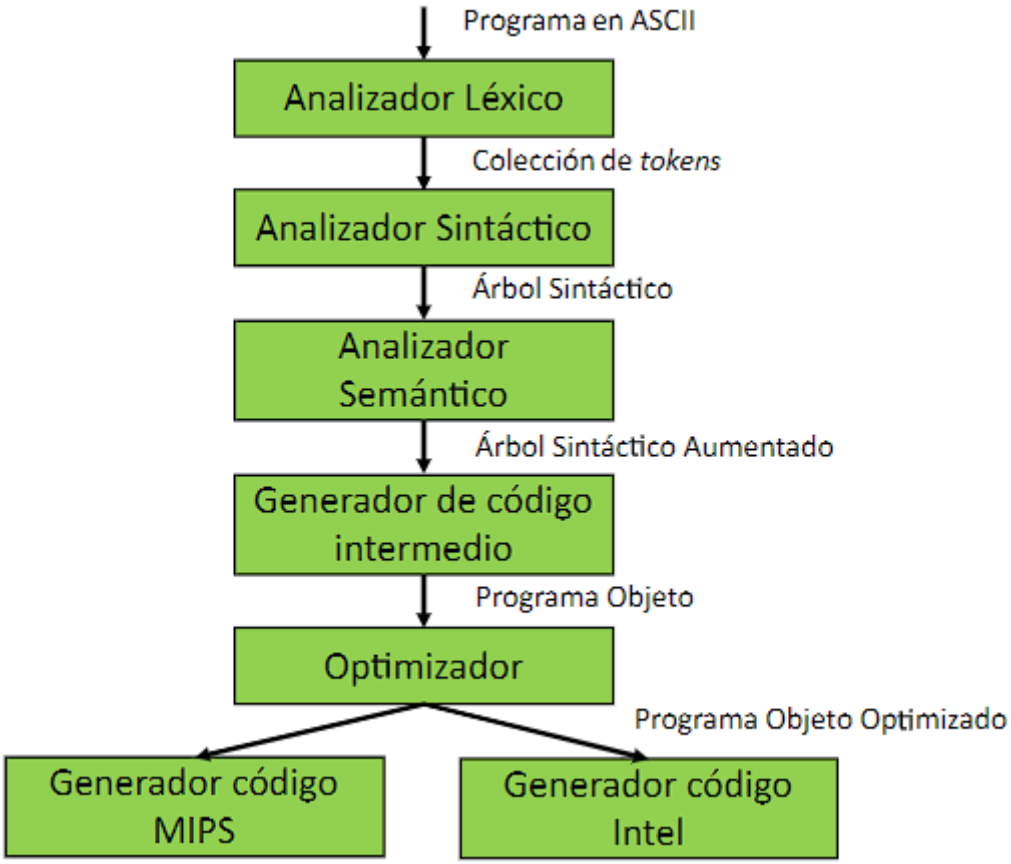


Figura 7.1: Patrón Filtro Tubería

A parte de sus ventajas para el desarrollo de herramientas basadas en scripts, también este patrón ha sido seleccionado porque el resto de patrones estudiados están mas orientados a una programación orientada a objetos y esto realizar grandes adaptaciones al patrón para que se adaptase el tipo de aplicación que se quería desarrollar.

Una vez conociendo el patrón vamos a modelar nuestra aplicación, para ello se ha decidido dividir los scripts en tareas secuenciales. Para el script de crear la red han surgido en total 4 subtareas: la creación, la configuración, la conexión y la monitorización de cada nodo como se puede apreciar en la figura 7.2. En cambio, para los otros dos scripts han surgido únicamente dos subtareas: infracción nodo, información conexiones, como se puede apreciar en la figura 7.3.

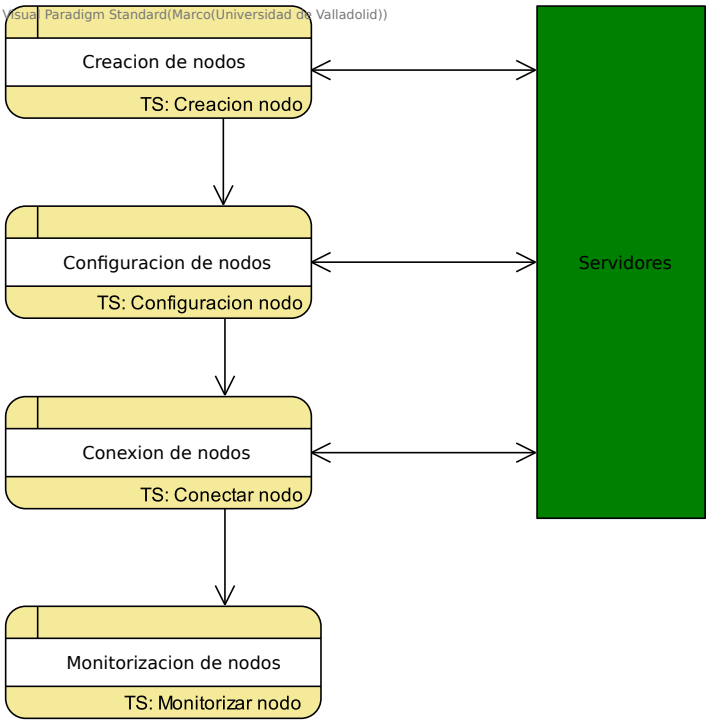


Figura 7.2: Patrón Filtro Tubería para Crear Red

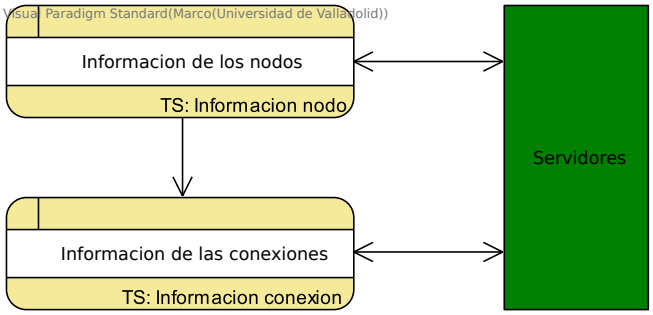


Figura 7.3: Patrón Filtro Tubería para Obtener Información

7.2 Patrones de diseño

El patrón arquitectónico define la estructura lógica del sistema, mostrando los componentes lógicos del mismo, como subsistemas y objetos, y las relaciones entre ellos, sin considerar el soporte físico que los sustentará, como bases de datos, clientes o servidores. El patrón arquitectónico sirve de base para el patrón de diseño, que define el comportamiento y diseño de los diferentes subsistemas, refinándolos y proporcionando directrices específicas. En resumen, el patrón arquitectónico establece la estructura general del sistema, mientras que el patrón de diseño brinda detalles más específicos y concretos sobre la implementación de cada subsistema.

En este caso vista la arquitectura lógica vamos a seleccionar un patrón de diseño que se adecue a este. En el contexto de nuestro sistema, hemos optado por utilizar el patrón Transaction Script, el cual organiza cada una de las tareas o subtareas en transacciones individuales. Por ejemplo, en una biblioteca, podríamos tener dos grandes casos de uso: prestar y devolver libros. Aplicando el patrón Transaction Script, cada uno de estos casos de uso puede encapsularse en un Transaction Script que se encargue de llevar a cabo dicha tarea específica. Si dividimos una tarea en subtareas más pequeñas, se generarán más Transaction Scripts, uno para cada subtarea.

Es importante destacar que el patrón Transaction Script no está orientado a objetos, lo que lo hace ideal para nuestro sistema. En nuestro diseño, hemos decidido representar cada filtro, es decir, cada subtarea o subsistema, utilizando un Transaction Script que será responsable de llevar a cabo sus respectivas funcionalidades. De esta manera, podemos organizar y gestionar de manera efectiva las diferentes operaciones y lógica de negocio de nuestro sistema, proporcionando una estructura clara y modular.

La Figuras 7.2 representa la arquitectura lógica del script de creación. La tarea principal, que es la creación de una red, se ha dividido en varias subtareas representadas por 4 filtros conectados entre sí mediante tuberías: la creación de los nodos, la conexión de los nodos, la configuración de los nodos y la monitorización de los nodos. Además, todos los filtros están conectados al servidor `GNS3`, ya que existe un flujo constante de información con él.

En el nivel del patrón de diseño, cada filtro se representa mediante un Transaction Script, el cual se encarga de realizar la tarea asignada al filtro y, al finalizar, llama al siguiente Transaction Script transmitiéndole la información necesaria para que realice su propia tarea. Los datos de entrada para el proceso son un archivo JSON proporcionado por el usuario, que contiene la configuración a seguir. El resultado final, es decir, los datos de salida, seria es un proyecto `GNS3` completamente utilizable por el usuario.

Para el segundo srcipt (figura 7.3) únicamente se han creado dos subtareas, la obtención de la información del nodo y sus conexiones. A través de conexiones al servidor `GNS3` o al servicio de docker. Una vez obtenida este información se devolverá un JSON con la configuración del laboratorio o se imprimirá la información del nodo en función del caso de uso que nos encontremos.

Esta arquitectura basada en filtros y Transaction Scripts permite una estructura modular y organizada, donde cada filtro se encarga de una tarea específica y se comunica de manera secuencial con los demás filtros para lograr la creación exitosa de la red en `GNS3`.

7.3 Arquitectura Física

La arquitectura física del sistema se basa en un entorno distribuido que consta de varios componentes interconectados, cada uno con un rol específico en el funcionamiento del sistema. En primer lugar, se encuentran los servidores `GNS3`, respectivamente, que proporcionan la capacidad de gestionar y controlar los nodos virtuales de la red, así como la creación y gestión de contenedores. Para ellos se ha desplegado dos servidores especializados.

Además de los servidores `GNS3`, hay otro servidor dedicado que juega un papel crucial en la ar-

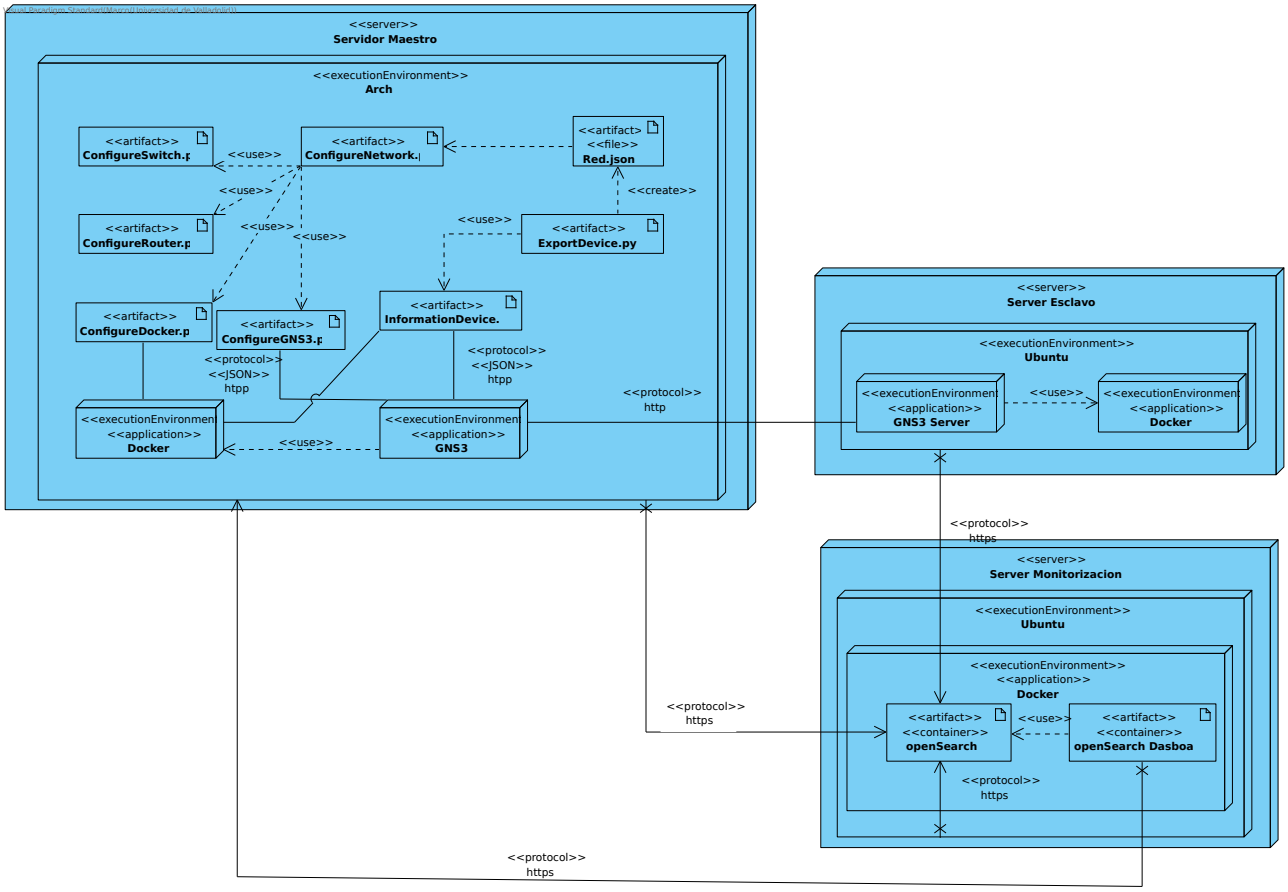


Figura 7.4: Diagrama de despliegue del sistema

quitectura: el servidor de monitorización. Este servidor alberga la aplicación de monitorización OpenSearch, que se utiliza para recopilar y analizar los datos de la red. OpenSearch proporciona capacidades avanzadas de monitoreo y generación de informes, permitiendo obtener una visión detallada del tráfico de red, la salud de los componentes y otras métricas relevantes.

Cabe mencionar que los servidores `GNS3` y el servidor de monitorización se comunican entre sí a través de la red, utilizando protocolos estándar para intercambiar información y comandos necesarios para la creación y configuración de la red virtual.

Por ultimo, en el servidor principal se ha decidido ejecutar los scripts creados, todo esto lo podemos apreciar en la figura 7.4.

En resumen, la arquitectura física del sistema está compuesta por dos servidores `GNS3`, un servidor de monitorización con la aplicación OpenSearch y un servidor adicional para alojar la aplicación. Estos componentes trabajan en conjunto para brindar la funcionalidad requerida, permitiendo la creación, configuración y monitorización de la red virtual.

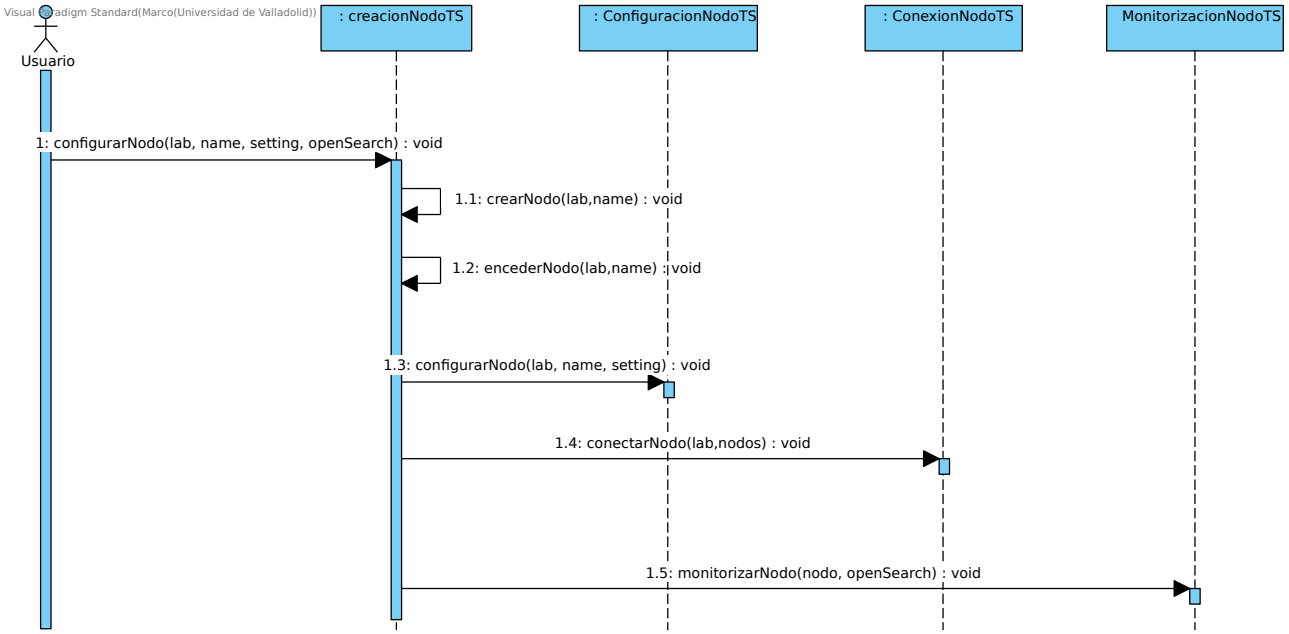


Figura 7.5: Diagrama de secuencia para creación de nodos

7.4 Diagrama de secuencia

En el contexto del diseño de sistemas, los diagramas de frecuencia juegan un papel importante al proporcionar una representación visual de las interacciones frecuentes entre los componentes. Estos diagramas nos permiten identificar y comprender mejor las partes del sistema que están más involucradas en las operaciones recurrentes. Al mostrar la frecuencia de las interacciones, podemos identificar patrones, cuellos de botella y áreas de optimización en el sistema.

La utilidad de los diagramas de frecuencia radica en su capacidad para mostrar las relaciones más activas y repetitivas entre los componentes del sistema. Esto puede me ha ayudado a tomar decisiones informadas sobre cómo optimizar el rendimiento y la eficiencia del sistema. Además, los diagramas de frecuencia también pueden me han sido útiles para identificar oportunidades de reutilización de código o para determinar qué partes del sistema son críticas en términos de rendimiento y deben ser mejoradas.

En los siguientes diagramas se va a poder ver el uso de los Transaction Scripts definidos durante el diseño. Cabe destacar que todos los nodos tienen el mismo desarrollo pero en función del tipo se configuraran o exportaran las características asociadas, también hay que remarcar que el usuario introduce un JSON y a partir de el se va a llamando los derivados scripts en función de lo introducido en el y por ultimo como cada nodo es independiente se crea un hilo para que se puedan configurar en paralelo.

En resumen, el diseño del sistema ha sido cuidadosamente planificado y estructurado para garantizar un funcionamiento eficiente y escalable. La utilización de patrones arquitectónicos y el enfoque en la separación de responsabilidades han permitido una organización clara y modular de los componentes del sistema. El uso del patrón Transaction Script ha facilitado la gestión de la lógica de negocio y la ejecución de tareas específicas de manera independiente. Además, la arquitectura física ha sido

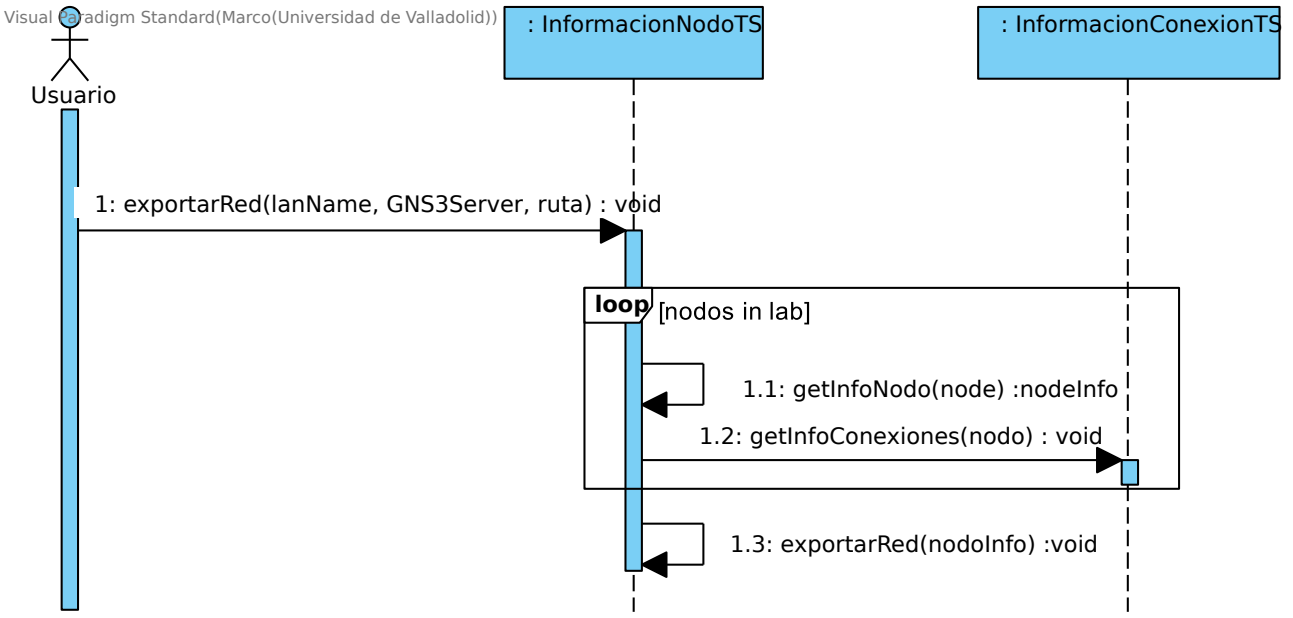
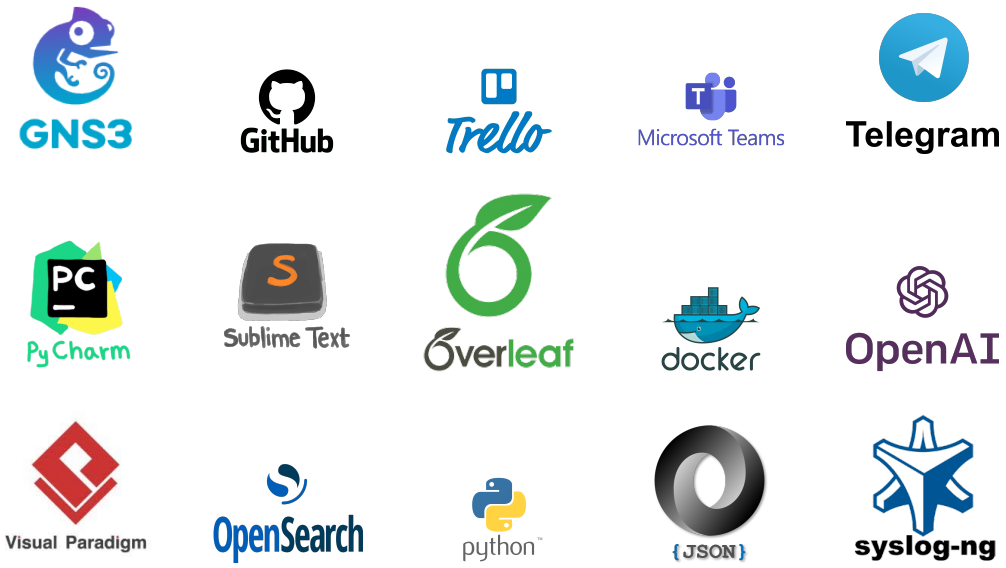


Figura 7.6: Diagrama de secuencia para obtener informacion

diseñada considerando la integración de tecnologías como `GNS3` y Docker, aprovechando al máximo sus capacidades para la creación y configuración de nodos virtuales. En conjunto, estos elementos han permitido un diseño robusto y flexible, capaz de adaptarse a las necesidades del sistema y ofrecer un rendimiento óptimo.

Capítulo 8

Implementación



Cuadro 8.1: Logos de las herramientas empleadas en este trabajo

8.1 Herramientas de Desarrollo

8.1.1 GNS3

GNS3 (Graphical Network Simulator 3) es una herramienta de simulación de redes para el diseño, prueba y resolución de problemas en redes de computadoras. Esta herramienta se utiliza en proyectos de red, como en el desarrollo de redes empresariales, de campus y de proveedores de servicios.

GNS3 es una herramienta de software libre y gratuito que proporciona una simulación de red en tiempo real, lo que significa que los usuarios pueden experimentar con diferentes configuraciones de red sin afectar a una red real. La herramienta utiliza una combinación de dispositivos virtuales, como routers y switches, para simular una red virtual en un entorno de prueba.

Algunas de las **ventajas** que ofrece GNS3 son:

+ Permite a los usuarios experimentar con diferentes configuraciones de red y probar soluciones sin tener que gastar tiempo y recursos en la implementación en una red real.
+ Permite la creación de topologías de red complejas y la integración con otros sistemas de automatización de red, como Ansible.

GNS3 también tiene sus **desventajas** como:

- Requiere una cantidad significativa de recursos de hardware para ejecutar simulaciones de red complejas.
- Requiere un conocimiento técnico sólido y experiencia en el diseño y configuración de redes de computadoras.

En resumen, GNS3 es una herramienta de simulación de redes útil para el diseño, prueba y resolución de problemas en redes de computadoras. Ofrece una forma eficiente y segura de probar soluciones de red antes de implementarlas en una red real. Sin embargo, requiere un conocimiento técnico sólido y recursos de hardware adecuados para ejecutar simulaciones complejas.

8.1.2 GitHub

GitHub es una plataforma en línea de control de versiones y colaboración para el desarrollo de software. Permite a los usuarios almacenar, administrar y compartir su código fuente y proyectos de software. La plataforma es utilizada por millones de desarrolladores en todo el mundo y se ha convertido en una herramienta esencial en el proceso de desarrollo de software.

Las capacidades de GitHub incluyen un sistema de control de versiones, que facilita a los desarrolladores rastrear los cambios en su código fuente y colaborar en el mismo proyecto. Los usuarios también pueden contribuir a proyectos de código abierto, lo que fomenta la colaboración y el intercambio de ideas.

Ventajas:

+ Es fácil de usar y ofrece una amplia gama de herramientas de colaboración, como solicitudes de extracción, comentarios y etiquetas, lo que posibilita la colaboración entre desarrolladores.
+ Ofrece un sistema de seguimiento de problemas, lo que permite a los usuarios informar de problemas y colaborar en su resolución.

+ Ofrece una plataforma de alojamiento gratuito para proyectos de código abierto, lo que permite a los desarrolladores compartir su trabajo con la comunidad de forma gratuita.

Desventajas:

- No es adecuada para proyectos de software privados y comerciales, ya que requiere que los proyectos sean públicos.
- Puede ser menos adecuada para proyectos de software que no estén basados en código, como proyectos de diseño gráfico o documentos.

En resumen, GitHub es una plataforma esencial para el desarrollo de software, que ofrece una amplia gama de herramientas de colaboración y control de versiones. Su facilidad de uso y alojamiento gratuito para proyectos de código abierto lo han convertido en una herramienta popular para la comunidad de desarrolladores en todo el mundo. A pesar de algunas limitaciones, GitHub es una plataforma de referencia para la colaboración y el desarrollo de software en equipo.

8.1.3 Trello

Trello es una herramienta en línea de gestión de proyectos que utiliza el método `Kanban` para visualizar el flujo de trabajo. El software es utilizado por empresas y equipos de desarrollo de software en todo el mundo para colaborar en proyectos y mejorar la eficiencia del equipo.

Las capacidades de Trello incluyen la creación de tableros, listas y tarjetas para organizar el flujo de trabajo. Los usuarios pueden añadir tareas, notas y archivos adjuntos a las tarjetas, lo que facilita la colaboración y la comunicación entre los miembros del equipo. Además, Trello permite establecer plazos y recordatorios para cada tarea, lo que ayuda a los equipos a cumplir con los plazos del proyecto.

Ventajas:

+ Es una herramienta intuitiva y fácil de usar, lo que facilita la adopción por parte del equipo.
+ Permite una fácil visualización del progreso del proyecto y la gestión de tareas.
+ Ofrece integraciones con otras herramientas de software, como Microsoft Teams, lo que mejora la colaboración y la eficiencia del equipo.
+ Tiene una versión gratuita que ofrece muchas de las funcionalidades básicas para la gestión de proyectos.

Sin embargo, también existen algunas **desventajas** de Trello, como:

- La versión gratuita tiene algunas limitaciones en cuanto a la cantidad de tableros, tarjetas y archivos adjuntos que se pueden almacenar.

- La herramienta puede ser menos adecuada para proyectos muy grandes o complejos, ya que la organización puede volverse difícil.
- No es adecuada para la gestión de proyectos que requieren una gran cantidad de documentación y seguimiento.

En resumen, Trello es una herramienta de gestión de proyectos en línea altamente visual y personalízale que es utilizada por equipos de todo el mundo. Ofrece una amplia gama de capacidades para administrar proyectos y colaborar con otros miembros del equipo. Sus ventajas incluyen la facilidad de uso, la colaboración en tiempo real y la posibilidad de integrarse con otras herramientas de software, mientras que sus desventajas incluyen algunas limitaciones en la versión gratuita y la dificultad de gestionar proyectos muy grandes o complejos.

8.1.4 Microsoft Teams

Microsoft Teams es una plataforma de colaboración en línea que permite a los equipos comunicarse y colaborar en proyectos. Es una herramienta de chat, videoconferencia y colaboración en documentos, que integra varias aplicaciones de Microsoft, como Word, Excel, OneNote y PowerPoint.

Las capacidades de Microsoft Teams incluyen la capacidad de comunicarse en tiempo real mediante chat o videoconferencia, colaborar en documentos de forma simultánea, compartir archivos y administrar proyectos. Además, Microsoft Teams también cuenta con una gran cantidad de aplicaciones y complementos para ampliar sus capacidades.

Las **ventajas** de Microsoft Teams incluyen:

+ Ofrece una amplia gama de capacidades de colaboración y comunicación, que incluyen chat, videoconferencia y colaboración en documentos.
+ Integra varias aplicaciones de Microsoft en una sola plataforma, lo que facilita la colaboración y la gestión de proyectos.
+ Es una herramienta segura y confiable que cumple con los requisitos de seguridad y privacidad de la empresa.

Sin embargo, también existen algunas **desventajas** de Microsoft Teams, como:

- Puede haber problemas de estabilidad y rendimiento en función del número de usuarios y la carga de trabajo.
- La interfaz de usuario puede ser confusa para algunos usuarios, lo que puede dificultar su uso.
- La integración con otras herramientas y aplicaciones puede ser limitada.

En resumen, Microsoft Teams es una plataforma de colaboración en línea que ofrece una amplia gama de capacidades de comunicación y colaboración en documentos. Integra varias aplicaciones de

Microsoft en una sola plataforma y es una herramienta segura y confiable. Sus ventajas incluyen la amplia gama de capacidades, la integración de aplicaciones y la seguridad, mientras que sus desventajas incluyen problemas de estabilidad y rendimiento, interfaz de usuario confusa y limitada integración con otras herramientas y aplicaciones.

8.1.5 Telegram

`Telegram` es una aplicación de mensajería instantánea en línea que permite a los usuarios enviar mensajes, archivos y realizar llamadas de voz y vídeo. Fue desarrollado por los hermanos Dúrov en 2013 y ha ganado popularidad debido a sus características de seguridad y privacidad.

Las capacidades de `Telegram` incluyen la capacidad de enviar mensajes de texto, imágenes, vídeos, archivos y mensajes de voz. También permite la creación de grupos de chat con un número ilimitado de miembros, así como la posibilidad de crear canales para la transmisión de información a un público más amplio.

Las **ventajas** de `Telegram` incluyen:

+ Ofrece una gran cantidad de características de seguridad y privacidad, como el cifrado de extremo a extremo y la autodestrucción de mensajes.
+ Es una plataforma multiplataforma que se puede utilizar en diferentes dispositivos y sistemas operativos.
+ Permite la creación de grupos de chat con un número ilimitado de miembros y la creación de canales para la transmisión de información a un público más amplio.
+ No necesitas conocer el número de teléfono para poder contactar con otro usuario de la aplicación.

Sin embargo, también existen algunas **desventajas** de `Telegram`, como:

- La adopción de la aplicación puede ser limitada en comparación con otras aplicaciones de mensajería instantánea como `WhatsApp`.
- Algunas de las características de seguridad y privacidad, como la autodestrucción de mensajes, pueden no ser utilizadas por todos los usuarios.
- La plataforma no tiene la misma integración con otras aplicaciones que algunas de sus competidoras.

En resumen, `Telegram` es una aplicación de mensajería instantánea que ofrece una gran cantidad de características de seguridad y privacidad, así como la capacidad de crear grupos de chat y canales para la transmisión de información. Sus ventajas incluyen características de seguridad y privacidad, plataforma multiplataforma y capacidad de crear grupos de chat y canales, mientras que sus desventajas incluyen adopción limitada, características de seguridad y privacidad no utilizadas por todos los usuarios y limitada integración con otras aplicaciones.

8.1.6 PyCharm

PyCharm es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) utilizado principalmente para el desarrollo de aplicaciones en el lenguaje de programación Python. Fue desarrollado por la compañía JetBrains y se encuentra disponible en una versión de pago y una versión de código abierto llamada PyCharm Community Edition.

Entre las capacidades de PyCharm se encuentran el soporte para el desarrollo web con frameworks como Django y Flask, el autocompletado de código, la depuración de aplicaciones, el análisis de código y la integración con herramientas de control de versiones como Git y Mercurial.

Ventajas:

+ PyCharm cuenta con una amplia variedad de características que facilitan el desarrollo de aplicaciones en Python.
+ El soporte para frameworks web como Django y Flask permite una mayor productividad al momento de desarrollar aplicaciones web.
+ La función de autocompletado de código reduce el tiempo necesario para escribir código y evita errores por tipografía.
+ La integración con herramientas de control de versiones como Git y Mercurial facilita el trabajo en equipo y el control de cambios en el código.

Desventajas:

- PyCharm es un software pesado y puede requerir de una computadora potente para un uso fluido.
- La versión de pago puede ser costosa para algunos usuarios, pero incluye una licencia para estudiantes.
- Puede haber una curva de aprendizaje inicial para los usuarios que no estén familiarizados con la interfaz de PyCharm.

En conclusión, PyCharm es una herramienta potente y versátil para el desarrollo de aplicaciones en Python, especialmente para aquellos que trabajan en proyectos web utilizando frameworks como Django y Flask. Aunque puede requerir una computadora potente y puede tener un precio alto para algunos usuarios, sus características y su integración con herramientas de control de versiones hacen que sea una herramienta valiosa para el desarrollo de software en equipo.

8.1.7 Sublime Text

Sublime Text es un editor de texto avanzado diseñado para desarrolladores de software y programadores. Es compatible con Windows, macOS y Linux, y es conocido por su facilidad de uso, velocidad y potentes capacidades de edición de texto.

Sublime Text está escrito en C++ y utiliza una interfaz de programación de aplicaciones (API) personalizada que permite a los desarrolladores escribir complementos y personalizar el editor para satisfacer sus necesidades específicas. También incluye un motor de búsqueda y reemplazo avanzado, soporte para múltiples selecciones y un sistema de comandos de teclado personalizado que permite a los usuarios realizar tareas comunes de edición de texto de manera eficiente.

Sublime Text tiene una amplia variedad de capacidades que lo convierten en una herramienta de programación poderosa. Algunas de sus principales características incluyen:

- Soporte para múltiples lenguajes de programación, incluyendo Python, Java, C++, HTML, CSS y JavaScript.
- Resaltado de sintaxis avanzado que facilita la lectura del código y la identificación de errores.
- Autocompletado inteligente que sugiere palabras clave, funciones y variables mientras se escribe el código.
- Múltiples selecciones que permiten editar varias partes del código simultáneamente.
- Edición de texto rápida y precisa que permite a los desarrolladores trabajar de manera eficiente.
- Complementos y paquetes personalizados que amplían la funcionalidad del editor.

A continuación, se detallan las principales **ventajas e inconvenientes** de Sublime Text:

+ Interfaz de usuario intuitiva y fácil de usar.
+ Potentes capacidades de edición de texto que permiten a los desarrolladores trabajar de manera eficiente.
+ Personalízale a través de complementos y paquetes para satisfacer las necesidades específicas de los usuarios.
+ Amplio soporte de la comunidad y documentación en línea.
- No es de código abierto, lo que significa que no se puede acceder al código fuente y modificarlo.
- El precio de la licencia es relativamente alto en comparación con otros editores de texto, aunque no requiere licencia para poder usarlo
- No tiene todas las características que ofrecen algunos de sus competidores, como la integración con Git. Esta se incluye en una aplicación aparte conocida como sublime merge.

Sublime Text es un editor de texto avanzado que ofrece una gran cantidad de características y capacidades para los desarrolladores de software. Aunque tiene algunas desventajas, sus ventajas superan con creces las desventajas, lo que lo convierte en una herramienta popular entre los programadores de todo el mundo. Si bien existen alternativas más económicas y de código abierto, Sublime Text sigue siendo una opción popular para aquellos que buscan un editor de texto avanzado y personalízale.

8.1.8 Overleaf

Overleaf es un editor de documentos de LaTeX en línea que permite a los usuarios escribir, compilar y compartir documentos de manera colaborativa. Es compatible con todos los sistemas operativos y no requiere la instalación de software adicional en la computadora del usuario.

Overleaf se basa en LaTeX, un lenguaje de marcado utilizado para la creación de documentos científicos y técnicos. Los usuarios pueden escribir en LaTeX directamente en el editor en línea de Overleaf y ver los resultados en tiempo real. Además, Overleaf utiliza un sistema de compilación en la nube, lo que significa que el procesamiento de documentos se realiza en servidores remotos en lugar de en la computadora del usuario.

Overleaf tiene una amplia variedad de capacidades que lo convierten en una herramienta de edición de documentos de LaTeX poderosa. Algunas de sus principales características incluyen:

- Edición de documentos de LaTeX en tiempo real.
- Compilación en la nube de documentos de LaTeX.
- Colaboración en tiempo real con otros usuarios.
- Integración con servicios de almacenamiento en la nube como Dropbox y Google Drive.
- Plantillas y ejemplos de documentos para facilitar el proceso de escritura.
- Soporte para paquetes de LaTeX personalizados.

A continuación, se detallan las principales **Ventajas e inconvenientes** de Overleaf:

+ Fácil acceso y uso a través de un navegador web sin necesidad de instalar software adicional.
+ Compilación en la nube rápida y eficiente que ahorra tiempo y recursos en la computadora del usuario.
+ Posibilidad de colaborar en tiempo real con otros usuarios, lo que permite una mayor eficiencia y productividad en proyectos colaborativos.
+ Amplia variedad de plantillas y ejemplos de documentos para facilitar el proceso de escritura.
+ Posibilidad de integración con servicios de almacenamiento en la nube para facilitar el acceso y la gestión de documentos.
- Requiere una conexión a Internet estable para utilizar todas sus capacidades.
- Al ser una plataforma en línea, los documentos están almacenados en la nube y no en la computadora del usuario, lo que puede plantear preocupaciones de privacidad y seguridad.
- Las capacidades de personalización son limitadas en comparación con un editor de LaTeX de escritorio tradicional.

Overleaf es una herramienta de edición de documentos de LaTeX en línea que ofrece una gran cantidad de capacidades para los usuarios. Aunque tiene algunas desventajas, sus ventajas superan con creces las desventajas, lo que lo convierte en una herramienta popular entre los usuarios de LaTeX de todo el mundo. Si bien existen alternativas de escritorio de LaTeX, Overleaf es una opción popular para aquellos que buscan un editor en línea fácil de usar y colaborativo para trabajar en proyectos de LaTeX.

8.1.9 Docker

Docker es una tecnología de virtualización de contenedores que permite la creación y distribución de aplicaciones en un entorno aislado y portátil. Docker utiliza el sistema operativo del host para compartir recursos de hardware y software, lo que lo hace más eficiente que la virtualización tradicional de máquinas virtuales.

Las capacidades de Docker incluyen la creación y gestión de contenedores, el uso de imágenes y repositorios de imágenes, la integración con herramientas de orquestación de contenedores y la posibilidad de ejecutar aplicaciones en diferentes entornos.

Ventajas:

+ **Portabilidad**: Las aplicaciones en contenedores de Docker se pueden ejecutar en cualquier entorno que tenga Docker instalado, lo que facilita la migración de aplicaciones entre entornos.
+ **Eficiencia**: Docker utiliza el sistema operativo del host para compartir recursos de hardware y software, lo que lo hace más eficiente que la virtualización tradicional de máquinas virtuales.
+ **Escalabilidad**: Docker permite la creación y gestión de múltiples contenedores para una aplicación, lo que facilita la escalabilidad horizontal de la aplicación.
+ **Reutilización**: Las imágenes de Docker se pueden reutilizar para crear múltiples contenedores, lo que facilita la creación y gestión de entornos de desarrollo, pruebas y producción.

Inconvenientes:

- **Complejidad**: Docker puede ser complejo de configurar y gestionar, especialmente para aplicaciones complejas que requieren múltiples contenedores.
- **Seguridad**: La seguridad de los contenedores depende de la configuración correcta de Docker y de la seguridad del sistema operativo del host.
- **Rendimiento**: Aunque Docker es más eficiente que la virtualización tradicional de máquinas virtuales, aún puede haber una sobrecarga de rendimiento al ejecutar aplicaciones en contenedores.

En resumen, Docker es una tecnología de virtualización de contenedores que ofrece portabilidad, eficiencia, escalabilidad y reutilización, pero puede ser compleja de configurar y gestionar, y la seguridad y el rendimiento pueden ser un problema. A pesar de estos inconvenientes, Docker es una

tecnología muy popular en la actualidad debido a sus numerosas ventajas en el despliegue de aplicaciones.

8.1.10 OpenIA

OpenAI es una organización de investigación en inteligencia artificial con el objetivo de crear inteligencia artificial de propósito general segura y beneficiosa para la humanidad. Una de las herramientas más conocidas de OpenAI es ChatGPT, un modelo de lenguaje basado en transformer que puede generar texto coherente y humano como respuesta a preguntas o comentarios de los usuarios.

Las capacidades de ChatGPT incluyen la capacidad de generar respuestas coherentes y contextuales a partir de un texto de entrada, la comprensión de diferentes lenguajes y el aprendizaje continuo a través de la retroalimentación de los usuarios.

Ventajas:

+ **Eficiencia**: ChatGPT puede generar respuestas en tiempo real, lo que lo hace ideal para aplicaciones de atención al cliente y chatbots.
+ **Personalización**: ChatGPT puede ser personalizado y entrenado para adaptarse a las necesidades específicas de una empresa o industria.
+ **Escalabilidad**: ChatGPT puede ser utilizado en múltiples canales de comunicación y puede manejar grandes volúmenes de interacciones de usuario.
+ **Aprendizaje continuo**: ChatGPT puede aprender y mejorar a través de la retroalimentación de los usuarios y la recopilación de datos.

Inconvenientes:

- **Limitaciones lingüísticas**: Aunque ChatGPT puede comprender y generar texto en varios idiomas, aún puede haber limitaciones en términos de gramática y sintaxis en idiomas menos conocidos o complejos.
- **Sesgo**: ChatGPT puede ser sesgado en función de los datos utilizados para entrenar el modelo, lo que puede generar respuestas parciales o discriminatorias.
- **Privacidad**: ChatGPT puede recopilar información personal del usuario, lo que plantea preocupaciones de privacidad.
- **Accesibilidad**: Aunque ChatGPT es una herramienta poderosa, su uso puede estar limitado por la complejidad técnica y el costo asociado.

En resumen, OpenAI y ChatGPT son herramientas de inteligencia artificial con capacidad de generar texto coherente y personalízale. Aunque presentan numerosas ventajas en términos de eficiencia, personalización, escalabilidad y aprendizaje continuo, también presentan limitaciones lingüísticas, sesgo, preocupaciones de privacidad y accesibilidad. Como tal, es importante considerar cuidadosamente el uso de estas herramientas y abordar cualquier preocupación ética o técnica asociada.

8.1.11 Visual Paradigm

Visual Paradigm es una herramienta integral de modelado y diseño de software que ofrece una amplia gama de funcionalidades para el desarrollo de sistemas y aplicaciones. Permite a los desarrolladores visualizar, diseñar y documentar los diferentes aspectos de un proyecto de software, incluyendo diagramas de clases, diagramas de secuencia, diagramas de actividad, entre otros. Visual Paradigm proporciona una interfaz intuitiva y fácil de usar, lo que facilita la creación y modificación de modelos de software de manera eficiente.

Ventajas:

- **Versatilidad**: Visual Paradigm es una herramienta versátil que cubre una amplia gama de necesidades de modelado y diseño de software, lo que la hace adecuada para proyectos de diferentes tamaños y complejidades.
- **Colaboración en equipo**: La herramienta permite a los equipos de desarrollo colaborar de manera efectiva en la creación y modificación de modelos de software, facilitando la comunicación y el intercambio de ideas entre los miembros del equipo.
- **Generación automática de código**: Visual Paradigm ofrece la capacidad de generar automáticamente código fuente a partir de los modelos creados, lo que acelera el proceso de desarrollo y reduce la posibilidad de errores.
- **Amplia gama de notaciones y estándares**: La herramienta soporta múltiples notaciones y estándares de modelado, lo que permite a los desarrolladores adaptarse a las prácticas y convenciones establecidas en su campo de trabajo.

Desventajas:

- **Curva de aprendizaje**: Dado que Visual Paradigm ofrece numerosas funcionalidades y opciones de configuración, puede requerir un tiempo de aprendizaje y familiarización antes de poder utilizar todas sus capacidades de manera efectiva.
- **Costo**: Visual Paradigm es una herramienta comercial, por lo que su adquisición puede implicar un costo para los proyectos y equipos de desarrollo con recursos limitados.

En resumen, Visual Paradigm es una herramienta integral de modelado y diseño de software que ofrece numerosas ventajas para el desarrollo de proyectos. Con su amplia gama de diagramas y herramientas, brinda versatilidad y permite visualizar, diseñar y documentar todos los aspectos importantes de un proyecto de software. Además, facilita la colaboración en equipo, fomentando la comunicación y el intercambio de ideas, y ofrece la generación automática de código, lo que acelera el proceso de desarrollo y reduce errores. En resumen, utilizar Visual Paradigm mejora la eficiencia del desarrollo de software al proporcionar una plataforma completa y colaborativa.

8.1.12 OpenSearch

OpenSearch es un conjunto distribuido, basado en la comunidad, con licencia de Apache 2.0, totalmente de código abierto para búsqueda y análisis. Se usa para una amplia gama de situaciones, como el monitoreo de aplicaciones en tiempo real, el análisis de registros y la búsqueda en sitios web. OpenSearch proporciona un sistema altamente escalable para proporcionar acceso y respuesta rápidos a volúmenes grandes de datos con una herramienta de visualización integrada, OpenSearch Dashboards, lo que facilita a los usuarios analizar sus datos. OpenSearch cuenta con tecnología de la biblioteca de búsqueda Apache Lucene y admite diversas capacidades de búsqueda y análisis, como la búsqueda de k vecinos más cercanos (KNN), SQL, detección de anomalías, Machine Learning Commons, análisis de rastreos, búsqueda de texto completo y más.

OpenSearch se creó como una alternativa de código abierto a Elasticsearch y Kibana, que anteriormente eran marcas registradas de Elastic. En abril de 2021, Elastic cambió la licencia de Elasticsearch y Kibana de Apache 2.0 a Server Side Public License (SSPL), lo que llevó a AWS a bifurcar la versión de Elasticsearch anterior y lanzarla como OpenSearch.

OpenSearch le permite ingerir, proteger, buscar, añadir, ver y analizar datos fácilmente para una variedad de casos de uso, como el análisis de registros, la búsqueda de aplicaciones, la búsqueda empresarial y más. Con OpenSearch se puede beneficiar de tener un producto de código abierto al 100 % que puede usar, modificar, ampliar, monetizar y revender de la forma que quiera. Existe un creciente número de socios del proyecto OpenSearch que ofrecen una variedad de servicios, como soporte profesional, características mejoradas y servicios administrados de OpenSearch. El proyecto OpenSearch continúa brindándole un conjunto seguro de búsqueda y análisis de alta calidad con un

OpenSearch es una opción atractiva como alternativa a Elasticsearch debido a su enfoque de código abierto. Al elegir OpenSearch, los usuarios pueden aprovechar las capacidades de indexación, búsqueda y análisis de datos similares a Elasticsearch, mientras se benefician de la libertad y transparencia que brinda el código abierto, evitando las restricciones de licencia asociadas con Elasticsearch.

8.2 Implementación

8.2.1 Python

Python es un lenguaje de programación interpretado de alto nivel, diseñado para ser fácil de leer y escribir. Es ampliamente utilizado en el mundo de la informática, especialmente en aplicaciones científicas y de análisis de datos. Python es conocido por su simplicidad, elegancia y eficiencia.

Las capacidades de Python incluyen:

- **Versatilidad**: Python se puede utilizar para una amplia gama de aplicaciones, desde la creación de aplicaciones web hasta la automatización de tareas y la realización de análisis de datos complejos.
- **Librerías**: Python cuenta con una amplia variedad de librerías y herramientas de terceros para facilitar la programación en diferentes áreas.
- **Sintaxis sencilla**: Python tiene una sintaxis sencilla y fácil de leer, lo que facilita la comprensión del código y su mantenimiento.
- **Interpretación en tiempo real**: Python es un lenguaje interpretado, lo que permite una rápida retroalimentación y corrección de errores en tiempo real.
- **Multiplataforma**: Python puede ser utilizado en diferentes sistemas operativos, lo que lo hace altamente portátil.

Ventajas:

+ **Facilidad de uso**: Python es fácil de aprender y utilizar, lo que lo hace ideal para principiantes en programación.
+ **Eficiencia**: Python es un lenguaje de alto nivel que proporciona una mayor productividad con menos líneas de código que otros lenguajes de programación.
+ **Librerías**: Python cuenta con una amplia variedad de librerías y herramientas de terceros para facilitar la programación en diferentes áreas.
+ **Comunidad activa**: Python tiene una gran comunidad de usuarios y desarrolladores que proporcionan soporte y contribuyen al desarrollo de nuevas herramientas y librerías.
+ **Versatilidad**: Python se puede utilizar en diferentes áreas, desde el desarrollo web hasta el análisis de datos y la inteligencia artificial.

Inconvenientes:

- **Velocidad**: Aunque Python es fácil de utilizar, no es tan rápido como otros lenguajes de programación, lo que puede ser un problema en aplicaciones que requieren un alto rendimiento.

- **Escalabilidad**: Aunque Python es flexible, no es tan escalable como otros lenguajes de programación, lo que puede ser un problema en aplicaciones con grandes volúmenes de datos.
- **Falta de tipado** : Python es un lenguaje de programación dinámico, lo que significa que no tiene una verificación de tipo en tiempo de compilación. Esto puede llevar a errores de tiempo de ejecución si no se tiene cuidado.
- **Compatibilidad**: Python 2 y Python 3 son versiones incompatibles del lenguaje, lo que puede ser un problema si se trabaja con código que fue escrito en una versión anterior.

En resumen, Python es un lenguaje de programación popular y versátil que ofrece numerosas ventajas en términos de simplicidad, flexibilidad, comunidad activa y librerías y frameworks. Sin embargo, también presenta limitaciones en términos de velocidad, escalabilidad, falta de tipado y compatibilidad con versiones anteriores. Como tal, es importante evaluar cuidadosamente si Python es el lenguaje de programación adecuado para un proyecto en particular.

8.2.2 JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero y fácil de leer y escribir. Es comúnmente utilizado para transmitir datos entre un servidor y una aplicación web, pero también se puede utilizar en otros contextos.

Las capacidades de JSON incluyen la capacidad de representar datos complejos de una manera simple y legible, la interoperabilidad con otros lenguajes y herramientas, y la capacidad de ser analizado y generado por programas fácilmente.

Ventajas:

+ **Legibilidad**: JSON es fácil de leer y entender, lo que facilita la depuración y el mantenimiento del código.
+ **Interoperabilidad**: JSON es compatible con otros lenguajes y herramientas, lo que permite la transferencia de datos entre diferentes sistemas.
+ **Simplicidad**: JSON es un formato de intercambio de datos simple y fácil de escribir, lo que reduce el tiempo y el esfuerzo requeridos para implementarlo.
+ **Eficiencia**: JSON es un formato de datos compacto que reduce la cantidad de datos que se transmiten a través de la red.

Inconvenientes:

- **Limitaciones**: JSON no es adecuado para todos los tipos de datos, lo que puede requerir la implementación de otras soluciones para manejar casos específicos.
- **Seguridad**: JSON no proporciona seguridad incorporada, lo que puede ser un problema si se utiliza para transmitir datos sensibles.

- **Deserialización**: La deserialización de JSON puede ser costosa en términos de recursos de CPU y memoria, lo que puede afectar el rendimiento en aplicaciones de alta carga.

En resumen, JSON es un formato de intercambio de datos popular y útil que ofrece numerosas ventajas en términos de legibilidad, interoperabilidad, simplicidad y eficiencia. Sin embargo, también presenta limitaciones en términos de limitaciones de datos, seguridad y deserialización costosa. Como tal, es importante evaluar cuidadosamente si JSON es el formato adecuado para un proyecto en particular.

8.2.3 Syslog-ng

Syslog-ng es una solución de código abierto que se utiliza para la recolección, procesamiento y almacenamiento de logs en sistemas distribuidos. Proporciona una infraestructura sólida y flexible para centralizar y gestionar registros de múltiples fuentes, lo que facilita la monitorización y el análisis de eventos en tiempo real. Mediante su arquitectura modular y su capacidad de filtrado y enriquecimiento de logs, Syslog-ng permite una personalización avanzada y una integración fluida con otros sistemas y herramientas de gestión.

Ventajas:

- **Flexibilidad**: Syslog-ng ofrece una amplia gama de opciones de configuración y personalización, lo que permite adaptarlo a las necesidades específicas de cada entorno.
- **Escalabilidad**: Esta herramienta es capaz de gestionar grandes volúmenes de logs de manera eficiente, lo que la hace adecuada para entornos distribuidos y de alto rendimiento.
- Seguridad: Syslog-ng ofrece mecanismos de autenticación y cifrado que garantizan la integridad y confidencialidad de los registros, protegiendo así la información sensible.
- **Interoperabilidad**: Syslog-ng es compatible con diversos protocolos y formatos estándar, lo que facilita la integración con otros sistemas y herramientas de gestión.

Desventajas

- **Curva de aprendizaje**: Dado que Syslog-ng ofrece numerosas opciones de configuración, puede requerir un tiempo de aprendizaje y familiarización antes de aprovechar todo su potencial.
- **Mantenimiento**: Al tratarse de una herramienta de código abierto, es necesario realizar actualizaciones y revisiones periódicas para asegurar su correcto funcionamiento y mantenerse al día con las últimas mejoras y parches de seguridad.

En resumen, este Trabajo utiliza la herramienta Syslog-ng, destacando sus ventajas en términos de flexibilidad, escalabilidad, seguridad e interoperabilidad. Además, se abordan las desventajas relacionadas con la curva de aprendizaje y el mantenimiento.

Capítulo 9

Pruebas

Durante la fase de pruebas, que es una etapa fundamental en el desarrollo de cualquier software, se ha tratado de validar el correcto funcionamiento de los módulos y garantizar la calidad del producto final. En nuestro caso, el diseño basado en Transaction Script nos brinda una gran ventaja en cuanto a la facilidad de realizar pruebas. Cada filtro implementado en nuestro sistema, representado por un Transaction Script, encapsula una tarea específica y se encarga de llevarla a cabo de manera autónoma. Esta modularidad nos permite realizar pruebas unitarias exhaustivas en cada uno de los filtros, verificando su funcionamiento individualmente. Además, al seguir un enfoque basado en scripts, podemos simular distintos escenarios y configuraciones, lo que facilita la validación de la interoperabilidad entre los diferentes filtros y la correcta integración de todo el sistema.

A posteriori de validar cada modulo funcionara por si solo se paso por otras dos fases antes de concluir que el trabajo cumplía todos los requisitos propuestos. La primera de estas dos fases fue la integración de todos los módulos para poder crear laboratorios completos y no únicamente configurar los apartados específicos de cada nodo, estas pruebas se realizaron en un sistema local y con un único nodo de `GNS3` y docker. Cuando se consiguió este hito y se configuraron todos los servidores se paso la ultima fase de pruebas, que fue la integración de nuestra aplicación para los casos en que el sistema pueda ser distribuido y los nodos se puedan virtualizar en cualquiera de los servidores disponibles.

En el siguiente diagrama (figura 9.1) se puede observar de una forma mas visual como se ha ido pasando por todas estas fases. En un alto nivel las fases han sido:

1. Validar de manera individual de cada posible configuración y creación de nodos, como por ejemplo las ips, rutas, etc. sentando las bases de la aplicación.
2. Comprobar que se pudiera configurar de manera completa cada nodo.
3. Comprobar las conexiones y validar que las comunicaciones entre ellos.
4. Crear una red de prueba de forma local.

5. En cuanto se concluyese la configuración de los servidores, hacer pruebas para fases distribuidas.

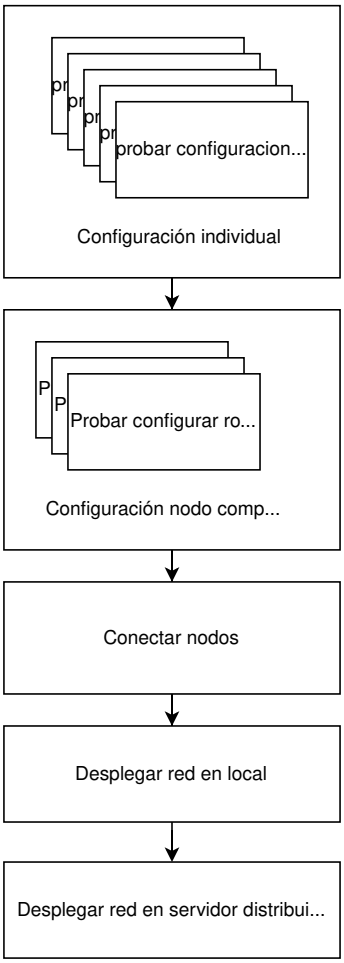


Figura 9.1: Diagrama con las fases de pruebas

Estas pruebas las podemos clasificar de las siguiente manera, en función de lo comprueben serian:

- **Pruebas de Unicidad** [44] - Las pruebas de unicidad, también conocidas como pruebas unitarias, se centran en la verificación del correcto funcionamiento de componentes o módulos individuales de un sistema de software. Estas pruebas se realizan de forma aislada para comprobar que cada unidad funciona según lo esperado.
- **Pruebas de Integración** [43] - Las pruebas de integración se llevan a cabo para verificar la correcta interacción y funcionamiento conjunto de diferentes componentes o módulos de un sistema de software. Estas pruebas evalúan cómo se combinan y comunican las diferentes partes del sistema, detectando posibles problemas de integración.
- **Pruebas del Sistema** [45] - Las pruebas del sistema se enfocan en evaluar el sistema de software en su conjunto, verificando que cumpla con los requisitos y funcionalidades especificadas. Estas pruebas se realizan en un entorno similar al de producción y abarcan pruebas funcionales, de rendimiento, de seguridad y otras pruebas necesarias para asegurar la calidad y correcto desempeño del sistema.

Para ejemplificar las fases que se acaban de comentar vamos a ver este proceso la creación y configuración de la ip de un router hasta la creación completa de una red que incluya esta necesidad. Se puede apreciar en la siguiente tabla 9.1, donde podemos apreciar cada prueba, en que consiste y el resultado obtenido de ella. Podemos obtener dos posibles resultados, OK o KO, siendo el primero la finalización de la prueba de forma satisfactoria, y en el segundo caso acabando la prueba de forma errónea.

Nombre	**Descripción**	**Resultado**
Crear router	Crear un nuevo router en la red.	OK
Configurar IP del router	Asignar la dirección IP introducida al router.	OK
Configurar router completo	Configurar todos los parámetros de un router existente.	OK
Crear una red en local	Crear una red local con múltiples nodos conectados entre si.	OK
Crear una red en varios servidores	Crear una red donde cada nodo se pueda encontrar en un servidor diferente.	OK

Cuadro 9.1: Resultado de las pruebas realizadas para el Router

Una vez explicado como se pasara evolucionaría el sistema según se va validando, vamos a mostrar las pruebas del ejemplo anterior para el sistema de caja negra [39]. Una prueba de caja negra [39] es una técnica de prueba de software en la que se evalúa el comportamiento de un sistema sin conocer su estructura interna o su lógica subyacente. Se enfoca en la entrada y salida del sistema, tratándolo como una caja negra en la que se prueban los datos de entrada y se analiza la correspondiente salida esperada.

Se ha tomado la decisión de ir validando las pruebas con pruebas de caja negra [39] porque gracias a estas pruebas se puede comprobara funcionalidad de la aplicación y encontrar errores que no se han valorado durante el resto de etapas.

Código	PU01
Nombre	Crear router
Descripción	Crear el router seleccionado en la red.
Precondición	El template del router existe.
Acciones realizadas	1. Se conecta al servidor `GNS3` 2. Se crea un laboratorio donde se va a simular el router 3. Se crea el router escogido por el usuario 4. Se enciende el router creado
Resultado	OK

Cuadro 9.2: Prueba de caja negra para Crear un Router

Código	PU02
Nombre	Configurar IP del router
Descripción	Asignar la dirección IP introducida al router.
Precondición	El template del router ha sido creado.
Acciones realizadas	1. Se conecta al router creado previamente 2. Se configura la IP y mascara seleccionadas
Resultado	OK

Cuadro 9.3: Prueba de caja negra para Configurar IP del Router

Código	PI01
Nombre	Configurar router completo
Descripción	Configurar todos los parámetros de un router existente.
Precondición	El template del router existe.
Acciones realizadas	1. Se conecta al servidor `GNS3` 2. Se crea un laboratorio donde se va a simular el router 3. Se crea el router escogido por el usuario 4. Se enciende el router creado 5. Se conecta al router 6. Se configuran los parámetros seleccionados por el usuario
Resultado	OK

Cuadro 9.4: Prueba de caja negra para Configurar router completo

Código	PS01
Nombre	Crear una red en local
Descripción	Crear una red local con múltiples nodos conectados entre si.
Precondición	El usuario tiene conexión con el servidor `GNS3`
Acciones realizadas	1. Se conecta al servidor `GNS3` 2. Se crea o abre el laboratorio escogido por el usuario 3. Se crean los componentes escogido por el usuario 4. Se encienden los nodos creados 5. Se conectan los nodos entre si 6. Se configuran los parámetros de cada nodo
Resultado	OK

Cuadro 9.5: Prueba de caja negra para Crear una Red en Local

Código	PS02
Nombre	Crear una red en varios servidores
Descripción	Crear una red donde cada nodo se pueda encontrar en un servidor diferente.
Precondición	El usuario tiene conexión con el servidor `GNS3`
Acciones realizadas	1. Se conecta al servidor `GNS3` Maestro 2. Se crea o abre el laboratorio escogido por el usuario 3. Se crean los componentes escogido por el usuario, independientemente del servidor `GNS3` 4. Se encienden los nodos creados 5. Se conectan los nodos entre si 6. Se configuran los parámetros de cada nodo
Resultado	OK

Cuadro 9.6: Prueba de caja negra para Crear una Red en varios Servidores

Capítulo 10

Conclusiones

En primer lugar, mencionar que se han cumplido todos los objetivos esenciales que se han propuesto para este proyecto, pudiendo finalizar con todos los requisitos que se han identificado en la fase de análisis, incluso los propuestos por el tutor de este trabajo de fin de grado.

Con este trabajo se ha conseguido demostrar la potencia, utilidad y libertad que tiene la herramienta de simulación `GNS3` para poder simular redes empresariales, todo esto se puede observar gracias a los scripts dedicados a este ámbito. Por otro lado para llevar un control de lo que ocurre en nuestro sistema se ha llevado a cabo el uso de otras aplicaciones, se ha demostrado lo útil que son herramientas de registro en una red informática y que junto con una suite de análisis se puede llevar un control eficaz de lo que ocurre en los sistemas. Por ultimo, se ha podido observar como gracias a herramientas de código abierto, como las que han sido usadas en este proyecto, se puede construir proyectos de alta calidad y de gran potencia como el que se ha puesto en marcha con este trabajo.

Personalmente, el resultado final del proyecto ha cumplido con las expectativas iniciales, aunque según se ha ido avanzado en el proyecto y se iban adquiriendo nuevos conocimientos se ha visto que todavía se pueden moldear mas utilidades para nuestro sistema que para lo que ha sido diseñado. Porque aunque gracias a las tecnologías y herramientas utilizadas se ha podido crear un script para el diseño y monitorización de redes virtuales que se puede utilizar en varios ámbitos, se ha observado que todavía estas herramientas permiten realizar muchas mas utilidades que las previstas en un inicio.

Durante el desarrollo del proyecto se ha tenido una comunicación directa con el tutor a través de varias formas, desde reuniones hasta por medio de aplicaciones de mensajería. Cabe destacar que este ha estado disponible para cualquier necesidad que surgiese durante el desarrollo, aportando formas para resolver problemas, aportando su visión y necesidades que tiene que tener esta herramienta y comunicándose con el servicio técnico para solicitar las máquinas donde se ha realizado el despliegue del sistema.

Para acabar, durante toda la duración de este proyecto se ha conseguido poner a prueba, mejorar y aprender todos los conocimientos que se han instruido durante estos años en la universidad. Siendo capaz de utilizar todos los conocimientos que se imparten en distintas asignaturas de esta carrera,

observando su utilidad y como son de necesarios para el desarrollo de proyectos en el ámbito de software. En el ámbito personal, me ha servido para desarrollarme en muchos ámbitos, ver mi potencial y aprender nuevos aspectos de la informática. Como finalización de esta valoración final me gustaría aclarar que me quedo totalmente satisfecho con mi trabajo en el proyecto realizado.

10.1 Trabajo futuro

Entre las posibles mejoras y añadidos que se podrían aplicar a este TFG. El código que se encuentra disponible en un repositorio de GitHub [18], para que cualquier persona que quiera usarlo, modificarlo o mejorarlo tenga la posibilidad y se sienta libre de hacerlo.

Como se ha comentado en las conclusiones durante el desarrollo de este trabajo se pudieron observar que este sistema todavía puede mejorar y cumplir nuevos requisitos que no han sido abordados en este proyecto.

A continuación, voy a plantear algunas mejoras que se me fueron ocurriendo para el apartado de la creación de redes virtuales: añadir la posibilidad de automatizar la configuración de nuevos sistemas operativos, poder utilizar otros sistemas de virtualización que nos permite `GNS3`, monitorizar de forma automática los equipos virtualizados, etc. También se podría plantear la creación de una pagina para que los usuarios pudieran subir su redes y máquinas para compartirlo entre los miembros de la comunidad.

Por último, dentro del sistema desplegado, se podría añadir ciertas mejoras como por ejemplo asegurar las conexiones entre los equipos a través de certificados autofirmados, modificar el accesos al servidor `GNS3` a través de una conexión cifrada, etc.

Apéndices

Apéndice A

Manual de Instalación

En este manual se va a enseñar como preparar los equipos para su correcto funcionamiento con los scripts creados, se va a explicar como instalarlo en un equipo Linux pero el procedimiento es similar para cualquier otro Sistema Operativo utilizando los paquetes específicos de dicho sistema.

A.1 Preparar equipo Local

Para el correcto funcionamiento de los scripts se necesita tener instalado `Python3` y una serie de librerías, estas se encuentran en el archivo requirements.txt y se pueden instalar a través de pip.

Se van a tener que seguir los siguientes pasos:

1. Clonar repositorio CyberCrunch
2. Instalar pyhon3 y pip.
3. Instalar librerías del fichero requirements.txt con pip.

Listing A.1: Preparar equipo local

```
git clone https://github.com/marcocacho/CyberCrunch
sudo apt update
sudo apt install python3 python3-pip
pip install -r CyberCrunch/requirements.txt
```

Una vez usados esos comandos el equipo ya esta listo para usar la aplicación sin problemas, para aprender a construir una red, expórtala o ver información podemos ir al apéndice de Manual de Usuario.

A.2 Preparar equipo para la Virtualización

Para preparar un equipo donde se pueda ejecutar el programa se va a necesitar los servicios de Docker y `GNS3Server`, opcionalmente se puede instalar el gui de `GNS3` y docker.

Los pasos a seguir son los siguientes:

1. Añadir el repositorio de `GNS3`
2. Instalar requisitos necesarios para su uso
3. Añadir repositorio de docker
4. Instalar `GNS3` y docker
5. Añadir el usuario a los nuevos grupos

Listing A.2: Instalar servicios

```
sudo add-apt-repository ppa:GNS3/ppa
sudo apt-get install apt-transport-https ca-certificates curl \
software-properties-common
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) stable"
sudo apt install GNS3-gui GNS3-server docker-ce
udo usermod -aG group user_name ubridge libvirt kvm wireshark docker
```

Una vez instalado las aplicaciones vamos a configurar las herramientas para poder utilizarlas.
Configurar `GNS3`:
En la figura A.1 podemos ver un ejemplo de como configurar nuestro servidor `GNS3`, nada mas instalar la aplicación de virtualización `GNS3` vamos a tener un fichero similar pero debemos modificar alguno de los parámetros para su correcto funcionamiento, este fichero lo podemos encontrar por ejemplo en la ubicación .config/gns3/2.2/gns3_server.conf. Algunos parámetros pueden ser:

- **host:** Si vamos a ejecutar el programa de un ordenador externo se tendrá que poner la IP por la que se quiere escuchar
- **port:** Por defecto `GNS3` usa el puerto 3080 si queremos usar otro lo podemos modificar
- **auth:** Si queremos activar el uso de usuario y contraseña para conectarme al equipo
- **user y password:** Datos del usuario y contraseña que se van a utilizar si se activa la autenticación.
- **ubridge_path:** ruta al servicio ubridge, por defecto no suele dar problemas pero en caso contrario se tendra que modificar.

```
[Server]
host = 192.168.1.2
port = 3080
images_path = /opt/GNS3/images
projects_path = /opt/GNS3/projects
report_errors = True
auth = True
user = user
password = pasword1234.
path = /usr/bin/GNS3server
ubridge_path = /usr/bin/ubridge
appliances_path = /home/usuario/GNS3/appliances
additional_images_paths =
symbols_path = /home/usuario/GNS3/symbols
configs_path = /home/usuario/GNS3/configs
auto_start = True
allow_console_from_anywhere = True
protocol = http
console_start_port_range = 5000
console_end_port_range = 10000
udp_start_port_range = 10000
udp_end_port_range = 2000
```

Figura A.1: Fichero configuración GNS3_server.conf

Configurar docker:
En la figura A.3 podemos ver un ejemplo de como configurar nuestro servidor docker, en este caso no se va a crear ningún archivo de configuración pero si se quiere conectar al servicio desde una maquina externa va a ser necesario, en cualquier otro caso esta configuración puede ser omitida.

Para configurar el servicio docker de forma sencilla con crear y añadir en la ruta /etc/docker/kk.json.

Listing A.3: Fichero configuración kk.json

```
1 {
2   "hosts": ["tcp://192.168.1.2:2376", "unix:///var/run/docker.sock"],
3   "tls": false
4 }
```

Activar y Arrancar Servicios:
Una vez hemos configurado ambos servicios vamos a proceder a activar y ejecutar los demonios para poder empezar a usarlos. Para ellos ejecutamos los siguientes comandos.

Listing A.4: activas servcios

```
sudo systemctl enable GNS3server
sudo systemctl start GNS3server
sudo systemctl enable docker
sudo systemctl start docker
```

Servidor Distribuido:
Esta configuracion previa se tendra que seguir en todos los servidor que se quieran usar, después se tendrá que conectar ambos servidores con `GNS3`, esto se puede hacer como se aprecia en la siguiente imagen A.2.

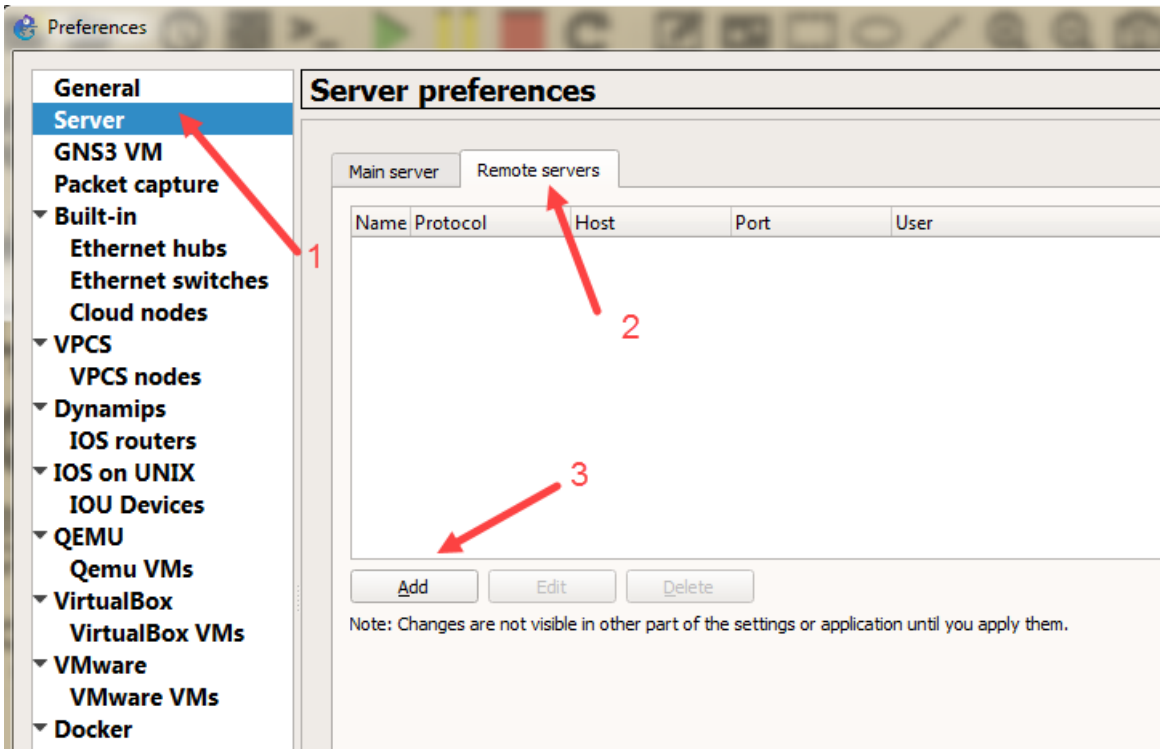


Figura A.2: Conectar servidor remoto

Una vez tenemos esto ya se podría usar el programa con nuestros servidores de forma autónoma.

A.3 Preparar equipo para la Monitorización

Por último, vamos a enseñar a configurar los equipos para monitorízalos y a instalar una herramienta para tratar los datos para poder utilizarlos.

En este caso se optado por usar las herramientas de syslog, mas concretamente se ha usado el servicio de syslog-ng, para la monitorización en los equipos y para poder trabajar con los datos recopilados

se ha usado OpenSearch, las servicios proporcionados se han configurado en dockers en la máquina dedicada a este fin.

Configurar Syslog:
Vamos a empezar por la monitorización de los equipos a través de syslog-ng, para ello tenemos que instalar este servicio.

Listing A.5: Instalar syslog-ng

```
sudo apt update
sudo apt install syslog-ng
sudo systemctl enable syslog-ng
```

Una vez instalado vamos a mostrar como poder enviar los logs a nuestra máquina con OpenSearch, esta herramienta permite monitorizar todo tipo de eventos que se producen en el equipo, en este caso únicamente se va a mostrar los eventos mas comunes pero en la wiki oficial podemos observar muchas otras opciones que podemos usar en función de nuestras necesidades.

El archivo de configuración por defecto lo encontramos en la ruta /etc/syslog-ng/syslog-ng.conf, aunque nuestra configuración se va a encontrar en el un archivo .conf de la ruta /etc/syslog-ng/conf.d/ y para que el servicio añada nuestros archivos específicos se tiene que añadir al final del archivo syslog-ng.conf la linea: @include /etc/syslog-ng/conf.d/*.conf".

El archivo de .conf se compone de 3 partes: el origen (source), el destino (destination) y el log. Se puede apreciar un ejemplo en el cuadro A.6.

Listing A.6: Fichero configuración syslog-ng

```
source src {
    system();
    internal();
};

destination d_opensearch_http {}
        elasticsearch-http(
                index("equipos_fisicos")
                type("")
                url("https://IP:9200/_bulk"
                tls(peer-verify(no))
                user("admin")
                password("admin")
                template("$(format-json --scope rfc5424
                                --scope dot-nv-pairs
                                --rekey .*
                                --shift 1
                                --scope nv-pairs
                                --exclude DATE
                                --key ISODATE
```

```
                                    @timestamp=${ISODATE}
                                    @equipo=\"maestro\"
                                    @hostname=\${HOST})")
        );
};

log {
    source(src);
    destination(d_opensearch_http);
    flags(flow-control);
};
```

Una vez hemos modificado la configuración reiniciamos el servicio con: **`sudo systemctl restart syslog-ng`**

Configurar OpenSearch:

Para instalar opensearch se necesita realizar un par de requisitos previos, el primero tener docker instalado al lanzar los servicios a través de el y tener un mapeo de memoria suficiente. Para instalar docker realizar lo mismo que se ha visto anteriormente:

Listing A.7: Instalar docker

```
sudo apt update
sudo apt install docker
sudo systemctl enable docker
sudo systemctl start docker
```

Para poder utilizar OpenSearch se necesita un mapeo de memoria mínimo de 262144, para ello vamos a comprobar si nuestro mapeo de memoria esta activo y es suficiente. Si no es suficiente o no esta activo tendremos que deshabilitamos la paginación de memoria y el rendimiento de intercambio en el host para mejorar el rendimiento: **`sudo swapoff -a`**, después aumentamos el número de mapas de memoria disponibles para OpenSearch:

Listing A.8: Ajustar mapeo memoria

```
# Editar el archivo de configuración de sysctl
sudo vi /etc/sysctl.conf

# Agregar una línea para definir el valor deseado
# o cambiar el valor si la clave ya existe,
# y luego guardar los cambios.

vm.max_map_count=262144
```

```
# Recargar los parámetros del kernel utilizando sysctl
sudo sysctl -p

# Verificar que el cambio se haya aplicado verificando el valor
cat /proc/sys/vm/max_map_count
```

Una vez hecho esto nuestro sistema esta disponible para levantar los dockers, para ellos primero nos bajamos las imágenes de la siguiente forma:

Listing A.9: Descagar Imagenes OpenSearch

```
docker pull opensearchproject/opensearch:latest
docker pull opensearchproject/opensearch-dashboards:latest
```

Una vez hecho esto podríamos crear a mano los contenedores, pero para una mayor comodidad se va a utilizar un docker-compose.yml para facilitar el despliegue. Un ejemplo seria el siguiente, que se recomienda para empezar en la web oficial:

,

Listing A.10: Fichero docker-compose.yml

```
1  version: '3'
2  services:
3     # This is also the hostname of the container within the Docker network
4     # (i.e. https://opensearch-node1/)
5    opensearch-node1:
6      # Specifying the latest available image - modify if you want a
       ↪ specific version
7      image: opensearchproject/opensearch:latest
8          container_name: opensearch-node1
9          environment:
10             # Name the cluster
11             - cluster.name=opensearch-cluster
12             # Name the node that will run in this container
13             - node.name=opensearch-node1
14             # Nodes to look for when discovering the cluster
15             - discovery.seed_hosts=opensearch-node1,opensearch-node2
16             # Nodes eligible to serve as cluster manager
17             - cluster.initial_cluster_manager_nodes=opensearch-node1,
                 ↪ opensearch-node2
18             - bootstrap.memory_lock=true # Disable JVM heap memory
                 ↪ swapping
19             # Set min and max JVM heap sizes to at least 50\% of system
                 ↪ RAM
20             - OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m
```

```
21      ulimits:
22        memlock:
23          soft: -1 # Set memlock to unlimited (no soft or hard limit)
24          hard: -1
25        nofile:
26          soft: 65536 # Maximum number of open files for the opensearch
              ↪ user - set to at least 65536
27          hard: 65536
28      volumes:
29        - opensearch-data1:/usr/share/opensearch/data # Creates volume
            ↪ called opensearch-data1 and mounts it to the container
30      ports:
31        - 9200:9200 # REST API
32        - 9600:9600 # Performance Analyzer
33      networks:
34        - opensearch-net # All of the containers will join the same Docker
            ↪ bridge network
35    opensearch-node2:
36      image: opensearchproject/opensearch:latest # This should be the same
          ↪ image used for opensearch-node1 to avoid issues
37      container_name: opensearch-node2
38      environment:
39        - cluster.name=opensearch-cluster
40        - node.name=opensearch-node2
41        - discovery.seed_hosts=opensearch-node1, opensearch-node2
42        - cluster.initial_cluster_manager_nodes=opensearch-node1,
            ↪ opensearch-node2
43        - bootstrap.memory_lock=true
44        - "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m"
45      ulimits:
46        memlock:
47          soft: -1
48          hard: -1
49        nofile:
50          soft: 65536
51          hard: 65536
52      volumes:
53        - opensearch-data2:/usr/share/opensearch/data
54      networks:
55        - opensearch-net
56    opensearch-dashboards:
57      # Make sure the version of opensearch-dashboards matches the version
          ↪ of opensearch
```

```
    # installed on other nodes
    image: opensearchproject/opensearch-dashboards:latest
    container\_name: opensearch-dashboards
    ports:
      - 5601:5601 # Map host port 5601 to container port 5601
    expose:
      - "5601" # Expose port 5601 for web access to OpenSearch Dashboards
    environment:
      OPENSEARCH_HOSTS: '[ "https://opensearch-node1:9200",
        ↪ "https://opensearch-node2:9200"]' # Define the OpenSearch
        ↪ nodes that OpenSearch Dashboards will query
    networks:
      - opensearch-net

volumes:
  opensearch-data1:
  opensearch-data2:

networks:
  opensearch-net:
```

Una vez tenemos nuestro propio lanzador lo podemos utilizar ejecutando el siguiente comando:

```
docker-compose up -d
```

Con esto ya tendríamos nuestros equipo de monitorización montado, dejando todos los equipos listos para ejecutar el código, para alojar las redes virtuales y para monitorizarlo. Cabe destacar que no tienen porque ser equipos diferentes, aunque en este TFG si ha sido para demostrar las capacidades de cada aplicación por separado.

Apéndice B

Manual de Usuario

Este apéndice esta dedicado a enseñar a los usuarios de esta aplicación sepan como utilizar las herramientas programadas y saber como escribir su fichero JSON para su correcto funcionamiento.

Vamos a empezar por su primer uso, el despliegue de laboratorios. Para que la aplicación funcione correctamente, como requisito previo los servidores que se van a utilizar tienen que tener conectividad con la máquina donde se ejecute la aplicación y que las máquinas a virtualizar existan en el entorno de pruebas.

El fichero JSON tiene que tener el siguiente formato B.1, únicamente son obligatorios los campos de “labName” y “GNS3” para poder crear o abrir el proyecto, el resto de campos se pueden incluir o no en función de los requisitos del proyecto que se quiere diseñar.

Listing B.1: Formato Fichero JSON

```
{
    "labName": name
    "GNS3": {
        "_comment": "datos del servidor GNS3",
        "ip": <ip del servidor de GNS3>,
        "port": puerto abierto del servidor de GNS3,
        "user": usuario (si no hay usuario o contraseña, dejar como ""),
        "pass": la_clave_que_toque,
    },
    "openSearch": {
        "ip": ip de la máquina donde se almacenan los logs,
        "port": puerto de la máquina abierto,
    },
    "nat": {
        "router": name,
```

```
16          "iface": interfaz del router reservada para el NAT,
17          "nat": interfaz del NAT
18      },
19      "components": lista con los componentes (si se trata de un nombre de
            ↪ un nodo, se van a diferenciar 3 posibles nodos distintos:
            ↪ router, switch o docker),
20      "connection\_list": Lista de conexiones con el siguiente formato por
            ↪ cada interfaz que se quiere conectar: [
21          {
22              "name": "nombre",
23              "interface": interfaz,
24          },
25          {
26              "name": "nombre",
27              "interface": interfaz,
28          }
29      ]
30
31
32  #Configuración de los componentes:
33
34  #Para un router:
35   "name": { nombre del router
36   "machineType": "router",
37   "template": template del router,
38   "OS": tipo de SO del router,
39   "gateway": ip router salida por defecto,
40   "interfaces": lista de interfaces con: {"iface": interfaz, "ip": ip,
         ↪ "netmask": máscara de la red},
41   "acls": lista de las ACLs con: { "list": número de la lista, "action":
         ↪ acción, "origin": ip origen,
42   "orNetmask": máscara origen, "dest": ip destino, "destNetmask": máscara
         ↪ destino, "protocol": protocolo,
43   "protocol": protocolo por nombre o número},
44   "interfaces_acl": lista de las ACLs y donde se aplican con: {
         ↪ "interface": interfaz, "list_acl": lista con las ACLs: {"list":
         ↪ número de la lista, "action": cuando se aplica} }
45   }
46
47  #Para un switch:
48   "name": { nombre del switch
49   "machineType": "switch",
50   "template": template del switch,
```

```
51  "OS": tipo de SO del switch,
52  "vlans": lista con las VLANs que se aplican con: {"number": número de
        ↪ la VLAN o trunk,
53  "interfaces": lista con las interfaces}
54  } \\
55
56 #Para un docker:
57  "name": { nombre del docker
58  "machineType": "docker",
59  "template": template del docker,
60  "docker_port": puerto en el que está levantado el servicio Docker en la
        ↪ máquina (opcional, sino será usado 2375)
61  "iface": nombre de la interfaz,
62  "ip": ip,
63  "netmask": mascarade la red,
64  "gateway": ip puerta de enlace
65  }
```

Una vez visto el modelo de como se puede crear os voy a enseñar a crear una red con varios segmentos de red, equipos y switches como se representa en la siguiente figura B.1

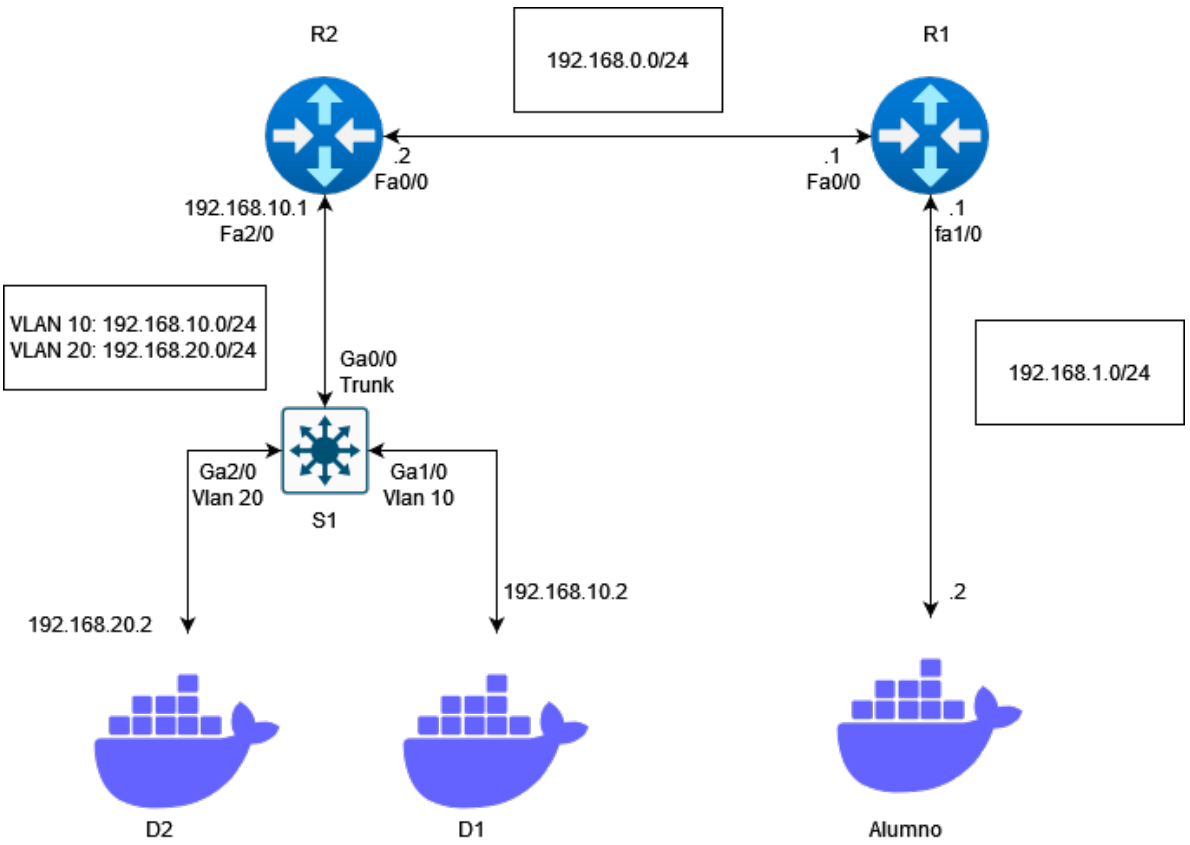


Figura B.1: Red de Ejemplo

Para poder configurar esta red B.1 tenemos que empezar dando un nombre al laboratorio y poniendo los datos del servidor `GNS3` maestro donde se va a virtualizar los nodos, en este nuestro seria:

Listing B.2: Requisitos laboratorio Para un laboratorio

```
"labName": "red_Prueba",
  "gns3": {
    "ip": "10.0.49.10",
    "port": "3080",
    "user": "admin",
    "pass": "admin"
  }
```

Ahora vamos a ver como crear la configuración de los nodos, por ejemplo la de R1. Lo primero de todo tenemos que escoger el template, el tipo de dispositivo que es y el sistema operativo, este caso seria: **template:** Cisco 7200, **machineType:** router y **OS:** cisco_ios.

Después tenemos que seleccionar las interfaces se van a configurar, según la imagen estas son:

- **Inteface:** Fa 0/0 - **IP:** 192.168.0.1 - **Netmask:** 255.255.255.0
- **Inteface:** Fa 1/0 - **IP:** 192.168.1.1 - **Netmask:** 255.255.255.0

Por ultimo, para los routers se tiene que escoger el encaminamiento que para este laboratorio se va a usar OSPF, que nos exige añadir dentro de las rutas la IP, Wildcard y Area, en este caso nos quedaria algo

- **Area:** 1 - **IP:** 192.168.0.1 - **Wildcard:** 0.0.0.255
- **Area:** 1 - **IP:** 192.168.1.1 - **Wildcard:** 0.0.0.255

Como resultado se deberá crear algo como vemos en la figura B.3

Listing B.3: Ejemplo configurar Router

```
 "RM-1": {
      "machineType": "router",
      "template": "Cisco 7200",
      "OS": "cisco_ios",
      "interfaces": [
        {
          "iface": "fa0/0",
          "ip": "192.168.0.1",
          "netmask": "255.255.255.0"
        },
        {
          "iface": "fa1/0",
```

```
          "ip": "192.168.1.1",
          "netmask": "255.255.255.0",
          "nat": "inside"
        }
      ],
      "pathingType": "ospf",
      "routes": [
        {
          "ip": "192.168.1.0",
          "wildcard": "0.0.0.255",
          "area": "1"
        },
        {
          "ip": "192.168.0.0",
          "wildcard": "0.0.0.255",
          "area": "1"
        }
      ]
    }
```

Ahora vamos configurar un switch, en nuestro diagrama B.1 solo se usa ”SE-1”. Para los switches también tenemos que escoger el template, el tipo de dispositivo que es y el sistema operativo, este caso seria: **template:** Cisco IOSvL2 esclavo, **machineType:** switch y **OS:** cisco_ios.

En este caso los switches únicamente se puede configurar las distintas vlans, este caso la lista de interfaces únicamente contiene un único valor.

- **Number:** Trunk - **Name:** N/A - **Lista Interfaces:** Gi 0/0
- **Number:** 10 - **Name:** 192.168.1.1 - **Lista Interfaces:** Gi 1/0
- **Number:** 20 - **Name:** 192.168.1.1 - **Lista Interfaces:** Gi 2/0

Como resultado se deberá crear algo como vemos en la figura B.4

Listing B.4: Ejemplo configurar Switch

```
"SE-1": {
      "machineType": "switch",
      "template": "Cisco IOSvL2 esclavo",
      "OS": "cisco_ios",
      "vlans": [
        {
          "number": "trunk",
          "interfaces": [
            "Gi 0/0"
```

```
          ]
        },
        {
          "number": "10",
          "name": "Vlan 10",
          "interfaces": [
            "Gi 1/0"
          ]
        },
        {
          "number": "20",
          "name": "Vlan 20",
          "interfaces": [
            "Gi 2/0"
          ]
        }
      ]
    }
```

Como ultimo tipo de nodo posible vamos a configurar los dockers que hacen de maquinas finales, como por ejemplo Alumno. Al igual que todos hay que empezar indicando los mismos apartados, pero para este caso seria **machineType:** docker, **template:** ubuntu_tfg.

También se tendrá que añadir los datos de la conexión a la red:
iface: eth0 - **IP:** 192.168.20.2 - **netmask:** 255.255.255.0 - **gateway:** 192.168.20.1

Quedando como resultado algo parecido a la figura B.5

Listing B.5: Ejemplo configurar Dokcer

```
"Alumno": {
     "machineType": "docker",
     "template": "ubuntu_tfg",
     "iface": "eth0",
     "ip": "192.168.1.2",
     "netmask": "255.255.255.0",
     "gateway": "192.168.1.1"
   }
```

Ahora vamos a indicar las conexiones entre los equipos, si nos fijamos entre R1 y R2 podemos apreciar:

- **Name**: R2 - **Interface:** FastEthernet0/0
- **Name**: R1 - **Interface:** f0/0

Podemos ver un ejemplo completo de un laboratorio completo en único archivo en el repositorio de gitHub en el archivo redPrueb.json

Para poder utilizar ese archivo para configurar la red de manera automática se tendría que hacer como en la imagen B.2, haciendo uso del fichero ConfigureNetwork.py del proyecto:

```
[usuario@tfg4910 CyberCrunch]$ python3 ConfigureNetwork.py redPrueba.json
Se empieza a configurar el laboratorio
Created: SE-1 -- Type: qemu -- Console: 5001
Created: Maestro -- Type: docker -- Console: 5026
Created: E2 -- Type: docker -- Console: 5005
Created: E1 -- Type: docker -- Console: 5003
Created: RM-1 -- Type: dynamips -- Console: 5025
Created: RE-2 -- Type: dynamips -- Console: 5000
```

Figura B.2: Ejecución del script ConfigureNetwork.py

Una vez tenemos una red se va puede obtener la información de un nodo del laboratorio o expórtalo para poder compartirlo, en este caso vamos a usar el fichero LabInfoUtils.py del proyecto. Si queremos obtener información de algún nodo del laboratorio,debemos proporcionar los datos para conectarnos al Servidor `GNS3` y el nombre del laboratorio junto con el nombre del nodo. Podemos ver un ejemplo en la figura B.3. Y si en cambio se quiere exportar el laboratorio en vez de proporcionar el nombre del nodo se dará el nombre del fichero con el que se quiere guardad el laboratorio escogido, como se puede observar en la imagen B.4 Para ambos casos los datos del servidor `GNS3` son un diccionario como el de los datos de entrada con el mismo formato que para crear. un ejemplo seria:

Listing B.6: Ejemplo conector con GNS3

```
{
    "ip": "10.0.49.10",
    "port": "3080",
    "user": "admin",
    "pass": "admin"
}
```

```
[usuario@tfg4910 CyberCrunch]$ python3 LabInfoUtils.py print redPrueba '{    "ip": "10.0.49.10",    "port": "3080",    "user": "admin",    "pass": "Gns32mil23."  }' RM-1
Name: RM-1
Type: router
Protocol: ospf 1
Status: started

Links:
Interface           Destination Name    Destination Interface
FastEthernet0/0     RE-2                FastEthernet0/0
FastEthernet1/0     Maestro             eth0
FastEthernet2/0     NAT                 nat0

Interfaces:
Interface           Ip
FastEthernet0/0     192.168.0.1
FastEthernet1/0     192.168.1.1
FastEthernet2/0     192.168.122.118
```

Figura B.3: Ejecución del script para imprimir información de un nodo

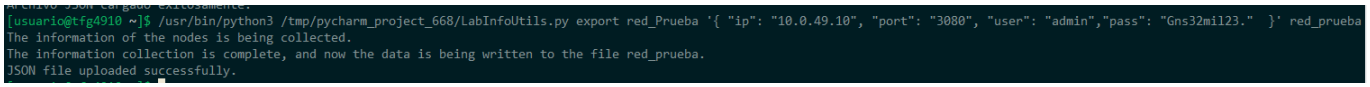


Figura B.4: Ejecución del script para exportar información de un nodo

Apéndice C

Manual del Desarrollador

En este ultimo manual vamos a explicar aspectos a tener en cuenta para prevenir futuros errores con el fin de que usuarios y otras personas que quieran participar en el desarrollo del proyecto puedan continuarlo. Podemos encontrar el trabajo realizado en el repositorio de GitHub publico llamado CyberCrunch [18].

Me gustaría empezar recomendando el desarrollo y uso de estos scripts en equipos Linux, para asegurar su correcto funcionamiento, aunque si se cumplen con los requisitos mínimos este debería funcionar en cualquier otro equipo con otro sistema operativo aunque esto puede causar errores como el uso de librerías diferentes que deberían ser sustituidas por las adecuadas al sistema.

El código que nos descarguemos de GitHub [18] esta compuesto los programas ejecutables LabInfoUtils.py y ConfigureNetwork.py que son con los que interactúa el usuario para exportar, crear la red o recopilar la información de un nodo, para esto encontramos un archivo dedicado a imprimir por pantalla de un nodo llamado PrettyPrint.py, y para recopilar la información de los nodos desplegados se usa InformationDevice.py. Para la opción de crear un nuevo laboratorio, nodo y configurarlo se usan los archivos encontrados en NetworkDeployment, creando un nuevo archivo para cada tipo de dispositivo posible.

Una vez entendido cómo esta estructurado el código si se quiere modificar algo se recomienda seguir esta organización. Por ejemplo, si se finalizase el modulo de NetworkDeployment/ConfigureMachine.py que se encarga de configurar maquinas virtuales de VirtualBox se recomienda seguir ampliando las funcionalidades dentro de este archivo, y para recopilar la información únicamente modificar la forma de conectarse con el nodo en el archivo InformationDevice.py. O si se quiere añadir nuevos sistemas operativos a configurar en routers y switches, se puede seguir usando Netmiko [17] para conectarnos y si utilizan unos pasos similares a cisco continuar con las funciones existentes modificando los comando en función del sistema.

Por último, toda persona que quiera modificar, añadir nuevos casos de uso o simplemente quiera mejorarlo siéntase libre. Durante todo el desarrollo se ha tratado de diseñar las funciones mas simples posible para facilitar la comprensión tratando de ayudar a próximos usuarios.

Bibliografía

[1] GFOSS - Open Technologies Alliance. *phpmyadmin_honeypot*. https://github.com/gfoss/phpmyadmin_honeypot. Consultado el 20 de noviembre de 2022. 2022.

[2] Amazon Web Services. *Virtualización*. https://aws.amazon.com/es/what-is/virtualization/. Consultado el 10 de mayo de 2023. 2023.

[3] David Banville. *gns3fy: A Python API Library for GNS3*. https://davidban77.github.io/gns3fy/. Consultado el 20 de marzo de 2023. 2023.

[4] CESUR Formación. *Ciberseguridad y Formación*. https://www.cesurformacion.com/blog/cibeseguiridad-formacion. Consultado el 10 de mayo de 2023. 2023.

[5] *Cisco*. https://www.cisco.com/c/es_es/index.html. Consultado el 20 de febrero de 2023. 2028.

[6] Cowrie Community. *Cowrie*. https://github.com/cowrie/cowrie. Consultado el 20 de noviembre de 2022. 2022.

[7] *Cuckoo Sandbox*. https://cuckoosandbox.org/. Consultado el 20 de noviembre de 2022. 2022.

[8] Cymmetria. *StrutsHoneypot*. https://github.com/Cymmetria/StrutsHoneypot. Consultado el 20 de noviembre de 2022. 2022.

[9] *Debunking the Myth*. https://www2.fireeye.com/debunking-the-myth.html. Consultado el 20 de noviembre de 2022. 2022.

[10] *Docker*. https://www.docker.com/. Consultado en 10 de febrero de 2023. 2023.

[11] *FortiSandbox*. https://www.fortinet.com/lat/products/sandbox/fortisandbox. Consultado el 20 de noviembre de 2022. 2022.

[12] *GitHub*. https://github.com/. Consultado el 30 de enero de 2023. 2023.

[13] *GNS3*. https://www.gns3.com/. Consultado el 15 de marzo de 2023. 2023.

[14] GNS3. *ubridge*. https://github.com/GNS3/ubridge. Consultado el 30 de mayo de 2023. 2023.

[15] *Hack The Box*. https://www.hackthebox.com/. Consultado a 15 de noviembre de 2022. 2022.

[16] *ITSecGames*. http://www.itsecgames.com/. Consultado el 15 de noviembre de 2022. 2022.

[17] ktbyers. *netmiko*. https://github.com/ktbyers/netmiko. Consultado el 30 de mayo de 2023. 2023.

[18] marcocacho. *CyberCrunch*. https://github.com/marcocacho/CyberCrunch. Consultado el 22 de junio de 2023. 2023.

[19] *Microsoft Teams*. https://www.microsoft.com/es-es/microsoft-teams/log-in. Consultado el 20 de enero de 2023. 2023.

[20] *OpenAI*. https://openai.com/. Consultado 15 de marzo de 2023. 2023.

[21] *OpenSearch*. https://opensearch.org/. Consultado 5 de mayo de 2023. 2023.

[22] *Overleaf*. https://www.overleaf.com/. Consultado el 20 de enero de 2023. 2023.

[23] *OWASP Mutillidae II*. https://owasp.org/www-project-mutillidae-ii/. Consultado el 15 de noviembre de 2022. 2022.

[24] *PC Componentes*. https://www.pccomponentes.com/. Consultado el 30 de marzo de 2023. 2023.

[25] *PyCharm*. https://www.jetbrains.com/es-es/pycharm/. Consultado el 30 de enero de 2023. 2023.

[26] *Red Hat*. https://www.redhat.com/en/technologies/virtualization/enterprise-virtualization. Consultado el 17 de noviembre de 2022. 2022.

[27] *Root-me*. https://www.root-me.org/?lang=es. Consultado el 15 de noviembre de 2022. 2022.

[28] Telekom Security. *tpotce*. https://github.com/telekom-security/tpotce. Consultado el 20 de noviembre de 2022. 2022.

[29] *Sublime Text*. https://www.sublimetext.com/. Consultado el 31 de enero de 2023. 2023.

[30] *syslog-ng*. https://www.syslog-ng.com/. Consultado 7 de mayo de 2023. 2023.

[31] *Telegram*. https://web.telegram.org/. Consultado el 25 de febrero de 2023. 2023.

[32] *Terraform*. https://www.terraform.io/. Consultado el 17 de noviembre de 2022. 2022.

[33] *Trello*. https://trello.com/es. Consultado el 30 de enero de 2023. 2023.

[34] *TryHackMe*. https://tryhackme.com/. Consultado el 15 de noviembre de 2022. 2022.

[35] *VirtualBox*. https://www.virtualbox.org/. Consultado el 17 de noviembre de 2022. 2022.

[36] *Visual Paradigm*. https://www.visual-paradigm.com/. Consultado 10 de abril de 2023. 2023.

[37] *VMware*. https://www.vmware.com/es.html. Consultado el 17 de noviembre de 2022. 2022.

[38] *What is Cyber Security?* https://kaspersky.com/resource-center/definitions/what-is-cyber-security. Consultado el 10 de mayo de 2023. 2023.

[39] Wikipedia. *Caja negra (sistemas)*. https://es.wikipedia.org/wiki/Caja_negra_(sistemas). Consultado el 30 de mayo de 2023. 2023.

[40] Wikipedia. *Entorno de pruebas (informática)*. https://es.wikipedia.org/wiki/Entorno_de_pruebas_(inform%C3%A1tica). Consultado el 15 de mayo de 2023. 2023.

[41] Wikipedia. *Honeypot*. https://es.wikipedia.org/wiki/Honeypot. Consultado el 15 de mayo de 2023. 2023.

[42] Wikipedia. *Kanban*. https://es.wikipedia.org/wiki/Kanban. Consultado el 30 de enero de 2023. 2023.

[43] Wikipedia. *Prueba de integración*. https://es.wikipedia.org/wiki/Prueba_de_integraci%C3%B3n. Consultado el 30 de mayo de 2023. 2023.

[44] Wikipedia. *Prueba unitaria*. https://es.wikipedia.org/wiki/Prueba_unitaria. Consultado el 30 de mayo de 2023. 2023.

[45] Wikipedia. *System testing*. https://en.wikipedia.org/wiki/System_testing. Consultado el 30 de mayo de 2023. 2023.

[46] *WildFire*. https://www.paloaltonetworks.com/network-security/wildfire. Consultado el 20 de noviembre de 2022. 2022.