ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Integración de la Inteligencia Artificial en la Gestión de Redes: Aplicación Práctica y Potenciales Mejoras en la Tarea de Gestión

Integration of Artificial Intelligence into Network Management: Practical Application and Potential Improvements in the Management Task

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Alejandro Miguel Franquelo

Noviembre - 2024

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Alejandro Miguel Franquelo

Director del TFG: José Ángel Irastorza Teja

Título: 'Integración de la Inteligencia Artificial en la Gestión de Redes: Aplicación Práctica y Potenciales Mejoras en la Tarea de Gestión'

Title: 'Integration of Artificial Intelligence into Network Management: Practical Application and Potential Improvements in the Management Task'

Presentado a examen el día: 14 de Noviembre de 2024

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): JESUS MARIA IBAÑEZ DIAZ
Secretario (Apellidos, Nombre): ALBERTO ELOY GARCÍA GUTIÉRREZ
Vocal (Apellidos, Nombre): AMPARO HERRERA GUARDADO
Este Tribunal ha resuelto otorgar la calificación de: .....
Fdo.: El Presidente Fdo.: El Secretario

Fdo.: El Vocal Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector Trabajo Fin de Grado N.º
(a asignar por Secretaría)

Agradecimientos.

Este Trabajo de Fin de Grado representa el cierre de una etapa, y quería agradecer a todas las personas que han estado presentes durante este proceso.

En primer lugar, a José Ángel, mi tutor del TFG, gracias por confiar en mí y proporcionarme ayuda y consejos para poder llegar a lograr esta meta en mi vida.

A mi familia, que siempre ha estado presente dándome ánimos durante todo este proceso. Y también por brindarme la oportunidad y las herramientas para poder llegar a conseguir este ansiado título.

También quiero mencionar a mis compañeros, con los que durante toda esta etapa he compartido muchas horas de clase. Además, hicieron que esta experiencia con su apoyo y compañerismo fuera mucho más llevadera.

Y en especial, quiero dar las gracias a mi novia, que ha estado junto a mi durante estos 4 años dándome ánimo y fuerza en los momentos más difíciles. Gracias por apoyarme y estar a mi lado en cada paso de este viaje y por haber sido mi fuente de inspiración cuando lo necesitaba.

Resumen.

En el marco actual, los usuarios demandan constantemente que las capacidades de la red estén en su máximo nivel de eficiencia. Esta exigencia conlleva una evolución constante en las tecnologías utilizadas en la gestión de redes. Dados los avances actuales en inteligencia artificial y su aplicación en diversas áreas, se considera relevante su potencial aplicación e integración en la gestión de redes, más aún cuando históricamente, tareas como la gestión de fallos han hecho uso de herramientas de IA, como los sistemas expertos.

Así, en este trabajo se estudia la integración de la inteligencia artificial en la arquitectura de las plataformas de gestión de redes. Para lograr este objetivo, se ha llevado a cabo un análisis exhaustivo con el fin de seleccionar la plataforma de gestión más idónea, siendo de software libre o comercial, que se integre de manera óptima con las herramientas de IA. El propósito primordial de esta integración es mejorar la operatividad y facilitar la toma de decisiones del gestor humano en la gestión continua de las redes de comunicaciones.

Para ejemplificar este enfoque, se ha estudiado el uso de TensorFlow, una herramienta líder en inteligencia artificial, y Zabbix, una plataforma de gestión de redes de comunicaciones versátil y actualizada. Ambas tecnologías se han integrado en un demostrador como prueba de concepto.

Abstract.

In the current context, users constantly demand that network capabilities operate at their highest level of efficiency. This demand leads to a continuous evolution of the technologies used in network management. Given the current advances in artificial intelligence and its application in various fields, its potential application and integration into network management is considered relevant, especially since tasks such as fault management have historically utilized AI tools, such as expert systems.

Thus, this work studies the integration of artificial intelligence into the architecture of network management platforms. To achieve this goal, a thorough analysis has been conducted to select the most suitable management platform, whether open-source or commercial, that integrates optimally with AI tools. The primary purpose of this integration is to improve operational efficiency and facilitate decision-making for human managers in the continuous management of communication networks.

To illustrate this approach, the use of TensorFlow, a leading artificial intelligence tool, and Zabbix, a versatile and up-to-date network management platform, has been studied. Both technologies have been integrated into a demonstrator as a proof of concept.

Contenido.

Indice de figuras.

Lista de abreviaturas.

API - Application Programming Interface

DNN - Deep Neural Network

GPU - Graphics Processing Unit

IA - Inteligencia Artificial

ICMP - Internet Control Message Protocol

IoT - Internet of Things

LSTM - Long Short-Term Memory

PU - Processing Unit

RAM - Random Access Memory

TFG - Trabajo de Fin de Grado

1. Introducción.

Hoy en día, las comunicaciones son un aspecto esencial en el día a día. Esto hace que el buen funcionamiento de las redes de comunicaciones sea un aspecto crucial, para ello existen herramientas de gestión de red, que pueden utilizase en cualquier tipo de situación, desde una pequeña empresa hasta las grandes operadoras de telecomunicaciones.

En este TFG se propone la integración de GNS3, Zabbix y TensorFlow, a través de la cual se presenta una solución al cumplimiento de las exigencias que presentan los usuarios a las entidades que les proveen el servicio. Esta integración tiene como objetivo evitar fallos a través de la predicción del comportamiento de la red, con el fin de anticiparse ante estos.

GNS3 es una herramienta de simulación de redes, y en esta integración desempeña esta función. A través de esta herramienta se obtienen datos de un entorno controlado. Estos parámetros son recogidos por Zabbix, que es una plataforma de gestión de red, la cual se encarga de recopilar y almacenar los datos. TensorFlow se encarga de detectar los patrones generados por estos datos, y se genera con él un modelo de IA que permite realizar las predicciones.

A través de la unión de estas herramientas, en este TFG se presenta el gran abanico de posibilidades que ofrece la IA como herramienta para proponer soluciones. Esta integración que se presenta en este TFG pretende representar una muestra del potencial que tiene la IA.

1.1. CONTEXTO Y MOTIVACIÓN.

Las plataformas de gestión de redes tienen como función la supervisión del estado de una red, esto juega un papel muy importante ya que reflejan el estado a tiempo real de estas. Esta herramienta funciona recopilando los datos de la red, analizándolos y en caso de que se detecte alguna anomalía, se generan alertas con el fin de solucionar este problema. En la gestión de red clásica, esta herramienta se encarga en su totalidad de la obtención y almacenamiento de datos y alertas ocurridas, estas últimas, se configuran directamente por el gestor de red en base a sus conocimientos y experiencia. Sin embargo, en el contexto en el que actual, la evolución constante de

las redes de comunicaciones es un hecho que no se puede refutar. Esto causa una serie de nuevos desafíos en el entorno que se deben de afrontar.

La complejidad de la gestión de las redes, la incorporación de nuevos dispositivos más complejos, nuevos protocolos y tecnología más avanzada, está causando un aumento en la complejidad del manejo de estas herramientas.

Las amenazas de seguridad son un aspecto importante, ya que la complejidad en las redes dificulta la detección de posibles intrusos en la red.

La escalabilidad de las redes, el aumento del tamaño y de su complejidad, está haciendo que cueste adaptar las herramientas tradicionales de gestión a la nueva situación, ya que resulta más difícil de asegurar el rendimiento por la cantidad de parámetros a gestionar.

La evolución de la tecnología dificulta cumplir con los nuevos estándares, y adaptarse a la incorporación de nuevos elementos tecnológicos puede tener cierta complejidad y requerir de recursos tecnológicos de los que no se dispone en ese momento.

Los errores humanos son un problema debido al aumento de la complejidad, ya que lo que anteriormente podía suponer un pequeño error, hoy en día puede suponer un problema notorio en la red. [1]

Ante estos desafíos, las herramientas han ido avanzando, y la IA se presenta como una herramienta que puede complementar a otras, facilitando muchas tareas, ya que la IA ayuda a solucionar diferentes problemas de maneras muy variadas.

La IA es capaz de automatizar procesos repetitivos que aprende, de este modo, y así libera a las personas de tareas repetitivas para que se concentren en otras actividades.

La inteligencia artificial, al no ser un ser vivo, no necesita descanso, esto le permite realizar tareas en cualquier momento, solucionando y atendiendo problemas que puedan surgir en momentos de ausencia de un operario. Siendo una herramienta muy precisa en muchas situaciones.

Además, maneja grandes volúmenes de información, de los cuales se nutre para formar su conocimiento y operar de manera eficiente y precisa, ya que con mayor número de datos se cometen menos errores.[2]

En la actualidad, la constante evolución plantea diferentes situaciones que requieren ser afrontadas. En el ámbito de la gestión de redes, la integración de la IA a las herramientas de gestión de red presenta una solución robusta frente a la variabilidad y necesidad de aprendizaje constante y rápido, creando una herramienta capacitada para las nuevas necesidades que surgen en nuestro entorno.

1.2. OBJETIVOS DEL ESTUDIO.

En este trabajo de fin de grado se expone la integración de la inteligencia artificial en la gestión de redes, con el fin de mostrar el potencial que tiene la IA en este ámbito. La capacidad de esta combinación de herramientas se debe al contexto en el que se encuentra. Esta integración presenta una visión prometedora, ya que permite cumplir los diferentes objetivos propuestos por cada empresa y, a su vez, reduce los desafíos que puedan surgir.

A lo largo de este documento, se evalúan las diferentes características de las herramientas de gestión de red, en las cuales se identifican las posibilidades y la facilidad de proporcionar una integración con una inteligencia artificial. Algunos ejemplos de las plataformas de gestión de red que se tratan más adelante son Zabbix, Nagios o LibreNMS. En base a las facilidades de integración que ofrecen las plataformas de gestión, se elige una herramienta de desarrollo de IA, con el fin de lograr la mayor compatibilidad posible. Para hacer posible el demostrador, se necesita una plataforma para la simulación de redes, a la que accede la herramienta de gestión de red con el fin de recopilar los datos.

Tras la elección de las plataformas y herramientas se diseña un demostrador, donde se explica la configuración e instalación de los elementos que lo componen. En este caso, se toma como ejemplo de modelo de inteligencia artificial dedicado a la predicción de valores, con el fin de mostrar las prestaciones que tiene este modelo. Para ello, se muestran los datos obtenidos para corroborar lo mencionado anteriormente.

1.3. METODOLOGÍA.

La metodología que se utilizara para este trabajo de fin de grado se detalla a continuación. Para la integración de GNS3, Zabbix y TensorFlow el proceso se divide en diferentes fases de desarrollo.

La primera es el análisis de manera teórica de las herramientas que se tiene a nuestra disposición, como se menciona en capítulos anteriores, se tienen en cuenta sus prestaciones y, en base a ello, se seleccionarán las indicadas para el desarrollo de esta integración.

Tras haber elegido la plataforma de simulación de redes, se procede con su instalación y configuración. Con el entorno preparado, se continúa creando la topología de red que cumpla con los requisitos necesarios del demostrador. Esta topología incluye diferentes elementos de red, como routers o switches. Tras haber creado la topología, se configuran los elementos para que sea funcional y simule con precisión una red física compuesta por equipos reales.

Con la red preparada, se procede con la instalación y configuración de la herramienta de gestión de red. Se instalan los agentes en los dispositivos que se desean gestionar para así poder recopilar los datos sobre el funcionamiento de estos equipos, con el fin de mas tarde puedan ser utilizarlos. En caso de que sea necesario se configuran más parámetros que serían de bastante relevancia en la gestión de red clásica, pero para la integración propuesta no tiene una relevancia significativa.

Para completar la integración, se desarrollan los scripts que permitan la obtención y tratamiento de los datos, con el fin de poder crear un modelo de inteligencia artificial que genere las predicciones de los valores. Cada uno de estos scripts tiene una funcionalidad, encargándose de las diferentes partes , como la obtención de los datos del dispositivo, la creación del modelo de IA y la utilización de este en las predicciones.

Además, cuando esté completado, se añaden unas alertas a través de correo electrónico, que cuando las predicciones superen un umbral, se emiten avisos con el fin de anticipar los posibles fallos.

Con el demostrador preparado, se somete a este a pruebas con el fin de asegurar el buen funcionamiento. Se evalúan los resultados obtenidos, y, ajustando las configuraciones y parámetros, y repitiendo las pruebas, se logra la validación del demostrador.

1.4. ESTRUCTURA DEL DOCUMENTO.

Este documento se divide en varios capítulos, organizados con el fin de facilitar la comprensión de la integración.

Capítulo 1: Introducción. En él se presenta una muestra del tema que se trata en este TFG, incluyendo un contexto histórico y las motivaciones que llevan a realizar este trabajo. Además, se indica cómo se procede con el desarrollo del trabajo.

Capítulo 2: Fundamentos teóricos. En este capítulo se abordan todos los aspectos teóricos, desde la simulación de redes hasta las plataformas que se utilizan para ello. También se analiza la gestión de red y una parte de las herramientas que existen en este ámbito. Además, se desarrollan nociones sobre inteligencia artificial y las plataformas de desarrollo donde se generan estos modelos. Por último, se realiza una comparativa determinando cuáles son las herramientas óptimas para realizar esta integración.

Capítulo 3: Estado del arte. En este capítulo se describen las tecnologías existentes y el estado actual de estas en el ámbito tecnológico, donde se mencionan sus hallazgos y contribuciones.

Capítulo 4: Implementación de la integración. Aquí se muestra la preparación del entorno y la configuración de la integración, presentando los diferentes pasos y medidas que se llevan a cabo con el fin de lograr el éxito en la integración.

Capítulo 5: Resultados y análisis. En este capítulo se realiza un análisis de los resultados y se determinan los beneficios de la integración y las posibles mejoras que se incluyen.

Capítulo 6: Conclusiones. En este capítulo se presentan las conclusiones a las que se llega el estudio y las líneas de investigación que ofrece esta integración tan completa.

Anexos: Al final de este documento, se incluyen varios anexos donde se detalla el código utilizado en el diseño del modelo de inteligencia artificial que se desarrolla en este proyecto.

2. Estado del arte.

La integración de la inteligencia artificial en el ámbito de la gestión de redes se encuentra actualmente en el objetivo de investigación de diferentes empresas debido a su potencial. Esta integración pretende mejorar la eficiencia, la seguridad y reducir el tiempo de respuesta a los posibles fallos que puedan surgir en la red. En este capítulo se van a revisar algunas investigaciones realizadas en este campo.

La investigación del desarrollo de las redes 6G es un tema actual, en el que se presenta la idea de incluir la inteligencia artificial y la ciencia de datos. Las generaciones predecesoras en ciertas situaciones no poseían la suficiente capacidad de adaptación para absorber picos de tráfico. Debido a esto, en [3] se propone la idea de la integración de la inteligencia artificial en diferentes capas que son la detección inteligente, el análisis de datos, el control y la aplicación inteligentes. A través de ellas se pretende lograr detectar patrones y realizar una gestión de los recursos. Además, también se investiga sobre la implementación de la IA en 'Edge Computing' para el desarrollo del hardware y el ahorro de la energía.

La detección de anomalías en la red mediante técnicas de aprendizaje automático es un tema sobre el que la Universidade da Coruña ha decidido investigar. Han realizado pruebas con diferentes modelos con diferentes tipos de aprendizaje que tratan de detectar intrusiones con la IA, ya que cuenta con gran facilidad de manejo de grandes cantidades de datos y capacidad para el análisis de ellos. En su estudio, utilizaron técnicas como Random Forest, Naive Bayes y Redes Neuronales Profundas (DNN). [4]

También se han realizado estudios sobre diferentes algoritmos en situaciones particulares, como puede ser un algoritmo bioinspirado asistido por IA para redes de comunicación seguras de IoT [5]. En este artículo se muestra el desarrollo con el que se ha logrado la implementación del algoritmo bioinspirado en un modelo de IA con el fin de aumentar la seguridad en la red, ya que este tipo de problemas está en un creciente aumento debido a la proliferación de los ataques utilizados y el gran volumen de datos que se manejan en la actualidad. En esta investigación, se muestra cómo se incluye esta tecnología en los procesos de autenticación y autorización en la red de comunicaciones IoT.

La RMIT University (Royal Melbourne Institute of Technology University), debido a la gran evolución de la red en la actualidad y la gran cantidad de dispositivos conectados, hace que los enfoques tradicionales no sean suficiente para mitigar los ataques que se realizan. Por ello, plantea el uso del aprendizaje automático con el fin de poder detectar cualquier tipo de intrusión en la red. Además, se emplea la inteligencia artificial con el fin de detectar varios tipos de anomalías en la red, aunque principalmente se centra en el ámbito de la seguridad. [6]

La latencia es un requisito critico en la gestión de red para los servicios a partir del 5G. La latencia ha sido un aspecto analizado en las redes de comunicaciones debido a su importancia. En la situación actual, debido a las investigaciones que se realizan sobre las comunicaciones de retrodispersión ambiental de manera inalámbrica, ha surgido la necesidad de cumplir los requerimientos de latencia y aumentar la tasa de datos. La herramienta de aprendizaje profundo se utiliza con el fin de mejorar el rendimiento y que se cumplan con las restricciones anteriormente mencionadas. [7]

En agosto de este año, se ha publicado un artículo sobre el uso de la inteligencia artificial en la clasificación de tráfico en redes definidas por software [8]. En él se muestra que los enfoques convencionales, como la identificación del tráfico basada en los puertos, se vuelven soluciones ineficaces debido al uso y las medidas de seguridad que se toman en la red. En este estudio, se han aplicado técnicas de IA en redes definidas por software, en él se ha demostrado que es una manera eficiente de identificar y clasificar tráfico. Sin embargo, también se ha demostrado en este estudio que la inteligencia artificial posee varias limitaciones en la detección de tráfico cifrado, la inspección de carga útil.

Investigadores del centro tecnológico de Telecomunicaciones de Cataluña junto con otras entidades, han realizado un estudio sobre cómo utilizar la inteligencia artificial para crear una solución escalable para gestionar y orquestar una gran cantidad de “network slices”. El enfoque que se propone en este estudio implica dividir los planos de gestión y orquestación con el fin de escalar, optimizar y mejorar la infraestructura de la red a través del uso de múltiples bucles de control distribuidos y basados en IA, que pueden operar a nivel de nodo, a nivel de slice, entre slices o a nivel del dominio de orquestación. [9]

En las redes de próxima generación, la agilidad, la automatización y la inteligencia de la red están en el foco de investigación, ya que se busca proporcionar la gestión sin intervención humana y redes que se optimicen por sí mismas utilizando la inteligencia artificial. Además, también se aborda sobre la integración de esta herramienta en diferentes puntos de la red, como puede ser en la nube o en el borde [10]. En este estudio, además, se realizan experimentos basados en un nuevo diseño de un marco de inteligencia en el borde que orquesta y despliega microservicios de IA. Se usa un modelo que predice el flujo de red de borde en tiempo casi real. Los resultados obtenidos muestran que el modelo de pronóstico implementado predice con precisión el rendimiento y la latencia del flujo de red.

La radio definida por software (SDR) es una tecnología que se utiliza para desarrollar e implementar protocolos de comunicación inalámbrica, donde funcionalidades tradicionales de hardware se implementan mediante software. En [11] se aborda su integración con la inteligencia artificial. La utilidad que se le da a esta herramienta es conseguir la autogestión de la red. Para ello, se determinan parámetros básicos que se deben gestionarse, que son la frecuencia del canal, el ancho de banda y el tipo de modulación. Por una parte, la radio definida por software se encarga de recoger y analizar los componentes de la señal, y la IA detecta la situación en la que se encuentra el sistema cuando hay datos no deseados. Con esta configuración, se consigue que el índice de conectividad y el área de cobertura se optimicen casi al doble, lo que es un aumento muy notorio.

En la plataforma de gestión Zabbix, se encuentran diferentes herramientas que presentan iniciativas que apuntan hacia la integración de la inteligencia artificial en estas plataformas. En la página web de Zabbix [12], se indica que en la versión 6.0 se incluyen técnicas de seguimientos de tendencias para la detección de anomalías. Se utilizan métodos como promedio móvil ponderado, desviación estándar y desviación absoluta media, que permiten la detección automática de comportamientos anómalos en la red a través de estas herramientas implementadas en esta versión.

En resumen, la integración de la IA en diferentes aspectos de la gestión de redes es un tema innovador que suscita un gran interés en la comunidad científica actual. Esta integración representa una estrategia funcional para mejorar la eficiencia, la seguridad y la experiencia del usuario.

3. Fundamentos teóricos.

En el marco actual, la gestión de red es un elemento esencial para asegurar el buen funcionamiento de las redes de comunicaciones. En este proyecto se desarrolla un demostrador que incorpora un simulador, que recoge la topología de la red, la cual será gestionada por una plataforma de gestión estándar a la que se le añade una capa de inteligencia que asista al gestor humano a realizar las tareas cotidianas de supervisión y control de la red. En este capítulo se analizan diversas opciones de software para la implementación de la plataforma de gestión, con el objetivo de seleccionar la más adecuada para el proyecto. La inteligencia artificial también forma una parte importante, y se estudian las opciones de las plataformas de diseño de modelos de IA con el fin de predecir los valores. Estos valores se comparan con un umbral predeterminado, y en caso de que exista una diferencia mayor a la esperada, se envía una alerta a través de correo electrónico al gestor de la red.

3.1. SIMULACION DE REDES.

La simulación de red hace uso de una herramienta de software que permite crear redes y simular su comportamiento como si fueran reales. Este entorno virtual resulta muy útil en casos de estudio, ya que facilita el diseño y la modificación de redes de manera cómoda, sin incurrir en grandes gastos económicos. La reproducción de una red en un entorno virtual antes de implementarla en un entorno real ayuda a evitar errores y, en algunos casos, previene que estos lleguen a ocurrir. [13]

La simulación de red presenta numerosas ventajas. En el ámbito académico, posibilita cometer errores sin generar problemas graves y permite realizar pruebas e investigaciones gracias a la facilidad con la que se ejecuta en distintos entornos. [14]

Las prácticas en simulación de redes facilitan el aprendizaje de nuevas tecnologías, además de ofrecer la oportunidad de probar nuevas ideas, como se plantea en este trabajo de fin de grado. En este caso, se valora la facilidad de acceso a las configuraciones de los diferentes dispositivos y la visión completa y compacta que ofrece de la red.

En base a todo lo mencionado, los simuladores de redes se presentan como unas herramientas muy completas y útiles en diversas situaciones, tanto de prueba como

de aprendizaje, que permiten simular desde routers o switches hasta elementos de red mucho más complejos.

3.1.1. GNS3 (Graphical Network Simulator 3).

GNS3 es una herramienta de simulación de redes de código abierto que permite a los usuarios, diseñar, probar y simular diferentes tipos de redes, con el objetivo de aprender o solucionar problemas. GNS3 permite la simulación de muchos dispositivos, incluyendo los de fabricantes reconocidos en el ámbito de las redes, gracias a su interfaz intuitiva, como se ve en la Figura 1.

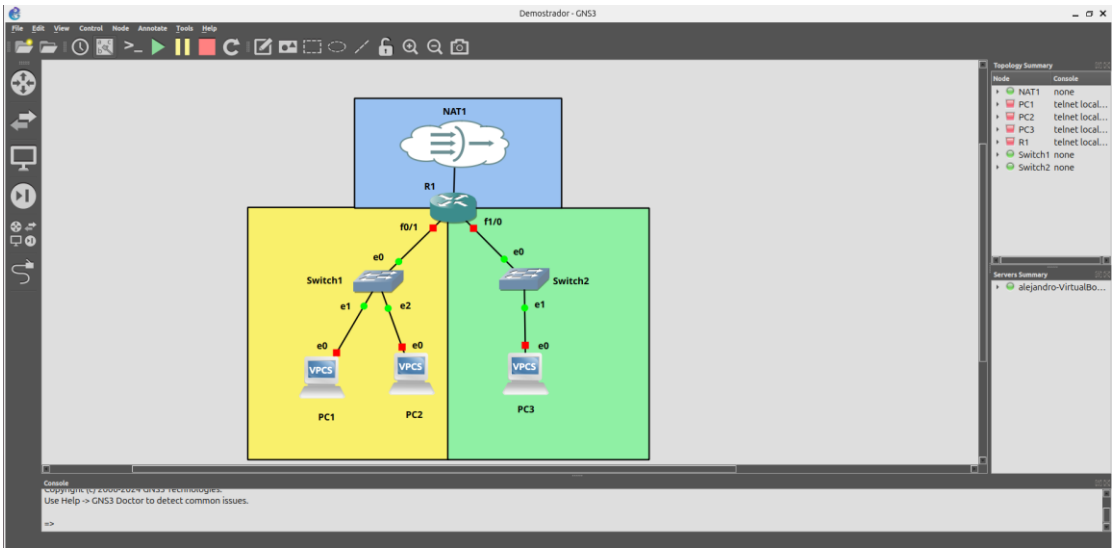

Figura 1: Entorno GNS3, Fuente: propia.

GNS3 es una herramienta reconocida en el ámbito de la formación y certificación, y que además es utilizada en diversas asignaturas del Grado de Tecnologías de Ingeniería de Telecomunicación, mención de Telemática, de la Universidad de Cantabria. Esta plataforma es compatible con una gran cantidad de gestores de red, por ello, y junto a su flexibilidad y capacidades hacen que sea una buena candidata para su integración en el demostrador final.

Entre sus diversas características merece destacar su gran comunidad activa y en constante crecimiento, que facilita a las personas que se están iniciando solucionen sus problemas, o puedan simular laboratorios de red ya configurados, esto permite que el aprendizaje en esta plataforma sea más intuitivo que en otras. Además, GNS3

es compatible con una gran variedad de dispositivos de diferentes fabricantes, aunque principalmente se utilicen los dispositivos del fabricante Cisco por su gran penetración en el mercado y la facilidad de acceder a los sistemas de los equipos a simular. También se pueden simular dispositivos de diferentes proveedores y cada vez más común dispositivos de licencia abierta proporcionados por algunos fabricantes . Ahora también se puede probar la utilización de varias tecnologías virtualizadas de red como puede ser máquinas virtuales o contenedores Docker.

La única limitación que posee esta herramienta es el hardware donde se aloja, ya que no posee un límite de dispositivos a simular, así que solo depende de la CPU y de la memoria del host donde se simule. También posee soporte nativo para Linux sin tener la necesidad de un software de virtualización adicional. GNS3 permite dispositivos emulados y simulados, los emulados son los cuales imitan el hardware de un dispositivo, y los simulados son los que simulan las características y la funcionalidad de un dispositivo. En rasgos generales la emulación al ser una representación es menos compleja y se utiliza en ámbitos de aprendizaje y pruebas básicas, y la simulación al usar software real, la reproducción es más exacta, pero posee mayor complejidad y requiere más recursos, esto hace que se utilice en pruebas más avanzadas y en configuraciones reales. [15]

3.1.2 Cisco Packet Tracer

Cisco Packet Tracer al igual que GNS3 es una herramienta de simulación de redes desarrollada por Cisco, está enfocada para el ámbito de la educación, concretamente en su programa Cisco Networking Academy. Se permite crear, simular y visualizar diferentes tipos de redes con diferentes dispositivos, desde routers hasta switches, donde se proporciona una experiencia cercana a la realidad, ya que posee varias limitaciones respecto al hardware físico. [16]

Entre las características más destacables de Cisco Packet Tracer se encuentra la capacidad de creación de topologías de red mediante una interfaz de arrastrar y soltar, que facilita el uso de la herramienta de manera significativa como se observa en la Figura 2. También se encuentra de manera simulada una interfaz de línea de comandos, que ayuda a familiarizarse con los entornos reales de trabajo, facilitando el aprendizaje de los comandos básicos y la configuración de equipos Cisco. El software

soporta diferentes protocolos de red como pueden ser RIP, OSPF, EIGRP, BGP, entre otros, esto hace que sea una herramienta bastante asemejada a la realidad. [17]

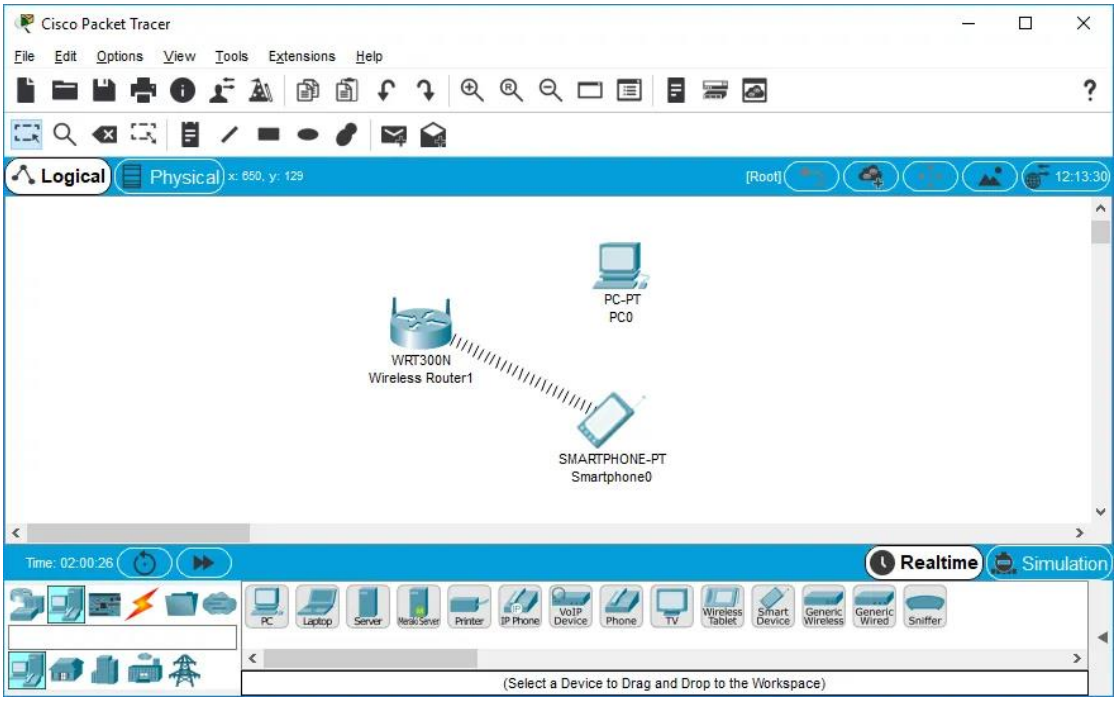

Figura 2: Entorno Cisco Packet Tracer, Fuente: https://www.telectronika.com/tutoriales/packet-tracer-interfaz-usuario/

Cisco Packet Tracer posee una versión gratuita para estudiantes de Cisco Networking Academy, pero para usuarios que no forman parte de este programa posee limitaciones en varios aspectos. Esta herramienta permite guardar diferentes configuraciones, aunque posee algunas limitaciones en cuanto a la simulación de características avanzadas y no está diseñada para la implementación de redes en entornos de producción. Para estas tareas Cisco recomienda herramientas más avanzadas como son GNS3 o EVE-NG que realizan simulaciones más precisas y detalladas. [18]

3.2. GESTION DE RED.

La gestión de red es el proceso de controlar y mantener la red de manera operativa, asegurando su rendimiento y seguridad. Las plataformas encargadas de esta tarea supervisan de forma continua el estado de la red, recopilando datos sobre los diferentes parámetros de esta. Estas plataformas, además de asegurar los

requerimientos necesarios para el funcionamiento de la red, también se pueden comprobar los requerimientos que ponga la empresa u organización.

La gestión de red tiene varios beneficios como son la detección inmediata de problemas y la optimización del rendimiento de la red, ya que se consigue tener una imagen completa de la red a través de estas plataformas. Esto hace que el cumplimiento de normas sea más sencillo de aplicar gracias al control de la infraestructura. [19]

La gestión de red se puede ver complementada con diferentes herramientas, como puede ser la inteligencia artificial, ya que puede simplificar esta tarea y aumentar su rendimiento. Al manejar grandes volúmenes de datos, la inteligencia artificial complementa la gestión de red en varios casos, como pueden ser los cuellos de botella, donde la IA analiza el tráfico y lo puede llegar a redirigir con el fin de prevenir la congestión de la red. También permite la mejora de la flexibilidad y la movilidad en entornos móviles, ya que proporciona una gestión eficiente de los recursos organizándolos y asignándoles prioridades. Además, proporciona seguridad a la red ya que es capaz de detectar patrones de ataque, y también hace que se ahorren costes, ya que optimiza los recursos para que no se infrautilicen o sobreutilicen y estén disponibles en todo momento, esto de manera directa lleva a una reducción de los gastos operativos y energéticos de los dispositivos utilizados [20]

3.2.1. Zabbix.

Zabbix es una plataforma de gestión de red de código abierto y gratuito diseñada para la recopilación de los datos, el procesamiento de estos, generación de alertas y la visualización de los datos de manera numérica o en gráficos. Esta herramienta permite una visión completa del estado la red en el momento actual, las alertas hacen que la gestión sea mucho más cómoda debido a que informan donde y que problema está ocurriendo. [21]

Entre sus características más destacables se encuentra la posibilidad de generar análisis avanzados de la red, siendo estos personalizables, además, incluye la detección automática de redes, la gestión de la configuración y la generación de informes.

Una de las funciones que se puede encontrar en Zabbix, es la gestión en tiempo real, que ofrece información del estado de la red en cada momento. La personalización que presenta la plataforma permite modificar desde las notificaciones recibidas hasta el dashboard que se muestra en la pantalla principal, tal como se muestra en la Figura 3. También se permite la modificación de las alertas, que envían notificaciones cuando detectan variaciones en los parámetros de gestión monitorizados por la plataforma. Además, se proporciona documentación y soporte para el usuario.

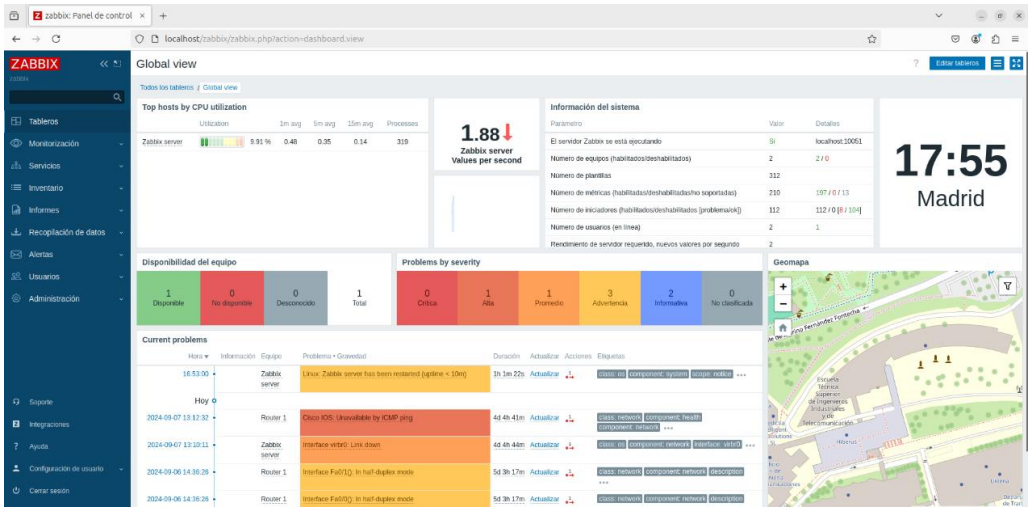

Figura 3: Dashboard Zabbix , Fuente: propia.

Igualmente, Zabbix dispone de una API propia que facilita la integración con otro tipo de tecnologías. Esta API permite a los usuarios realizar integraciones con diferentes plataformas, con el fin de crear un ecosistema adaptado a cada entorno para gestionar de la red. A través de esta API, se encuentran integraciones ya realizadas por otros usuarios que se comparten a través de una amplia comunidad de desarrolladores en internet.

Zabbix además de ser una buena aplicación para la gestión de red, presenta algunas dificultades para su uso, como es su curva de aprendizaje. Adaptarse a esta herramienta requiere algo de tiempo, debido a la cantidad de funciones y configuraciones se pueden realizar en este entorno. El proceso de instalación si no se está familiarizado, puede presentar cierta dificultad, aunque con los conocimientos técnicos básicos no debería presentar ningún problema. Por último, la mayor desventaja que posee es el uso de gran cantidad de recursos, que puede llegar a afectar en el rendimiento del sistema host que lo sustenta.

En estos momentos Zabbix todavía no incorpora una herramienta de inteligencia artificial de manera nativa, pero permite a través de su API la integración con este tipo de herramientas. Aun así, posee diferentes herramientas con funciones que intentan asemejarse a las que puede realizar una inteligencia artificial, lo que muestra intenciones que prometen que, en un futuro no muy lejano, estas dos tecnologías formen una integración que revolucione el escenario actual de la gestión de las redes tal y como se conoce hoy en día. [22]

3.2.2 LibreNMS.

LibreNMS es una plataforma de gestión de red de código abierto que proporciona descubrimiento automático de los dispositivos de la red, alertas, un panel personalizable a gusto del gestor de red, tal y como se observa en la Figura 4, recopilación de datos con múltiples protocolos y una característica muy importante para destacar es que integra una copia de seguridad en el dispositivo. LibreNMS permite la integración con diversas tecnologías a través de su API, y cuenta con una comunidad muy activa que facilita la resolución de problemas en caso de que surjan.[23]

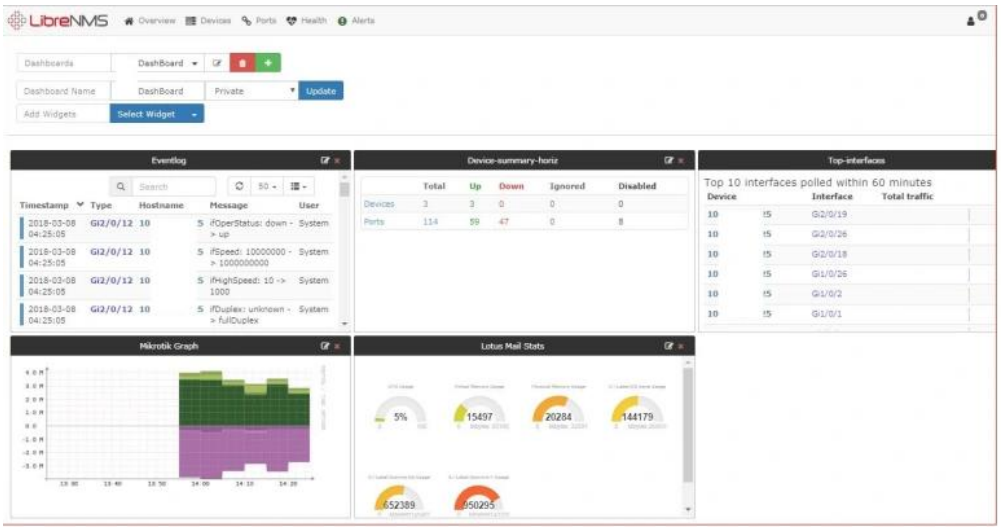

Figura 4: Dashboard LibreNMS, Fuente: https://aacable.wordpress.com/tag/nms/

Las principales características sobre LibreNMS incluyen la compatibilidad con una amplia gama de dispositivos y protocolos, lo que permite que sea utilizada en gran

variedad de entornos. Su interfaz web es intuitiva y facilita la visualización de datos, así como la detección de posibles problemas en la red. También incluye alertas que notifican los problemas de la red a través de diferentes medios como pueden ser por correo electrónico, SMS o incluso notificaciones automáticas a un dispositivo móvil. Todo ello permite la supervisión del rendimiento de la red proporcionando detalladamente la información de los dispositivos. Como se mencionó anteriormente, esta plataforma es capaz de descubrir nuevos dispositivos e incluirlos en la herramienta de gestión, lo que facilita la expansión de la red. Además, LibreNMS posee aplicaciones para Android y iPhone con funcionalidades básicas para los gestores de la red. [24]

LibreNMS no incluye herramientas de inteligencia artificial integradas en la plataforma y presenta una API que permite utilizar dashboards externos como la herramienta Grafana. También permite su integración otras herramientas que permitan la gestión de incidencias lo que hace que se optimice el trabajo y se agilice las repuestas ante los diferentes incidentes. Además, permite crear notificaciones a través de otras aplicaciones como pueden ser Telegram. [23]

3.2.3 Nagios.

Nagios es una plataforma de gestión de red de código abierto que supervisa toda la infraestructura de la red para asegurar que los sistemas, las aplicaciones, los servicios y los procesos funcionen correctamente. Esta herramienta gracias a su arquitectura y flexibilidad permite soportar muchos plugins lo que hacen de ella una herramienta fácil de complementar cuando el plugin está diseñado, en caso contrario se complica ya que diseñar y programar un plugin no es una tarea rápida ni sencilla. Nagios es adecuado para redes de todo tipo, desde pequeñas redes, hasta redes empresariales. [25]

Nagios, al ser de código abierto, ha incentivado a los usuarios a elegir esta herramienta para desarrollar proyectos propios. Esto ha hecho que con el paso del tiempo se hayan desarrollado muchos proyectos e infinidad de plugins para diferentes necesidades de los usuarios. Algunos de estos plugins se encuentran en versiones de pago, pero la mayor parte de ellos son gratuitos, la instalación de estos permite modificar la interfaz de manera sencilla sin tener la necesidad de programarla de manera laboriosa. Ya que

su funcionamiento se basa en incluir una o varias funcionalidades trabajando en conjunto con Nagios como se ve en Figura 5.

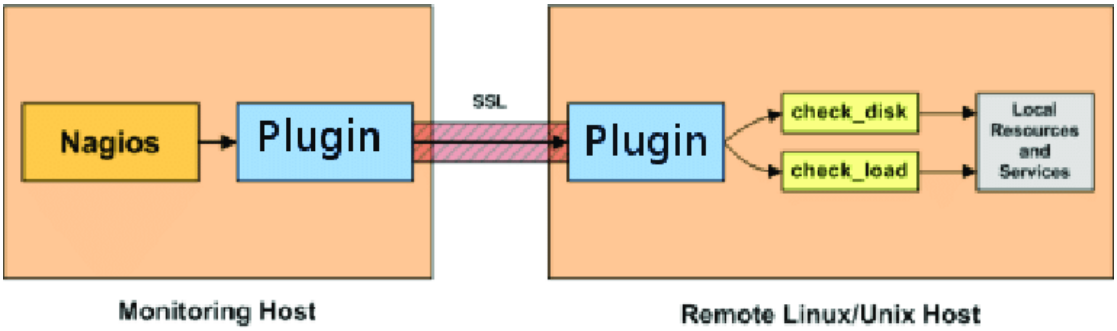

Figura 5: Funcionamiento plugin Nagios, Fuente: https://www.researchgate.net/figure/Nagios-Monitoring-plugin-3_fig7_280672766

El principal inconveniente de Nagios reside en la dificultad de configuración del servidor y de los diferentes tipos de objeto, lo cual principalmente se debe a la falta de documentación en español, lo que ralentiza todos los procesos de despliegue. Por ello, se recomienda empezar por herramientas más amigables para el usuario y que no requieran tantos conocimientos técnicos. [26]

Al igual que las otras dos plataformas anteriormente mencionadas, Nagios no posee herramientas de inteligencia artificial integradas. Pero en esta plataforma se pueden encontrar diferentes plugins que pueden tener funcionalidades similares a las realizadas por una inteligencia artificial. Un ejemplo de plugin similar es la gestión basada en eventos, que se permite detectar situaciones en la infraestructura de red previamente determinadas y actuar de manera programada ante ellas.

3.3. INTELIGENCIA ARTIFICIAL.

La definición de inteligencia artificial varía según el contexto en el que se enmarque. La definición propuesta por IBM es la que mejor se adapta a la conceptualización de inteligencia artificial que se pretende emplear en este TFG. Según IBM 'La inteligencia artificial es una tecnología que permite a ordenadores y máquinas simular la inteligencia humana y su capacidad para resolver problemas.' [26].

La inteligencia artificial se basa en datos que se utilizan para generar y reconocer patrones. Para ello, se emplean algoritmos que los detectan y, mediante software,

crean modelos que permiten su aplicación. Aun así, hay varios tipos de IA, y cada una funciona de una manera totalmente diferente. En el desarrollo del demostrador se usa una rama de la IA llamada machine learning, que permite a los sistemas aprender y mejorar automáticamente a partir de la experiencia, sin necesidad de ser programados explícitamente para cada tarea. [28]

3.3.1. TensorFlow.

TensorFlow es una librería de código abierto desarrollada por Google para el desarrollo y despliegue de modelos de machine learning y deep learning, ofreciendo herramientas para crear, entrenar e implementar modelos de forma eficiente. Esta plataforma hace lo más sencillo posible la creación de estos modelos. Un modelo en machine learning es una representación matemática, que mediante algoritmos de aprendizaje detecta patrones para poder realizar predicciones. En deep learning, los modelos suelen ser redes neuronales profundas, que ajustan sus parámetros durante el entrenamiento para minimizar errores y mejorar la capacidad de generalización. TensorFlow además es compatible con un ecosistema de potentes librerías y de complementos que abre un abanico de posibilidades con las que experimentar. Otro aspecto destacado es la amplia variedad de algoritmos de aprendizaje que ofrece TensorFlow. Además, cuenta con una gran cantidad de recursos en su página web que, a través de ejemplos prácticos, facilitan el aprendizaje y el primer contacto con esta tecnología. [29]

Las características principales de TensorFlow son su API de alto nivel, que facilita el desarrollo de modelos y su manejo de cálculos de manera eficiente, lo que permite el uso de grandes volúmenes de datos. Su API permite implementar aprendizaje automático tanto de alto como de bajo nivel, con capacidad de entrenamiento y cálculo en CPU y GPU. Además, se integra fácilmente con Keras, una API de alto nivel que ha sido de gran utilidad en el desarrollo del demostrador de este TFG.

TensorFlow se utiliza en gran variedad de proyectos y aplicaciones, como el procesamiento del lenguaje natural, el reconocimiento de imágenes, análisis predictivo y el control de vehículos autónomos. Para cada uno de estos casos se debe realizar un tipo específico de entrenamiento que permita al modelo realizar la tarea encomendada a este. [30]

Finalmente, cabe destacar que TensorFlow cuenta con una amplia documentación y una activa comunidad en línea, lo que facilita el soporte y la iniciación en esta plataforma. A pesar de que este software se actualiza de manera continua, se anticipa un rápido crecimiento en los próximos años, lo que ha llevado a muchas empresas a adoptar esta plataforma. [31]

3.3.2 PyTorch.

PyTorch es una librería de código abierto desarrollada por Meta Platforms Inc., que ofrece facilidad de uso y flexibilidad. Es una herramienta que permite crear modelos a diferentes escalas en base a las necesidades. El ecosistema de PyTorch es un ecosistema robusto, ya que posee una comunidad activa de desarrolladores e investigadores, que enriquecen la herramienta con el fin de hacerla completa en todos los ámbitos. Además, esta plataforma esta optimizada para hacerla funcionar en GPUs (Graphics Processing Unit). Su popularidad está creciendo últimamente debido a las nuevas mejoras que se están implementando, esto está haciendo que se utilice en ciertos entornos de investigación. [32]

PyTorch es una de las plataformas de desarrollo de modelos de inteligencia artificial más populares, debido a que se utiliza Python, un lenguaje muy popular en nuestros días. PyTorch es fácil de aprender en comparación a otras tecnologías, lo que permite al desarrollador tener mayor productividad debido a su sencillez y su corta curva de aprendizaje. La depuración del código es sencilla ya que solo se necesitan herramientas de depuración de Python, esto facilita mucho su utilización, posee gran cantidad de bibliotecas útiles y es posible ejecutarlo en la nube, lo que permite a los desarrolladores trabajar desde cualquier parte del mundo, por ello no se requiere tener la capacidad de cómputo necesaria en el hardware que están utilizando. [33]

3.4 ELECCION DE PLATAFORMAS.

Para crear este demostrador se ha decidido utilizar GNS3 como simulador de redes, Zabbix como plataforma de gestión de redes y como herramienta para el desarrollo de modelos de inteligencia a TensorFlow. Esta elección se fundamenta en la necesidad de una integración eficiente. A continuación, detallaré las razones por las que se optó por estas plataformas en lugar de las otras alternativas anteriormente expuestas.

GNS3 es una herramienta poderosa en el ámbito de la gestión de redes, destacándose por su capacidad de crear representaciones fieles de entornos reales. Esto resulta especialmente valioso para realizar pruebas donde los datos son un aspecto crucial. La familiaridad que he adquirido a lo largo de mis años de estudios con esta aplicación ha sido un factor importante, ya que he podido experimentar de primera mano la diversidad de usos que ofrece. Además, la amplia gama de dispositivos que se pueden utilizar y la facilidad con la que se llevan a cabo ciertas configuraciones y acciones son aspectos que también merecen ser destacados. Zabbix es la plataforma de gestión de red elegida, y su principal ventaja radica en su API, que facilita enormemente la integración con la inteligencia artificial para la extracción de datos, siendo mucho más robusta y flexible que la de, por ejemplo, LibreNMS. Además, dado que la IA se desarrolla en Python, Zabbix cuenta con una biblioteca específica llamada PyZabbix, diseñada para interactuar con su API. Como plataforma de gestión de red, Zabbix es muy completa y al igual que las otras plataformas anteriormente comparadas, ofrece alertas configurables por el gestor de red y gráficos detallados de los datos obtenidos. En defintiva, aunque LibreNMS y Nagios son plataformas punteras para la gestión de redes, sus APIs no se integran tan fácilmente con TensorFlow y por eso son descartadas para su integración en el demostrador.

TensorFlow, se ha consolidado como una de las bibliotecas de inteligencia artificial más maduras y robustas disponibles. Permite ejecutar modelos tanto en la CPU (Unidad Central de Procesamiento) como en la GPU, lo cual es relevante en este demostrador, ya que no todos los dispositivos cuentan con una tarjeta gráfica (GPU) potente capaz de soportar la inteligencia artificial. Además, TensorFlow facilita la creación de modelos de inteligencia artificial optimizados para tareas específicas, como las predicciones, que son cruciales para el desarrollo del demostrador que se implementa en este trabajo.

En el ámbito de las plataformas para el desarrollo de modelos de inteligencia artificial, PyTorch es una biblioteca muy completa y utilizada en diversos campos. Sin embargo, carece de ciertas herramientas que ofrece TensorFlow. Aunque PyTorch está optimizado para su rendimiento en GPU, en este caso era preferible que la integración se realizara en CPU. Además, TensorFlow está mejor preparado para manejar grandes volúmenes de datos en comparación con PyTorch. Por estas razones, optar por la plataforma de Google resulta en una solución más robusta y escalable

En resumen, la integración de GNS3, Zabbix y TensorFlow proporciona un entorno capaz de simular redes de manera realista, añadiendo una capa de gestión integral en tiempo real de las métricas seleccionadas. Además, incorpora una capa de inteligencia que permite realizar predicciones, apoyando al operador de red. Esta combinación genera una solución más completa y eficaz que cualquier otra alternativa disponible.

4. Implementación del demostrador.

Para la implementación de este demostrador, se preparará el entorno con el objetivo de facilitar la integración y simplificar el proceso. Por lo tanto, este capítulo se divide en dos secciones: la primera aborda la preparación del entorno, y la segunda se centra en la configuración utilizada para esta integración. En la Figura 6 se muestra un diagrama con la integración de todas las herramientas que componen el demostrador implementado en este trabajo de fin de grado. En el diagrama se recogen las aplicaciones y servicios que engloban la integración y las interconexiones entre todas ellas. En este capítulo se describe de forma detallada el procedimiento utilizado para realizar esta integración. Como se observa en esta figura, se superponen las implementaciones de tres planos funcionales, el primero de ellos, el plano de red, implementado por la plataforma GNS3 que es el encargado de simular la red y sus diferentes estados. Por encima de este plano se sitúa el plano de gestión, que es implementado por la plataforma de gestión de red Zabbix, encargada de mostrar el estado actual de la red en todo momento. Y, por último, el plano de inteligencia representado por la librería TensorFlow y encargado de aportar la parte de inteligencia artificial en este demostrador.

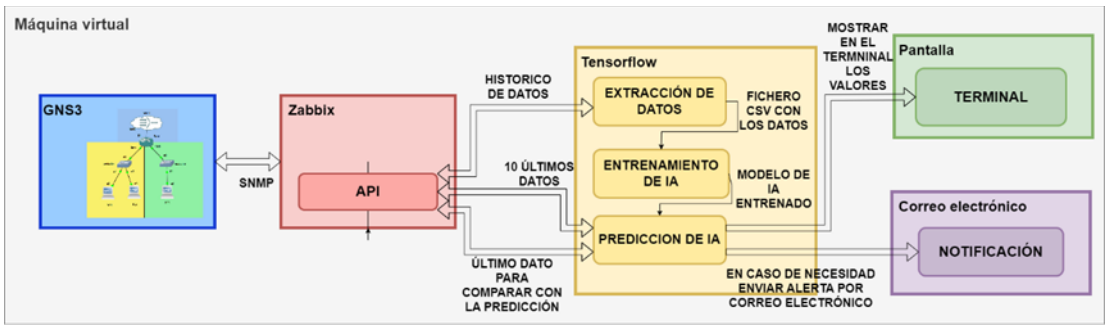

Figura 6: Diagrama de la integración , Fuente: propia.

4.1. PREPARACIÓN DEL ENTORNO

En este capítulo se van a detallar los procesos de la creación de la máquina virtual y las diferentes instalaciones (GNS3, Zabbix y TensorFlow) que se van a realizar de los diferentes programas utilizados en el demostrador.

Creación de la máquina virtual.

Para la implementación de la integración en un entorno controlado, se va a implementar la arquitectura que se ve en la Figura 7. Para ello, se parte de un entorno común en diferentes dispositivos con Windows 11 como sistema operativo, como hipervisor se utiliza VirtualBox para crear la máquina virtual, donde se utiliza Ubuntu como sistema operativo. Primero, se descarga e instala VirtualBox desde su sitio oficial. Luego, se configura una máquina virtual, asignando los recursos, es decir, el número de CPU, la cantidad de memoria RAM y , la cantidad de disco duro virtual, que en este caso se utilizara 4 CPU, 4096 MB de memoria RAM y 50GB de disco duro. Después, se descarga la imagen ISO de Ubuntu de su página oficial, y se utiliza para instalar el sistema operativo dentro de la máquina virtual. Una vez instalado Ubuntu siguiendo las indicaciones de la página web oficial, se actualizan los paquetes. Una vez hecho esto, el entorno está preparado para instalar las herramientas necesarias, como GNS3, Zabbix y TensorFlow, con ello, el entorno de desarrollo está perfectamente preparado para la integración.

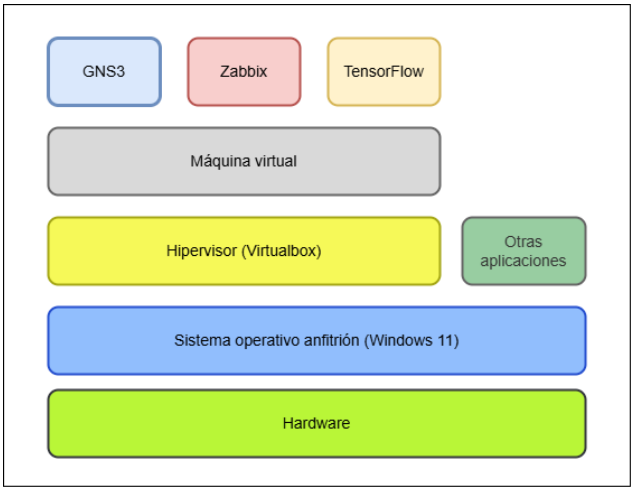

Figura 7: Arquitectura de la virtualización, Fuente: propia.

Instalación de GNS3.

La primera herramienta que se va a instalar es GNS3, ya que es la encargada de simular la red, y sin ella nada del demostrador es posible. Con esta aplicación se consigue el entorno de red que replica una infraestructura real. Para instalar el GNS3 se deben de seguir una serie de pasos, lo primero seria agregar a través del comando `sudo

`add-apt-repository ppa:gns3/ppa'` el repositorio oficial de GNS3, a través de este comando se obtiene el software necesario para poder tener acceso a la última versión de GNS3. A continuación, con el fin de asegurar de que todos los paquetes estén actualizados y no ocurran problemas, se ejecutara `'sudo apt update'` como medida preventiva.

Una vez que esté todo preparado, se procederá con la instalación de GNS3 usando el comando `'sudo apt install gns3-gui gns3-server'`. Este comando descarga e instala la interfaz gráfica y el servidor de GNS3, partes necesarias para la utilización de este. Una vez se haya finalizado este proceso, se comprueba que la herramienta está bien instalada, para ello la se abre desde el menú de aplicaciones o en el terminal se ejecuta el comando `'gns3'`. Tras ello se debe de observar cómo se abre la plataforma y se tiene disponible el entorno para utilizarlo como se observa en Figura 8.

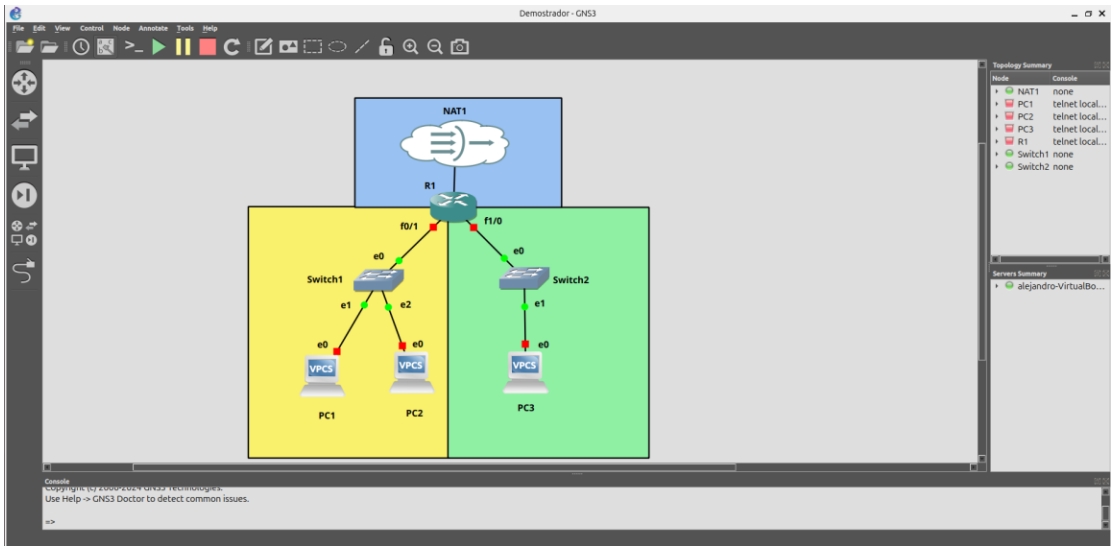

Figura 8: Entorno de GNS3 , Fuente: propia.

Instalación de Zabbix.

Teniendo preparado la parte de la simulación de red, el siguiente aspecto necesario es la gestión de red, ya que sin ella sería imposible recopilar los datos a tiempo real. Para instalar Zabbix, siendo el proceso más largo que con la anterior herramienta, se enttra en modo super usuario con el comando `'sudo su'`. Luego con el comando `'wget https://repo.zabbix.com/zabbix/6.4/ubuntu/`

pool/main/z/zabbix-release/zabbix-release_6.4-1+ubuntu24.04_all.deb' se descarga el paquete de Zabbix que contiene los archivos necesarios para proceder con su instalación. Y al igual que con GNS3 se actualiza la lista de paquetes para asegurar que todo en óptimas condiciones con 'apt update'.

Teniendo el entorno preparado, se instala Zabbix y sus componentes ejecutando 'apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-sql-scripts zabbix-agent'. Para el poder usar Zabbix también hay que instalar MySQL Server un sistema de gestión de bases de datos con el comando 'apt-get install mysql-server -y' para gestionar la base de datos que tiene Zabbix. Una vez que se tiene instalado MySQL, se inicia con 'mysql -uroot -p' y se configura la base de datos de Zabbix creando una nueva base de datos con 'create database zabbix character set utf8mb4 collate utf8mb4_bin;' y un usuario con los permisos necesarios 'create user zabbix@localhost identified by 'ubuntu';' y 'grant all privileges on zabbix.* to zabbix@localhost;'.

Después de esto, se importa el esquema inicial de la base de datos utilizando 'zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --default-character-set=utf8mb4 -uzabbix -p zabbix'. Luego, se ajustan algunas configuraciones adicionales en MySQL, se edita el archivo de configuración de Zabbix 'nano /etc/zabbix/zabbix_server.conf' para establecer la contraseña de la base de datos, y se recargan los servicios con 'systemctl daemon-reload'. Finalmente, se reinicia y se habilitan los servicios de Zabbix y Apache con 'systemctl restart zabbix-server zabbix-agent apache2 y systemctl enable zabbix-server zabbix-agent apache2'. Con estos pasos, el entorno de Zabbix queda instalado, y para acabar con su configuración y que esté listo para ser utilizado a través del navegador, se accede a través del navegador al lugar donde se tenga instalado Zabbix, en el caso de este demostrador es http://localhost/zabbix y aparecerá la pantalla de bienvenida como se ve en la Figura 9.

Figura 9: Configuración de idioma , Fuente: propia.

Tras ello se continua con el próximo paso y se realiza la comprobación de los requisitos previos con el fin de asegurar que todo está instalado correctamente como se ve en la Figura 10.

Figura 10: Comprobación de requisitos previos , Fuente: propia.

Al continuar con el siguiente paso, se presenta como se ve en Figura 11 la configuración de la base de datos, en ella se configuran los parámetros necesarios, en este caso se observan ya configurados en la imagen.

Figura 11: Configuración de la base de datos , Fuente: propia.

Despues en la Figura 12 se ve que se indica configurar unos últimos ajustes como son el nombre, zona horaria y el tema.

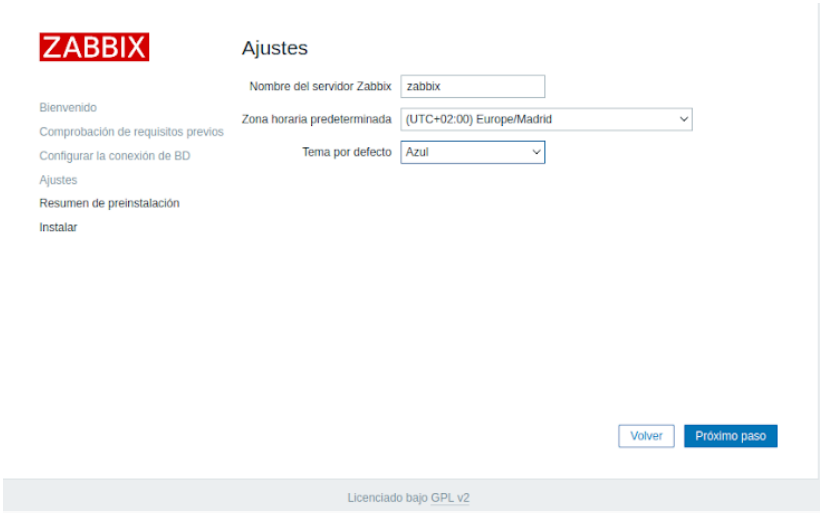

Figura 12: Ajustes finales, Fuente: propia.

Por último, se indica el resumen de los datos más relevantes como se observa en la Figura 13, se comprueban y así se acaba de configurar el entorno, cuando finaliza surge otra pantalla como se ve *en* Figura 14 y así finaliza el proceso.

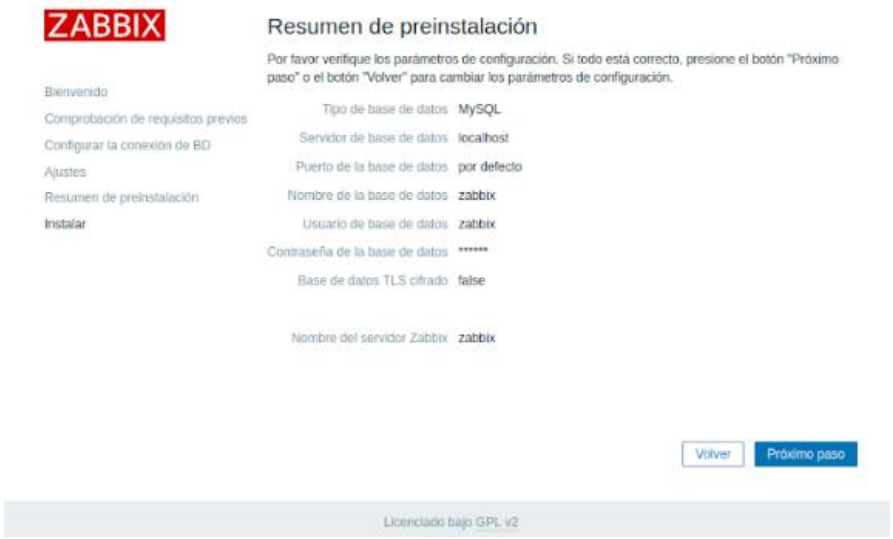

Figura 13: Resumen preinstalación, Fuente: propia.

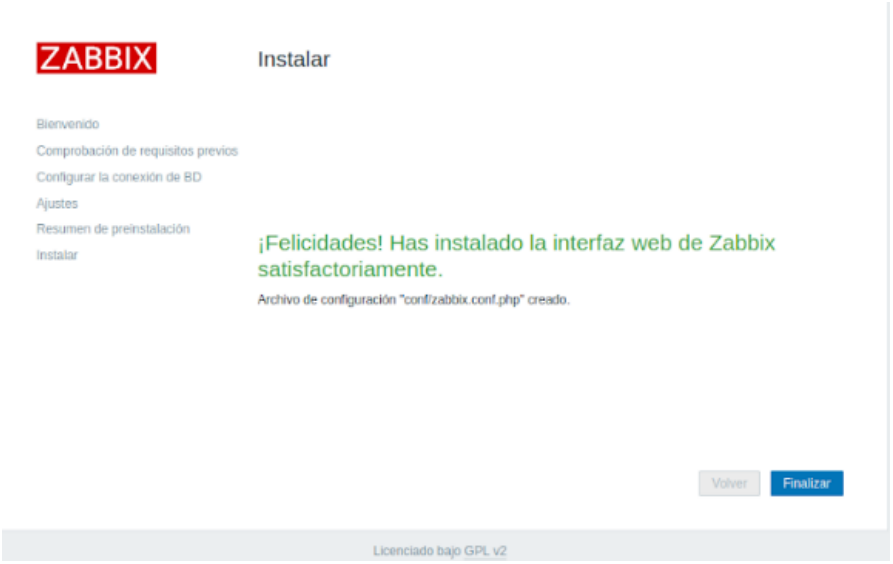

Figura 14: Instalación completada, Fuente: propia.

Teniendo todo preparado, accediendo al enlace anteriormente indicado, ya sale la página de inicio de sesión como se ve en la Figura 15, y con las credenciales configuradas anteriormente se puede acceder a la herramienta.

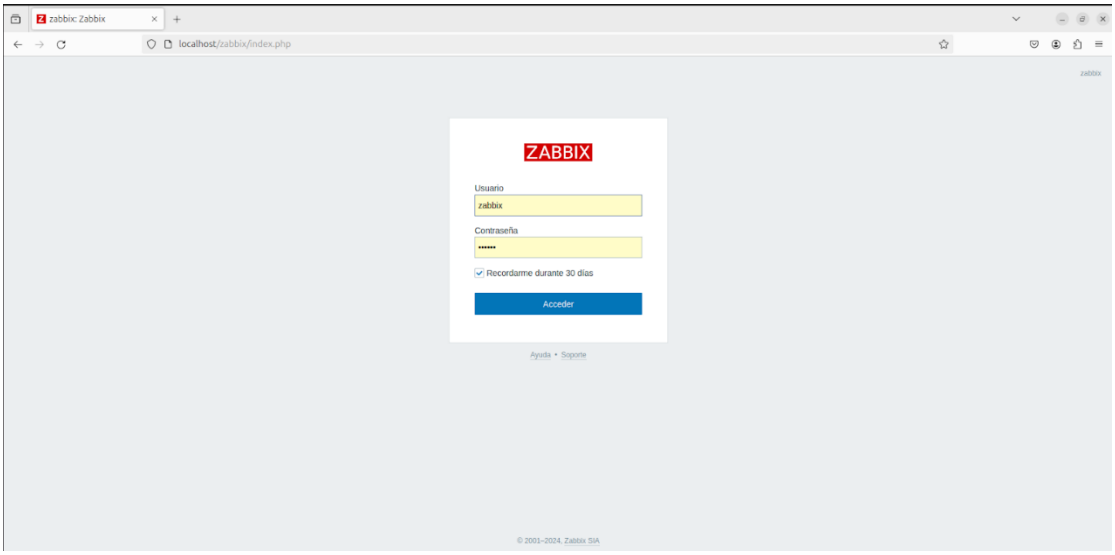

Figura 15: Pantalla de acceso, Fuente: propia.

Para finalizar se entra en el entorno de Zabbix donde se encuentra el panel inicial con diferentes métricas como se ve en la Figura 16.

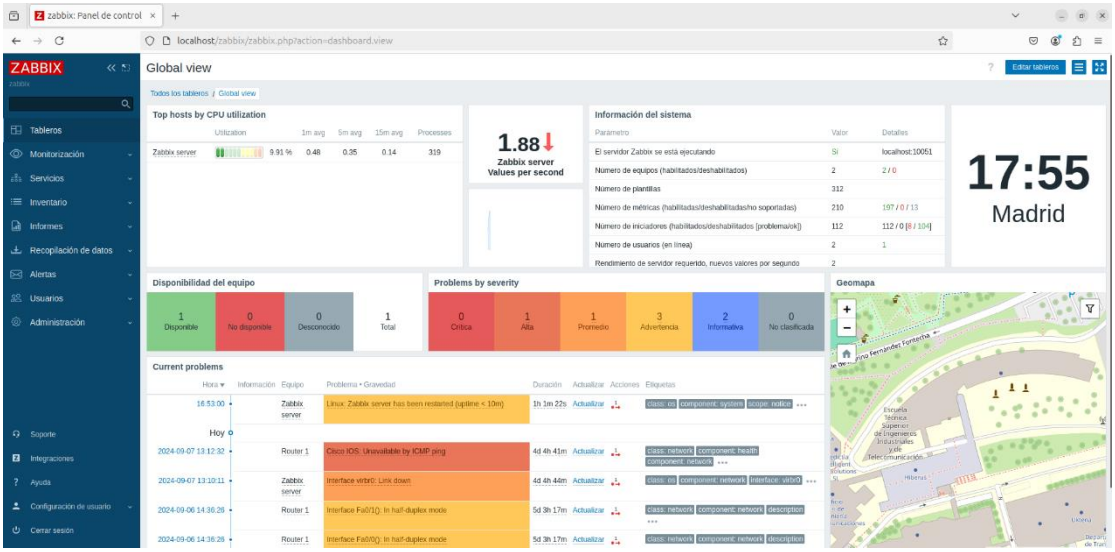

Figura 16: Entorno de Zabbix , Fuente: propia.

Instalación de TensorFlow.

La última herramienta que queda por instalar es TensorFlow, que es un entorno de trabajo que permite crear modelos de inteligencia artificial, este entorno se va a crear en Python.

Para la instalación Python se ejecuta en el terminal `'sudo apt install python3 python3-venv'` con ello se garantiza la última versión y la capacidad de crear entornos aislados.

Después de tener Python instalado, se procede a instalar varias librerías con el comando `'pip install pandas numpy TensorFlow scikit-learn matplotlib pyzabbix'`. A continuación, se describen las funcionalidades que presentan cada una de estas librerías.

Las librerías `pandas` y `numpy` se utilizan en el ámbito de la manipulación y el análisis eficiente de datos, y se encargan del tratamiento de grandes volúmenes de estos, en este caso para los datos obtenidos por Zabbix que va a utilizar la inteligencia artificial. Además, `numpy` proporciona estructuras de datos de alto rendimiento para cálculos numéricos y pandas facilita el análisis de los datos.

`TensorFlow`, es la librería encargada del desarrollo de la inteligencia artificial, ya que se encarga del desarrollo y entrenamiento de los modelos. Esta librería permite crear y entrenar las redes neuronales que son capaces de realizar las predicciones o detectar anomalías en caso de que se opte por esta capacidad, esta librería es esencial para el desarrollo del demostrador.

`Scikit-learn` es otra librería de inteligencia artificial, encargada de del aprendizaje automático. Esta librería proporciona las herramientas para la clasificación, la regresión, el agrupamiento y otras técnicas que complementan el uso de TensorFlow, especialmente se centra en este trabajo en la preparación de datos y el ajuste de los modelos.

Finalmente, la librería `pyzabbix` permite interactuar de manera eficiente con la API de Zabbix, haciendo más sencilla la extracción de los datos y la automatización de

ciertas tareas. Con esta librería se pueden obtener métricas en tiempo real de Zabbix para procesarlas con Python y así utilizar los modelos de IA.

Después de tener instaladas estas librerías se actualiza el sistema con `'sudo apt update'` para poder asegurar se tiene la última versión de todos los paquetes. Para facilitar el uso de esta tecnología sin interferir o generar problemas, se crea un entorno virtual aislado llamado 'zabbix_env' con el comando `'python3 -m venv zabbix_env'`. Este entorno va a permitir que no haya interferencias con otras configuraciones del sistema.

Para utilizar este entorno se debe ejecutar en el terminal el comando `'source zabbix_env/bin/activate'`. Esto hace que las operaciones que se realicen posteriormente se encuentren dentro del entorno de trabajo, facilitando la integración sin interferir con otras haciendo que así se mejore la fluidez con la que puede operar la inteligencia artificial.

4.2. CONFIGURACION DE LA INTEGRACIÓN

En este capítulo, con las herramientas ya instaladas, se procede a configurar todo el entorno. Se distinguirá entre la configuración de los dispositivos de red (capa de red), la configuración de la plataforma de gestión (capa de gestión), incluyendo el método de generación del tráfico de red, y finalmente, la integración y configuración de las herramientas de inteligencia (capa de inteligencia), conformando así la arquitectura completa del demostrador.

En la Figura 17, se muestra la superposición de las tres capas introducidas anteriormente junto con las plataformas y herramientas que las componen. La *capa de red*, mediante la herramienta de simulación GNS3, implementa los distintos dispositivos de red interconectados permitiendo la conectividad entre todos ellos, por encima se le superpone la *capa de gestión* representada por la plataforma de gestión Zabbix que interconecta los agentes de gestión de los dispositivos gestionados con el gestor principal (Zabbix). Por encima de ambas se sitúa la *capa de inteligencia* representada por una serie de librerías de Python entre las que destaca la librería TensorFlow que realiza el tratamiento inteligente de los datos obtenidos desde la capa de gestión.

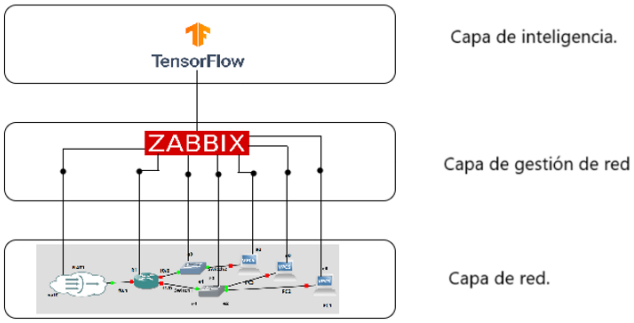

Figura 17: Arquitectura de capas, Fuente: propia.

Configuración de la capa de red.

La configuración de la capa de red supone el despliegue e implementación de la topología de red del demostrador sobre la herramienta de simulación de redes GNS3. El escenario que se va a crear se muestra en la Figura 18. Está compuesto de un router, dos switches, tres máquinas virtuales simples (VPCS) que representarán los ordenadores de usuario o PCS, y un nodo (nube) con la funcionalidad de NAT.

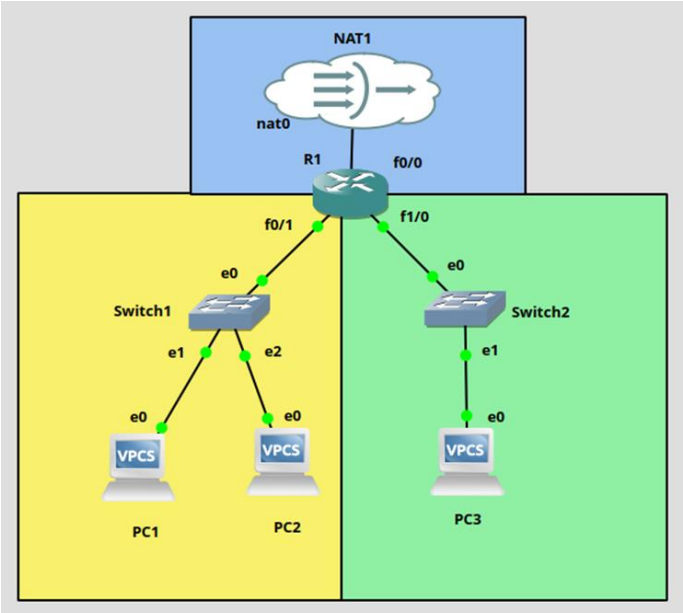

Figura 18: Red simulada, Fuente: propia.

Para crear este escenario es necesario arrastrar desde el menú izquierdo de la ventana principal de GNS3 cada uno de los dispositivos necesarios para completar la topología requerida en el demostrador.

En este caso, se colocan los tres PC en la parte baja y con la herramienta de conectar se crean los enlaces que interconecta los PC con los conmutadores ethernet correspondientes, finalmente se añade el nodo NAT que será el encargado de conectar la red simulada con el equipo host o anfitrión de la máquina virtual. GNS3 por defecto no posee ningún dispositivo de encaminamiento o router, para solucionar este problema se debe utilizar el asistente para cargar la imagen previamente descargada, en este caso la imagen de un router Cisco C3725. Una vez realizada esta operación, el router aparece en el menú de la izquierda, donde se encuentran todos los dispositivos o también en la sección de routers. Finalmente se arrastra este router Cisco al entorno y se hacen las conexiones de red con la herramienta dedicada para ello.

Teniendo ya la disposición de la red formada y conectada, se procede con la configuración individual de cada dispositivo. Para ello, se abren las consolas correspondientes a los PC, a los switches y al router. Primero se configuran los PC ya que solo requieren que se les configure la dirección ip. En el caso de PC1, se verifica inicialmente la configuración de red mediante el comando `'show ip'`, se observa que no tiene dirección IP asignada. Luego, se asigna la dirección IP 192.168.100.11 con la máscara de red 255.255.255.0 y se configura la puerta de enlace como 192.168.100.1 con el comando `'ip 192.168.100.11 255.255.255.0 gateway 192.168.100.1'`. Esta configuración se guarda con el comando `'write memory'`. Para PC2, se sigue un procedimiento similar: se revisa la configuración de red usando `'show ip'` y, al no tener dirección IP asignada, se establece la IP 192.168.100.12, la máscara 255.255.255.0 y la puerta de enlace 192.168.100.1 con `'ip 192.168.100.12 255.255.255.0 gateway 192.168.100.1'`, se guardan después los cambios con `'write memory'`. Finalmente, para PC3, se repiten estos pasos; primero se verifica la configuración de red con `'show ip'` y se asigna la dirección IP 192.168.110.11 con la máscara 255.255.255.0 y la puerta de enlace 192.168.110.1 mediante `'ip 192.168.110.11 255.255.255.0 gateway 192.168.110.1'`, guardando también la configuración con `'write memory'`.

La configuración del router Cisco C3725 se comienza habilitando el modo privilegiado con 'enable' y se entra al modo de configuración global con 'conf t'. Luego, se configuran las interfaces del router, en FastEthernet0/0 se accede con 'interface fastEthernet 0/0' y se asigna la IP 192.168.122.10 ya que al ser la puerta que conecta GNS3 con el host, esto se hace a través de la interfaz 'virbr0' del host Ubuntu que tiene como ip la 192.168.122.1, por ello se utiliza una ip de la misma red, y además también se debe utilizar también la misma máscara 255.255.255.0 y esto se fija con el comando 'ip address 192.168.122.10 255.255.255.0' y se habilita la interfaz con 'no shut'; en FastEthernet0/1 se accede con 'interface fastEthernet 0/1' y se establece la IP 192.168.100.1 con la misma máscara usando 'ip address 192.168.100.10 255.255.255.0' y se habilita con 'no shut'; y en FastEthernet1/0 se accede con 'interface fastEthernet 1/0' y se configura la IP 192.168.110.1 con la máscara 255.255.255.0 con 'ip address 192.168.110.10 255.255.255.0' y se habilita con 'no shut'. Tras estas configuraciones, se guarda la configuración utilizando 'write memory'.

Tras todo este proceso se comprueba que las interfaces estén correctamente configuradas y operativas con los comandos 'show ip interface brief' y 'show ip route'. Se añade una ruta por defecto para el tráfico que no pertenece a ninguna red local con el comando 'ip route 0.0.0.0 0.0.0.0 192.168.122.1' y se guarda la configuración nuevamente con 'write memory'

Finalmente, en el host donde se encuentra alojado GNS3, es necesario configurar rutas para asegurar la correcta conectividad las subredes. Para ello, se ejecutan los comandos 'sudo ip route add 192.168.100.0/24 via 192.168.122.10' y 'sudo ip route add 192.168.110.0/24 via 192.168.122.10'. Con estos pasos completados, la red queda configurada correctamente en GNS3, permitiendo la comunicación entre todos los dispositivos conectados y proporcionando acceso al host e Internet mediante el nodo NAT.

Se verifica la conectividad entre los PCS utilizando el comando 'ping', y se prueba la conexión a Internet para asegurarnos de que el NAT funciona correctamente.

Configuración de la capa de gestión de red.

Para gestionar mediante SNMP el router simulado en GNS3 con Zabbix es necesario realizar configuraciones tanto en el router como en Zabbix. Primero, se prepara el router simulado en GNS3 habilitando el protocolo SNMP, para ello hay que definir una comunidad SNMP que actúa como contraseña compartida entre el router y la plataforma de gestión. Esto se consigue mediante el comando '`snmp-server community public ro`', que establece una comunidad llamada 'public' con permisos de solo lectura. Luego, se configura el servidor de Zabbix para recibir traps SNMP desde el router utilizando el comando '`snmp-server host 192.168.122.1 version 2c public udp-port 161`', donde se especifica la dirección IP del servidor de Zabbix, utilizando SNMP versión 2c, la comunidad 'public' y el puerto UDP 161. Tras realizar estos ajustes, es importante guardar la configuración en el router con '`write memory`'.

Con el router configurado, se accede a la interfaz web de Zabbix para añadirlo como un nuevo host. Para ello se entra en la sección 'Configuration' > 'Hosts', se selecciona 'Create host' y se proporciona el nombre del host, asignándolo a un grupo en este caso 'Network Devices'. En el capítulo de interfaces, se añade una interfaz SNMP con la dirección IP configurada anteriormente en el router, configurando el puerto 161 y seleccionando SNMPv2, que es la versión configurada previamente en el router. Luego, se asocian las plantillas adecuadas, como 'Template Net SNMPv2' y 'Template Cisco IOS SNMP', para definir los ítems de gestión. Una vez guardada la configuración, Zabbix comenzará a recopilar datos del router.

Simulación del tráfico base

La métrica que se va a utilizar es el tráfico entrante por la interfaz f0/0 del router. Para poder realizar pruebas de tráfico hacia esta red, se ha realizado una simulación del tráfico de red típico de un día laboral utilizando el comando '`sudo hping3 -1 192.168.x.x`'. Este comando utiliza hping3, que es una herramienta de generación de paquetes TCP/IP utilizada en nuestro caso para enviar paquetes ICMP a la red simulada con la dirección IP 192.168.x.x. La opción -1 indica que se envían paquetes de tipo ICMP, ya que hay más opciones de tipo de tráfico que se pueden escoger. Los paquetes de tráfico ICMP son comúnmente utilizados para verificar la conectividad de la red y medir la latencia. La finalidad de esta simulación es emular el

comportamiento normal de la red durante un día de actividad que se muestra en la Figura 19.

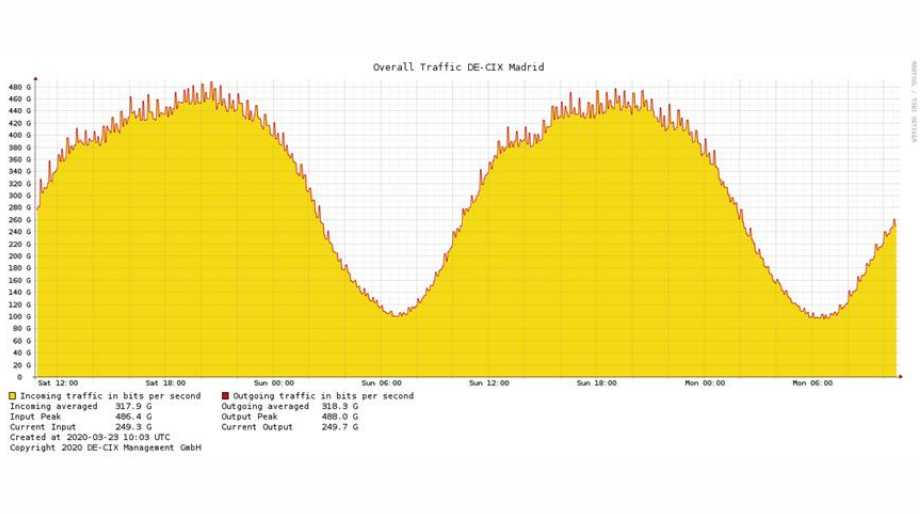

Figura 19: Tráfico diario, Fuente: DE-Cix https://blog.orange.es/red/horas-valle/

En la Figura 20 se muestra la aproximación generada en el demostrador con el fin de ser usada como muestra. Esta es una reproducción ciertamente afín a lo que se representa, que para el demostrador y las demandas de este son convenientes. Al generar este tipo de tráfico, se permite al sistema de gestión, en este caso Zabbix, analizar patrones de uso, medir el rendimiento y detectar posibles problemas.

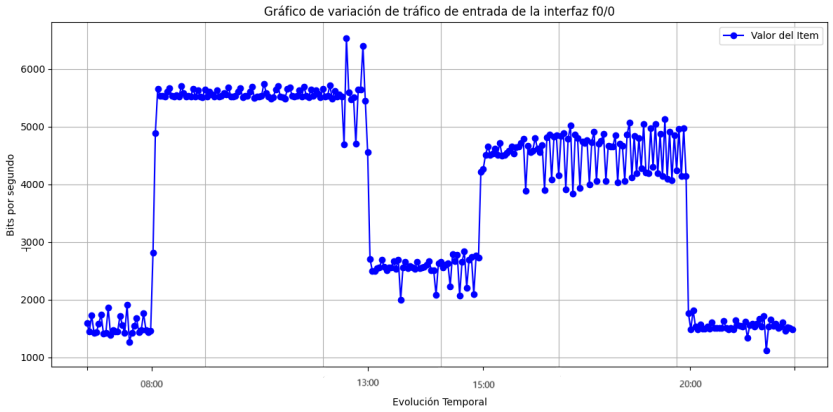

Figura 20: Simulación de tráfico diario, Fuente: propia.

Configuración de la capa de inteligencia

La integración de TensorFlow con Zabbix, va a permitir utilizar el modelo de IA con el fin de realizar predicciones del tráfico entrante por la interfaz f0/0 del router. Esta integración se divide en cinco scripts de Python que interaccionan entre sí y con otras plataformas, como se ve en la Figura 21. Estos scripts se utilizan de manera secuencial con el fin de obtener los datos necesarios de Zabbix. Los datos son utilizados para que se cree el modelo y se ponga en funcionamiento para predecir las posibles anomalías que puedan ocurrir en el tráfico y generar las alertas correspondientes.

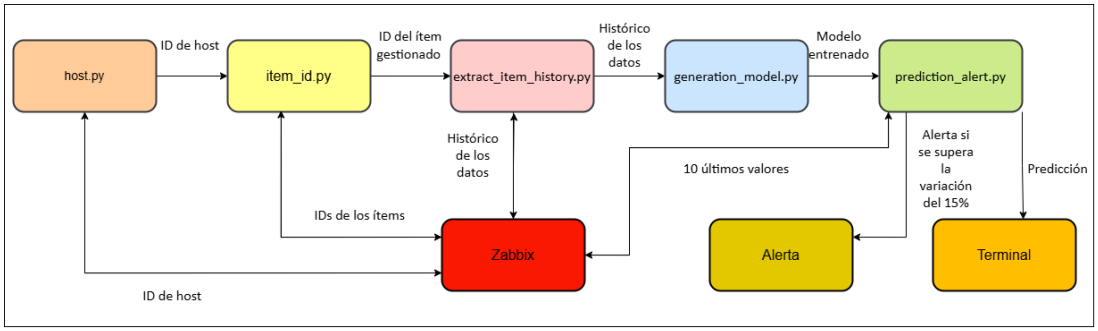

Figura 21: Interacción entre scripts, Fuente: propia.

El primer script, 'host.py', se encarga de conectarse a Zabbix a través de su API, y obtener el ID de host sobre el que se desea obtener los datos. Este ID es necesario ya que permite identificar a un dispositivo de manera única dentro del entorno en el que se encuentra Zabbix.

El segundo script, 'item_id.py', utiliza el ID del host obtenido con el primer script para identificar al dispositivo, y así obtener la lista de los ítems o métricas. Los ítems o métricas son por ejemplo el uso de la CPU, la tasa de tráfico que va por los interfaces o el uso de la memoria. Y este script se encarga de obtener el su id único para utilizarlo más adelante.

El tercer script, 'extract_item_history.py', se encarga de obtener el histórico de datos de la métrica seleccionada. Este paso es de los más importantes, ya que la información que se obtiene en este momento de los datos va a ser la que nutrirá la inteligencia artificial y hará que funcione de manera más o menos precisa, por ello un número elevado de datos y una buena calidad en estos hará que el error que cometa la IA aumente o se reduzca significativamente.

El cuarto script, 'generation_model.py', se encarga de generar el modelo de TensorFlow entrenado, basándose en los datos anteriormente obtenidos, por ello la importancia de estos. Este modelo aprende los patrones y tendencias que se presentan en este conjunto de datos, con el fin de poderlos detectar y predecir los valores futuros con la mayor exactitud posible.

Finalmente, el quinto script, 'prediction_alert.py', es el encargado de utilizar el modelo entrenado y generado con el script anterior. Este script utilizando el modelo, y los últimos 10 datos registrados en la base de datos de Zabbix, realiza las predicciones de la métrica que se está trabajando. Si se detecta una variación que excede un umbral definido, en caso de este demostrador será configurado en un 15%, generará una alerta al gestor de la red a través de correo electrónico. Esto permite una respuesta proactiva y evita problemas antes de que impacten significativamente en el rendimiento de la red.

La unión de estos cinco scripts conforma una integración robusta entre las herramientas, que permite aplicar las técnicas de inteligencia artificial a la gestión de red, para hacer que potenciales problemas se solucionen con antelación y se resuelvan de manera anticipada. Lo que hace que la gestión de red clásica se vea complementada por esta herramienta, y facilite las situaciones al gestor de red. Estos scripts se detallarán de manera detallada a continuación.

El script *host.py* mostrado en el Anexo I, se conecta a la API de Zabbix utilizando la librería de alto nivel pyzabbix, que autentica al usuario con las credenciales configuradas para acceder a los datos. Tras establecer la conexión y obtener el token de autenticación, se envía una petición para obtener la lista de hosts registrados en Zabbix junto a su identificador, especificando que solo se requiere obtener los campos hostid (identificador único del host) y name (nombre del host). Esto se logra a través de la línea '`hosts = zapi.host.get(output=['hostid', 'name'])`'. Posteriormente, el script itera sobre los hosts obtenidos y los imprime por pantalla, mostrando el ID y el nombre de cada uno. Esta información es muy importante como se mencionó anteriormente, debido a que sin ella no sería viable la identificación de los dispositivos y no sería posible obtener la lista de ítems ni los datos que recogen estas métricas. En este script se incluye el manejo de errores de conexión a través de la API con el servidor del gestor de red.

El script *item_id.py*, mostrado en el Anexo II, conecta a la API de Zabbix utilizando la librería de alto nivel pyzabbix, que autentica al usuario con las credenciales configuradas para acceder a los datos. Tras establecer la conexión y obtener un token de autenticación, el script utiliza el hostid obtenido por el primer script, para solicitar las métricas junto a sus identificadores con el fin de poder tener su referencia. Esto se consigue con la línea `'items = zapi.item.get(hostids=host_id, output=['itemid', 'name', 'key_', 'lastvalue'])'`, donde se detalla que solo se necesitan los campos itemid (identificador único del ítem), name (nombre del ítem), key_ (clave del ítem) y lastvalue (último valor registrado).

Una vez obtenidos los ítems, el script los convierte en un formato más manejable mediante un DataFrame (estructura de datos con dos dimensiones) de la biblioteca pandas, y los guarda en un archivo CSV. Esto permite tener los datos organizados y listos para su posterior análisis. Además, las excepciones están presentes para alertar de posibles errores al conectarse con la API de Zabbix, asegurando que se notifiquen fallos en caso de que no se puedan obtener los ítems. Este proceso es crucial, ya que el itemid es necesario para el tercer script que extraerá el histórico de datos y continuará con el flujo de trabajo.

El script *extract_item_history.py*, conecta a la API de Zabbix utilizando la librería de alto nivel pyzabbix, que autentica al usuario con las credenciales configuradas para acceder a los datos históricos del ítem que se está tratando en específico, cuyo itemid fue obtenido en el script anterior. Tras establecer la conexión y autenticarse, el script envía una solicitud para recuperar los datos históricos del ítem definido, dentro de un rango de tiempo determinado. Esto se logra mediante la línea `'history = zapi.history.get(itemids=item_id, time_from=time_from, time_till=time_till, output='extend', history=value_type, limit=max_samples)'`, que solicita las muestras históricas limitadas a un número máximo y de un tipo de dato específico (value_type), que depende del ítem.

El rango de tiempo se calcula utilizando las librerías datetime y pandas, y se determina el tipo de dato del ítem para solicitar correctamente el historial de datos. Con todo ello, los datos son convertidos en un formato manejable mediante un DataFrame de pandas, que luego se guarda en un archivo CSV. Este archivo contiene las columnas itemid y value, que son esenciales para entrenar el modelo de inteligencia artificial en pasos posteriores. El script incluye manejo de excepciones para notificar errores que

puedan ocurrir durante la conexión o la extracción de datos, asegurando que el proceso de recuperación de información histórica sea robusto y confiable.

El script *generation_model.py*, descrito en el Anexo IV, se encarga de entrenar el modelo de inteligencia artificial utilizando los datos históricos del ítem que se ha seleccionado, que han sido previamente extraídos y guardados en un archivo CSV. Este script utiliza la biblioteca TensorFlow para crear un modelo de redes neuronales de tipo LSTM (Long Short-Term Memory), que es ideal para analizar series temporales, como son por ejemplo las métricas de rendimiento extraídas de Zabbix.

El entrenamiento del modelo comienza cargando los datos desde el archivo CSV anteriormente generado, esto lo realiza a través de la función cargar_datos, que utiliza pandas para leer el archivo. Luego, los datos se preprocesan con la función preprocesar_datos, que es la encargada de escalar los valores utilizando MinMaxScaler para normalizarlos en un rango de 0 a 1, un paso que mejora el rendimiento de los modelos LSTM. También se organizan los datos en ventanas de tiempo definidas por time_steps, que son secuencias de valores anteriores que se usarán para predecir el siguiente valor.

El modelo LSTM se crea con la función entrenar_modelo, que utiliza una arquitectura de capas LSTM y Dropout para evitar el sobreajuste durante el entrenamiento. El modelo se entrena con los datos preprocesados, donde la pérdida se minimiza utilizando el algoritmo de optimización Adam y la función de pérdida de error cuadrático medio (mean_squared_error).

Finalmente, el modelo entrenado se guarda en un archivo .keras, junto con el scaler que fue utilizado para normalizar los datos. Esto asegura que los nuevos datos puedan ser escalados de la misma manera para hacer las predicciones. El manejo de los parámetros de entrenamiento, como el número de épocas y el tamaño del lote, permite ajustar el rendimiento del modelo según las necesidades del sistema. Este paso es fundamental en la integración, ya que el modelo LSTM entrenado aprenderá a reconocer los patrones en base a los datos históricos del ítem gestionado, lo que permitirá detectar posibles anomalías en el comportamiento futuro del sistema, necesario para la generación de alertas en los siguientes pasos del proceso.

El script ‘prediction_alert.py’, descrito en el Anexo V, tiene como función principal aplicar el modelo de inteligencia artificial entrenado en el script anterior, con el fin de realizar las predicciones en tiempo real sobre los datos extraídos de Zabbix, y genera alertas en caso de que se detecten variaciones significativas. El proceso se inicia conectándose a través de la API de Zabbix para obtener los últimos diez valores históricos del ítem que se está gestionando, y se utiliza un modelo LSTM entrenado para predecir el siguiente valor.

Para realizar la predicción, el script carga el modelo y el scaler generados en el script anterior mediante la función cargar_modelo_y_scaler, también se tiene la función obtener_datos_historicos_zabbix, que extrae los últimos diez valores del ítem como he mencionado antes y obtener_ultimo_valor_zabbix, obtiene el valor más reciente registrado por Zabbix para esa métrica. El modelo utiliza esta información para predecir el siguiente valor y con el ultimo valor obtenido se realiza la comparación y se calcula el error cometido.

Para evaluar la precisión de la predicción, en el script se incluye el cálculo del porcentaje de error absoluto medio (MAPE) mediante la función calcular_mape. Además, se compara la predicción con un promedio esperado ajustado manualmente según la hora del día, usando la función obtener_promedio_segun_hora_manual. Si la diferencia entre el valor predicho y el promedio esperado supera un umbral del 15%, el script envía una alerta por correo electrónico. Esto se logra a través de la función enviar_correo, que se conecta a un servidor SMTP y envía un correo con la información sobre la variación detectada. El formato escogido para el correo incluye, el valor predicho, el promedio esperado y la diferencia en bits por segundo y en porcentaje, como se ve en la Figura 22, que muestra un correo recibido en la bandeja de entrada del administrador en el proceso de pruebas del demostrador.

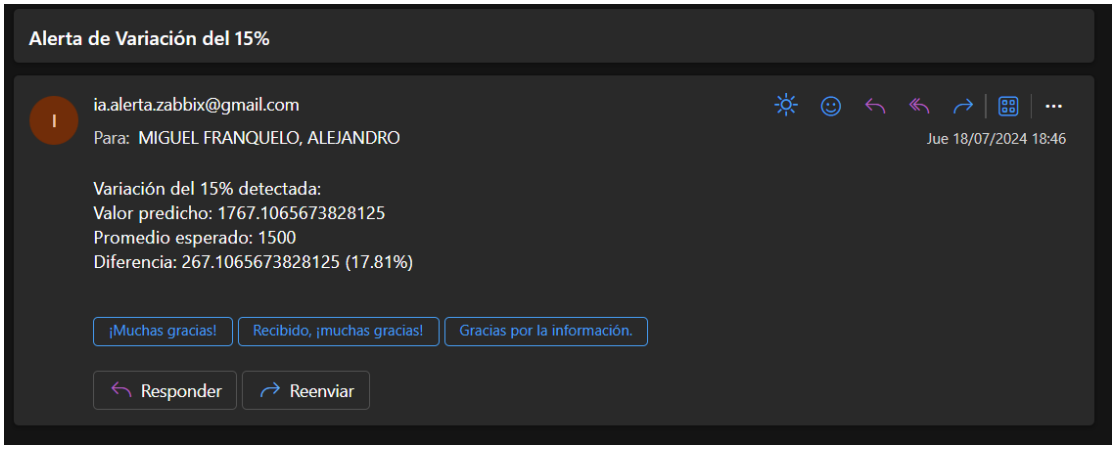

Figura 22: Alerta por correo electrónico, Fuente: propia.

Este script permite la gestión proactiva de los datos de red, notificando automáticamente a los administradores cuando se detectan anomalías significativas, lo que facilita la toma de decisiones rápidas para prevenir problemas de rendimiento. La lógica principal del script incluye la posibilidad de definir una hora manualmente, lo que permite adaptar la gestión a diferentes franjas horarias y contextos específicos de uso, que para el demostrador es muy importante para abrir la posibilidad de mostrar diferentes franjas horarias.

5. Resultados y análisis.

Este trabajo fin de grado pretende mostrar las capacidades que posee la integración de GNS3, Zabbix y TensorFlow en ámbito de la gestión de red, ya que, aparte de permitir gestionar la red en el momento actual, permite anticiparnos a los posibles problemas que puedan surgir. Esta integración pretende mejorar y facilitar la gestión de red a las personas encargadas de ello, liberándoles de trabajo y permitiendo la gestión proactiva de los problemas que puedan surgir, haciendo que estos se minimicen. Para mostrar sus capacidades se han realizado 3 casos de prueba modificando las situaciones del entorno y viendo cómo responde la integración en los posibles casos.

5.1. CASO DE PRUEBA 1

En este caso se ha determinado un uso esperado de la red, el modelo de inteligencia artificial debe realizar las predicciones y no debe generar ninguna alerta ya que el funcionamiento es el esperado.

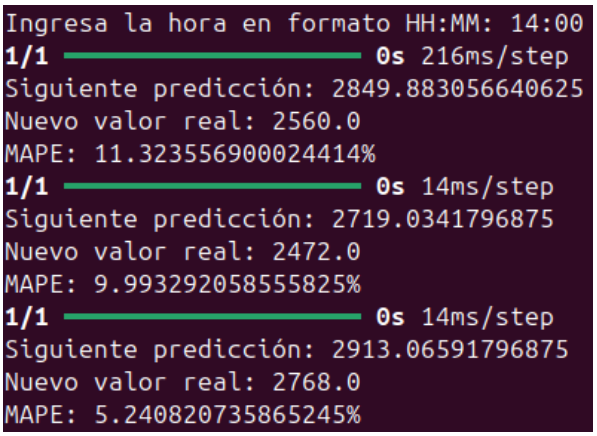

Figura 23: Predicciones de la prueba con tráfico esperado, Fuente: propia.

Para realizar las predicciones el script se encarga de obtener los últimos 10 datos y se utiliza el modelo para obtener las predicciones. Esta se compara con el valor promedio esperado y de no cumplir una variación superior al 15% se genera una alerta. Tras ello, se obtiene el ultimo valor generado en la red a través del gestor de red y este se compara con la predicción para saber el error cometido. Este proceso se repite en cada una de las predicciones realizadas.

Como se puede ver en la Figura 23 la hora en la que se está simulando la situación son las 14:00, en la Figura 20 se puede observar que le tráfico promedio esperado a esa hora es de unos 2600 bits/s. La diferencia entre este promedio y las tres predicciones realizadas por el modelo de IA no es superior al 15% en ninguna de las predicciones, por ello no se generan alertas, ya que no se considera una variación notoria. El error cometido (MAPE) en las predicciones está en torno a un 10% aunque se puede ver reducido en alguna situación.

5.2. CASO DE PRUEBA 2

En este caso se ha determinado un uso inferior de la red al esperado, el modelo de inteligencia artificial deberá realizar las predicciones al igual que en caso anterior, pero a diferencia de este se deben generar las alertas, ya que el funcionamiento no es el previsto.

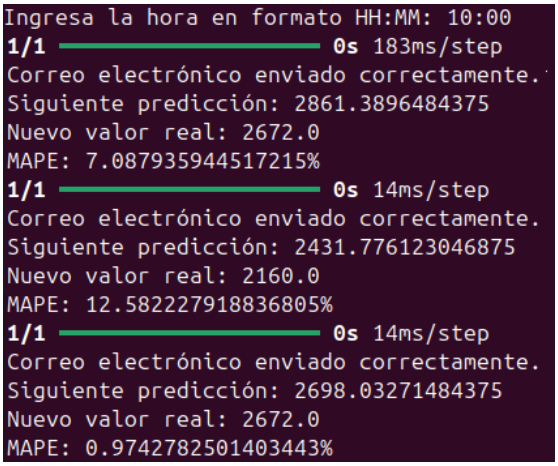

Figura 24: Predicciones de la prueba con tráfico por debajo de lo esperado, Fuente: propia.

Como se puede determinar en la Figura 24la hora en la que se está simulando la situación son las 10:00, en la Figura 20 se puede observar que le tráfico promedio esperado a esa hora es de unos 5600 bits/s. La diferencia entre este promedio y las tres predicciones realizadas por el modelo de IA son superiores al 15%, por ello se generan las alertas para cada una de estas. El error cometido (MAPE) en las predicciones está en torno a un 10% al igual que en el caso anterior, y de la misma manera se puede ver reducido en alguna situación.

Mientras se realizaban las predicciones como se observa en la Figura 24 aparece el mensaje “Correo electrónico enviado correctamente”, esto indica que la alerta ha sido enviada al correo electrónico del gestor de la red. El mensaje que recibe el administrador se puede observar en la Figura 25 donde se detallan los parámetros relevantes para analizar por la persona encargada. Las variables mostradas en el correo de alerta son la predicción realizada por el modelo, el promedio esperado y la diferencia en bits/s y porcentaje que se ha detectado para poder determinar la gravedad de la situación. En esta situación los datos muestran una variación por encima de lo esperado.

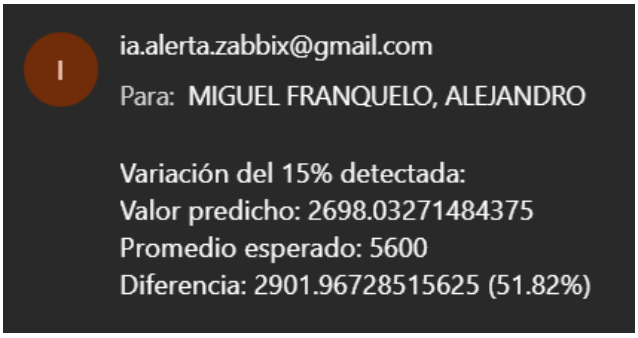

Figura 25: Alerta superación del umbral por arriba, Fuente: propia.

5.3. CASO DE PRUEBA 3

En este caso se ha determinado un uso superior de la red al esperado, el modelo de inteligencia artificial deberá realizar las predicciones al igual que en los casos

anteriores, pero a diferencia del primero y al igual que el segundo se tiene que generar las alertas ya que el funcionamiento no es el previsto.

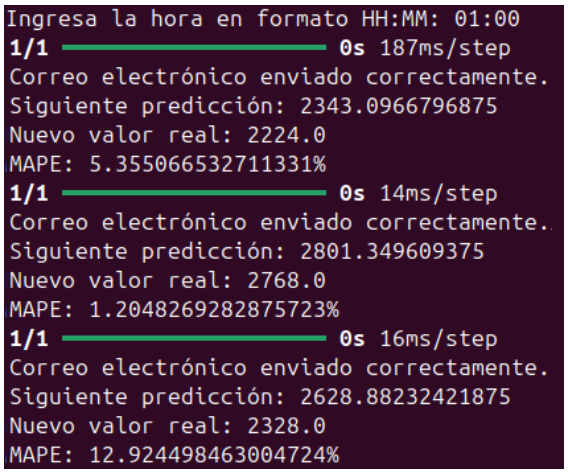

Figura 26: Predicciones de la prueba con tráfico por encima de lo esperado, Fuente: propia.

Como se puede ver en la Figura 26 la hora en la que se está simulando la situación son las 01:00, en la Figura 20 se puede observar que le tráfico promedio esperado a esa hora es de unos 1500 bits/s. La diferencia entre este promedio y las tres predicciones realizadas por el modelo de IA son superiores al 15% por ello se generan las alertas para cada una de estas. El error cometido (MAPE) en las predicciones está en torno a un 10% al igual que en los casos anteriores y de la misma manera que en los otros casos se puede ver reducido en alguna situación.

Mientras se realizaban las predicciones como se observa en la Figura 26 aparece el mensaje “Correo electrónico enviado correctamente”, esto indica que la alerta ha sido enviada al correo electrónico del gestor de la red. El mensaje que recibe el administrador se puede observar en la Figura 27 donde se detallan los parámetros relevantes para analizar por la persona encargada. Las variables mostradas en el correo de alerta son la predicción realizada por el modelo, el promedio esperado y la diferencia en bits/s y porcentaje que se ha detectado para poder determinar la gravedad de la situación. En esta situación los datos muestran una variación por debajo de lo esperado.

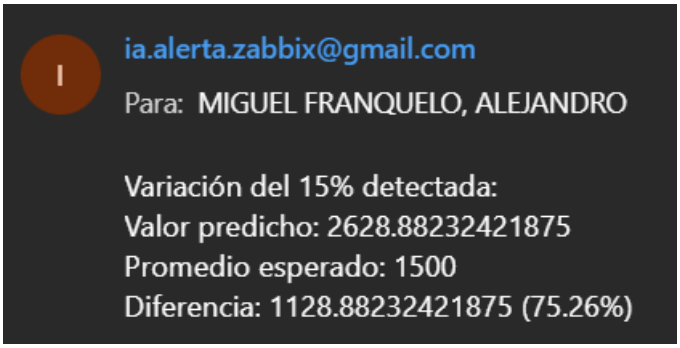

Figura 27: Alerta de superación del umbral por debajo, Fuente: propia.

Como se ha podido ver en estos tres casos, la integración de estas tres herramientas realiza un correcto funcionamiento, realizando las predicciones rondando un 10% de error entre la predicción y el valor real, viéndose en ciertos casos reducido al 1%. Las alertas funcionan de manera correcta enviando los parámetros y avisando al gestor de la situación, demostrando la utilidad que posee esta integración.

6. Conclusiones y líneas futuras.

Como se ha mostrado a lo largo de este trabajo de fin de grado, la integración de GNS3 como simulador de redes, de Zabbix como gestor de red y TensorFlow como plataforma de desarrollo de modelos de inteligencia artificial, ha permitido realizar de manera satisfactoria un demostrador que realiza predicciones de los valores de una métrica, y generar alertas de manera automática. En este demostrador se puede observar como la inteligencia artificial es una herramienta que puede complementar varias actividades.

GNS3 ha permitido que se generaran las variaciones de tráfico sin problema alguno, lo que ha facilitado la recopilación de variedad de datos para entrenar la inteligencia artificial. Además, la obtención de los datos en tiempo real ha sido fluida y de calidad, lo que ayuda mucho a la generación de las predicciones con del modelo creado por TensorFlow.

Teniendo la topología funcionando de manera correcta, Zabbix se ha configurado de manera satisfactoria gestionando las métricas requeridas en los dispositivos simulados. Con esta herramienta se pudo generar un conjunto de datos que formaran el histórico que más delante se requiere. La recopilación de estos datos se hizo de manera que se asegurase su calidad, esto hizo que se reflejara en el rendimiento del modelo de inteligencia artificial.

La API desarrollada por Zabbix, hizo que la integración se volviera más sencilla que si se hubiera realizado con otras plataformas de gestión de red, por ello gracias a su fácil comprensión ha sido una ayuda notoria en todo el proceso de integración de estas herramientas en el demostrador.

El proceso de obtención de los datos y el entrenamiento del modelo permitió identificar los patrones del histórico de datos de la métrica que se trató, en este caso el tráfico entrante por la interfaz f0/0 del router. El modelo aprendió los patrones y fue capaz de generar las predicciones de manera precisa y en tiempo real.

Una parte más sencilla pero no menos importante fue la generación de las alertas en caso de que superasen el umbral definido anteriormente por el gestor. En este demostrador el umbral está definido en el script a una variación mayor al 15% entre

los valore predichos y los predefinidos. Cuando se detecta la situación, se activa la alerta.

El sistema de alertas está integrado con un servidor de correo electrónico que envía las notificaciones a los administradores de la red en tiempo real. Como se ha visto anteriormente estos correos electrónicos proporcionan la información necesaria al gestor de la red para poder valora el error y así poder actuar en base a ello tomando las medidas oportunas.

El análisis de los resultados obtenidos con esta integración proporciona una mejora significativa en la repuesta ante posibles problemas que puedan ocurrir en la red. Durante las diferentes pruebas que se realizaron con el demostrador, se realizaron variaciones en el tráfico, el modelo de inteligencia artificial fue capaz de predecir de manera correcta estas situaciones, lo que lleva a pensar que es una herramienta útil que anticipa de forma segura la posibilidad de errores.

El modelo de TensorFlow mostró una buena precisión ya que el valor predicho entraba dentro de los valores generados por la métrica. Y las alertas generadas con las predicciones cuando superaban la variación máxima permitida, se enviaron en el momento requerido proporcionando los datos anteriormente mencionados para la evitar la degradación del servicio.

Como se ha observado esta integración proporciona innumerables beneficios como son los siguientes.

La prevención proactiva de problemas ya que el sistema provee la capacidad de predecir los comportamientos anómalos, y la generación de alertas permite la anticipación al problema, lo que permite reducir la respuesta ante fallos.

La optimización de la gestión debido a la combinación de herramientas que posee esta integración mejora la gestión tradicional complementándola, por ello ahora no solo se hace frente a os problemas actuales, sino también a posibles futuros y así hacer que la red este en mejores condiciones.

La automatización y el aumento de la eficiencia ya que las alertas generadas de manera automática permiten que los gestores de red tengan más tiempo disponible para otras tareas.

En conclusión, la integración propuesta ha demostrado ser una solución que permite solucionar de manera eficaz la gestión predictiva de la red. Haciendo que se proporcione un enfoque más avanzado en la gestión de las infraestructuras. Este sistema ha demostrado la capacidad que poseen estas tres herramientas integradas y lo resultados que podrían llegar a tener en un ámbito de operación serias notables.

Como líneas futuras de investigación en reacción a la gestión de redes se presentan a continuación algunas ideas.

Con el fin de seguir mejorando la integración de la inteligencia artificial en la gestión de redes, podría centrarse en realizar predicciones de fallos multidimensionales, analizando varias métricas de manera simultánea y a través de la inteligencia artificial encontrar patrones y tendencias que consigan predecir los fallos en la red y no solo con una métrica.

Otra dirección que se podría tomar es la optimización automática de los recursos de la red mediante la inteligencia artificial. Esta IA sería capaz de asignar recursos en tiempo real, optimizando el rendimiento y la eficiencia de la red sin necesidad de intervención manual. Lo cual sería interesante en ciertos entornos donde se ofrezcan servicios con alta variabilidad.

La utilización de la inteligencia artificial para automatizar las configuraciones en la red, la IA aprendería las configuraciones que tienen los dispositivos y cómo se comportan en las diferentes situaciones, así identificaría cual es la más correcta para cada situación y las utilizaría en base a ello.

En el ámbito de la seguridad se podría utilizar para aprender patrones de tráfico que se generen cuando hay amenaza y detectarlos, además podría incluirse que comprobara si todos los dispositivos están actualizados para que no haya vulnerabilidades.

Como ultima idea, se podría desarrollar un asistente virtual con inteligencia artificial que haga de soporte, con el fin de recomendar en tiempo real modificaciones para así mejorar los tiempos de respuesta y la calidad de las soluciones que pueda sugerir.

Estas líneas futuras no solo permitirían aprovechar al máximo las capacidades de la IA en la gestión de redes, sino que también abrirían nuevas posibilidades para innovar en la eficiencia operativa, seguridad y sostenibilidad de las infraestructuras tecnológicas.

Referencias.

[1] «Gestión de red», *Hpe.com*. [En línea]. Disponible en: https://www.hpe.com/es/es/what-is/network-management.html. [Accedido: 23-sep-2024].

[2] B. de CEUPE, «Características de la inteligencia artificial: ¿Cuáles son?», *Ceupe*. [En línea]. Disponible en: https://www.ceupe.com/blog/caracteristicas-de-la-inteligencia-artificial.html. [Accedido: 23-sep-2024].

[3] B. Nancharaiah, K. C. Ravi, A. K. Srivastava, K. Arunkumar, S. T. Siddiqui, y M. R. Arun, «Analysis of data science and AI-enabled 6G wireless communication networks», *Radioelectron. Commun. Syst.*, vol. 66, n.o 5, pp. 223-232, 2023.

[4] J. J. Estévez-Pereira, D. Fernández, y F. J. Novoa, «Network anomaly detection using machine learning techniques», en *3rd XoveTIC Conference*, 2020, vol. 7, p. 8.

[5] R. Alroobaea, R. Arul, S. Rubaiee, F. S. Alharithi, U. Tariq, y X. Fan, «AI-assisted bio-inspired algorithm for secure IoT communication networks», *Cluster Comput.*, vol. 25, n.o 3, pp. 1805-1816, 2022.

[6] S. Wang, J. F. Balarezo, S. Kandeepan, A. Al-Hourani, K. G. Chavez, y B. Rubinstein, «Machine learning in network anomaly detection: A survey», *IEEE Access*, vol. 9, pp. 152379-152396, 2021.

[7] F. Jameel, M. A. Jamshed, Z. Chang, R. Jantti, y H. Pervaiz, «Low latency ambient backscatter communications with deep Q-learning for beyond 5G applications», en *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, vol. 518, pp. 1-6.

[8] A. O. Salau y M. M. Beyene, «Software defined networking based network traffic classification using machine learning techniques», *Sci. Rep.*, vol. 14, n.o 1, 2024.

[9] L. Blanco *et al.*, «AI-driven framework for scalable management of network slices», *IEEE Commun. Mag.*, vol. 61, n.o 11, pp. 216-222, 2023.

[10] S. Zeb, M. A. Rathore, S. A. Hassan, S. Raza, K. Dev, y G. Fortino, «Toward AI-enabled NextG networks with edge intelligence-assisted microservice orchestration», *IEEE Wirel. Commun.*, vol. 30, n.o 3, pp. 148-156, 2023.

[11] F. Bargarai, A. Abdulazeez, V. Tiryaki, y D. Zeebaree, «Management of wireless communication systems using artificial intelligence-based software Defined Radio», pp. 107-133, 2020.

[12] ZABBIX, «BASELINE MONITORING AND ANOMALY DETECTION», *Zabbix.com*. [En línea]. Disponible en: https://assets.zabbix.com/files/events/meetup_20220427/Baseline_monitoring_anomaly_detection.pdf. [Accedido: 13-oct-2024].

[13] J. Jiménez, «Simuladores para virtualizar redes y aprender routing y switching», *RedesZone*, 11-mar-2022. [En línea]. Disponible en: https://www.redeszone.net/tutoriales/redes-cable/programas-simular-red/. [Accedido: 03-oct-2024].

[14] J. Torres, «Herramientas de software de simulación para redes de comunicaciones», UNIVERSIDAD NACIONAL DE LA PLATA, La Plata, Argentina., 2015.

[15] «Documentación», *Gns3.com*. [En línea]. Disponible en: https://docs.gns3.com/docs/. [Accedido: 03-oct-2024].

[16] Wikipedia contributors, «Packet Tracer», *Wikipedia, The Free Encyclopedia*, 26-sep-2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Packet_Tracer&oldid=1247816584.

[17] «Cisco Packet Tracer», *Netacad.com*. [En línea]. Disponible en: https://www.netacad.com/cisco-packet-tracer. [Accedido: 03-oct-2024].

[18] D. Leblanc, «Cisco Packet Tracer (64-bit)», *filehorse.com*, 29-may-2024. [En línea]. Disponible en: https://www.filehorse.com/download-cisco-packet-tracer-64/. [Accedido: 03-oct-2024].

[19] «¿Qué es la gestión de las redes?», *Redhat.com*. [En línea]. Disponible en: https://www.redhat.com/es/topics/management/what-is-network-management. [Accedido: 03-oct-2024].

[20] Orbit Consulting Group, «Redes de IT, deja que la IA trabaje por ti en su gestión», *Orbit Consulting*, 13-feb-2024. [En línea]. Disponible en: https://www.orbit.es/deja-que-la-ia-trabaje-por-ti-en-la-gestion-de-redes-it/. [Accedido: 03-oct-2024].

[21] «About Zabbix LLC», *Zabbix.com*. [En línea]. Disponible en: https://www.zabbix.com/about. [Accedido: 03-oct-2024].

[22] P. G. Miguel, «Zabbix monitoring software in-depth review», *The CTO Club*, 21-sep-2023. [En línea]. Disponible en: https://thectoclub.com/tools/zabbix-review/. [Accedido: 03-oct-2024].

[23] «LibreNMS Documentation», *Librenms.org*. [En línea]. Disponible en: https://docs.librenms.org/Support/Features/. [Accedido: 03-oct-2024].

[24] S. Durand, «LibreNMS: What is it and how does it work?», *Linux Guides, Tips and Tutorials | LinuxScrew*, 03-oct-2022. [En línea]. Disponible en: https://www.linuxscrew.com/librenms-what-is-it-and-how-does-it-work?utm_content=cmp-true. [Accedido: 03-oct-2024].

[25] «About», *Nagios Open Source*. [En línea]. Disponible en: https://www.nagios.org/about/. [Accedido: 03-oct-2024].

[26] «Nagios», *IONOS Digital Guide*, 12-oct-2023. [En línea]. Disponible en: https://www.ionos.es/digitalguide/servidores/herramientas/nagios-todos-los-procesos-de-red-a-tu-alcance/. [Accedido: 03-oct-2024].

[27] «¿Qué es la Inteligencia Artificial (IA)?», *Ibm.com*, 21-ago-2024. .

[28] J. Rojas, «Qué es y cómo funciona la Inteligencia Artificial», *Telefónica*, 25-nov-2022. .

[29] «Documentos», *TensorFlow*. [En línea]. Disponible en: https://www.tensorflow.org/about?hl=es-419. [Accedido: 03-oct-2024].

[30] K. Velusamy, «A basic introduction to TensorFlow», *Whizlabs Blog*, 15-nov-2023. .

[31] L. González, «¿Qué es TensorFlow? ¿Cómo funciona?», *Aprende IA*, 01-jun-2021. [En línea]. Disponible en: https://aprendeia.com/que-es-tensorflow-como-funciona/. [Accedido: 03-oct-2024].

[32] «Features», *PyTorch*. [En línea]. Disponible en: https://pytorch.org/features/. [Accedido: 03-oct-2024].

[33] «¿Conoces PyTorch? La herramienta de open source con la que puedes crear redes neuronales», *ITELLIGENT*, 26-ago-2020. [En línea]. Disponible en: https://itelligent.es/conoces-pytorch-herramienta-open-source-la-puedes-crear-redes-neuronales/. [Accedido: 03-oct-2024].

Anexo I: host.py

```
from pyzabbix import ZabbixAPI

# URL de la instancia de Zabbix
zabbix_server = 'http://localhost/zabbix'

# Credenciales de Zabbix
username = 'Admin'
password = 'zabbix'

try:
  # Conectar a la API de Zabbix
  zapi = ZabbixAPI(zabbix_server)
  zapi.login(username, password)

  # Obtener el token de autenticación
  auth_token = zapi.auth

  # Imprimir el token de autenticación
  print(f'Token de autenticación: {auth_token}')

  # Realizar una solicitud a la API de Zabbix para obtener la lista de
hosts
  hosts = zapi.host.get(output=['hostid', 'name'])

  # Imprimir los hosts obtenidos
  for host in hosts:
    print(f'Host ID: {host['hostid']}, Host Name: {host['name']}')

except Exception as e:
  print(f'Error al conectar con la API de Zabbix: {e}')
```

Anexo II: ítem_id.py

```
from pyzabbix import ZabbixAPI
import pandas as pd

# Configuración de Zabbix
zabbix_server = 'http://localhost/zabbix' # URL de la instancia de
Zabbix
username = 'Admin' # Usuario de Zabbix
password = 'zabbix' # Contraseña de Zabbix

# ID del host del cual se desea listar los items
host_id = '10630' #El host ID deseado Router1

try:
  # Conectar a la API de Zabbix
  zapi = ZabbixAPI(zabbix_server)
  zapi.login(username, password)

  # Obtener los items del host
  items = zapi.item.get(hostids=host_id, output=['itemid', 'name',
'key_', 'lastvalue'])

  if items:
    # Convertir los datos a un DataFrame de pandas
    df = pd.DataFrame(items)

    # Guardar los datos en un archivo CSV
    csv_filename = f'host_{host_id}_items.csv'
    df.to_csv(csv_filename, index=False)

    print(f'Datos de los items extraídos y guardados en
{csv_filename}')
  else:
    print(f'No se encontraron items para el host con ID {host_id}')

except Exception as e:
  print(f'Error al conectar con la API de Zabbix: {e}')
```

Anexo III: extract_item_history.py

```
from pyzabbix import ZabbixAPI
import pandas as pd
from datetime import datetime

# Configuración de Zabbix
zabbix_server = 'http://localhost/zabbix' # URL de tu instancia de
Zabbix
username = 'Admin' # Usuario de Zabbix
password = 'zabbix' # Contraseña de Zabbix

# ID del item que se desea extraer
item_id = '47242' # Item ID deseado

# Rango de tiempo para la extracción
time_from = int((datetime.now() - pd.Timedelta(hours=5)).timestamp())
time_till = int(datetime.now().timestamp())

# Número máximo de muestras a extraer
max_samples = 1000

try:
  # Conectar a la API de Zabbix
  zapi = ZabbixAPI(zabbix_server)
  zapi.login(username, password)

  # Determinar el tipo de datos del item (historia tipo 0 para float, 1
para string, 3 para log, etc.)
  item_info = zapi.item.get(itemids=item_id, output=['value_type'])
  if not item_info:
    print(f'No se encontró información para el item con ID {item_id}')
    exit()

  value_type = int(item_info[0]['value_type'])

  # Obtener datos históricos del item con límite de muestras
  history = zapi.history.get(itemids=item_id, time_from=time_from,
time_till=time_till, output='extend', history=value_type,
limit=max_samples)

  if history:
    # Convertir los datos a un DataFrame de pandas
    df = pd.DataFrame(history)
```

```
    # Añadir la columna 'itemid' con el valor del item_id
    df['itemid'] = item_id

    # Seleccionar solo las columnas 'itemid' y 'value'
    df = df[['itemid', 'value']]

    # Guardar los datos en un archivo CSV
    csv_filename = f'item_{item_id}_history_new.csv'
    df.to_csv(csv_filename, index=False)

    print(f’Datos históricos del item extraídos y guardados en
{csv_filename}’)
  else:
    print(f’No se encontraron datos históricos para el item con ID
{item_id}’)

except Exception as e:
  print(f’Error al conectar con la API de Zabbix: {e}’)
```

Anexo IV: generation_model.py

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from TensorFlow.keras.models import Sequential
from TensorFlow.keras.layers import LSTM, Dense, Dropout
from TensorFlow.keras.optimizers import Adam
import joblib

# Función para cargar y preparar los datos
def cargar_datos(file_path):
  df = pd.read_csv(file_path)
  return df

# Función para preprocesar los datos
def preprocesar_datos(df, time_steps):
  # Escalado de los datos
  scaler = MinMaxScaler(feature_range=(0, 1))
  scaled_data = scaler.fit_transform(df['value'].values.reshape(-1, 1))

  # Preparar los datos para LSTM
  X, y = [], []
  for i in range(time_steps, len(df)):
    X.append(scaled_data[i-time_steps:i, 0])
    y.append(scaled_data[i, 0])
  X, y = np.array(X), np.array(y)

  # Reshape para LSTM [samples, time steps, features]
  X = np.reshape(X, (X.shape[0], X.shape[1], 1))

  return X, y, scaler

# Función para crear y entrenar el modelo LSTM
def entrenar_modelo(X_train, y_train, epochs=50, batch_size=1):
  model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X_train.shape[1],
X_train.shape[2])),
    Dropout(0.2),
    LSTM(50),
    Dropout(0.2),
    Dense(1)
  ])
```

```
  model.compile(optimizer=Adam(learning_rate=0.001),
loss='mean_squared_error')
  history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, verbose=2)
  return model, history

# Función principal para cargar datos, entrenar modelo y guardar el
modelo entrenado
def entrenar_y_guardar_modelo(file_path, time_steps=10, epochs=50,
batch_size=1):
  # Cargar y preprocesar los datos desde el archivo CSV
  df = cargar_datos(file_path)
  X, y, scaler = preprocesar_datos(df, time_steps)

  # Entrenar el modelo LSTM
  lstm_model, history = entrenar_modelo(X, y, epochs=epochs,
batch_size=batch_size)

  # Guardar el modelo y el scaler
  lstm_model.save('lstm_model.keras')
  scaler_filename = 'scaler.pkl'
  joblib.dump(scaler, scaler_filename)

  print('Modelo LSTM entrenado y guardado correctamente.')

# Llamar a la función principal para entrenar y guardar el modelo
if __name__ == '__main__':
  file_path = 'item_47242_history_clock.csv' # Ruta al archivo CSV de
datos
  entrenar_y_guardar_modelo(file_path)
```

Anexo V: prediction_alert.py

```
from pyzabbix import ZabbixAPI
import numpy as np
import joblib
from datetime import datetime, time as dt_time
from TensorFlow.keras.models import load_model
from sklearn.metrics import mean_absolute_percentage_error
import time
import smtplib
from email.mime.text import MIMEText

# Configuración de Zabbix
zabbix_server = 'http://localhost/zabbix'
username = 'Admin'
password = 'zabbix'

# ID del item que deseas extraer
item_id = '47242'

# Configuración del correo electrónico
smtp_server = 'smtp.gmail.com'
smtp_port = 587
smtp_user = 'ia.alerta.zabbix@gmail.com'
smtp_password = 'dmdz gfis nfcd nfbu'
recipient_email = 'amf354@alumnos.unican.es'

# Función para enviar correo electrónico con mensaje multilínea y
porcentaje de variación
def enviar_correo(subject, body_lines):
  # Construir el cuerpo del mensaje
  body = '\n'.join(body_lines) # Unir las líneas con saltos de línea

  # Crear el mensaje MIME
  msg = MIMEText(body)
  msg['Subject'] = subject
  msg['From'] = smtp_user
  msg['To'] = recipient_email

  try:
    # Establecer conexión con el servidor SMTP
    with smtplib.SMTP(smtp_server, smtp_port) as server:
      server.starttls()
      server.login(smtp_user, smtp_password)
```

```
      server.sendmail(smtp_user, recipient_email, msg.as_string())
      print('Correo electrónico enviado correctamente.')
  except Exception as e:
    print(f'Error al enviar el correo electrónico: {e}')

# Función para obtener datos históricos de Zabbix
def obtener_datos_historicos_zabbix(item_id, time_steps):
  try:
    zapi = ZabbixAPI(zabbix_server)
    zapi.login(username, password)

    item_info = zapi.item.get(itemids=item_id, output=['value_type'])
    if not item_info:
      raise ValueError(f'No se encontró información para el item con ID
{item_id}')

    value_type = int(item_info[0]['value_type'])

    time_till = int(datetime.now().timestamp())
    time_from = time_till - (time_steps * 60)

    history = zapi.history.get(itemids=item_id, time_from=time_from,
time_till=time_till, output='extend', history=value_type,
sortfield='clock', sortorder='DESC', limit=time_steps)

    if history:
      values = [float(h['value']) for h in history]
      values.reverse()
      return values
    else:
      raise ValueError(f'No se encontraron datos históricos para el
item con ID {item_id}')

  except Exception as e:
    print(f'Error al obtener datos de Zabbix: {e}')
    raise

# Función para obtener el último valor de Zabbix
def obtener_ultimo_valor_zabbix():
  try:
    zapi = ZabbixAPI(zabbix_server)
    zapi.login(username, password)

    item_info = zapi.item.get(itemids=item_id, output=['value_type'])
```

```
    if not item_info:
      raise ValueError(f'No se encontró información para el item con ID
{item_id}')

    value_type = int(item_info[0]['value_type'])

    history = zapi.history.get(itemids=item_id, output='extend',
history=value_type, limit=1, sortfield='clock', sortorder='DESC')

    if history:
      last_value = float(history[0]['value'])
      return last_value
    else:
      raise ValueError(f'No se pudo obtener el último valor de Zabbix
para el item con ID {item_id}')

  except Exception as e:
    print(f'Error al obtener datos de Zabbix: {e}')
    raise

# Función para cargar el modelo LSTM y el scaler
def cargar_modelo_y_scaler(model_file='lstm_model.keras',
scaler_file='scaler.pkl'):
  model = load_model(model_file)
  scaler = joblib.load(scaler_file)
  return model, scaler

# Función para predecir el siguiente valor con el modelo LSTM
def predecir_siguiente_valor(model, scaler, data, time_steps):
  scaled_data = scaler.transform(np.array(data).reshape(-1, 1))
  X_test = np.array([scaled_data[-time_steps:]])
  X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
  next_value_scaled = model.predict(X_test)
  next_value = scaler.inverse_transform(next_value_scaled.reshape(-1,
1))
  return next_value[0][0]

# Función para obtener el promedio esperado según la hora manual
ingresada
def obtener_promedio_segun_hora_manual(hora_manual):
  try:
    current_time = datetime.strptime(hora_manual, '%H:%M').time()
```

```
    if current_time >= dt_time(hour=20) or current_time <
dt_time(hour=8):
      return 1500
    elif current_time >= dt_time(hour=8) and current_time <
dt_time(hour=13):
      return 5600
    elif current_time >= dt_time(hour=13) and current_time <
dt_time(hour=15):
      return 2600
    elif current_time >= dt_time(hour=15) and current_time <
dt_time(hour=20):
      return 4600
    else:
      return None

  except ValueError:
    print('Formato de hora incorrecto. Utiliza el formato HH:MM.')
    return None

# Función para calcular el MAPE
def calcular_mape(real_value, predicted_value):
  return mean_absolute_percentage_error([real_value],
[predicted_value]) * 100

# Función principal para realizar predicciones en tiempo real con hora
manual
def realizar_predicciones_en_tiempo_real_con_hora_manual(model, scaler,
item_id, time_steps, hora_manual, max_predictions=3):
  data = obtener_datos_historicos_zabbix(item_id, time_steps)

  for _ in range(max_predictions):
    try:
      new_real_value = obtener_ultimo_valor_zabbix()
      data.append(new_real_value)

      if len(data) >= time_steps:
        next_value_predicted = predecir_siguiente_valor(model, scaler,
data, time_steps)
        mape = calcular_mape(new_real_value, next_value_predicted)
        promedio_esperado =
obtener_promedio_segun_hora_manual(hora_manual)

        if promedio_esperado is not None:
          variacion_maxima = promedio_esperado * 0.15
```

```
            diferencia = abs(next_value_predicted - promedio_esperado)

            if diferencia > variacion_maxima:
              # Construir las líneas del cuerpo del correo electrónico
              body_lines = [
                f'Variación del 15% detectada:',
                f'Valor predicho: {next_value_predicted}',
                f'Promedio esperado: {promedio_esperado}',
                f'Diferencia: {diferencia} ({(diferencia /
promedio_esperado) * 100:.2f}%)'
              ]

              # Enviar correo electrónico
              enviar_correo('Alerta de Variación del 15%', body_lines)

          print(f'Siguiente predicción: {next_value_predicted}')
          print(f'Nuevo valor real: {new_real_value}')
          print(f'MAPE: {mape}%')
        else:
          print(f'Recolección de datos insuficiente. Valores actuales:
{len(data)}/{time_steps}')

      except Exception as e:
        print(f'Error en la predicción: {e}')

      time.sleep(60)

# Lógica principal del script con entrada manual de hora
if __name__ == '__main__':
  lstm_model, scaler = cargar_modelo_y_scaler()
  time_steps = 10

  hora_manual = input('Ingresa la hora en formato HH:MM: ')
  realizar_predicciones_en_tiempo_real_con_hora_manual(lstm_model,
scaler, item_id, time_steps, hora_manual)
```