Implementación del gestor de monitoreo, mapas de topologías y estandarización de la configuración de los dispositivos y plataformas de la red de telecomunicaciones aeronáuticas a cargo de COCESNA, en Guatemala

Nicolas Urioste Rivera

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación del gestor de monitoreo, mapas de topologías y estandarización de la configuración de los dispositivos y plataformas de la red de telecomunicaciones aeronáuticas a cargo de COCESNA, en Guatemala

Trabajo de graduación presentado por Nicolas Urioste Rivera para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación del gestor de monitoreo, mapas de topologías y estandarización de la configuración de los dispositivos y plataformas de la red de telecomunicaciones aeronáuticas a cargo de COCESNA, en Guatemala

Trabajo de graduación presentado por Nicolas Urioste Rivera para optar al grado académico de Licenciado en Ingeniería Electrónica

Guatemala,

2025

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Agradezco, en primer lugar, a Dios, quien no solo me ha otorgado la fortaleza y la templanza necesarias para enfrentar los retos de este proyecto, sino que también me ha guiado en este camino. Gracias a las dificultades que ha puesto en mi vida, hoy soy el hombre que soy, fortalecido y moldeado por cada obstáculo superado.

A mi familia, en su totalidad, les expreso mi más sincero agradecimiento. Cada uno de ustedes ha jugado un papel fundamental en mi vida, brindándome su amor, su apoyo incondicional y los valores que me han guiado a lo largo de este camino. Su presencia, aunque a veces desde la distancia, ha sido una fuente constante de fuerza y motivación para seguir adelante.

A mi madre, le debo el aprendizaje más valioso: la bondad. De ella he recibido no solo el ejemplo de una vida íntegra y noble, sino que también la enseñanza de que el corazón puede mantenerse puro, sin maldad, incluso, en un mundo lleno de adversidades. Su amor incondicional y su capacidad para actuar con generosidad hacia los demás me han mostrado el camino hacia una vida guiada por la empatía y la compasión.

A mi padre, le agradezco profundamente por su apoyo incondicional a lo largo de mis estudios. Siempre ha estado ahí para motivarme a seguir adelante y a nunca rendirme, recordándome que en la vida uno debe luchar por lo que realmente desea. Sus palabras de aliento y su ejemplo de perseverancia han sido una fuente constante de inspiración para continuar en este camino.

A Marcelo y Natalia, mis amigos incondicionales, quienes han estado a mi lado sin importar la distancia que nos separe. Su amistad ha sido un pilar constante en mi vida, recordándome siempre que la verdadera conexión trasciende el espacio y el tiempo. A Matthew, quien ha sido mi compañero en tantas aventuras, agradezco profundamente por mostrarme el verdadero significado de la fuerza de voluntad y por acompañarme en cada paso del camino.

A Ronny Montenegro, Guillermo Cruz, Raúl Calo y Manuel Meneses, en COCESNA, les debo un agradecimiento profundo. A través de su orientación, paciencia y experiencia, he aprendido no solo sobre la disciplina técnica necesaria para este proyecto, sino también sobre la importancia del trabajo en equipo y el liderazgo. Su apoyo y confianza me han permitido desarrollar este trabajo con mayor seguridad y claridad. Gracias a ellos, he podido avanzar con firmeza y superar los desafíos que encontré en el camino.

A mi querido país, Guatemala, que ha sido mi hogar y mi inspiración. Aquí he encontrado no solo el entorno que me vio crecer, sino que también la motivación para esforzarme y contribuir con el conocimiento y las capacidades que he adquirido. Tu belleza, tu cultura y tu gente son un recordatorio constante de lo afortunado que soy de llamarte mi patria.

Finalmente, extiendo mi gratitud a todos aquellos que, de una forma u otra, han sido parte de este viaje. A mis amigos, compañeros y mentores, cuyas palabras de aliento y apoyo me han impulsado a seguir adelante. A todos los que, con su presencia, han contribuido a que este proyecto sea una realidad, les estaré siempre agradecido.

Y a ti, lector, por tomarte el tiempo de leer no solo mi proyecto, sino también estos agradecimientos. Espero sinceramente que haya sido de tu agrado y que, de alguna manera, encuentres valor en el trabajo que aquí presento.

Índice

Lista de figuras

Lista de cuadros

Resumen

El proyecto se centró en mejorar la gestión, mapeo de topologias y estandarización de la configuración de los nodos y plataformas para la red de telecomunicaciones, que transporta servicios aeronáuticos en el espacio aéreo guatemalteco, mediante la implementación del gestor de red SW NPM (SolarWinds Network Performance Monitor) y una DB (base de datos) montada en MS SQL (Microsoft SQL Server). La necesidad que cubrió este proyecto surgió del sistema de monitoreo de COCESNA (Corporación Centroamericana de Servicios de Navegación Aérea) Guatemala, que dependía de notificaciones manuales por parte de los usuarios cuando se presentaban fallas en la red, un método que no solo era ineficiente, sino que también comprometía la calidad y seguridad del servicio aeronáutico.

El objetivo general del proyecto fue configurar y optimizar SW NPM para facilitar la supervisión automatizada y eficiente de la red. Los objetivos específicos incluyeron la identificación precisa de discrepancias en la red, reducción de la intervención humana, escalabilidad del sistema, extracción y registro de datos críticos, generación automática de alertas en situaciones anómalas, notificación por correo en caso de fallos externos, y la creación de mapas de topología de la red actualizados.

El marco teórico abordó temas fundamentales como el protocolo SNMP (*Simple Network Management Protocol*), las DB, Docker para *dockerización* y API (*Application Programming Interface*), proporcionando un contexto técnico esencial para la implementación del gestor de red. Este proyecto no solo esperaba utilizar la licencia existente de SW NPM de manera efectiva, eficiente y organizada, sino también establecer un sistema proactivo de gestión de red que asegurara la operatividad continua y segura del espacio aéreo guatemalteco. La implementación de este sistema permitió a COCESNA, a través de la estación en Guatemala responder más rápidamente a los problemas de infraestructura de red, reducir el tiempo de inactividad y mejorar la experiencia del área técnica y operativa, alineándose con los estándares internacionales de seguridad, eficiencia en servicios aeronáuticos y un monitoreo con SLA (*Service Level Agreement*) con límite inferior de 99.8 %.

El proyecto evidenció una mejora significativa en la gestión y supervisión de la red de telecomunicaciones. La implementación de un sistema automatizado de monitoreo y notificación redujo de manera considerable la intervención humana, lo cual mejoró la eficiencia operativa y optimizó la capacidad de respuesta ante fallos, reduciendo el tiempo de respuesta en un 98.6 %. Además, la *dockerización* de la API para la generación de reportes automatizados contribuyó a la centralización y estandarización de los procesos, garantizando así una mayor consistencia y precisión en la información generada, también redujo el tiempo de redacción de dichos reportes en un 40 %. A pesar de estas mejoras, se identificaron limitaciones relacionadas con la dependencia del protocolo SMTP (*Simple Mail Transfer Protocol*) para la transmisión de alertas y con la capacidad del sistema para adaptarse a entornos más complejos, sugiriendo la necesidad de explorar nuevas tecnologías que potencien la eficiencia y escalabilidad del sistema de monitoreo.

En el futuro, se recomienda migrar hacia el protocolo SNMPv3, que proporciona mejoras significativas en términos de seguridad y privacidad, fortaleciendo la protección de la información de monitoreo. Asimismo, se plantea expandir el monitoreo para incluir dispositivos adicionales compatibles con SNMP, mejorando la visibilidad y el control sobre toda la infraestructura. También se sugiere el desarrollo de una nueva herramienta tecnológica utilizando Docker para monitorear específicamente el funcionamiento de los radioenlaces, lo cual permitiría una supervisión más detallada y efectiva de estos componentes críticos. Finalmente, se propone expandir los estándares de monitoreo implementados en Guatemala a toda la red de COCESNA en Centroamérica, logrando así una mayor cohesión y eficiencia en la gestión de la infraestructura de telecomunicaciones de la región.

Abstract

The project focused on improving the management, topology mapping, and standardization of the configuration of devices and platforms for the telecommunications network that carries aeronautical services in Guatemalan airspace, through the implementation of the SW NPM (SolarWinds Network Performance Monitor) Network Manager and a DB (data base) hosted on MS SQL (Microsoft SQL Server). The need addressed by this project arises from the current monitoring system of COCESNA (Corporación Centroamericana de Servicios de Navegación Aérea) Guatemala, which relied on manual notifications by users when network failures occured, a method that was not only inefficient but also compromised the quality and safety of aeronautical services.

The general objective of the project was to configure and optimize SW NPM to facilitate automated and efficient network supervision. Specific objectives included the precise identification of discrepancies in the network, reduction of human intervention, system scalability, extraction and logging of critical data, automatic generation of alerts during anomalous situations, email notifications in case of external failures, and the creation of an up-to-date network topology map.

The theoretical framework addressed fundamental topics such as the SNMP (Simple Network Management Protocol) protocol, databases, Docker for dockerization, and API (Application Programming Interface), providing essential technical context for the implementation of the Network Manager. This project aimed to effectively, efficiently, and in an organized manner, utilize the existing SW NPM license while establishing a proactive network management system that ensured the continuous and safe operation of Guatemalan airspace.

The implementation of this system enabled COCESNA, through the station in Guatemala, to respond more quickly to network infrastructure issues, reduce downtime, and improve the experience of the technical and operational areas, aligning with international standards for safety, efficiency in aeronautical services, and monitoring with an minimum SLA (Service Level Agreement) of 99.8 %.

The project demonstrated a significant improvement in the management and supervision of the telecommunications network. The implementation of an automated monitoring and notification system considerably reduced human intervention, which improved operational efficiency and optimized response capability in case of failures, reducing response time by 98.6 %. Furthermore, the dockerization of the API for automated report generation contributed to the centralization and standardization of processes, thus ensuring greater consistency and accuracy in the generated information, also reducing the time spent writing these reports by 40 %. Despite these improvements, limitations were identified related to the dependence on the SMTP (Simple Mail Transfer Protocol) protocol for alert transmission and the system's ability to adapt to more complex environments, suggesting the need to explore new technologies to enhance the efficiency and scalability of the monitoring system.

In the future, it is recommended to migrate to the SNMPv3 protocol, which provides significant improvements in terms of security and privacy, strengthening the protection of monitoring information. Additionally, it is proposed to expand monitoring to include additional SNMP-compatible devices, thereby improving visibility and control over the entire infrastructure. It is also suggested to develop a new technological tool using Docker to specifically monitor the operation of the radio link, which would enable more detailed and effective supervision of these critical components. Finally, it is proposed to expand the monitoring standards implemented in Guatemala to the entire COCESNA network in Central America, thereby achieving greater cohesion and efficiency in the management of the region's telecommunications infrastructure.

CAPÍTULO 1

Introducción

La gestión eficiente y confiable de las redes de telecomunicaciones constituye un pilar fundamental para garantizar la seguridad y la continuidad operativa de los servicios aeronáuticos en la región centroamericana. En el contexto de COCESNA (Corporación Centroamericana de Servicios de Navegación Aérea), el monitoreo de la infraestructura de telecomunicaciones enfrenta retos considerables debido a la falta de integración, estandarización y automatización de los procesos de supervisión. Estas carencias no solo afectan la eficiencia operativa, sino que también limitan la capacidad de respuesta ante incidentes críticos, poniendo en riesgo la seguridad del espacio aéreo.

En los últimos años, la demanda de sistemas de monitoreo robustos y estandarizados en las telecomunicaciones aeronáuticas se ha incrementado significativamente. Las redes de telecomunicaciones aeronáuticas son la columna vertebral para el control seguro y eficiente del tráfico aéreo, y su correcto funcionamiento es esencial para garantizar la seguridad y la continuidad operativa de las actividades aeronáuticas. La infraestructura de monitoreo actualmente implementada por COCESNA esta sujeta a mejoras en cuanto a la integración y automatización de procesos, lo cual dificulta la identificación oportuna de fallos y la implementación de acciones correctivas proactivas. Esto genera una brecha considerable entre el estado actual del monitoreo de la red y las mejores prácticas establecidas en la industria.

En este sentido, se ha identificado una brecha importante en el desarrollo e implementación de herramientas que permitan una integración completa y una automatización avanzada del monitoreo de redes aeronáuticas, especialmente en contextos operativos críticos como el de la gestión del tráfico aéreo. Aunque existen plataformas de monitoreo en el mercado que ofrecen soluciones generales, muy pocas de ellas han sido adaptadas específicamente para satisfacer las exigentes necesidades de la industria aeronáutica regional. Esto subraya la necesidad de desarrollar soluciones técnicamente avanzadas que permitan mejorar la eficiencia del monitoreo, la gestión de alertas y la respuesta ante incidentes en la red.

El presente proyecto tiene como objetivo principal implementar mejoras sustanciales en el monitoreo de la red de telecomunicaciones aeronáuticas de COCESNA mediante la actualización y optimización de la herramienta SW NPM, así como el desarrollo de una API montada en *docker* para la automatización de la generación de reportes. Esta solución permitirá a los operadores contar con información detallada en tiempo real, gestionar de manera proactiva las alertas de la red y optimizar la visibilidad del estado de la infraestructura. Adicionalmente, se buscará la estandarización de las configuraciones de los dispositivos en la red, asegurando la compatibilidad con los estándares internacionales y promoviendo una gestión más eficiente y segura. En conjunto, estas mejoras contribuirán a la creación de un entorno de monitoreo altamente confiable, tecnológicamente avanzado y alineado con las mejores prácticas de la industria aeronáutica, fortaleciendo así la seguridad y la eficiencia del espacio aéreo en la región.

CAPÍTULO 2

Antecedentes

2.1. Diagnóstico de la red de monitoreo en COCESNA

Por el crecimiento, la red de monitoreo de COCESNA se ha ido desactualizando y es sujeta de mejoras, lo que representa un desafío significativo para aumentar la eficiencia operativa y la seguridad de la red. El gestor de red actual, basado en SW NPM (SolarWinds Network Performance Monitor), no proporciona beneficios tangibles a los usuarios debido a su interfaz poco intuitiva e información desactualizada. Específicamente, toda la información relevante se concentra en una sola pestaña, lo que requiere tanto desplazamiento horizontal como vertical extenso para poder visualizarla completamente, dificultando así el acceso rápido y eficiente a los datos necesarios.

Además, la mayoría de las viñetas dentro de la interfaz del gestor no contienen la información correcta o están vacías, lo cual puede ser indicativo de problemas de configuración o de integración con los nodos. Este problema se agrava con el hecho de que la mayoría de los Nodos monitoreados a través de SNMP (*Simple Network Management Protocol*) están inactivos o no reportan datos útiles. Incluso entre los dispositivos activos, la información que se despliega frecuentemente incluye datos irrelevantes como interfaces que han sido desactivadas manualmente, lo cual puede generar confusión y dificultar la identificación y diagnóstico de fallas reales en la red.

De acuerdo a la situación de la gestión de la red, fue posible identificar oportunidades de mejora con respecto a la capacidad de respuesta proactiva de COCESNA ante incidentes. Con un enfoque optimizado en la supervisión y mantenimiento de la infraestructura, se puede reducir significativamente el riesgo de inactividad del sistema y evitar posibles interrupciones operativas, fortaleciendo así la seguridad y la confiabilidad de los servicios aeronáuticos. Mejorar las herramientas de monitoreo y gestión permitirá una operación más efectiva y el mantenimiento adecuado de la infraestructura crítica, alineándose con los estándares de la industria y promoviendo la eficiencia operativa. Estos problemas subrayan la necesidad crítica de revisar y mejorar la solución de gestión de red de COCESNA. La implementación de un sistema más moderno y eficiente no solo mejorará la capacidad de

monitoreo, sino que también facilitará una respuesta más rápida y precisa ante incidencias en la red, contribuyendo así a una operación más estable, segura y confiable del espacio aéreo nacional.

2.2. Escenesarios donde un gestor de red a reducido costos y aumento de la eficiencia energética

2.2.1. AT&T

Un ejemplo relevante que refleja el impacto positivo de la automatización y gestión avanzada en sistemas críticos es la implementación por parte de AT&T de una solución avanzada de gestión de energía y edificios, denominada *Energy and Building Management Solution* (EBMS). Esta solución integra datos de múltiples fuentes internas, incluyendo sensores de IoT y sistemas de gestión de edificios (*Building Management System*, BMS), para optimizar la eficiencia operativa y energética en sus instalaciones.

AT&T ha logrado monitorear y analizar datos de casi 800 edificios, abarcando aproximadamente 80 millones de pies cuadrados en 450 ciudades. El impacto de EBMS ha sido significativo, reduciendo los gastos anuales de energía de la empresa en un 10-20 %, prolongando la vida útil del equipo y mejorando la eficiencia operativa. Esta transformación de una gestión de instalaciones reactiva a una proactiva ha sido posible gracias al uso de reglas específicas que generan alertas cuando se detectan fallos en el equipo, indicando un funcionamiento ineficiente. Esta estrategia ha permitido un mantenimiento más proactivo, basado en información real, optimizando la gestión energética y la eficiencia de las operaciones [1].

2.2.2. Sony Online Entertainment

Sony Online Entertainment, una división de Sony Corp, es conocida por sus juegos multijugador masivos en línea como EverQuest. Sony enfrentó el desafío de gestionar más de 1,500 servidores para soportar hasta 100,000 jugadores simultáneamente en tres continentes. La infraestructura necesaria para mantener una experiencia de juego sin interrupciones y en tiempo real es monumental. Los desafíos específicos incluían la necesidad de monitorear constantemente el rendimiento del servidor y la red, detectar y resolver problemas en tiempo real, y manejar la carga variable y los picos de tráfico que son comunes en los juegos en línea.

Para abordar estos desafíos, Sony implementó una solución de monitoreo de red personalizada utilizando NetVigil de Fidelia Inc. NetVigil ofreció la flexibilidad necesaria para integrarse con las herramientas personalizadas de Sony y proporcionar una visión integral del rendimiento tanto de los dispositivos de *host* como de red. La solución permitió a Sony tener un control detallado y en tiempo real sobre su infraestructura, mejorando significativamente la eficiencia operativa. Esta implementación resultó en un sistema de gestión de red altamente eficiente y rentable, asegurando una experiencia de juego óptima para millones de usuarios en todo el mundo. Los beneficios incluyeron una reducción en el tiempo de inactividad del servidor, una mejor gestión de la capacidad y un rendimiento general mejorado del sistema [2].

CAPÍTULO 3

Justificación

En COCESNA Guatemala, el monitoreo de la red es actualmente un proceso complejo y muy demandante. La organización cuenta con un gestor de red, el cual puede mejorarse y optimizarse; el método predominante para identificar fallas en la red depende de que los usuarios notifiquen a la central cuando su servicio deja de funcionar. Este enfoque reactivo no solo retrasa la respuesta a incidentes, sino que también afecta la eficiencia y la calidad del servicio aeronáutico ofrecido [3].

Adicionalmente, la ausencia de una DB (base de datos) organizada para identificar con mayor facilidad la IP perteneciente al nodos, su ubicación y el modelo, complica el proceso de cambio de un equipo por otro y dificulta que toda el área operativa esté en sintonía con respecto a los elementos de configuración y de red.

A pesar de contar con una licencia perpetua de SW NPM, hasta abril de 2024, la herramienta no ha sido implementada eficazmente para aprovechar sus capacidades de monitoreo avanzado. Actualmente, el gestor de red no proporciona beneficios tangibles a los usuarios debido a su interfaz poco intuitiva y el despligue de información desactualizada. Este problema se agrava con el hecho de que muchos de los nodos monitoreados están inactivos o no reportan datos útiles, incrementando el riesgo de inactividad del sistema y fallas operativas [4].

Afortunadamente, SW NPM ya cuenta con una DB la cual se va alimentando conforme se colocan nuevos nodos dentro del gestor para monitorear. Esto significa que la implementación adecuada de SW no solo facilitará la automatización de la detección y notificación de fallas, sino que también proporcionará una visibilidad clara del estado de la red a través de mapas de topología actualizados y alertas automáticas. Implementar este sistema permitirá a COCESNA responder más rápidamente a los problemas de red, reducir el tiempo de inactividad y mejorar la experiencia del área operativa [5].

Este proyecto propone configurar y optimizar SW para facilitar la supervisión automatizada y eficiente de la red con alertas tempranas. La implementación adecuada del gestor de red utilizando SW no solo facilitará una respuesta más rápida a los problemas, sino que también mejorará la calidad general del servicio al permitir una gestión proactiva en lugar de reactiva. Por otro lado también se propone la creación de una API que permita la integración de SW NPM con otras herramientas de monitoreo y ayudar a la automatización de tareas. Así, se espera que este proyecto contribuya directamente a mejorar la estabilidad y confiabilidad de la red para la prestación de servicios aeronáuticos críticos en el espacio aéreo guatemalteco [1].

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Implementar un gestor de red con la aplicación SolarWinds el cual pueda facilitar el trabajo de los usuarios desplegando información crítica de la red.

4.2. Objetivos específicos

- Monitorear el estado de los equipos y crear una alerta que notifique al personal que esté de turno si uno de estos está presentando fallas.
- Identificar y etiquetar de manera clara y concisa los enlaces para facilitar el diagnóstico de una falla.
- Generar un mapa que muestre dónde se encuentra cada ubicación geográficamente y los enlaces entre ubicaciones.
- Generar un mapa de la topología de la red para cada ubicación geográfica.
- *Dockerizar* una API la cual permita generar reportes automáticos a partir de nodos que se están monitoreando.

CAPÍTULO 5

Alcance

Este proyecto de graduación se enfoca en la mejora integral de la gestión y supervisión de la red de telecomunicaciones aeronáuticas de COCESNA en Guatemala. Se implementará y optimizará el gestor de monitoreo SW NPM (SolarWinds Network Performance Monitor) y se estandarizará la configuración de los dispositivos y plataformas de la red. Los objetivos específicos incluyen la identificación precisa de discrepancias en la red, la minimización de la intervención humana a través de alertas automáticas, la creación de un mapa actualizado de la topología de la red y la generación de reportes automáticos utilizando una API montada en Docker.

El proyecto aprovechará la licencia existente de SW NPM, versión 2020.2, que será crucial para el monitoreo avanzado de la red. Adicionalmente, se aprovechara la DB de MS SQL (Microsoft SQL Server) Server, versión 2008, del mismo SW NPM, para gestionar los datos de monitoreo, asegurando una integración robusta y escalable. Se emplearán tecnologías como Docker para la *dockerización* de aplicaciones, facilitando un despliegue eficiente y ágil de las herramientas necesarias.

El alcance del proyecto se restringe exclusivamente a la red de telecomunicaciones aeronáuticas gestionada por COCESNA en Guatemala. No se incluirán otros sistemas o redes dentro de la organización. El proyecto se desarrollará en un plazo de nueve meses, dando inicio desde el mes de marzo y finalizando en noviembre de 2024. Se realizará con entregas puntuales conforme al cronograma establecido en la Figura 67, limitando las actividades a la implementación, optimización, y estandarización de las configuraciones dentro del ámbito definido. Se seguirá un flujo de trabajo conforme las Figuras 68 y 69, asegurando la calidad y el orden a lo largo de la realizacion del proyecto.

CAPÍTULO 6

Marco teórico

6.1. SDN (Red definida por software)

Una SDN (*Software Defined Networking*), es una arquitectura innovadora que revoluciona la gestión de redes de computadoras al separar el plano de control y el plano de datos. Esta separación arquitectónica permite una mayor flexibilidad, escalabilidad y eficiencia en la gestión de redes [6, p.11].

Para facilitar la comunicación entre ambos planos, se emplean protocolos y estándares como OF (OpenFlow). Esta separación arquitectónica permite una mayor flexibilidad, escalabilidad y eficiencia en la gestión de redes [7, p.17].

En la implementación práctica de la SDN, se destacan casos de uso como la gestión eficiente del ancho de banda, la configuración dinámica de la red y la optimización del rendimiento. La arquitectura SDN típica consta de un controlador SDN, dispositivos de red compatibles con OF y la interfaz que posibilita la comunicación entre ellos. Estos avances han llevado a casos exitosos de implementación en entornos empresariales, proveedores de servicios y centros de datos, evidenciando el impacto significativo de la SDN en la innovación de las redes de telecomunicaciones [8].

Con una infraestructura basada en SDN, un controlador central adquiere una perspectiva global de la red, permitiendo la toma de decisiones inteligentes en aspectos como el enrutamiento y la recuperación de errores. Esto se logra al configurar y dirigir las capacidades de manera eficiente [9, p.1]. Como resultado, es posible utilizar componentes de menor costo, ya que ejecutan menos procesos en comparación con una red tradicional [10].

6.2. SNMP

SNMP (*Simple Network Management Protocol*) es un protocolo estándar para monitorear y administrar dispositivos de red, como routers, switches, servidores y dispositivos de almacenamiento. SNMP facilita la supervisión y gestión remota de los equipos de red al permitir que un NMS (Netwrok Management System) recopile información de estos dispositivos y opcionalmente, envíe comandos de configuración o control.

La operación básica de SNMP se basa en el concepto de agentes y gestores [11]. Los agentes son procesos de software instalado en los dispositivos de red que recopilan información sobre el estado y el rendimiento del dispositivo, mientras que los gestores son aplicaciones o sistemas que recopilan, procesan y muestran esta información [12]. SNMP utiliza mensajes de solicitud y respuesta para intercambiar información entre agentes y gestores, lo que permite a los administradores de red monitorear y diagnosticar problemas en la red de manera eficiente.

La implementación de SNMP facilita la supervisión y gestión remota de los equipos de red al permitir que un NMS recopile información de estos dispositivos y les envíe comandos de configuración o control [13].

6.2.1. Versiones de SNMP

A la fecha de publicación de esta tesis existen tres versiones de SNMP:

- **SNMPv1**: la primera versión, que proporciona una estructura básica para la administración de redes [13].
- **SNMPv2**: introduce mejoras en el rendimiento, seguridad y manejo de errores. Esta versión, que es la que se utilizará en la implementación, incluye funcionalidades adicionales como el *bulk retrieval* (recuperación en masa) y *trap* V2, que mejoran la eficiencia en la comunicación de datos [14].
- **SNMPv3**: ofrece mejoras significativas en términos de seguridad y privacidad, utilizando técnicas de autenticación y cifrado [14].

Las mejoras en SNMPv2 y SNMPv3 han abordado diversas preocupaciones de rendimiento y seguridad que eran limitaciones en la versión original del protocolo [15].

6.2.2. OID (*Object Identifiers*) y MIB (*Management Information Base*)

El término OID (*Object Identifier*) hace referencia a los identificadores únicos asignados a cada variable de información que el SNMP puede manejar [16]. Los OID están organizados jerárquicamente y representan un camino en el árbol de información de la MIB (*Management Information Base*), que es una base de datos utilizada para gestionar los entes en una red de comunicaciones. La MIB contiene textos de descripción y OID que representan diferentes tipos de información que un dispositivo de red puede almacenar y transmitir [17].

6.2.3. *Traps* y *polling* en SNMPv2

Los *traps* son mensajes de alerta enviados desde un agente hacia el gestor para notificar sobre eventos específicos. A diferencia de SNMPv1, SNMPv2 permite el envío de *traps inform*, que son mensajes que requieren confirmación, asegurando que la información ha sido recibida. El *polling* refiere al proceso donde el gestor SNMP regularmente solicita información de los agentes utilizando mensajes **GET**. En SNMPv2, el polling puede utilizar solicitudes **GET-BULK** para reducir el número de mensajes requeridos para obtener grandes cantidades de datos, lo cual hace el proceso más eficiente [4].

6.3. Gestor de red

El gestor de red hace referencia a dos conceptos. **Primero**, es el proceso de configuración, monitoreo y administración del desempeño de la red. **Segundo**, es la plataforma utilizada para lograr las tareas previamente discutidas [18].

Al igual que la automatización es una evolución de la gestión, la automatización de la red es también una evolución de la gestión de la red. Los sistemas de gestión de la red ahora se basan en la automatización para administrar y gestionar los recursos y los servicios de la red.

Uno de los ejemplos más destacados de plataformas de gestión de red es SW NPM (SolarWinds Network Performance Monitor). SW NPM es una herramienta integral diseñada para facilitar la supervisión continua, el diagnóstico eficiente y la generación de informes de rendimiento de la red. Esta aplicación proporciona una vista en tiempo real del estado operativo de la red, permitiendo a los administradores detectar, diagnosticar y resolver problemas de red de manera proactiva. SW es conocido por su capacidad para escalar en grandes entornos de red y por su interfaz intuitiva que simplifica la gestión de redes complejas.

Para este proyecto, se utilizó SW NPM versión 2020.2.4, que incluye características avanzadas como la visualización de la topología de red, alertas configurables, y soporte para una amplia gama de dispositivos de red. Esta versión específica proporciona herramientas robustas para la automatización de tareas rutinarias y ofrece mejoras significativas en la eficiencia de la monitorización de redes mediante algoritmos optimizados para el análisis de datos de tráfico y la gestión de eventos.

La incorporación de SW NPM fue un punto crítico en la supersupervisión detallada, ya que permitió una respuesta rápida a incidentes, asegurando una gestión efectiva y eficiente de los recursos de la red.

6.4. Base de datos

Las DB (bases de datos) son sistemas estructurados que permiten el almacenamiento, manipulación y consulta de grandes cantidades de datos de manera eficiente y segura [19]. Estos sistemas son esenciales para la gestión de información en diversas aplicaciones, desde sitios web hasta aplicaciones empresariales y sistemas de gestión de red. El uso de DB facilita la organización de la información en formatos que permiten un acceso rápido y fiable, soportando transacciones que requieren alta disponibilidad y consistencia [20].

Dentro de las DB, se distinguen dos grandes categorías: las DB relacionales y las no relacionales. Las DB relacionales, como MS SQL (Microsoft SQL Server), utilizan tablas para organizar los datos, permitiendo relaciones complejas entre diferentes conjuntos de datos. Estas DB son conocidas por su robustez, escalabilidad y cumplimiento de las normas ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), que aseguran transacciones seguras y confiables [21].

Para este proyecto, se utilizará MS SQL como sistema de gestión de DB. MS SQL es conocido por su robustez, escalabilidad y características avanzadas que incluyen soporte para transacciones ACID, alta disponibilidad, y seguridad mejorada [22].

MS SQL ofrece una integración fluida con otros productos de Microsoft, lo que facilita su uso en entornos que ya utilizan tecnologías de Microsoft [22]. Además, MS SQL proporciona herramientas avanzadas para la administración y el desarrollo, como SSMS (MS SQL Management Studio), que permite a los administradores y desarrolladores gestionar de manera efectiva las bases de datos y realizar tareas de desarrollo de manera eficiente [20].

6.5. Interfaces de programación de aplicaciones

6.5.1. Definición y Evolución de las API

Las API (*Application Programming Interface*) actúan como intermediarios que facilitan la comunicación entre diferentes componentes de software, permitiendo que sistemas heterogéneos puedan trabajar en conjunto de manera eficiente [23][24].

El concepto de API ha evolucionado significativamente desde sus primeras implementaciones en los años 60, cuando se utilizaban principalmente para facilitar la interacción entre sistemas operativos y aplicaciones de software [25]. En la actualidad, las API son esenciales en el desarrollo de sistemas integrados, donde diferentes aplicaciones, que pueden estar desarrolladas en distintos lenguajes o plataformas, puedan comunicarse para cumplir con los requisitos funcionales de un entorno empresarial [26][27].

6.5.2. Tipos de API

Existen diferentes tipos de API, cada una diseñada para cumplir con necesidades específicas en el desarrollo de software:

- **API locales:** facilitan la comunicación dentro de un mismo sistema operativo o entre aplicaciones instaladas en la misma máquina. Un ejemplo de ello son las API de Windows, que permiten a las aplicaciones interactuar con el sistema operativo [28].
- **API web:** permiten la comunicación entre servidores y clientes a través de la web, utilizando protocolos como HTTP/HTTPS. Son fundamentales para la integración de servicios web modernos, como REST (Representational State Transfer) y SOAP (Simple Object Access Protocol) [29].
- **API de bibliotecas o *frameworks*:** proveen funciones y métodos predefinidos que pueden ser utilizados por los desarrolladores para realizar tareas específicas sin necesidad de codificar desde cero. Por ejemplo, las API de .NET o Java ofrecen funciones que facilitan la interoperabilidad entre aplicaciones desarrolladas en sus respectivos entornos [30].
- **API abiertas (*Open* API):** estas son accesibles públicamente y permiten a los desarrolladores externos integrar sus aplicaciones con servicios de terceros. Las API abiertas han sido un catalizador para la innovación, permitiendo la creación de nuevas aplicaciones que aprovechan los servicios de empresas como Google, Facebook, y Twitter [31].
- **API privadas:** estas API están diseñadas para ser utilizadas exclusivamente dentro de una organización. Son esenciales para garantizar que las aplicaciones internas puedan comunicarse de manera eficiente y segura, permitiendo una integración más profunda entre sistemas sin exponer interfaces a terceros [32].

6.5.3. Importancia de las API en la centralización y automatización de sistemas

En el desarrollo de sistemas centralizados, las API son fundamentales para permitir la comunicación y sincronización entre diferentes aplicaciones locales [33]. Esto es particularmente relevante en entornos donde múltiples aplicaciones deben colaborar para automatizar procesos y generar reportes de manera centralizada [34].

El uso de API en este contexto permite que datos generados por diversas aplicaciones sean consolidados en un único sistema central, donde pueden ser procesados y utilizados para la toma de decisiones [35]. Además, las API facilitan la automatización de procesos, permitiendo que tareas repetitivas que requieren la interacción de múltiples sistemas puedan ser ejecutadas de manera eficiente y sin intervención manual [36].

6.5.4. Uso de API para generación de reportes y automatización

Las API juegan un papel vital en la automatización de procesos y la generación de reportes, especialmente en entornos donde se requiere consolidar datos de diferentes fuentes [37]. En el caso de la red de telecomunicaciones de COCESNA, la API diseñada no solo facilitó la obtención de datos mediante *querys* sino que también mediante CLI (*Command Line Interface*), por último, organiza estos datos en un formato coherente y fácil de interpretar [38].

La automatización mediante API reduce significativamente la intervención manual, lo que no solo minimiza errores humanos, sino que también ahorra tiempo y recursos a la empresa [39]. Anteriormente, tareas como la recolección y organización de datos de múltiples sistemas requerían de un esfuerzo manual considerable, involucrando personal que debía realizar tareas monótonas y repetitivas que consumían una gran cantidad de tiempo. Con la implementación de esta API, COCESNA podrá eliminar la necesidad de destinar recursos humanos a estas tareas rutinarias, optimizando así el uso de su personal en actividades de mayor valor agregado [40].

Además de ahorrar tiempo, la reducción en la necesidad de mano de obra directa para tareas repetitivas se traduce en un ahorro de costos operativos. La automatización asegura que los reportes se generen con mayor rapidez y precisión, mejorando la eficiencia general y permitiendo que la empresa se enfoque en otras áreas estratégicas y operativas que requieren mayor atención [41].

6.5.5. Implementación de la API en el desarrollo del proyecto

En esta tesis, la API desarrollada se monta en un entorno Docker para garantizar un despliegue consistente y aislado. Su principal función es realizar Querys a una DB (base de datos) basada en MS SQL (Microsoft SQL Server), donde se almacenan datos obtenidos a través de SNMP (*Simple Network Management Protocol*) de nodos la red de COCESNA. Para los datos que no pueden ser obtenidos mediante *querys*, la API recurre al CLI directamente.

Posteriormente, la API procesa y organiza estos datos para presentarlos en un formato claro y fácil de entender. El propósito es asegurar que los datos sean verificados por el usuario antes de generar un reporte final. Este reporte, una vez confirmado, es formateado de acuerdo con los estándares de COCESNA y enviado a los responsables del monitoreo de la red. Este enfoque no solo mejora la eficiencia en la generación de reportes, sino que también asegura la precisión y relevancia de la información presentada, contribuyendo así al mantenimiento proactivo de la red de telecomunicaciones.

6.6. Docker

Docker es una plataforma de *dockerización* líder que permite a los desarrolladores y administradores de sistemas empaquetar aplicaciones y sus dependencias en unidades estándar llamadas contenedores (*dockers*). Estos contenedores son ligeros, portátiles y aseguran que el software se ejecute de manera consistente independientemente del entorno en el que se despliegue [42]. Docker proporciona una API robusta y una interfaz de línea de comandos para gestionar el ciclo de vida de los contenedores, incluyendo la construcción, prueba, despliegue y escalado [43]. La eficiencia de Docker se deriva del hecho de que los contenedores comparten el Kernel del sistema operativo host y no requieren un sistema operativo separado para cada contenedor, lo que reduce la sobrecarga de recursos y mejora la utilización del *hardware* [44].

La tecnología de Docker se ha destacado por su capacidad para simplificar el proceso de despliegue y escalamiento de aplicaciones en cualquier ambiente, desde *laptops* personales hasta sistemas en la nube y servidores de producción. Docker facilita una amplia gama de operaciones de software, permitiendo una rápida configuración y despliegue de aplicaciones, lo que es crucial en la implementación de prácticas de DevOps que buscan mejorar la eficiencia y la velocidad del desarrollo de software [45].

Una de las características más significativas de Docker es Docker Hub, un registro que permite a los usuarios almacenar y distribuir imágenes de contenedores. Esto facilita la colaboración entre equipos y la reutilización de software, componentes esenciales para la metodología de DevOps que busca mejorar la eficiencia y la velocidad del desarrollo de software [44].

6.6.1. Implementación de Docker en el desarrollo del proyecto

Para este proyecto se utilizará la versión de Docker 4.28.0. No fue seleccionado por ninguna particularidad, simplemente era la versión más reciente al inicio del proyecto. Se eligió continuar con esta durante el desarrollo para asegurar el funcionamiento de los contenedores y no tener que actualizarlos en caso dejasen de funcionar con versiones más recientes.

CAPÍTULO 7

Configuración de nodos e interfaces para la plataforma SolarWinds

En este capítulo se presenta el proceso detallado de configuración de los nodos para la plataforma SW NPM (SolarWinds Network Performance Monitor), utilizando mecanismos como el descubrimiento automático, filtros para seleccionar nodos relevantes y la organización mediante grupos y dependencias. SW NPM puede monitorear los nodos utilizando ICMP (*Internet Control Message Protocol*) o SNMP (*Simple Network Management Protocol*), según las necesidades específicas de la red. Este enfoque permite monitorizar de manera efectiva los elementos críticos de la red, minimizando la sobrecarga del sistema y maximizando la eficiencia de los recursos disponibles. Cada etapa del proceso se resalta para garantizar un monitoreo preciso y efectivo de la infraestructura.

7.1. Descubrimiento automático de nodos

El descubrimiento automático es fundamental para la gestión de redes grandes, ya que permite identificar y agregar nodos al sistema de monitoreo sin intervención manual continua. En este proyecto, el descubrimiento automático se configuró para ejecutarse de manera periódica, **específicamente el primer lunes de cada mes a las 10:00 a. m**, para garantizar que haya personal de turno que pueda supervisar y seleccionar los nuevos nodos detectados, asegurando que la base de datos de nodos estuviera siempre actualizada y alineada con los cambios en la infraestructura de la red. La automatización de este proceso no solo reduce el esfuerzo manual, sino que también minimiza los errores humanos que podrían surgir al agregar dispositivos manualmente.

Para optimizar este proceso, se utilizaron filtros personalizados para seleccionar los nodos más útiles para el monitoreo. Los criterios de selección incluyeron el tipo de dispositivo (por ejemplo, routers y switches críticos), la ubicación geográfica y la relevancia de los nodos en la topología general de la red. Esta selección cuidadosa permitió reducir la carga innece-

saria sobre el motor de polling, enfocarse en los elementos críticos que requerían supervisión constante, y no exceder la cantidad de nodos permitidos por la licencia. Planificar cuidadosamente el descubrimiento también ayudó a garantizar un uso eficiente de los recursos disponibles, evitando la sobrecarga del sistema.

El descubrimiento automático se configuró para ejecutarse de manera periodica, permitiendo una detección proactiva de nuevos dispositivos que pudieran ser añadidos a la red. De esta manera, la plataforma SW NPM siempre se mantuvo actualizada con los cambios en la infraestructura, proporcionando una vista completa y precisa de los nodos activos en todo momento. Además, la automatización en la detección de dispositivos facilitó la actualización continua del inventario, permitiendo una identificación rápida de dispositivos que ingresan o salen del sistema.

7.1.1. Pasos para configurar el descubrimiento automático de nodos

El proceso de configuración del descubrimiento automático de nodos en SW NPM se llevó a cabo siguiendo los siguientes pasos:

1. **Definir los ajustes de descubrimiento:** en estos ajustes, Figura 1, se puede definir el nombre del descubrimiento, lo cual será útil para ejecutarlo nuevamente en el futuro, junto con una descripción y otros parámetros que definirán el comportamiento del descubrimiento. No es necesario modificar estos últimos ajustes.

Figura 1: Detalles para la configuración del descubrimiento automático, en este caso se esta para la estación de REU (Retalhuleu)

2. **Especificar las IP a monitorear:** esto se puede hacer ingresando un rango, una subred o una dirección IP específica. En este caso se utilizó un rango de IP, Figura 2.

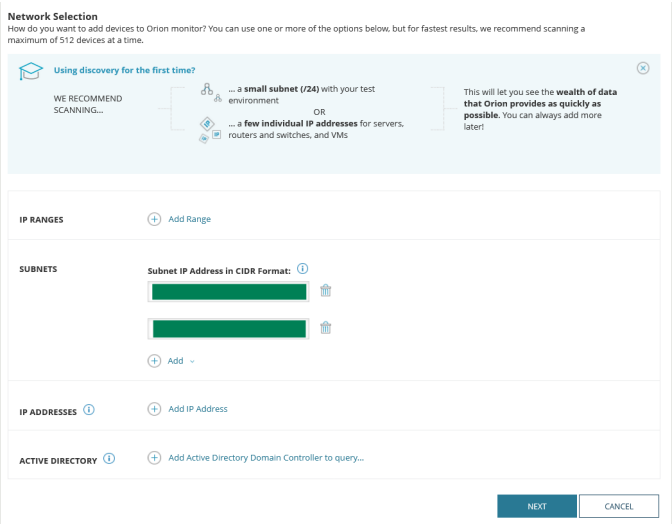


Figura 2: Especificación de las IP a monitorear en el descubrimiento automático. Las IP están sensuradas para proteger la seguridad de la red de COCESNA

3. **Seleccionar el método de *polling*:** se recomienda hacerlo mediante SNMP, Figura 3, ya que este protocolo permite una recopilación más detallada y eficiente de los datos de los dispositivos, además de ser un estándar en la industria para el monitoreo remoto. A diferencia de ICMP, SNMP proporciona una visión más amplia del estado operativo de los dispositivos al permitir la consulta de parámetros específicos, como la utilización de CPU y memoria. Esta capacidad resulta crucial en redes complejas como la de COCESNA, donde se requiere un monitoreo proactivo y detallado de cada nodo.

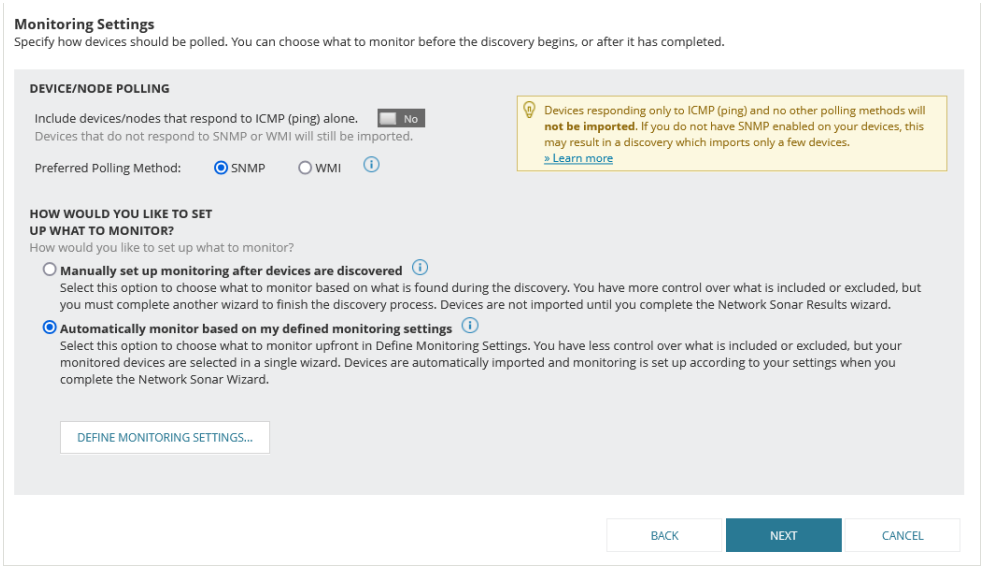


Figura 3: Selección del método de *polling* para el descubrimiento automático

4. **Monitoreo automático basado en ajustes definidos:** seleccionar la opción "Automatically monitor based on my defined monitoring settings" y definir los ajustes de monitoreo para reducir la cantidad de nodos e interfaces que deben seleccionarse manualmente previo al descubrimiento.

5. **Definir las interfaces de interés:** en estos ajustes, Figura 4, se excluyeron interfaces inactivas, VLAN, o puertos sin un rol significativo en la transmisión de datos críticos, asegurando que los recursos de monitoreo se centraran en las interfaces que realmente impactan el desempeño de la red.

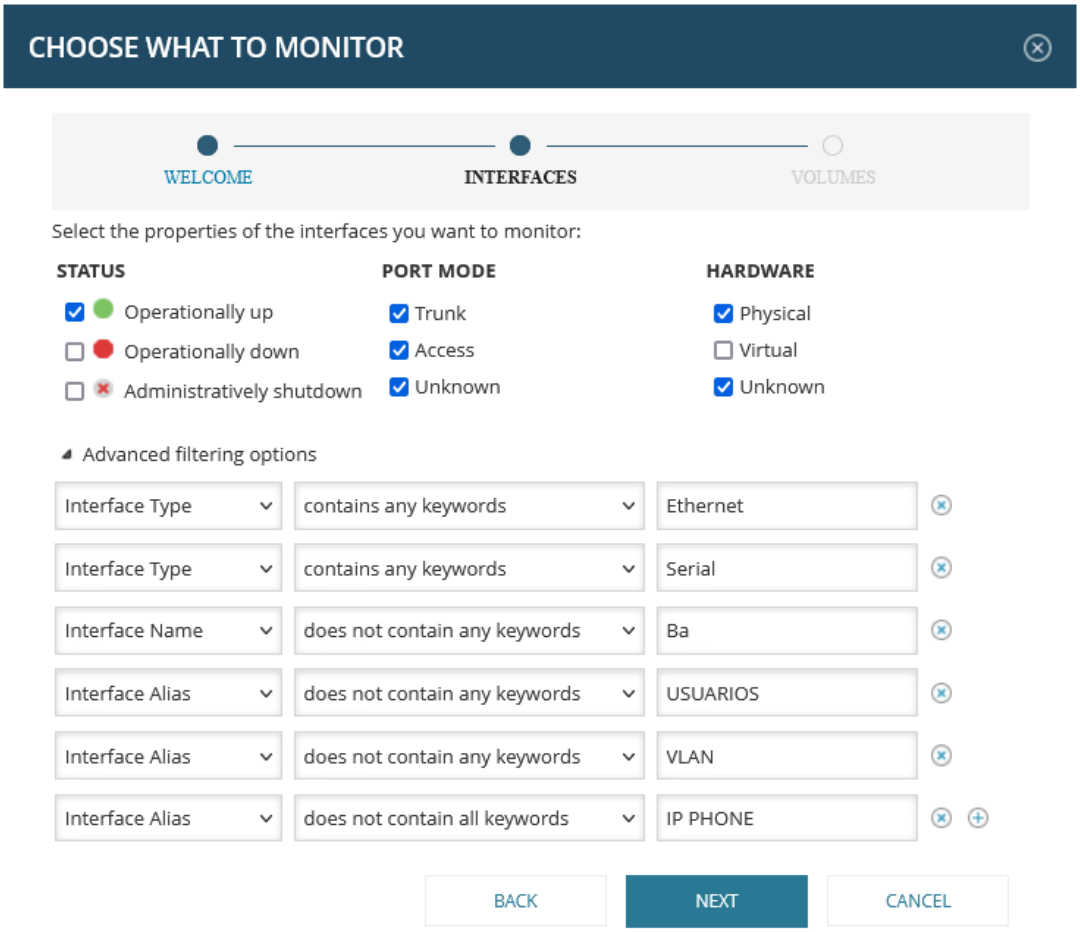


Figura 4: Configuración de filtros avanzados para el descubrimiento automático de nodos en SW NPM, que permite seleccionar interfaces según estado operativo, modo de puerto y tipo de *hardware*, junto con opciones detalladas por tipo de interfaz y alias. Los valores de la tercera columna hacen referencia a fragmentos de nombres de las interfaces asignados por COCESNA

6. **No alterar la selección de volúmenes por defecto:** los volúmenes se utilizan cuando se desea monitorear servidores o computadoras 5.

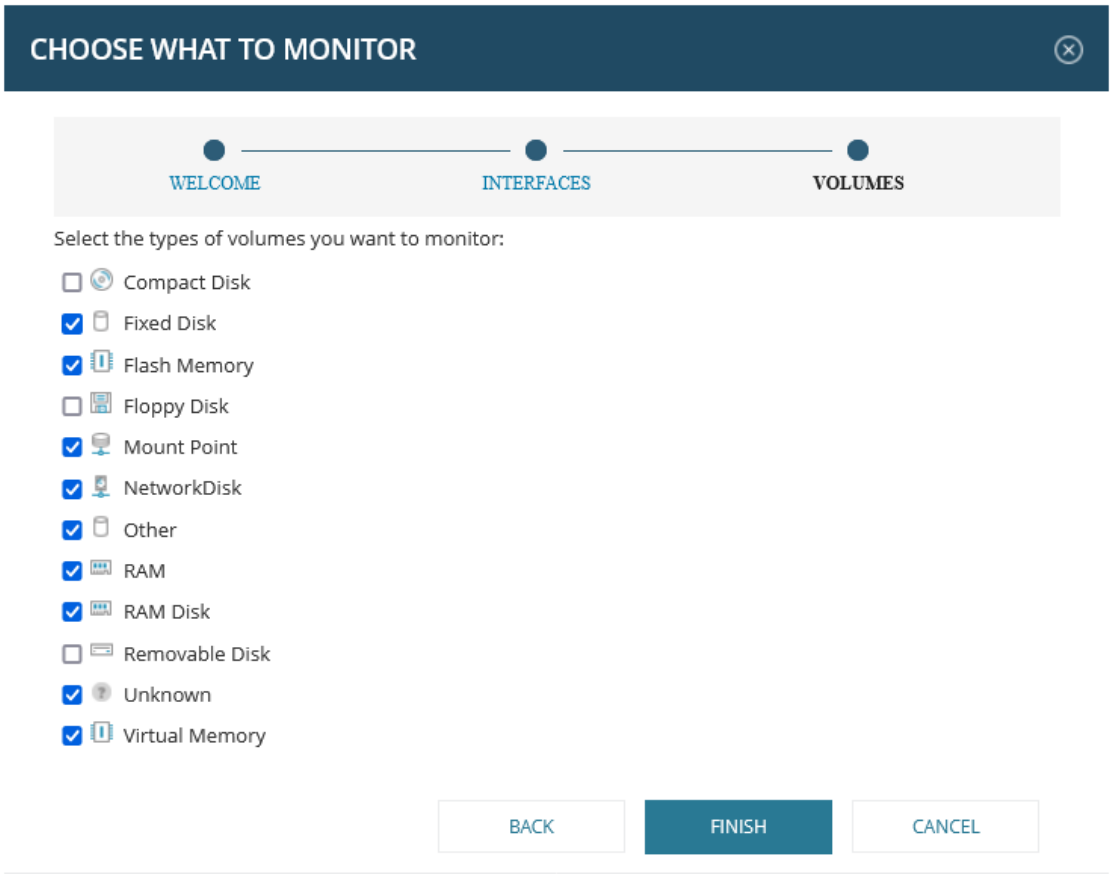


Figura 5: Configuración de volúmenes para el descubrimiento automático de nodos en SW NPM

7. **Revisar las credenciales SNMP:** revisar las credenciales SNMP en cada uno de los nodos, mediante CLI, para verificar si el protocolo SNMP se encuentra configurado. Si está configurado, se debe tomar nota de la comunidad SNMP y asegurarse de que no sea la que viene por defecto, ya que esto podría exponer a vulnerabilidades de seguridad. En caso contrario, se debe configurar una comunidad segura. En la Figura 6, se puede ver que hay cinco comunidades distintas configuradas para el monitoreo; el orden en que estas se colocan es de crucial importancia, ya que afecta directamente el tiempo que tarda en ejecutarse el descubrimiento y la carga que pondrá en SW NPM al hacerlo. La razón es que SW NPM intenta verificar si la comunidad es correcta para cada IP del descubrimiento; si no lo logra con la primera comunidad, pasará a la segunda, luego a la tercera, y así sucesivamente. En una red del tamaño de la de COCESNA, esto podría hacer que el proceso sea más prolongado de lo necesario.

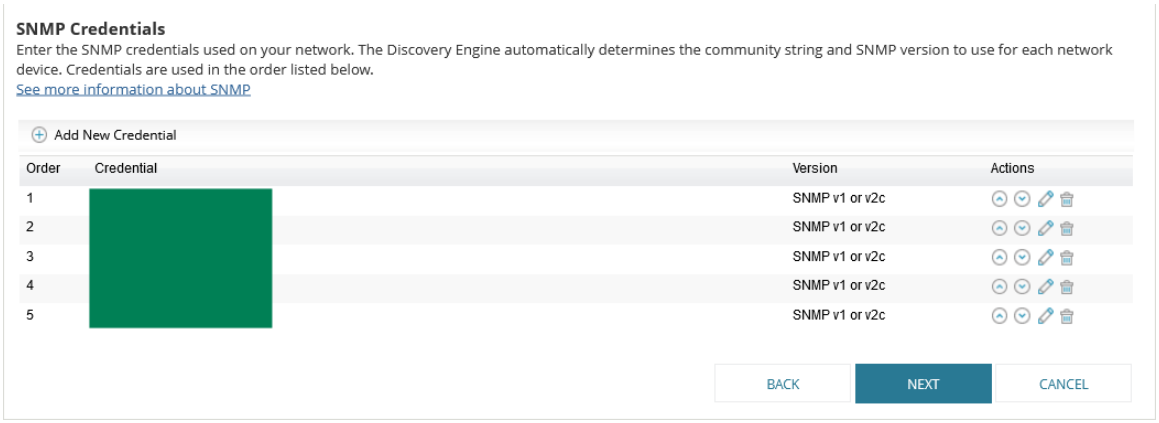


Figura 6: Configuración de credenciales SNMP para el descubrimiento automático de nodos en SW NPM. Se sensuraron las comunidades SNMP para proteger la seguridad de la red de COCESNA

8. **Ejecutar el descubrimiento:** al hacerlo, se despliega la ventana que proporciona información sobre el progreso del descubrimiento, Figura 7. Se puede ejecutar en segundo plano, y SW NPM creará una notificación indicando que se ha completado.

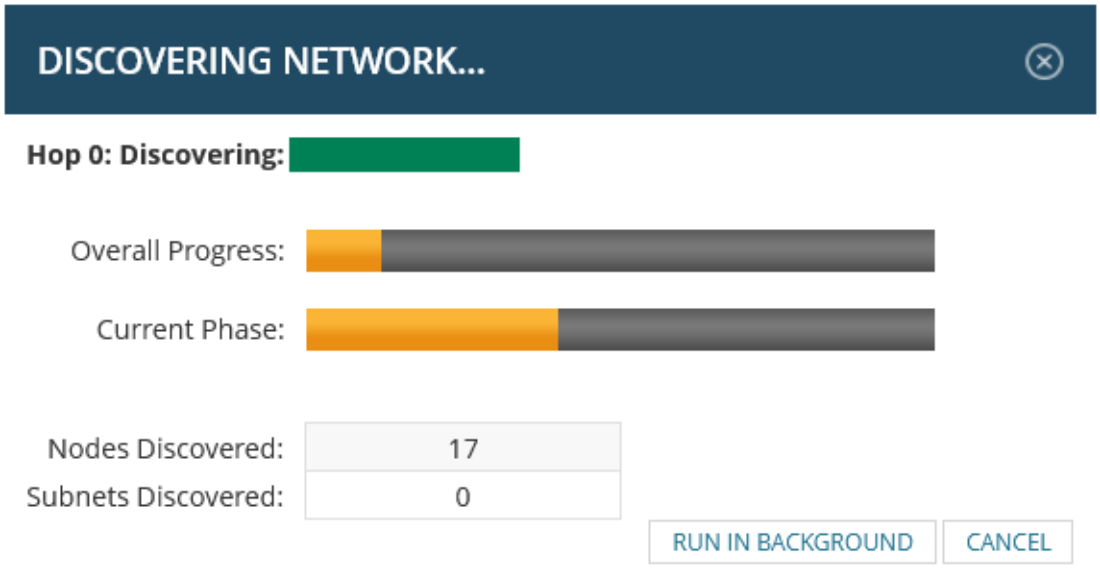


Figura 7: Ventana de progreso del descubrimiento automático de nodos en SW NPM

7.1.2. Resultados del descubrimiento automático

En esta sección se detalla el proceso manual para agregar los nodos descubiertos al sistema de monitoreo. Aunque el descubrimiento automático facilita la detección de dispositivos, es necesario un paso adicional para seleccionar y agregar los nodos e interfaces más relevantes. Esto asegura que el monitoreo esté enfocado en los elementos críticos de la red, garantizando la eficiencia en el uso de los recursos disponibles y evitando sobrepasar las limitaciones de la licencia.

1. **Visualizar los resultados del descubrimiento:** en la pestaña de "Scheduled Discovery Results", se despliegan los descubrimientos realizados por cada uno de los agentes de descubrimiento creados, Figura 8.

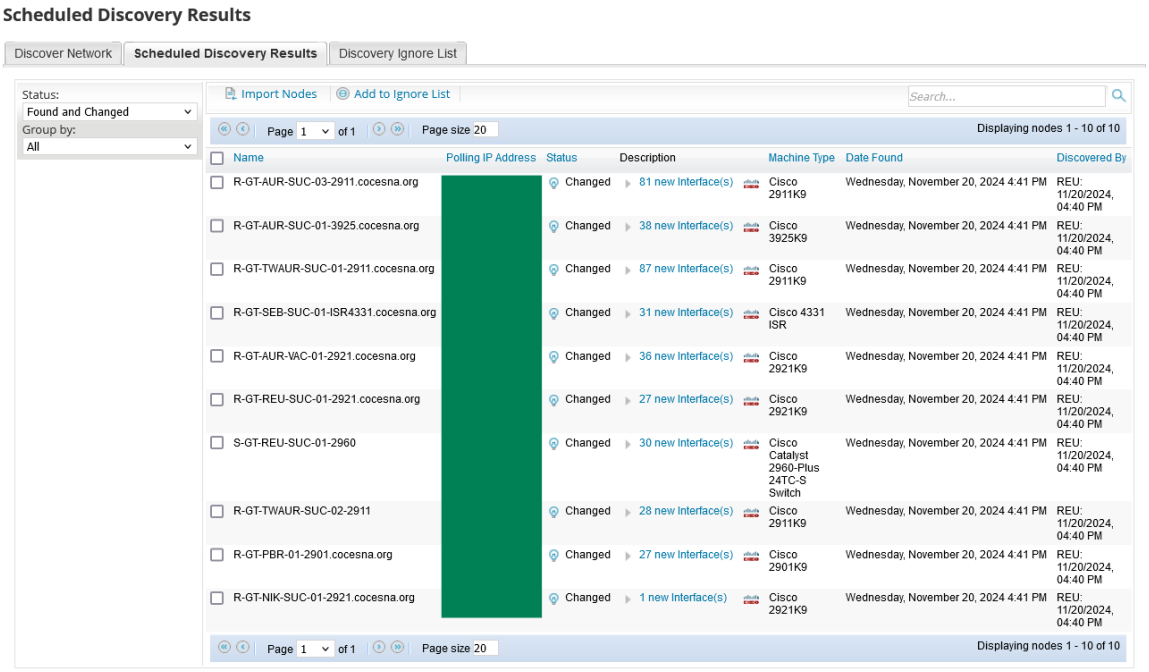


Figura 8: Resultados del descubrimiento automático de nodos en SW NPM

Aquí es donde se encuentra la parte manual del proceso del descubrimiento periódico. En los nodos se muestran las interfaces que pasaron los filtros especificados, Figura 4. Dado que COCESNA cuenta con una **licencia de SW NPM con un máximo de 150 nodos**, es necesario identificar cuáles de las interfaces son más críticas. El criterio utilizado fue el siguiente:

- Utilizar el comando del Cuadro 1 mediante CLI (*Command Line Interface*) en los nodos para identificar los vecinos conectados.

```
1 R-GT-SEB-SUC-01-ISR4331#show cdp neighbors
2 Capability Codes: R - Router, T - Trans Bridge, B - Source Route
      Bridge
3                   S - Switch, H - Host, I - IGMP, r - Repeater, P -
                        Phone,
```

```
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform
     Port ID
S-GT-COM-SEB-SUC-02-9200
                 Gig 0/2/0         171              S I   C9200L-24
                      Gig 1/0/24
S-GT-AUR-ACC-01-3750
                 Gig 0/0/0         144              S I   WS-C3750X
                      Gig 1/0/32
S-GT-COM-SEB-SUC-01-2960
                 Gig 0/0/1         175              S I   WS-C2960S
                      Gig 1/0/23

Total cdp entries displayed : 3
```

Cuadro 1: Comando para identificar los vecinos conectados a un nodo utilizando CLI

- Consultar con el encargado de la red para verificar si estos nodos eran necesarios o no.

2. **Seleccionar nodos e interfaces críticas:** una vez obtenidos los nodos e interfaces críticos, se pueden seleccionar independientemente para agregarlos al sistema de monitoreo, Figura 9. Se debe dar clic en la opción "Import Nodes".

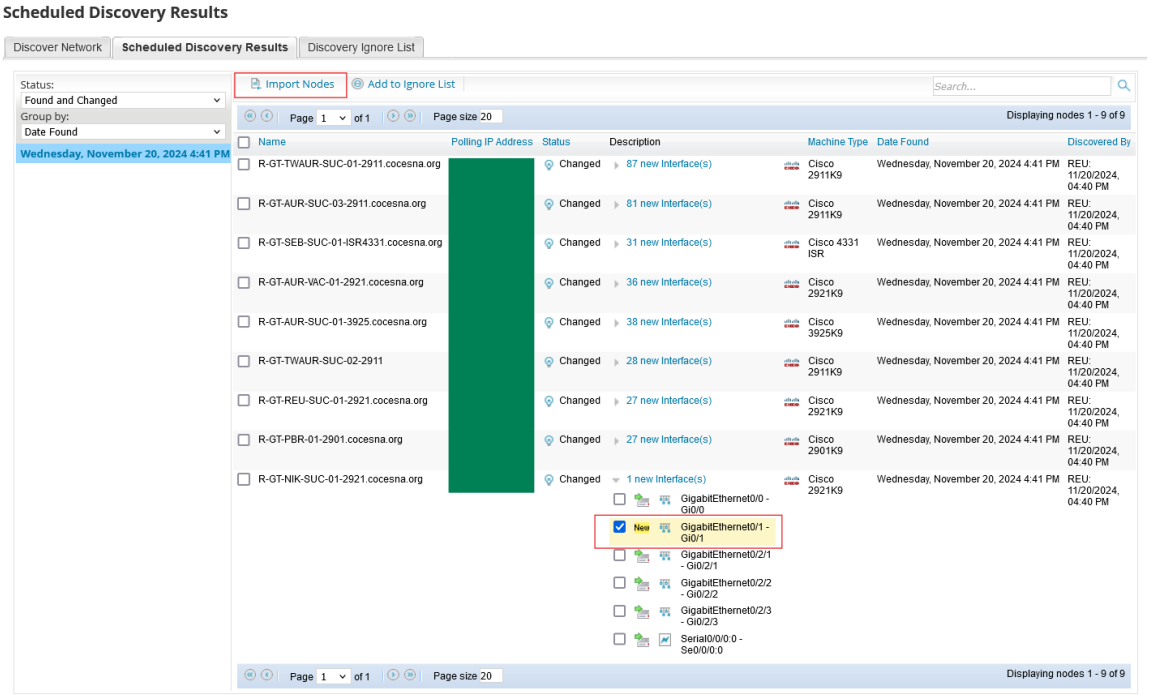


Figura 9: Importación de nodos descubiertos en SW NPM. Se seleccionan los nodos que se desean importar y se da click en "Import Selected"

3. **Configuración en el proceso de importación:** dentro del proceso de importación, se deben dejar los primeros tres pasos con la configuración por defecto. Cuando se llegue a la sección "Import Wizard", Figura 10, se debe verificar que el nodo seleccionado sea el correcto y que la información coincida con la del equipo de la red.

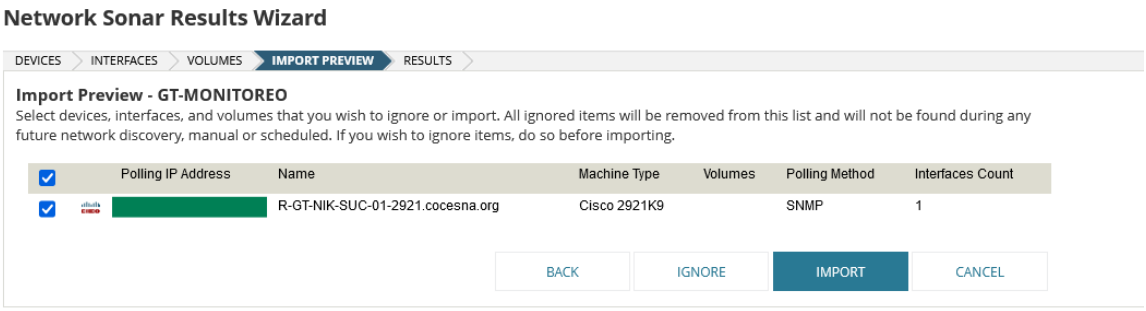


Figura 10: Confirmar la importación de nodos descubiertos en SW NPM. Se puede seleccionar los nodos que se desean importar y se da click en "Import"

4. **Ejecutar cambios y resumen de acciones:** por último, se ejecutarán los cambios y SW NPM desplegará un resumen de las acciones realizadas Figura 11. En este caso, si el nodo ya se encuentra en la base de datos de Orion, el sistema lo omitirá y continuará con el proceso. Por otro lado, si la interfaz no se encontraba en dicha base de datos, se alimentará la información necesaria para comenzar su monitoreo.

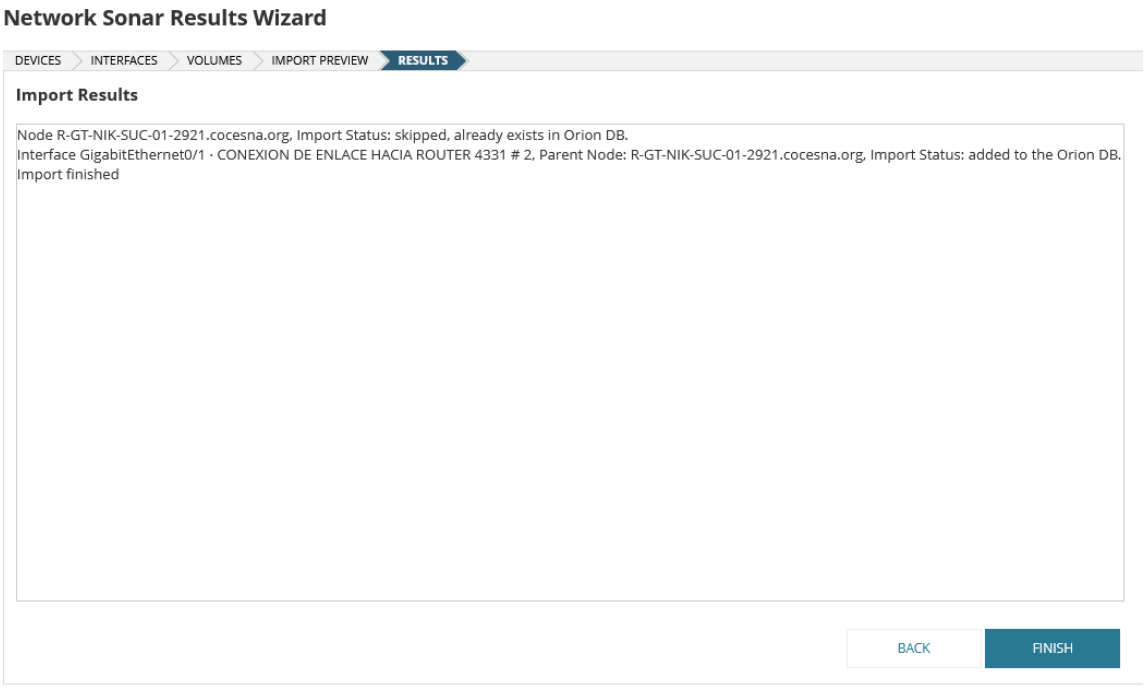


Figura 11: Resultados de la importación de nodos seleccionados y la respuesta de la base de datos

7.2. Filtros para la selección de interfaces descubiertas

El proceso de descubrimiento también incluyó la detección y selección automática de las interfaces de los nodos descubiertos. En este caso, se aplicaron filtros adicionales para seleccionar solo las interfaces activas y de mayor relevancia. Estos filtros se basaron únicamente en la criticidad y el propósito de la interfaz dentro del sistema de telecomunicaciones de CORCESNA. Este enfoque permitió concentrar los esfuerzos de monitoreo en aquellas interfaces que desempeñan un papel crucial en el rendimiento y la estabilidad de la red.

Por ejemplo, se excluyeron interfaces inactivas, VLAN, o puertos sin un rol significativo en la transmisión de datos críticos, asegurando que los recursos de monitoreo se centraran en las interfaces que realmente impactan el desempeño de la red. Esta selección precisa evitó la sobrecarga de la plataforma con datos innecesarios y garantizó que los administradores de red pudieran concentrarse en resolver problemas potenciales en puntos críticos de la infraestructura.

Además, el uso de filtros personalizados, como se presentan en la Figura 4 para la selección de interfaces permitió un monitoreo más granular, crucial para asegurar que los elementos más importantes de la red recibieran la atención adecuada. Se establecieron reglas específicas para priorizar las interfaces con mayor tráfico y aquellas que conectan segmentos críticos de la red, asegurando así la disponibilidad de datos en tiempo real para la toma de decisiones operativas.

7.3. Organización de nodos mediante grupos

Para simplificar el monitoreo y gestionar la complejidad de la red, los nodos fueron organizados en grupos. La clasificación se realizó tomando en cuenta factores como la ubicación geográfica, el rol del dispositivo en la red y el tipo de servicio que proporciona. Esta organización jerárquica facilitó una visión más clara de la infraestructura y mejoró la eficiencia en la gestión de alertas.

- **Grupos:** se crearon grupos principales basados en la región o zona donde se encuentran los dispositivos, lo cual permitió una vista organizada de la infraestructura a nivel regional. La creación de estos grupos facilita la identificación rápida de problemas específicos de cada región, esencial para una respuesta eficiente ante incidentes que afectan áreas geográficas particulares.
- **Subgrupos:** en este proyecto no se utilizaron subgrupos, aunque se reconoce su utilidad para facilitar la navegación en la plataforma SW NPM y mejorar la eficiencia en la gestión de alertas y el análisis del rendimiento. Los subgrupos podrían ser útiles para categorizar dispositivos según su rol específico dentro de un grupo, lo cual permitiría una administración más detallada y enfocada de los recursos de la red.

7.3.1. Configuración de grupos

Para llevar a cabo la organización de los nodos, se utilizó la interfaz de administración de SW NPM, la cual permite arrastrar y soltar los nodos en los grupos correspondientes. El proceso se llevó a cabo de la siguiente manera:

1. **Acceder a la sección de grupos:** para comenzar, navegue a la pestaña "Manage Groups" dentro de la plataforma SW NPM. Aquí encontrará la opción para crear nuevos grupos de nodos o editar los ya existentes, Figura 12.

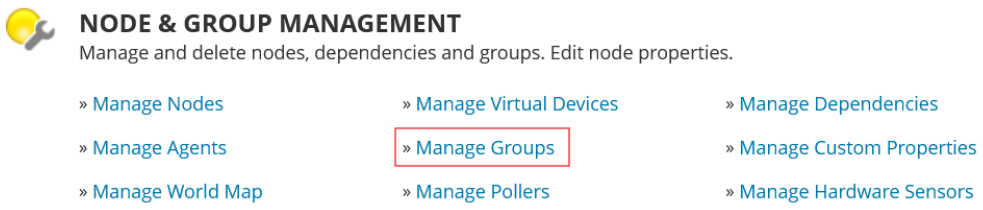


Figura 12: Menú de configuración de grupos en SolarWinds

2. **Definir un nuevo grupo:** haga clic en la opción "Add New Group". En esta sección, deberá proporcionar un nombre descriptivo para el grupo y una breve descripción que explique su propósito Figura 13. Es importante que el nombre del grupo sea claro y específico para facilitar la identificación y gestión de los nodos.

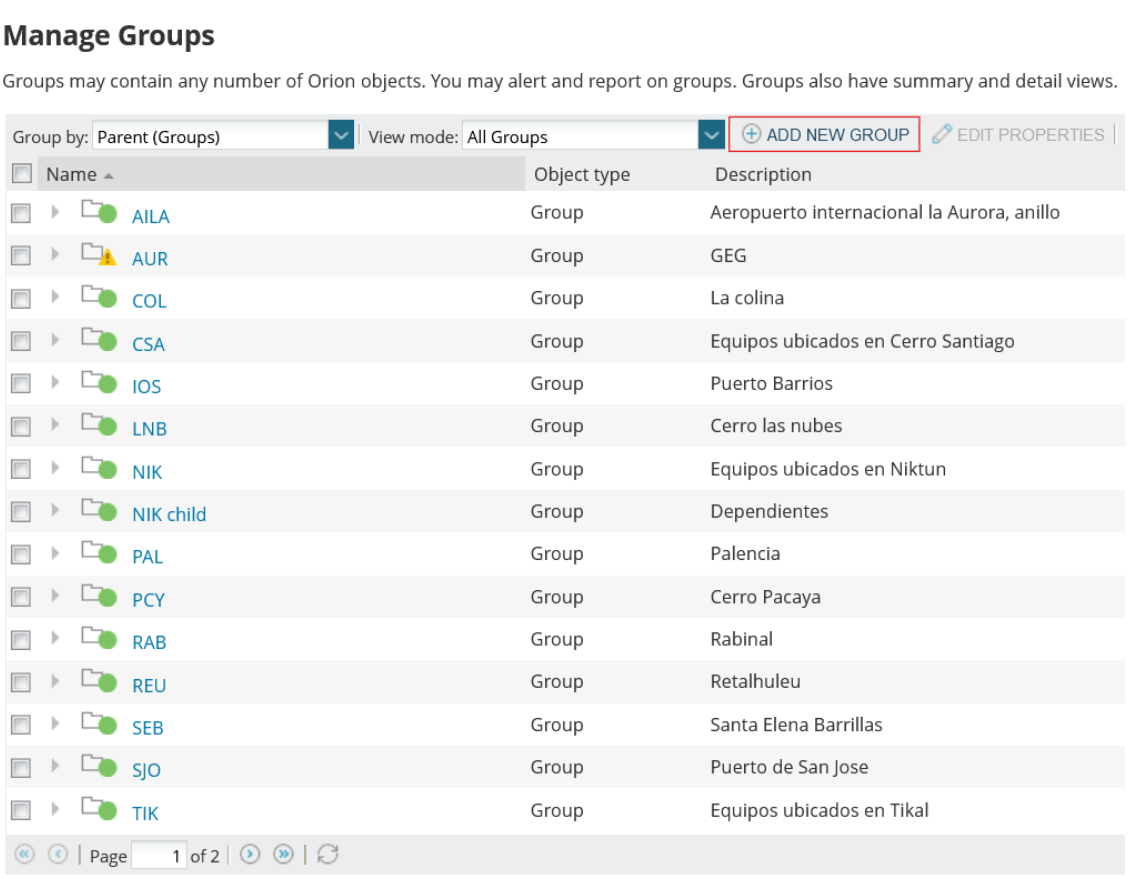


Figura 13: Grupos creados en SW NPM de las estaciones de COCESNA

3. **Asignar nodos al grupo:** al darle a "Create Group", Figura 13, se le asigna un nombre al grupo, Figura 14, es recomendable asignarle un nombre que indique la región a la que pertenecen los nodos. Luego, se seleccionan los nodos que que pueden pertenecer al grupo utilizando los filtro que proporciona SW NPM, Figura 15, y se le debe dar a la opciones de "Add Group" para agregar los nodos al grupo.

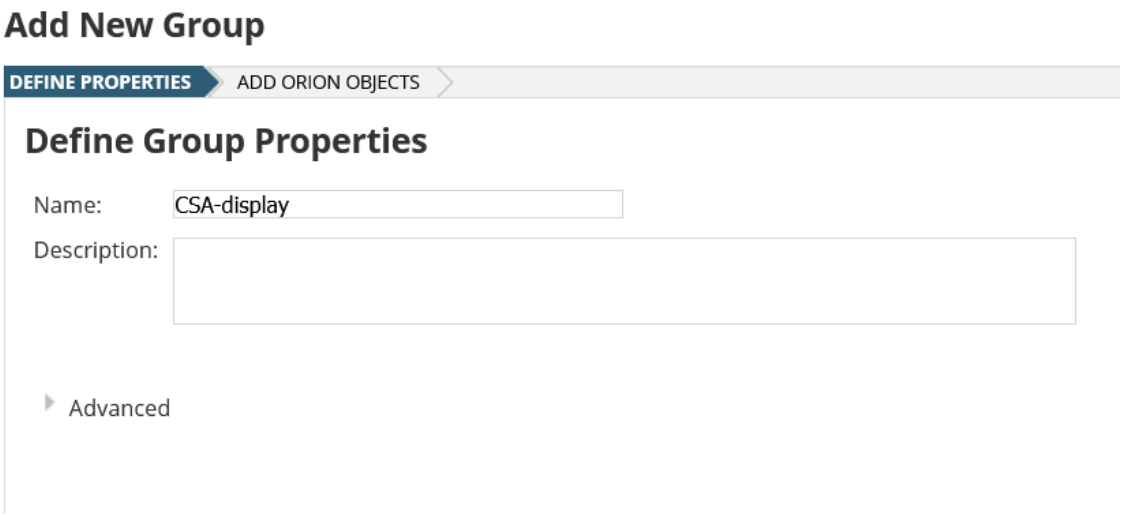


Figura 14: Propiededes que se le pueden asignar a un grupo en SW NPM

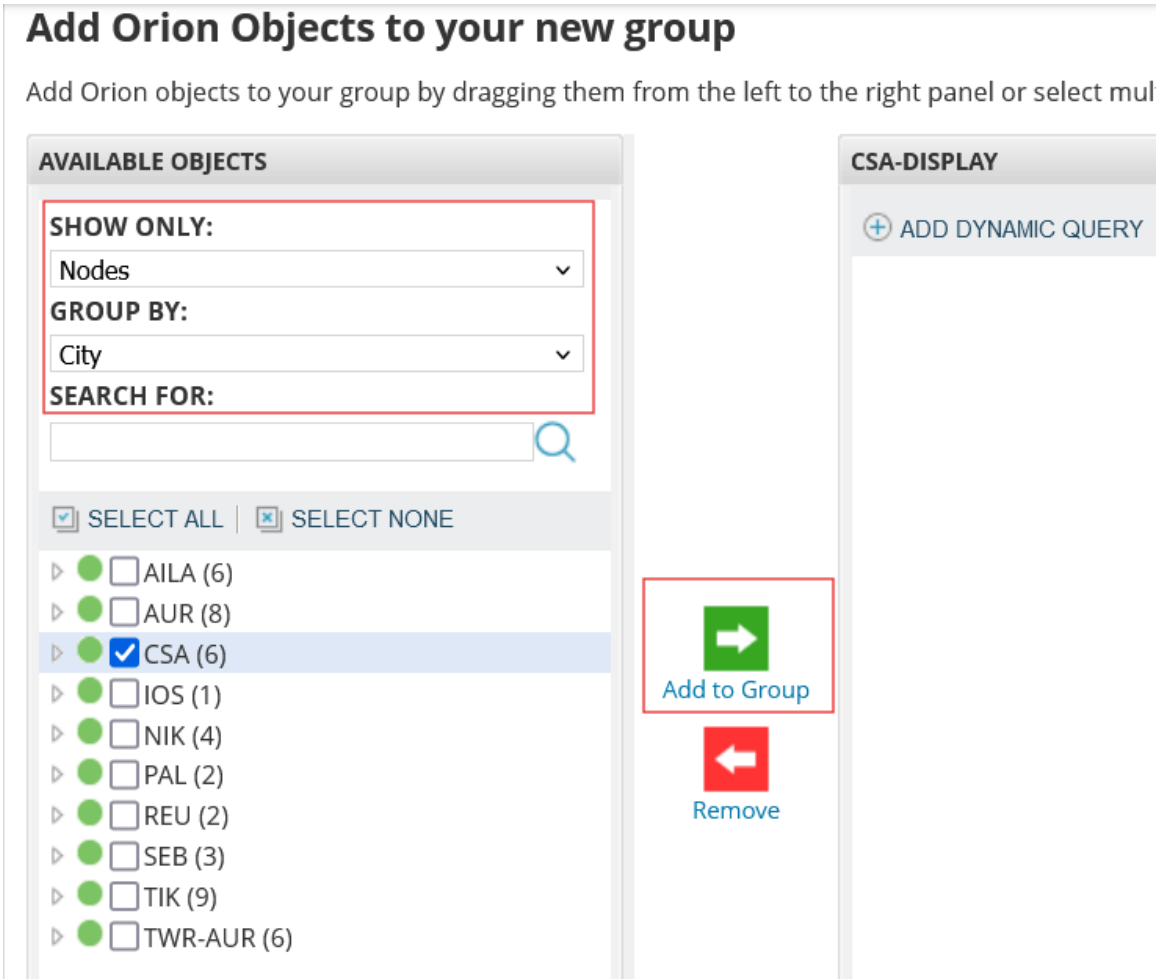


Figura 15: Filtros para la selección de nodos a asignar a un grupo

4. **Validación de configuración:** finalmente, se revisa la configuración de cada grupo para asegurar que todos los nodos estén correctamente organizados. Se puede verificar que se hayan agregado de manera correcta si los equipos aparecen dentro del grupo como se muestra en la Figura 16.

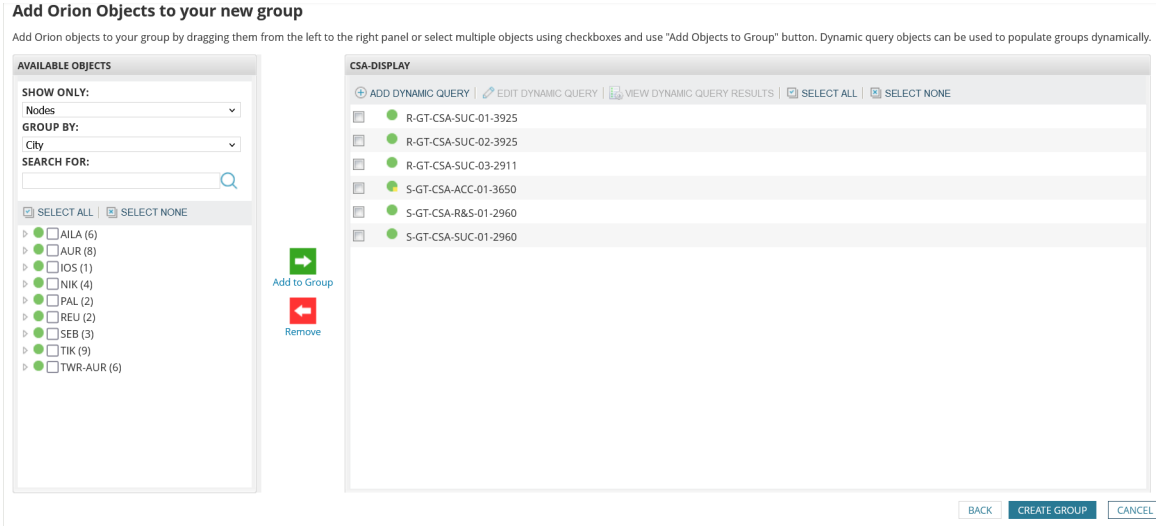


Figura 16: Nodos agregados a un grupo por medio de filtros

5. **Agregar nodos de manera automarica a los grupos:** para evitar tener que agregar los nodos manualmente a los grupos, se puede configurar un *dynamic query* para que SW NPM agregue los nodos automáticamente a los grupos en base a sus caracteristicas. En el caso de COCESNA se recomienda que se haga en base a la ciudad en la que se encuentran ya que al agregar un nodo nuevo, se le debe asignar una obligatoriamente, un ejemplo de esto se puede visualizar en la Figura 17.

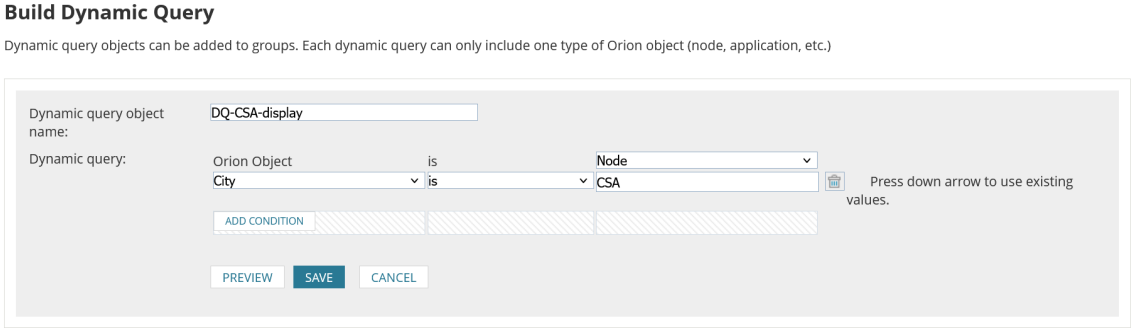


Figura 17: Parametros para la creación de un *dynamic query*

Este proceso asegura que los nodos se encuentren clasificados correctamente, facilitando el monitoreo eficiente y la identificación rápida de problemas en las diferentes regiones de la red.

7.4. Definición de dependencias entre nodos

Uno de los elementos clave en la configuración de la plataforma SolarWinds fue la creación de dependencias entre nodos. Las dependencias permiten gestionar las alertas de manera más efectiva, reduciendo la cantidad de alertas innecesarias y ayudando a identificar rápidamente el origen de los problemas. Para simplificar el proceso, se utilizó la herramienta de cálculo automático de dependencias, que permite definir las relaciones entre nodos de forma automática, optimizando así la gestión de alertas y la priorización de problemas. Esta funcionalidad es especialmente importante en redes grandes donde una falla en un nodo principal puede generar una cascada de alertas que dificulte la identificación del problema real.

Para activar la herramienta de cálculo automático de dependencias, se siguió el siguiente proceso.

1. **Acceder a la pestaña de dependencias:** entrar a la ventana de "Settings" se debe seleccionar la opción de "Mange Dependencies", como muestra la Figura 18 para crear las dependencias entre los nodos.

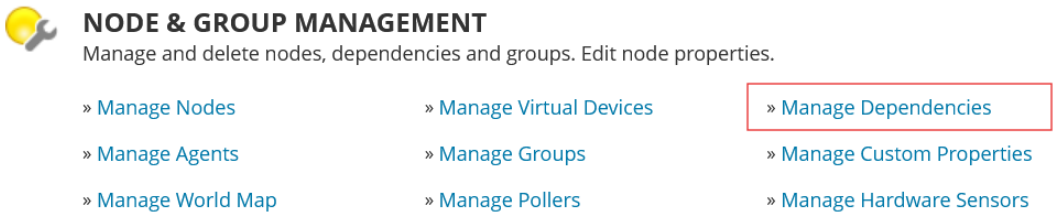


Figura 18: Menú de configuración de dependencias en SW NPM

2. **Configuración de dependencias:** dentro de la ventana de "Manage Dependencies", es posible visualizar todas las dependencias que se están utilizando en la red. Para la generación automática de dependencias, se debe seleccionar la opción de "Add Automatic Dependencies". En la Figura 19 se puede visualizar la ventana de configuración de dependencias y en rojo el switch para activar la generación automática de dependencias.

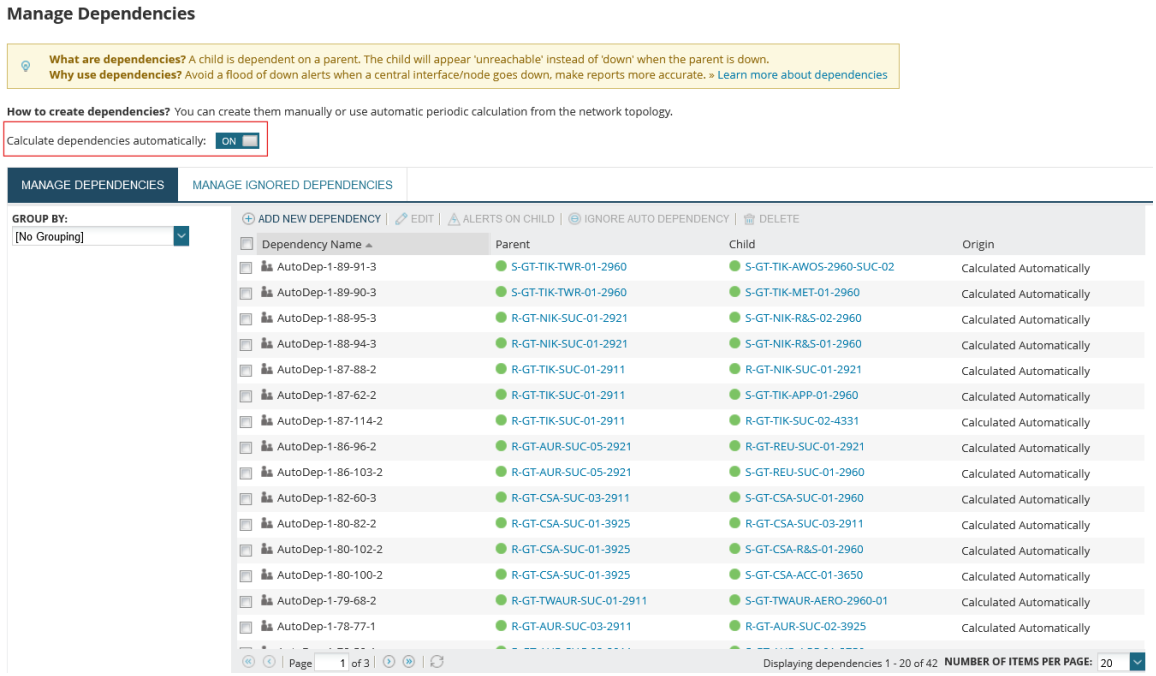


Figura 19: Ventana de configuración de dependencias en SW NPM

La definición de dependencias también permitió una mejor priorización de las acciones correctivas. Al identificar nodos críticos y definir sus dependencias, se facilitó la tarea de determinar qué problemas debían ser atendidos primero y cuál era la raíz de un problema específico. Esto aseguró que los recursos de resolución de problemas se utilizaran de manera óptima y que los incidentes se resolvieran de manera rápida y efectiva, minimizando así el tiempo de inactividad en la red.

7.5. Consideraciones finales

El proceso de configuración de los nodos y las interfaces en la plataforma SW NPM fue diseñado para maximizar la eficiencia del monitoreo y asegurar que los elementos críticos de la red estuvieran siempre supervisados. Utilizando una combinación de descubrimiento automático, filtros personalizados y la organización mediante grupos y dependencias, se logró optimizar la carga sobre el motor de polling y mejorar la capacidad de respuesta del sistema frente a eventos de la red.

Estas configuraciones proporcionan una mejor visibilidad del estado de la infraestructura y promueven un enfoque preventivo para la gestión de fallos, lo cual es esencial para mantener la estabilidad y el rendimiento de la red de telecomunicaciones de COCESNA. Al aprovechar herramientas de automatización y una organización jerárquica de los nodos, se pudo reducir significativamente el trabajo manual y mejorar la precisión del monitoreo. Esto garantizó que los recursos de la red se utilizaran de manera óptima y que las fallas se detectaran y resolvieran antes de que afectaran los servicios críticos.

Además, la capacidad de adaptar los parámetros de monitoreo según las necesidades específicas de la red proporcionó una flexibilidad fundamental para mantener la eficiencia operativa a largo plazo. La combinación de un monitoreo proactivo, una organización eficiente y una respuesta rápida y precisa a los incidentes contribuyó significativamente a la calidad y fiabilidad del sistema de telecomunicaciones gestionado por COCESNA.

CAPÍTULO 8

Configuración de alertas por correo

En este capítulo se describe el proceso de configuración de alertas por correo en la plataforma SolarWinds, asegurando una comunicación efectiva cuando se presentan eventos importantes en la red. La configuración de alertas adecuadas permite a los administradores de red recibir notificaciones en tiempo real, lo cual es esencial para mantener la estabilidad de la infraestructura y reaccionar rápidamente ante fallos o problemas que pudieran afectar el servicio.

8.1. Configuración de un servidor SMTP

Para poder enviar alertas por correo electrónico, es necesario configurar un servidor SMTP (*Simple Mail Transfer Protocol*) en SW NPM. Este servidor actuará como intermediario para enviar los correos generados por la plataforma a los destinatarios adecuados.

1. Acceder a la sección de configuración del servidor en la consola de SolarWinds.
2. Navegar a "Settings" (Configuraciones) y seleccionar "All Setting" 20.
3. En el apartado de "Notification Settings", seleccionar "SMTP Server" para configurar el servidor de correo saliente.
4. Ingresar la dirección del servidor SMTP, el puerto correspondiente (generalmente 25 o 587) y definir si se requiere autenticación 21.
5. Ingresar las credenciales necesarias si el servidor requiere autenticación (nombre de usuario y contraseña) 21.

6. Probar la configuración enviando un correo de prueba para asegurar que todos los parámetros son correctos y que la conexión es exitosa.

Una correcta configuración del servidor SMTP garantiza que las alertas puedan ser enviadas sin interrupciones, asegurando que el personal de monitoreo reciba notificaciones en tiempo real. Es importante tener en cuenta que **no se puede utilizar una dirección de correo con autenticación de dos factores (2FA)**, debido a la forma en la que SW NPM conecta con el servidor SMTP. Esta limitación se debe a que SW necesita una conexión directa y continua con el servidor SMTP, la cual no es compatible con los métodos de verificación adicionales que impone el 2FA.

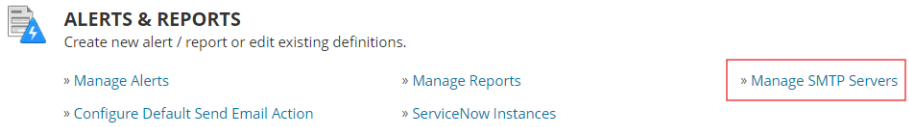


Figura 20: Configuración para protocolo SMTP

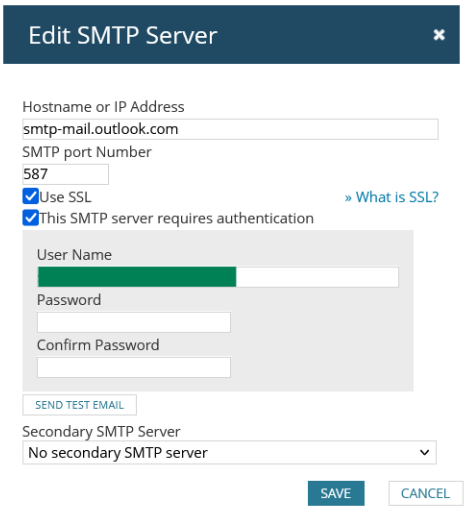


Figura 21: Ajustes para la configuración del servidor SMTP (Outlook)

8.2. Agregar una dirección de correo

El siguiente paso para configurar las alertas es agregar una dirección de correo electrónico a la cual se enviarán las notificaciones. Esta dirección puede ser individual o corresponder a un grupo de personas, dependiendo del tipo de alerta y el nivel de importancia del evento.

1. Acceder a la configuración de alertas en la consola de SW NPM.
2. Seleccionar la opción de agregar una nueva acción de correo electrónico.

3. Ingresar la dirección de correo y configurar los parámetros necesarios, como el asunto y el cuerpo del mensaje.

Es importante incluir información relevante en el cuerpo del correo, como el nombre del nodo afectado, la hora en que ocurrió el evento y el tipo de problema detectado. De esta manera, se asegura que el personal de monitoreo tenga suficiente contexto para tomar decisiones rápidas.

8.3. Cuerpo del correo

Al configurar el contenido del correo, es fundamental asegurarse de que el mensaje sea claro y contenga toda la información relevante. Para lograr esto, se utilizan variables de SW NPM que permiten incluir datos específicos del nodo o evento en el mensaje, tales como la IP del dispositivo, el nombre del nodo, y el estado del evento.

Estas variables permiten automatizar la generación del mensaje de alerta y personalizarlo de acuerdo con cada tipo de evento, lo cual facilita el análisis de la situación por parte del equipo de monitoreo.

Hay dos maneras de redactar el mensaje que va dentro del correo: texto simple o HTML. Para este proyecto se utilizó HTML, ya que permitía dar mejor formato al correo, haciéndolo más comprensible a simple vista para los destinatarios. El uso de HTML permitió incluir elementos visuales, como encabezados, listas y colores, que mejoran la organización del contenido, facilitando la identificación rápida de la información relevante y asegurando que las alertas sean interpretadas correctamente por los destinatarios. En la Figura 22 se puede apreciar un correo enviado con la configuración del Cuadro 2.

```
<html lang="es">
  <body>
    <p><span style="font-weight: bold;">Fecha de Activacion de la Alerta:</
        span> ${N=Alerting;M=AlertTriggerTime;F=DateTime}</p>

    <p>Se ha detectado una alta perdida de paquetes en el nodo <span style="
        font-weight: bold;">${N=SwisEntity;M=Caption}</span>, ubicado en <
        span style="font-weight: bold;">${N=SwisEntity;M=CustomProperties.
        City}</span>. El porcentaje de perdida de paquetes es de <span style
        ="color: red;">${PercentLoss}</span>%.</p>

    <p><span style="font-weight: bold;">Tiempo de respuesta promedio:</span>
         ${AvgResponseTime}</p>

    <p>El tiempo de respuesta varia en el siguiente rango:</p>
    <ul>
      <li><span style="font-weight: bold;">Minimo:</span> ${MinResponseTime}
          </li>
      <li><span style="font-weight: bold;">Maximo:</span> ${MaxResponseTime}
          </li>
    </ul>

    <hr>
```

```
16      <p>Para mas informacion sobre el nodo, haz clic en <a href="${
            NodeDetailsURL}" style="color: blue; text-decoration: underline;">
            aqui</a>.</p>
17      <p>Para confirmar la alerta, haz clic en <a href="${N=Alerting;M=
            AcknowledgeLink}" style="color: blue; text-decoration: underline;">
            aqui</a>.</p>
18      <hr>
19    </body>
20  </html>
```

Cuadro 2: Cuerpo de correo generado por SolarWinds en formato HTML

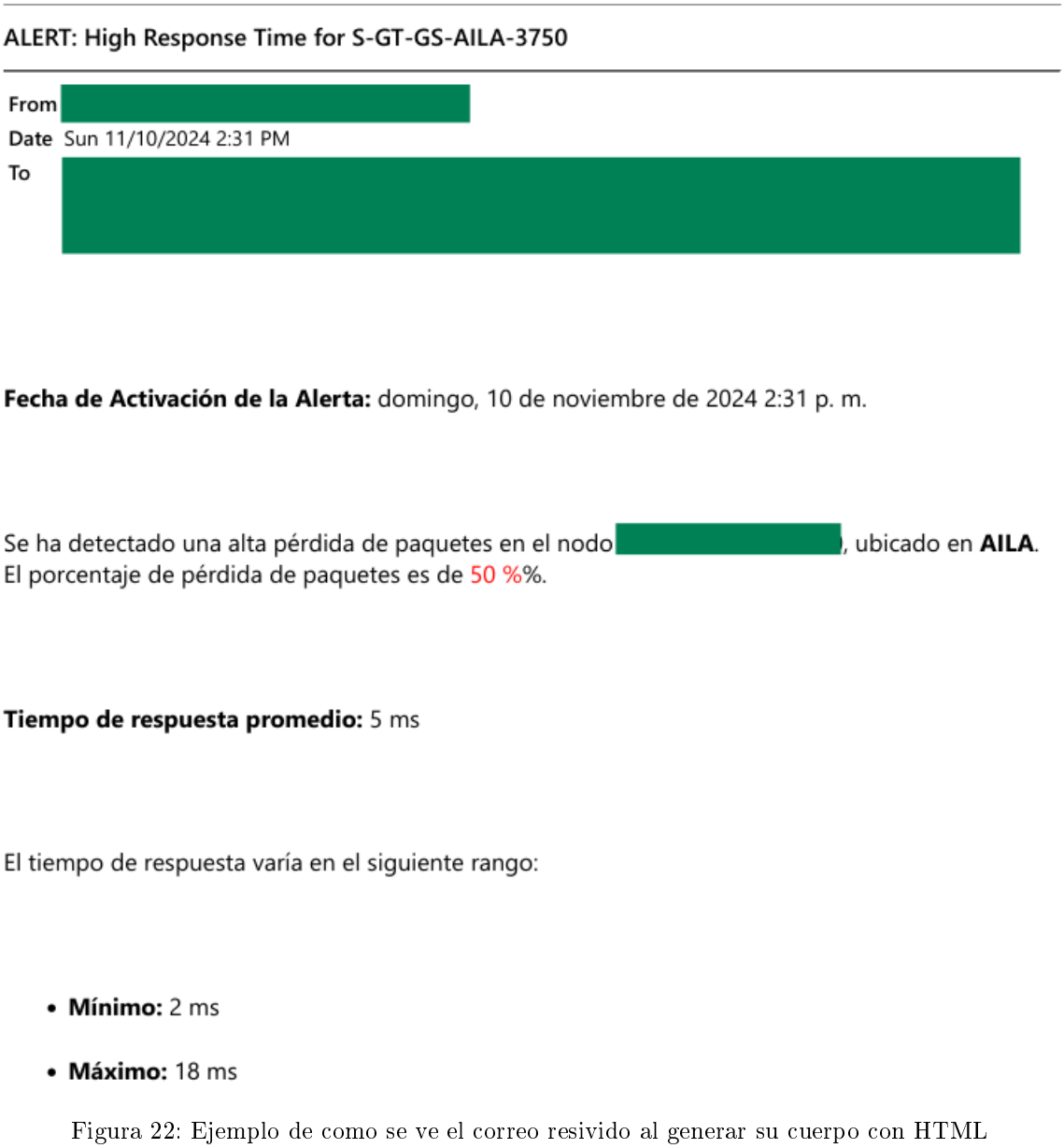
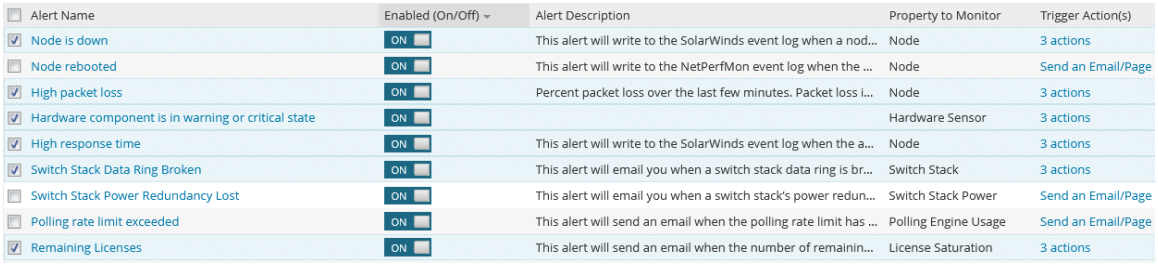


Figura 22: Ejemplo de como se ve el correo resivido al generar su cuerpo con HTML

8.4. Escalamiento de alertas y mejores prácticas

Uno de los aspectos más importantes al configurar alertas es la capacidad de escalamiento. En redes grandes y críticas como la gestionada por COCESNA, no todos los problemas tienen el mismo nivel de urgencia, por lo que es fundamental poder priorizar las alertas y escalarlas a distintos niveles de personal según la gravedad del problema.

- **Escalamiento de alertas:** se configuró que ciertas alertas de alta criticidad fueran enviadas inicialmente al personal de monitoreo. Si la alerta no se atiende dentro de un plazo determinado, se escala a un nivel superior, notificando a los supervisores o gerentes. Esto asegura una respuesta rápida y efectiva, especialmente cuando se trata de problemas que pueden afectar servicios esenciales.
- **Priorización de alertas:** se desactivaron alertas para evitar la saturación de alertas ("*alert fatigue*"). Solo se configuran notificaciones para eventos que requieren intervención humana, Figura 23. De esta manera, se evita que el personal se vea abrumado por mensajes innecesarios, permitiendo un enfoque en los incidentes que realmente requieren atención.



Alert Name	Enabled (On/Off)	Alert Description	Property to Monitor	Trigger Action(s)
Node is down	ON	This alert will write to the SolarWinds event log when a nod...	Node	3 actions
Node rebooted	ON	This alert will write to the NetPerfMon event log when the ...	Node	Send an Email/Page
High packet loss	ON	Percent packet loss over the last few minutes. Packet loss i...	Node	3 actions
Hardware component is in warning or critical state	ON		Hardware Sensor	3 actions
High response time	ON	This alert will write to the SolarWinds event log when the a...	Node	3 actions
Switch Stack Data Ring Broken	ON	This alert will email you when a switch stack data ring is br...	Switch Stack	3 actions
Switch Stack Power Redundancy Lost	ON	This alert will email you when a switch stack's power redun...	Switch Stack Power	Send an Email/Page
Polling rate limit exceeded	ON	This alert will send an email when the polling rate limit has ...	Polling Engine Usage	Send an Email/Page
Remaining Licenses	ON	This alert will send an email when the number of remainin...	License Saturation	3 actions

Figura 23: Selección de alertas necesarias para notificación por correo

- **Mejores prácticas:** se recomienda realizar pruebas periódicas de las configuraciones de alerta para asegurar que están funcionando correctamente. También es importante actualizar las reglas de escalamiento conforme evoluciona la infraestructura, de manera que los cambios en la red estén alineados con las políticas de notificación.

8.5. Configuración de las reglas de escalamiento

Para implementar el escalamiento de alertas, es necesario definir reglas de escalamiento dentro de la plataforma SW NPM. Estas reglas permiten que la alerta sea reenviada o escalada si no ha sido atendida en un periodo definido.

1. Acceder a la sección de configuración avanzada de alertas.
2. Seleccionar la opción de agregar una nueva regla de escalamiento, Figura 24.

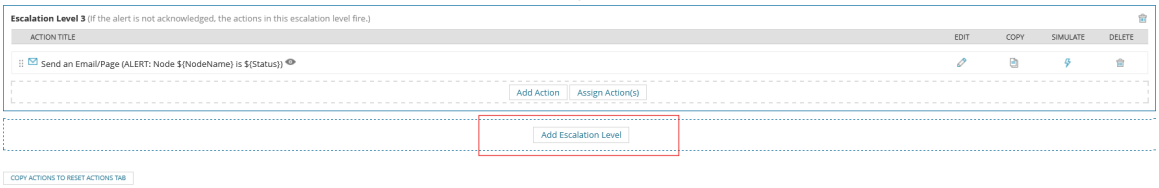


Figura 24: Configuración para agregar escalamiento a las alertas en SW NPM

3. Definir el primer nivel de notificación, especificando al personal de monitoreo.

4. Configurar la regla de escalamiento indicando el tiempo de espera antes de reenviar la alerta a un nivel superior, 25

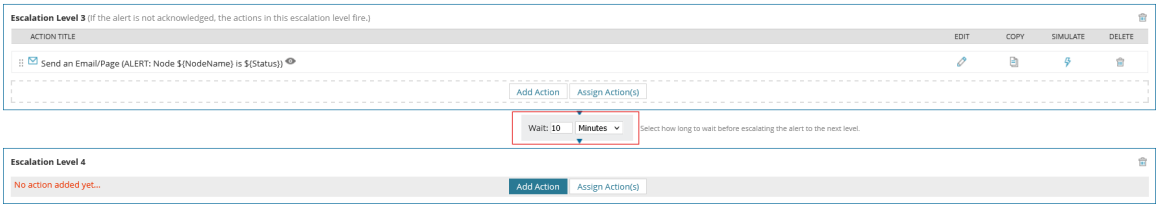


Figura 25: Ajustes de tiempo para activar el escalamiento, peude estar en minutos, horas o segundos

5. Repetir este proceso para tantos niveles de escalamiento como sea necesario. En el caso de COCESNA, se configuraron tres niveles de escalamiento: el primero, para el personal de monitoreo, el segundo, para los supervisores y el tercero, para los gerentes.

Este mecanismo de escalamiento es crucial para redes como la de COCESNA, donde la continuidad del servicio es fundamental y el tiempo de respuesta debe ser el mínimo posible.

8.6. Consideraciones finales

La correcta configuración de alertas por correo es un componente esencial para el monitoreo proactivo de la red de COCESNA. Al implementar estrategias de escalamiento, priorización, y uso de herramientas externas, se asegura que los problemas se detecten y se gestionen de forma rápida y efectiva. Esto no solo reduce el riesgo de interrupciones en el servicio, sino que también mejora la capacidad del equipo para responder ante eventos críticos, garantizando así la continuidad y la estabilidad de la infraestructura de telecomunicaciones.

CAPÍTULO 9

Configuración de mapas de topología en SW NPM

En este capítulo se detalla el proceso de configuración de los Mapa de topología en la plataforma SW NPM, asegurando una visualización clara y eficiente de la infraestructura de red. Los mapas de topología son herramientas fundamentales para la comprensión y monitoreo de la red, ya que permiten identificar visualmente la conectividad entre dispositivos y ayudan a detectar de manera proactiva posibles fallos o problemas de rendimiento.

9.1. Creación de mapas de topología con Orion Network Atlas

En esta sección se describe el proceso detallado para crear mapas de topología utilizando ONA (Orion Network Atlas), herramienta fundamental para la gestión y visualización de la infraestructura de red. Los mapas de topología permiten representar gráficamente la estructura de la red, identificando de manera visual el estado de los nodos y las interconexiones entre ellos, facilitando el monitoreo eficiente y el diagnóstico de problemas en la red de COCESNA.

A continuación, se presenta una guía paso a paso para crear un mapa de topología con ONA.

1. **Establecer conexión con el servidor de SW NPM:** abra la aplicación Network Atlas 26. Proporcione la dirección IP del servidor de SW NPM, el usuario y la contraseña, figura 27. Una vez establecida la conexión, podrá comenzar a crear y personalizar los mapas de topología de la red.

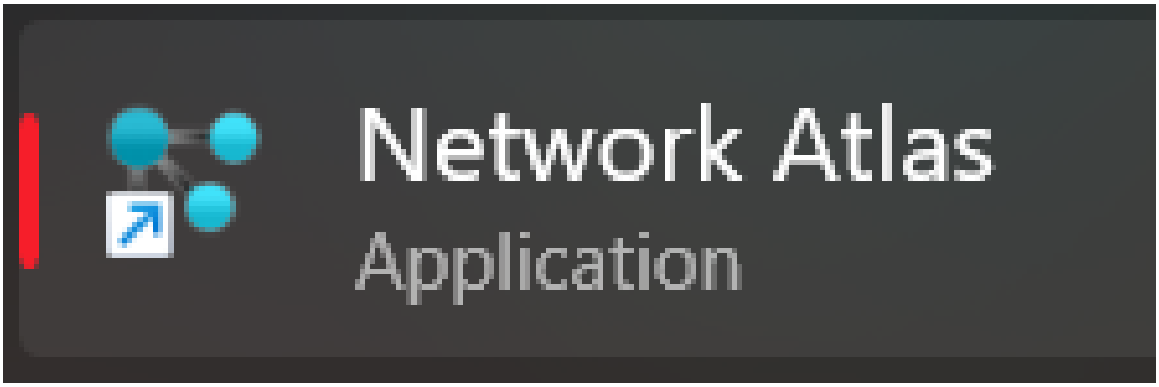


Figura 26: Logo de la herramienta Orion Network Atlas

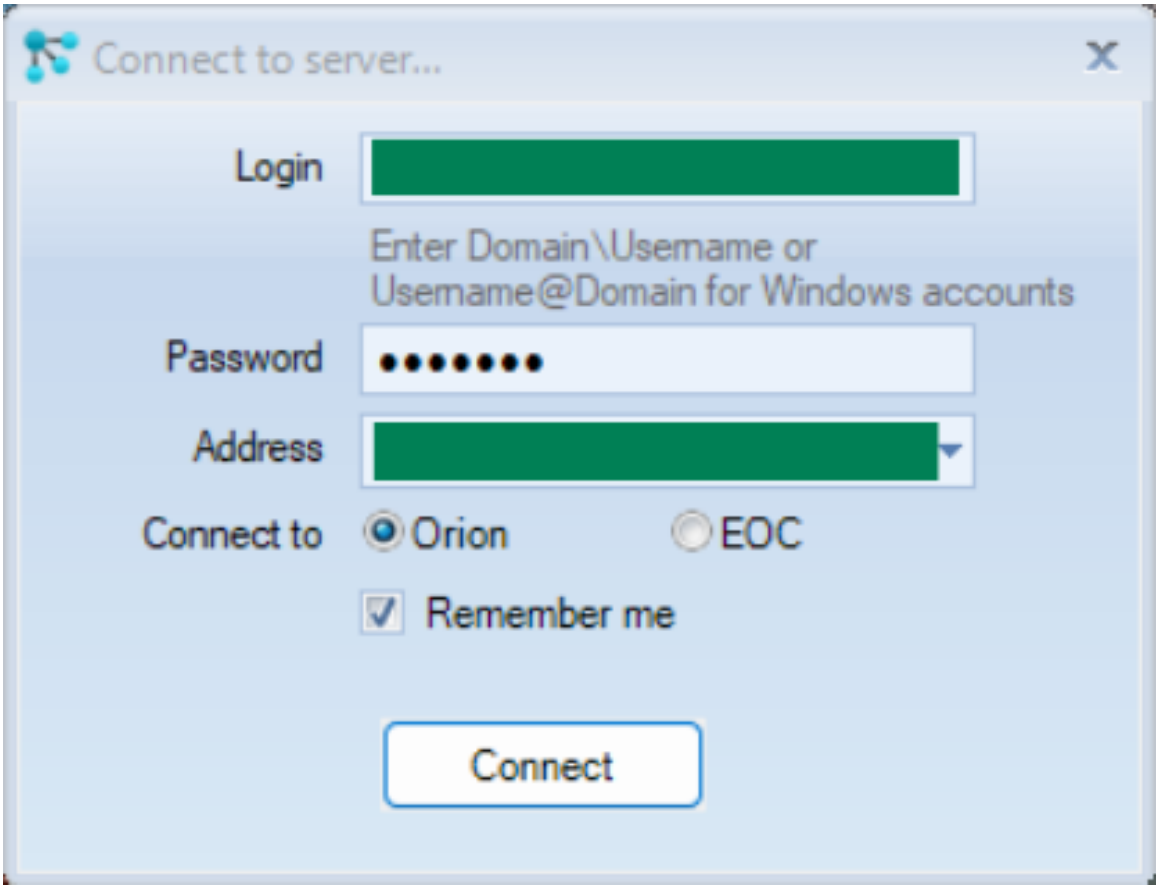


Figura 27: Conexión de Orion Network Atlas con el servidor de SolarWinds

2. **Navegar por la ventana principal de Network Atlas:** dentro de la aplicación, encontrará una ventana como la que se muestra en la Figura 28. Esta ventana posee tres elementos principales: Maps, Orion Groups y Orion Objects.

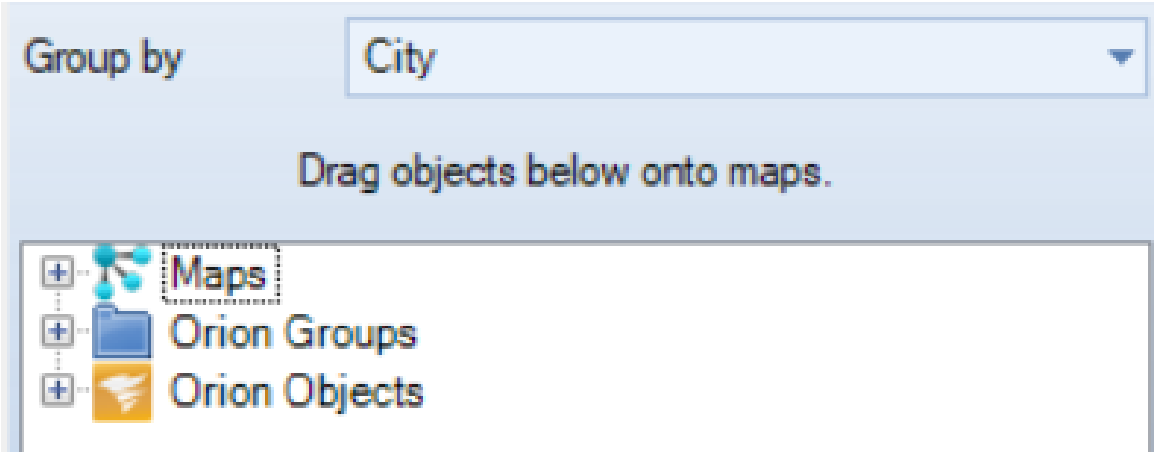


Figura 28: Categorías de dispositivos disponibles para la creación de mapas de topología

- **Maps:** contiene los mapas de topología creados previamente, Figura 29. Lo cual permite seleccionar y editar cualquiera de ellos.

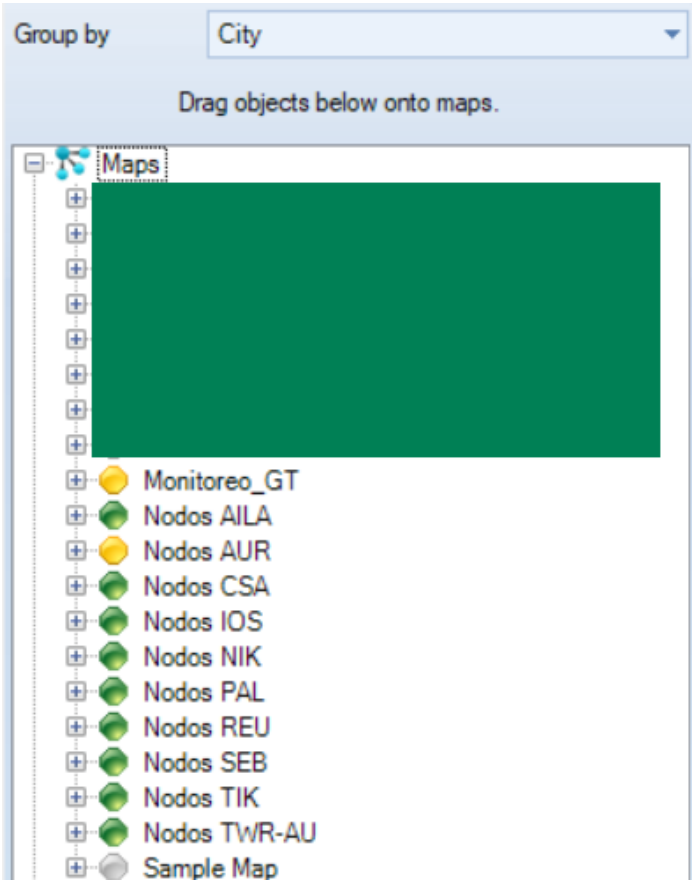


Figura 29: Categorías de mapas disponibles en Orion Network Atlas

- **Orion Groups:** representa los grupos creados utilizando la herramienta de grupos, Figura 30. Estos grupos son útiles para el mapa global, ya que el estado del grupo refleja el estado de sus miembros y están configurados para proporcionar una visión general del estado de los dispositivos.

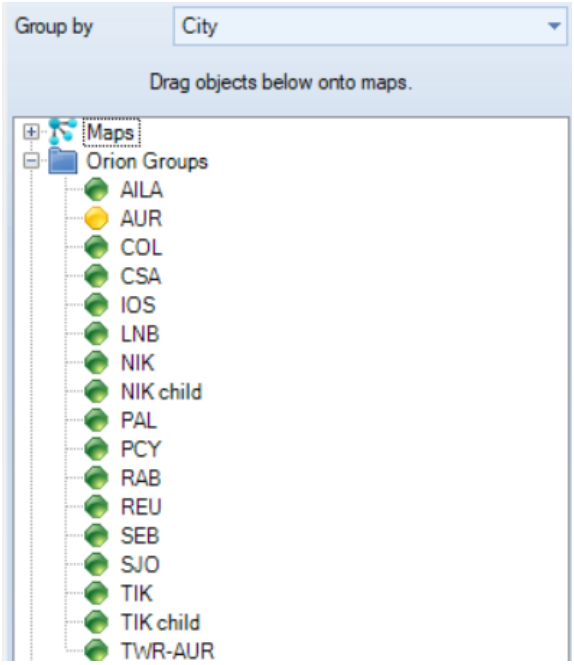


Figura 30: Categorías de grupos disponibles en Orion Network Atlas

- **Orion Objects:** contiene nodos, interfaces, sensores de voltaje y temperatura, Figura 31. Los objetos están organizados por la propiedad "City", facilitando su ubicación geográfica y gestión.

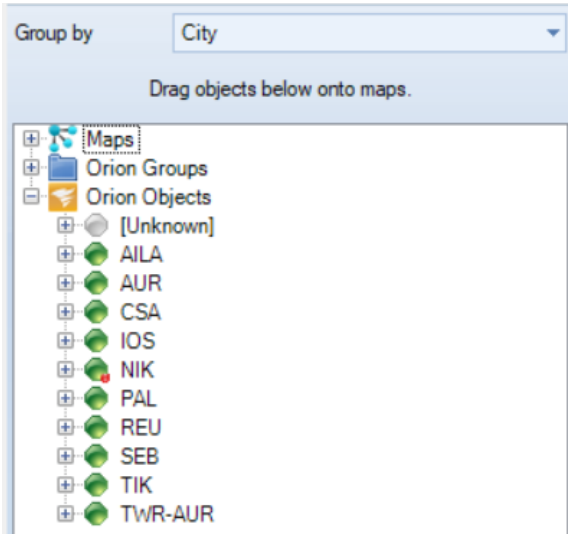


Figura 31: Categorías de objetos disponibles en Orion Network Atlas

3. **Seleccionar los nodos para el mapa:** para el propósito de esta demostración, se utilizarán los nodos pertenecientes a la estación SEB (Santa Elena Barrillas), Figura 32. Al expandir el elemento SEB, se despliegan los nodos asociados. Expanda un nodo para visualizar sus sensores e interfaces Figura 33.

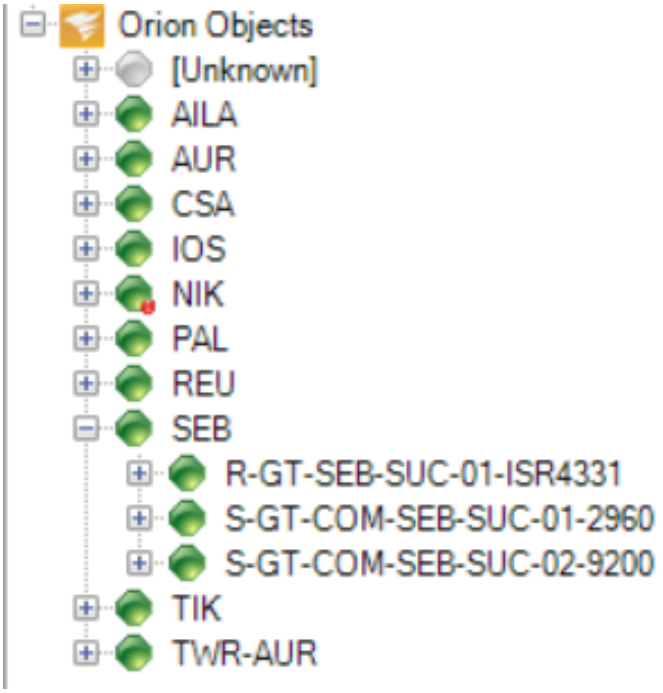


Figura 32: Nodos disponibles dentro de la categoría de objetos

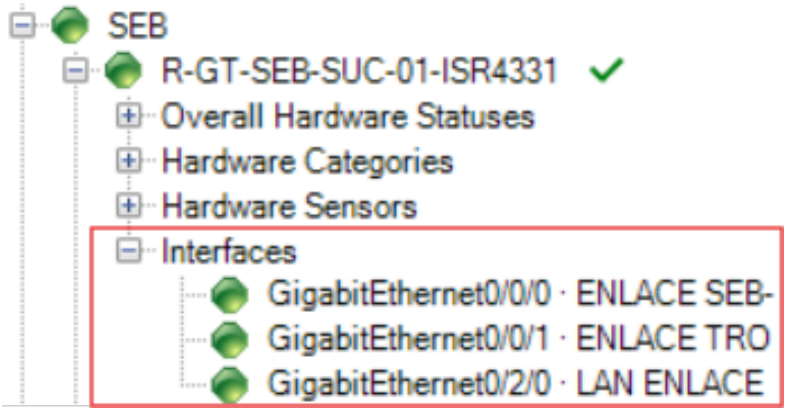


Figura 33: Interfaces disponibles dentro de la categoría de objetos

4. **Agregar elementos al mapa:** arrastre los elementos deseados (nodos, interfaces, etc.) desde la ventana de elementos al "canvas" en blanco, Figura 34. Puede agregar todos los nodos de una sola vez o seleccionarlos individualmente según lo requiera. Para agregar las interfaces, siga el mismo procedimiento que con los nodos, seleccionando la sección de interfaces del nodo deseado, Figura 35.

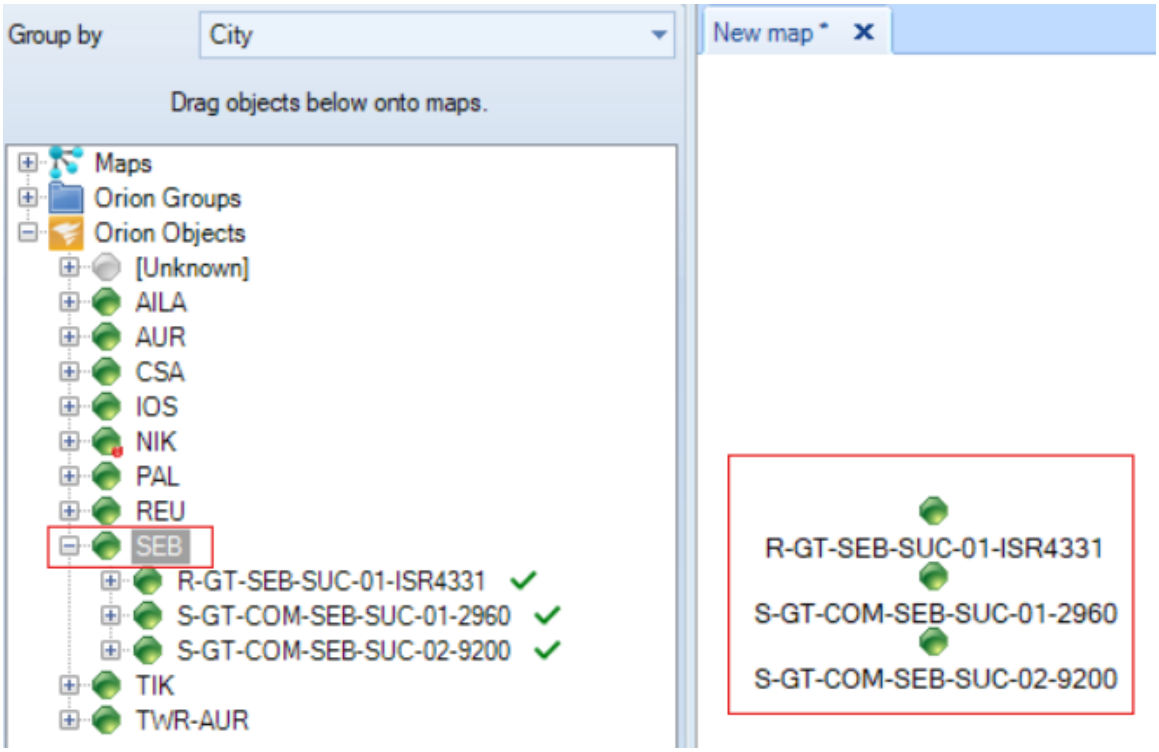


Figura 34: Arrastrar y soltar nodos en el área de trabajo de Orion Network Atlas

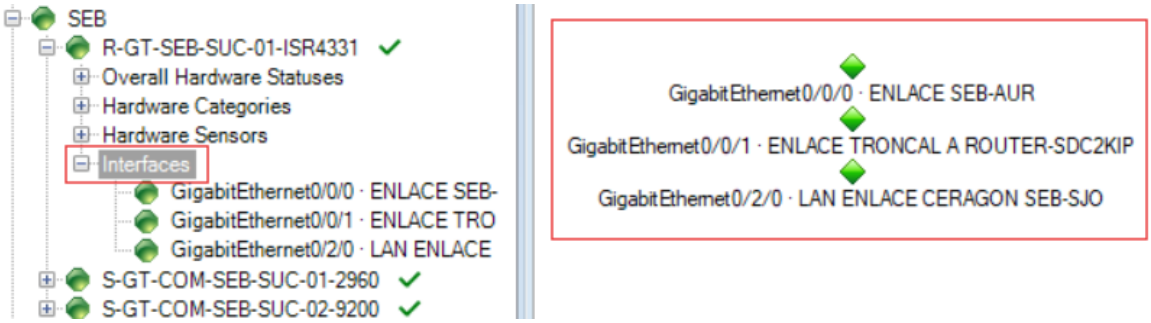


Figura 35: Inclusión de interfaces en el mapa de topología

5. **Modificar la representación de los nodos:** la representación por defecto de los nodos es un círculo, con colores que indican su estado (verde para buen estado, amarillo para falla menor, rojo para estado crítico). Para cambiar la apariencia del nodo, selecciónelo, haga clic derecho y elija "Select Graphic", Figura 36.

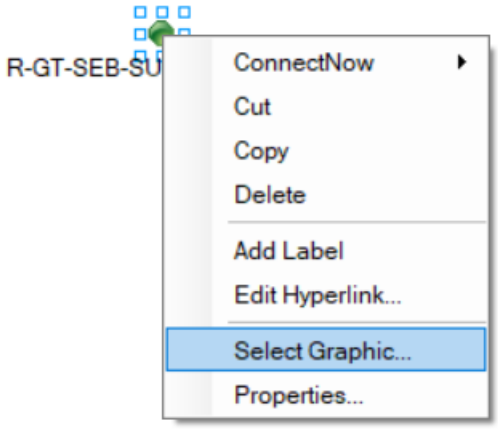


Figura 36: Personalización de los nodos en el mapa de topología

- Routers: seleccione la gráfica de "router" en la categoría "General", Figura 37.

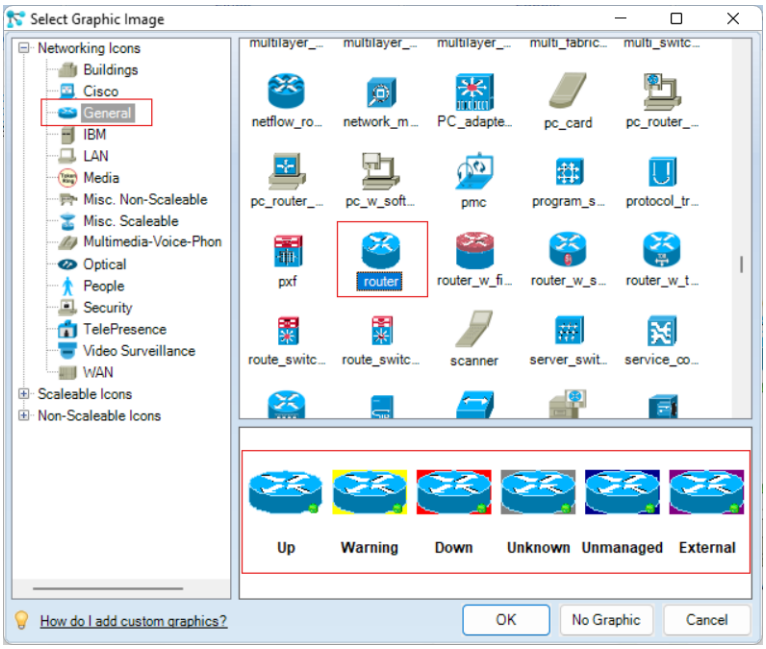


Figura 37: Modificar la representación de los routers en el mapa de topología

- Switches: seleccione la gráfica de "workgroup" en la categoría "General", Figura 38.

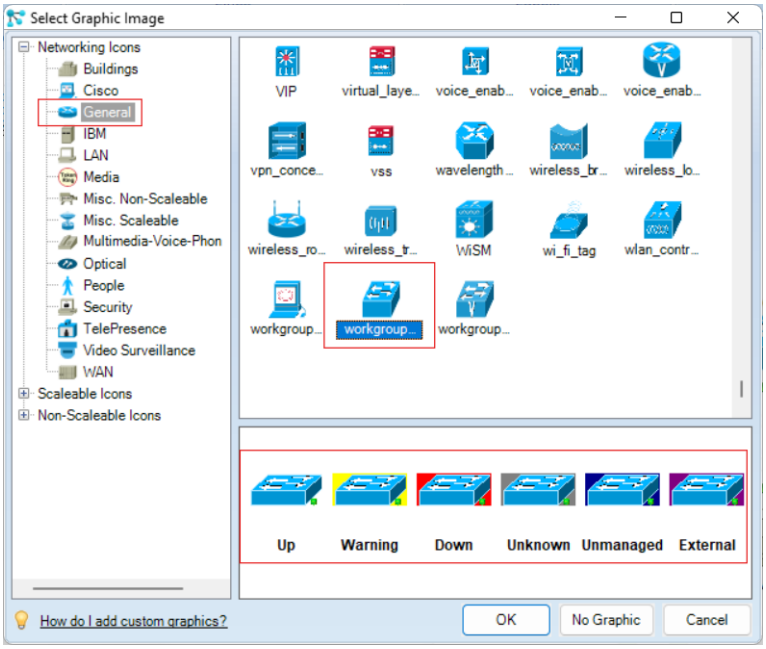


Figura 38: Modificar la representación de los switches en el mapa de topología

Los colores alrededor de la imagen indican el estado del nodo: transparente (todo bien), amarillo (alerta activa), rojo (apagado), gris (estado desconocido), azul (reconocido pero no monitoreado) y morado (elemento externo a la red).

6. **Establecer enlaces entre los nodos:** utilice la herramienta "Straight Line" para crear enlaces ("links") entre los nodos, Figura 39. **Se recomienda realizar la conexión entre interfaces para que el "link" despliegue la información correcta automáticamente** en SW NPM. Para determinar las interconexiones, utilice el comando del Cuadro 1 desde la CLI (*Command Line Interface*), lo cual le mostrará los vecinos conectados y las interfaces utilizadas.

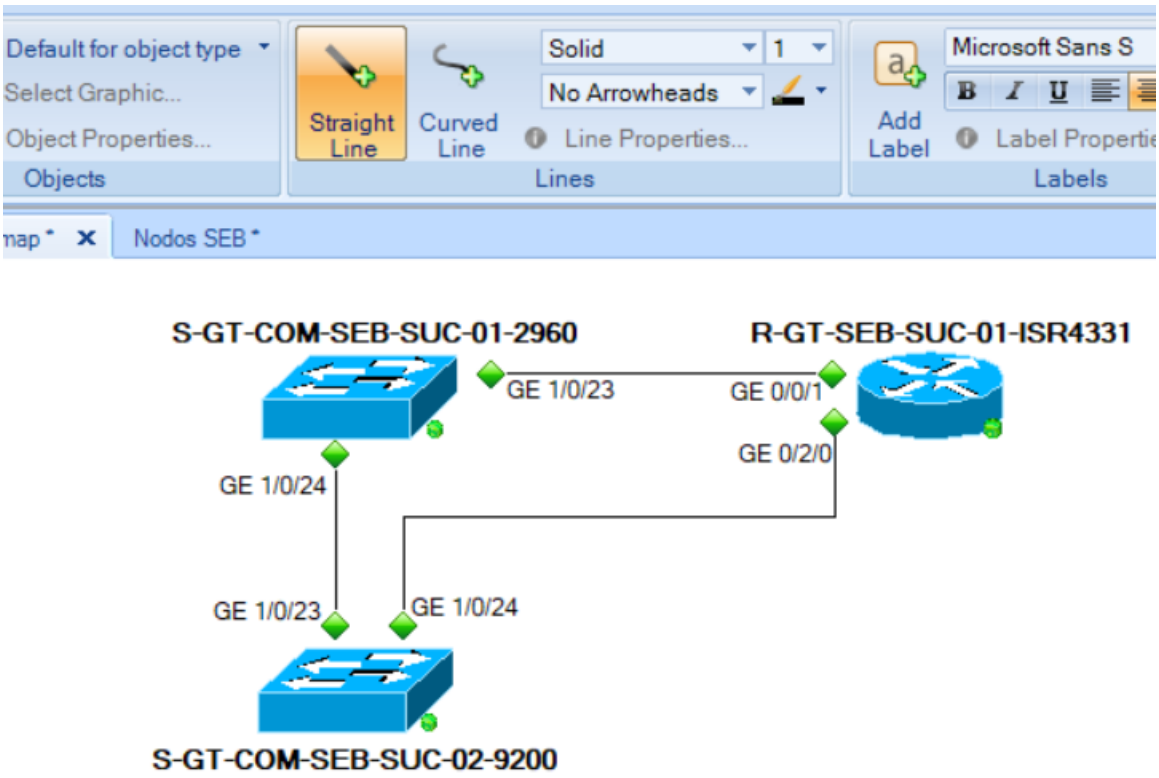


Figura 39: Creación de enlaces entre nodos en el mapa de topología

7. **Agregar funcionalidades adicionales a los enlaces:** en la viñeta "Connection Display Options", modifique las propiedades de los enlaces, Figura 40. Por defecto, no se despliega información adicional. Puede habilitar "Show Link Speed" para mostrar la velocidad de transmisión (en *bits* por segundo) o "Show Link Utilization" para mostrar la utilización del enlace. Para los mapas de COCESNA, se utilizó "Show Link Utilization" para identificar de un vistazo qué tan saturado se encuentra el enlace, Figura 41.

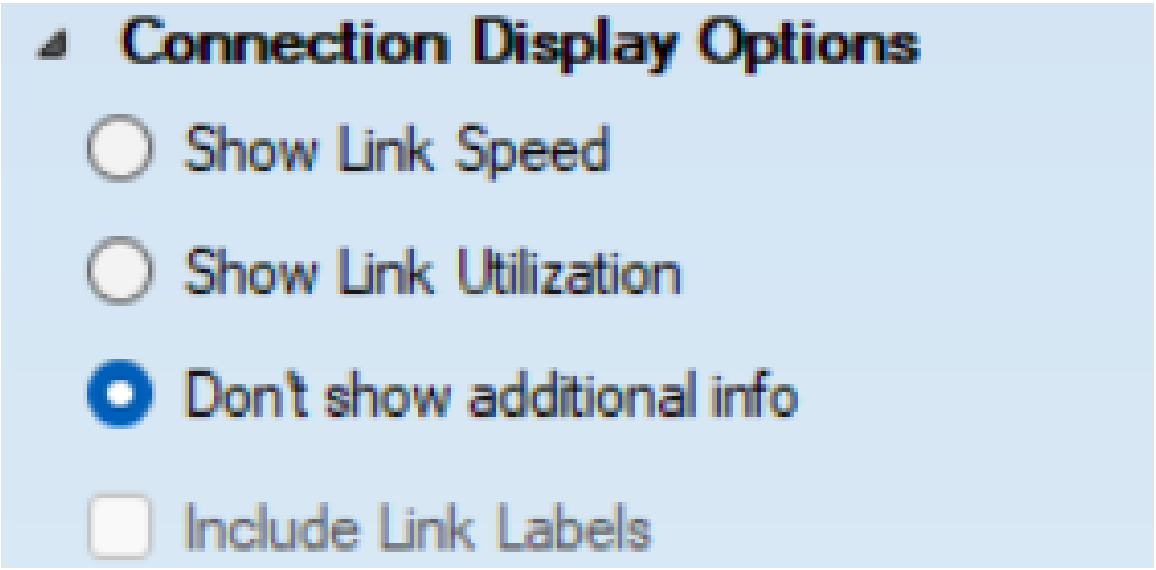


Figura 40: Personalización de los enlaces en el mapa de topología

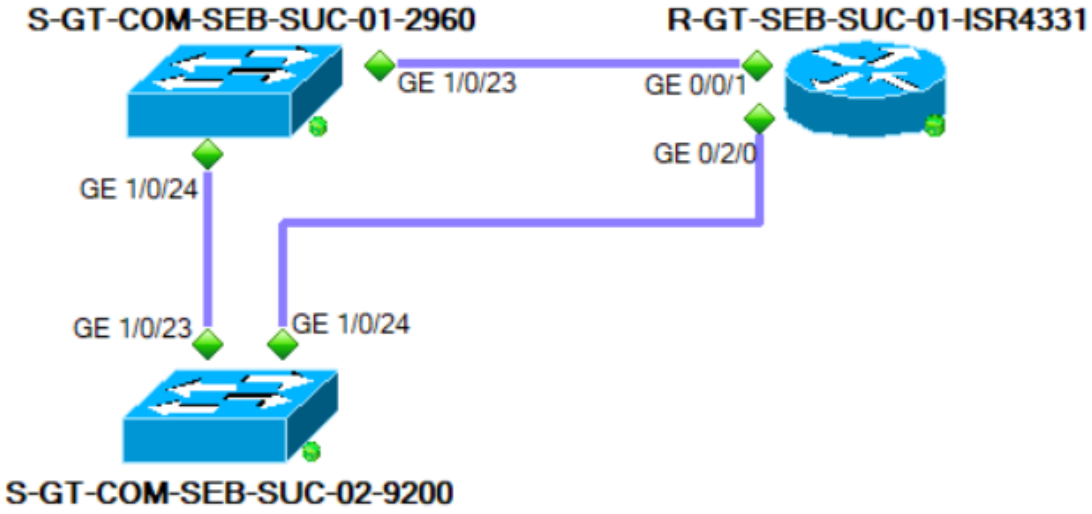


Figura 41: Mapa de topología con la utilización de enlaces

8. **Guardar el Mapa:** presione "CTRL+S" para guardar el mapa y asígnale un nombre descriptivo. En este caso, el mapa se guardó bajo el nombre "MapaEjemplo" 42.

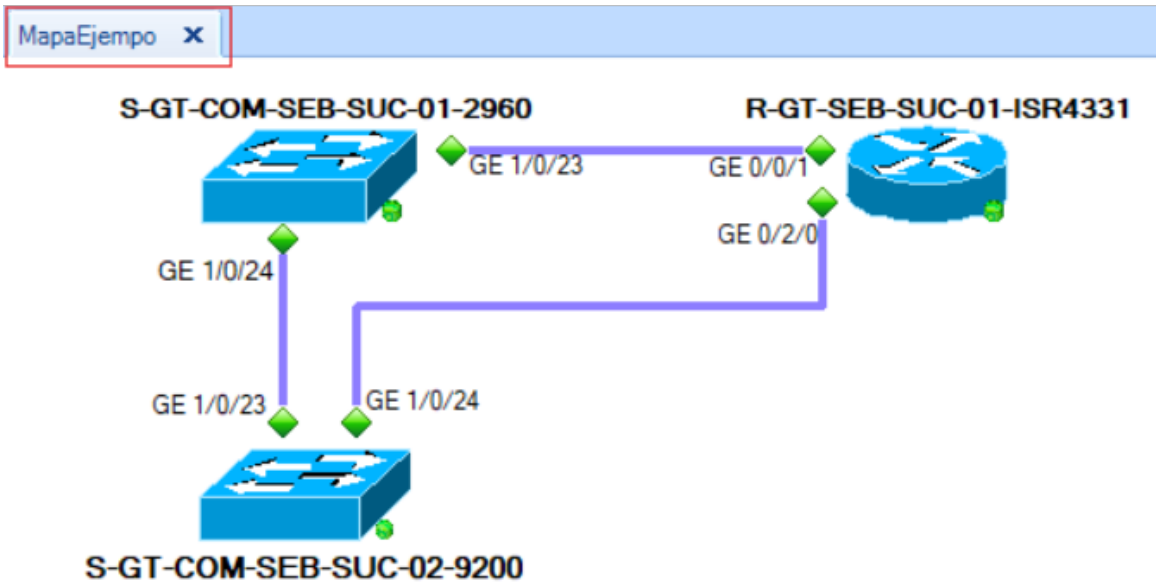


Figura 42: Mapa de topología finalizado con el nombre "MapaEjemplo"

9. **Agregar el mapa a la plataforma web de SW NPM:** acceda a la interfaz web de SW NPM. Seleccione uno de los cuadros de mapa y haga clic en "EDIT", Figura 43. En la lista de mapas creados en ONA, seleccione el mapa deseado y guarde la configuración, Figura 44. Finalmente, el mapa estará disponible para su despliegue en la plataforma web, Figura 45.

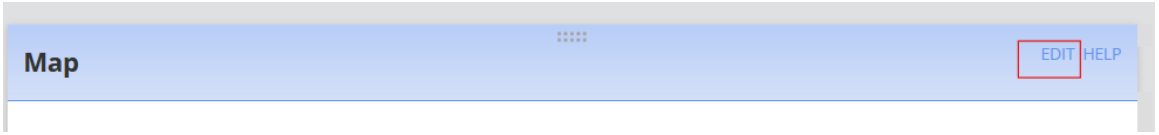


Figura 43: Cuadro de mapa dentro de la plataforma web de SolarWinds

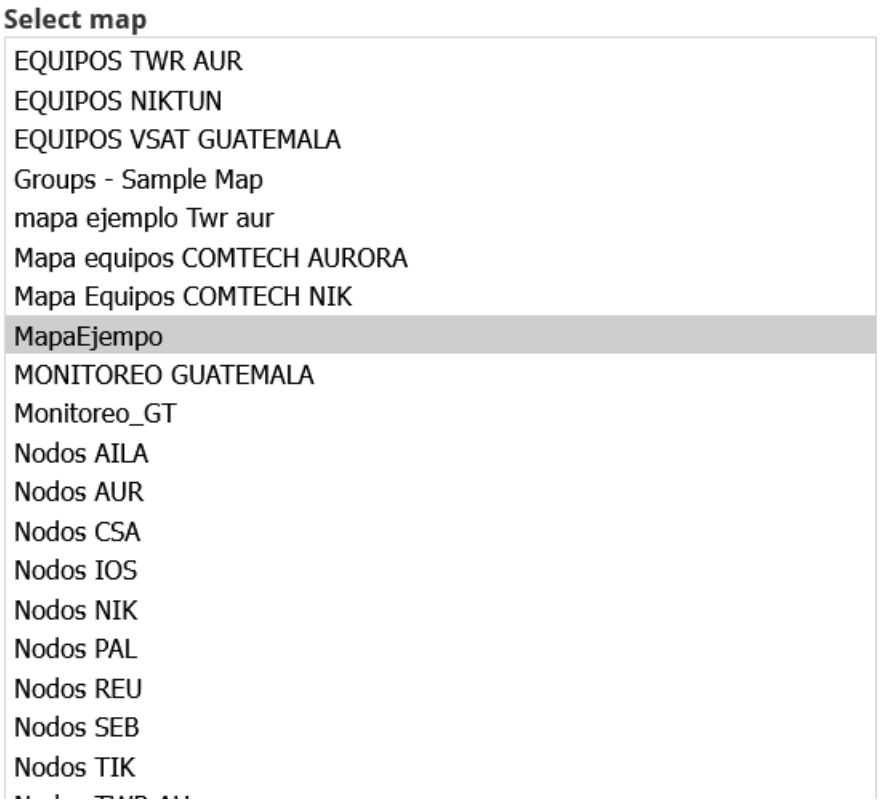


Figura 44: Selección de mapa en la plataforma web de SolarWinds

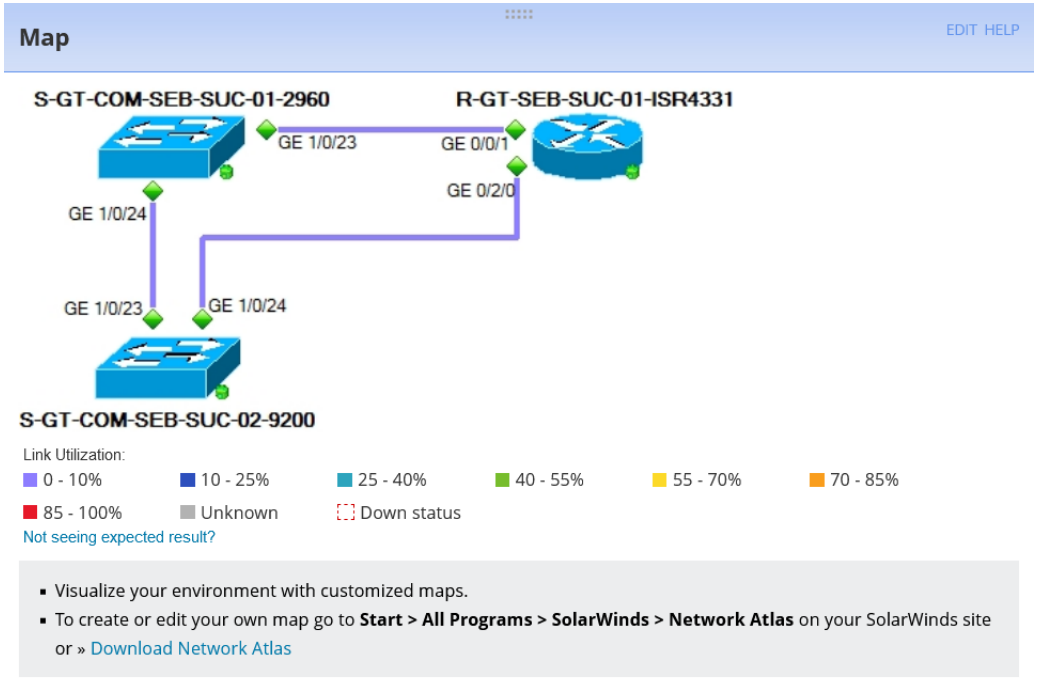


Figura 45: Despliegue del mapa "MapaEjemplo" en la plataforma web de SolarWinds

9.2. Uso de sensores de los nodos en los mapas de topología

Para mejorar la utilidad de los mapas de topología, es posible colocar elementos que muestren información en tiempo real sobre el estado de los sensores. Dichos elementos pueden ser colocados en cualquier posicion del canvas tal como si fuese un nodo o onterfaz, en la Figura 35 fuera del rectangulo rojo, se puede la opción de agregar un sensor. Algunas de las métricas que se pueden agregar incluyen:

- **Voltaje consumido:** visualizar el consumo de energía de los dispositivos ayuda a identificar posibles problemas de suministro eléctrico antes de que afecten el funcionamiento de la red.
- **Temperatura del dispositivo:** monitorear la temperatura de ciertos dispositivos puede prevenir daños permanentes debido al sobrecalentamiento, ayudando a garantizar una operación estable y segura.

Estas metricas **no son representadas numericamente**, sino que se muestran con un color que indica el estado del sensor (verde para buen estado, amarillo para falla menor, rojo para estado crítico).

9.3. Mejores prácticas para la configuración de mapas

La creación y mantenimiento de mapas de topología efectivos requiere seguir ciertas mejores prácticas:

- **Agrupar dispositivos por funcionalidad:** agrupar los dispositivos según su rol (acceso, distribución, core) facilita la visualización y permite identificar más rápidamente cualquier problema que afecte a una parte específica de la red.
- **Usar etiquetas claras:** utilizar etiquetas claras y precisas para los nodos y enlaces en el mapa es esencial para facilitar la comprensión. Las etiquetas deben incluir información relevante como el nombre del dispositivo, su ubicación y su rol en la infraestructura.
- **Pruebas periódicas:** realizar pruebas periódicas de los mapas de topología, asegurando que la información visualizada sea precisa y que todas las conexiones estén correctamente representadas. Esto implica verificar tanto la conectividad física como la lógica de los dispositivos en la red.

9.4. Ejemplos de mapas de topología realizados

Para ilustrar cómo se ven los mapas de topología creados durante este proyecto, se incluyen varias fotografías de los mapas realizados, Figuras de la 46 a la 55. Estas imágenes proporcionan ejemplos concretos de cómo se puede visualizar la infraestructura de red y la conectividad entre dispositivos en SW NPM ONA. Los mapas incluyen dispositivos críticos de la red de telecomunicaciones de COCESNA y muestran cómo se personalizó cada nodo con métricas específicas para facilitar la interpretación visual y la identificación de problemas potenciales. Estos ejemplos también sirven como evidencia del trabajo realizado en la configuración y personalización de los mapas de topología, destacando la utilidad de SW NPM para el monitoreo continuo.

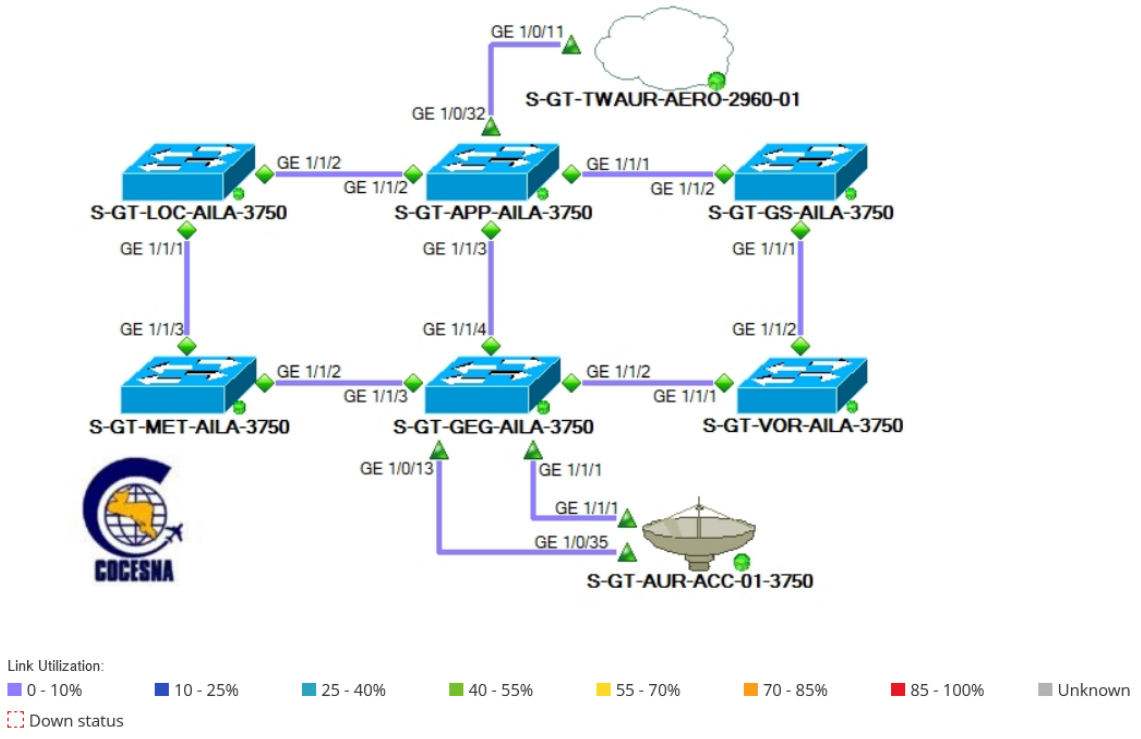


Figura 46: Mapa de topología de la estación AILA (Aeropuerto La Aurora), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

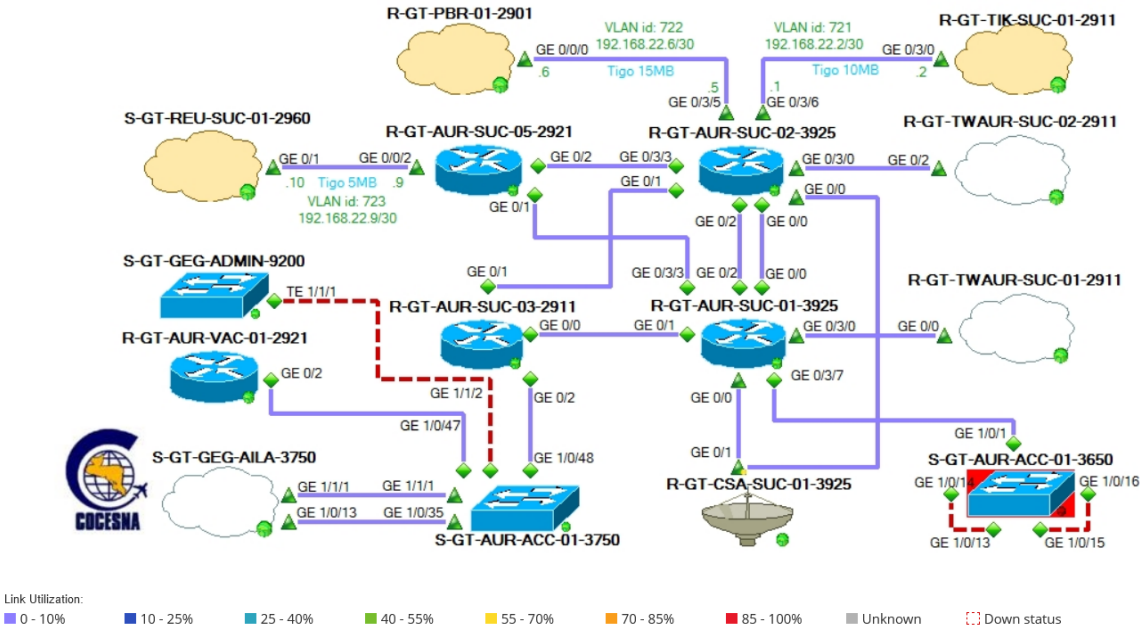


Figura 47: Mapa de topología de la estación AUR (Aeropuerto La Aurora), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

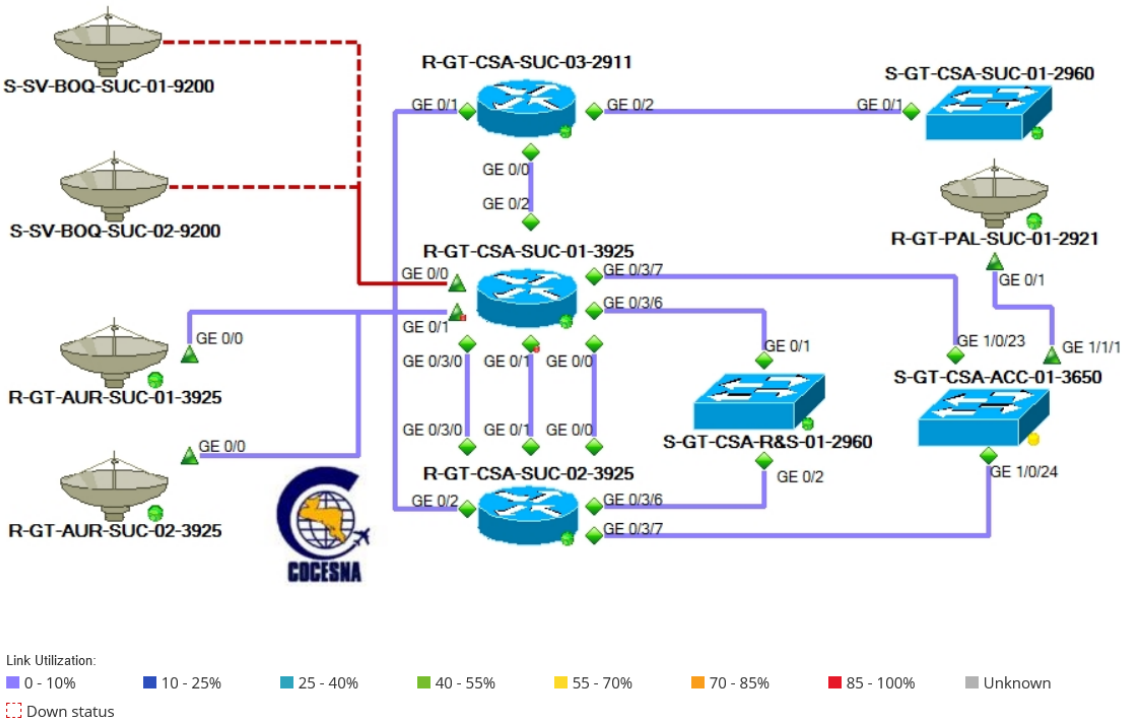


Figura 48: Mapa de topología de la estación CSA (Cerro Santiago), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

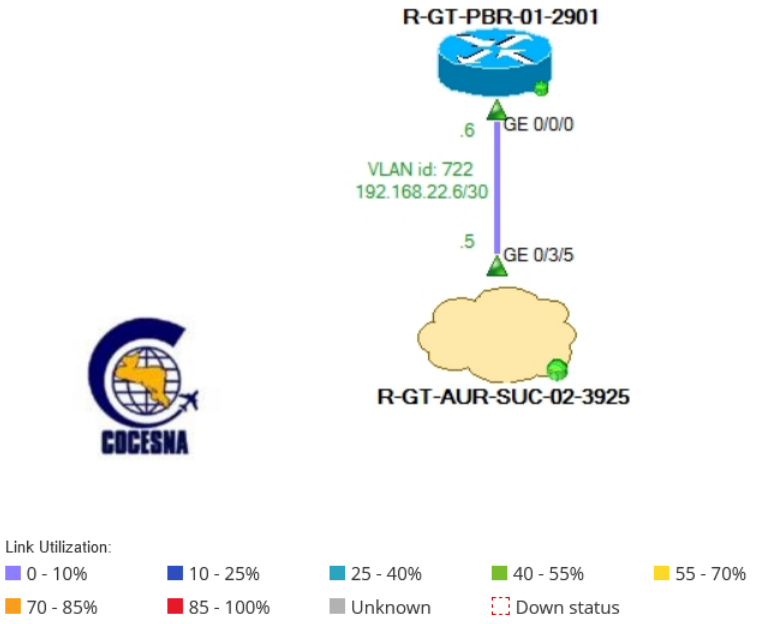


Figura 49: Mapa de topología de la estación IOS (Puerto Barrios), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

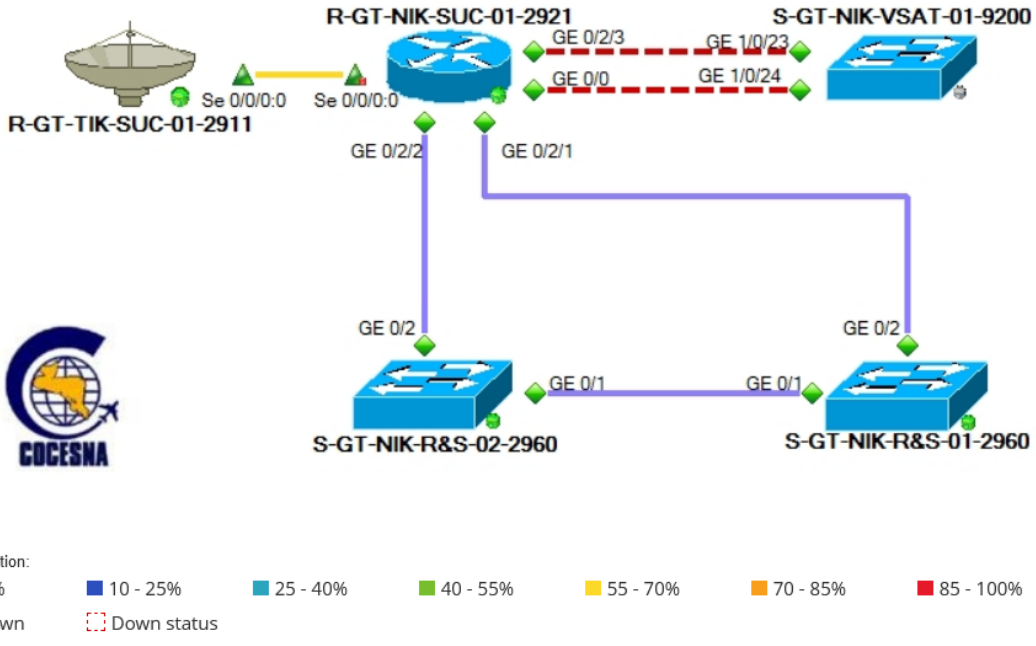


Figura 50: Mapa de topología de la estación NIK (Cerro Niktún), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

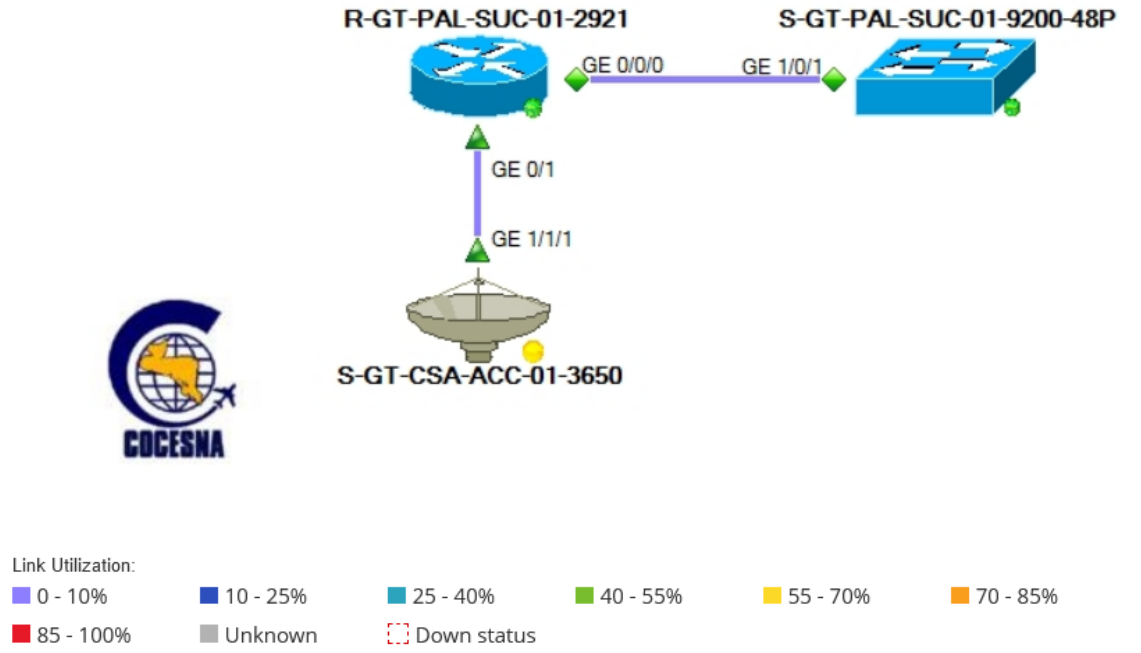


Figura 51: Mapa de topología de la estación PAL (Cerro Palencia), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

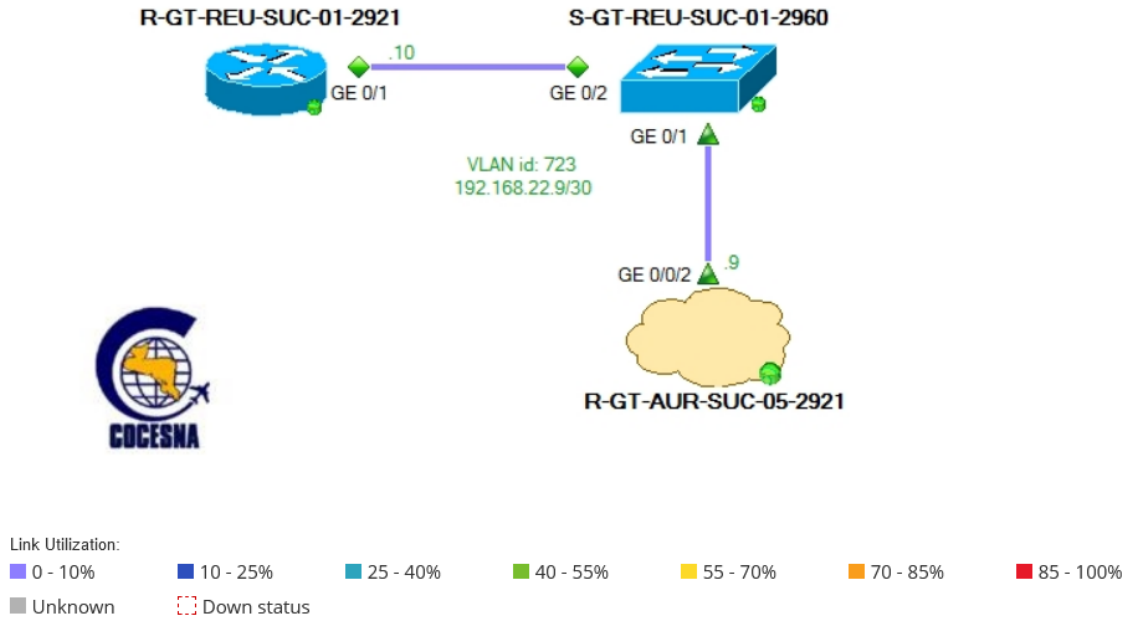


Figura 52: Mapa de topología de la estación REU (Retalhuleu), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

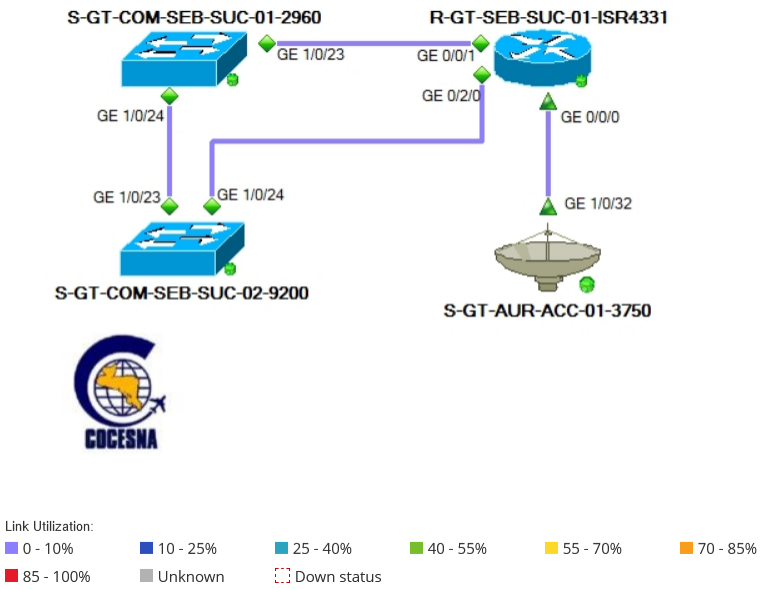


Figura 53: Mapa de topología de la estación SEB (Santa Elena Barrillas), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

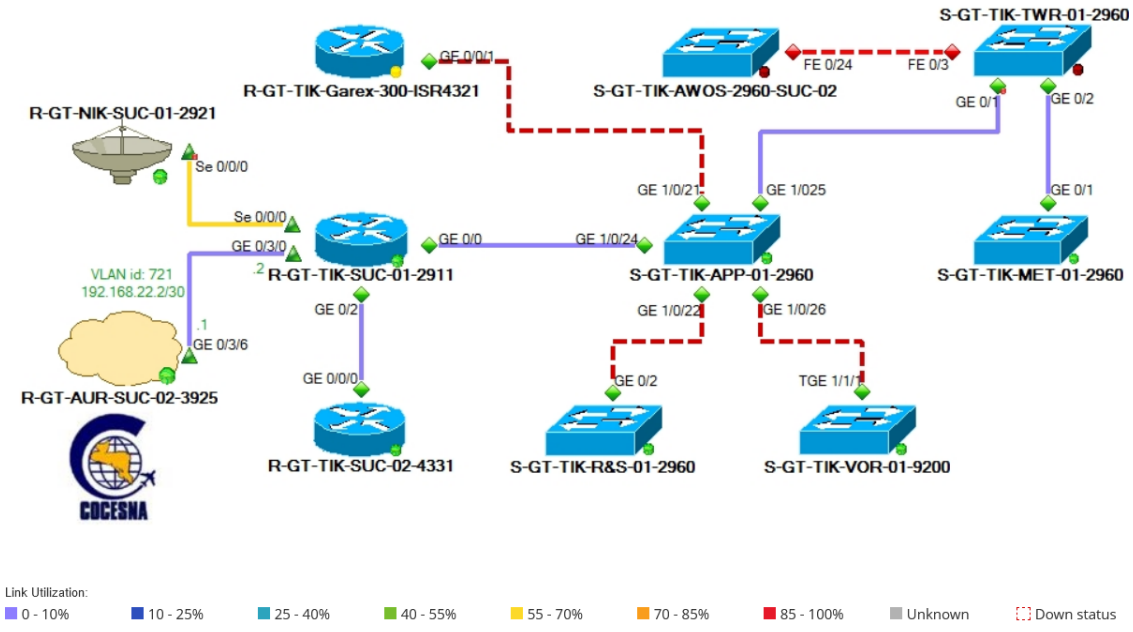


Figura 54: Mapa de topología de la estación TIK (Aeropuerto Internacional Mundo Maya), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

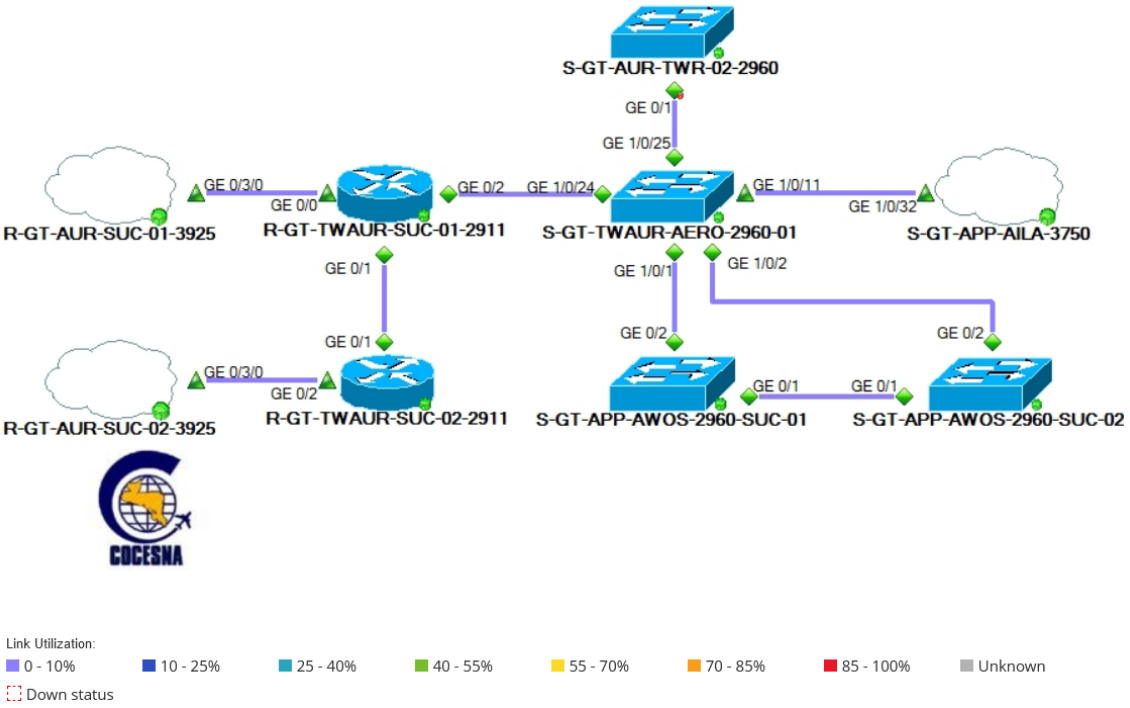


Figura 55: Mapa de topología de la estación TWR-AUR (Torre de Control AILA), que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

9.5. Importancia del mapa global de interconexión de estaciones de COCESNA

Un elemento clave dentro de la configuración de mapas de topología es el mapa global de interconexión de las estaciones de COCESNA, ver Figura 56. Este mapa muestra cómo se interconectan las distintas estaciones y es fundamental para tener una visión macro de toda la infraestructura. Al visualizar las conexiones entre las estaciones principales, los administradores pueden identificar de inmediato posibles puntos de fallo que podrían afectar a grandes áreas de la red.

El mapa global permite también evaluar la redundancia y resiliencia de la red, asegurando que existan rutas alternativas para el tráfico en caso de fallos. Tener esta vista integral facilita la planificación estratégica y el análisis de capacidad, garantizando que la red de telecomunicaciones de COCESNA continúe operando de manera óptima ante cualquier eventualidad.

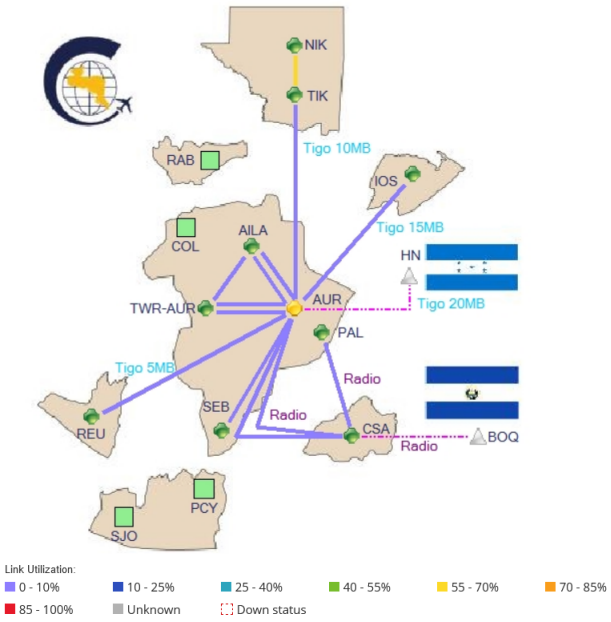


Figura 56: Mapa de topología de la estación COCESNA Guatemala, que ilustra la estructura de conexiones y la utilización de cada enlace, representada en un rango de colores que indica el porcentaje de ancho de banda utilizado

9.6. Enlaces en rojo desplegados en los mapas

Los enlaces en rojo de las Figuras 47, 50 y 54 y se debe a que hubo un problema de comunicación entre SW NPM (SolarWinds Network Performance Monitor) y la DB de MS SQL (Microsoft SQL Server) y detecta que no hay comunicación a pesar de que si la hay. La causa de este problema es que la version de la DB no tiene soporte con la version de SW NPM que se esta utilizando.

Por otro lado, los enlaces en rojo las Figuras 48 y 56 se debe a que no se posee la comunidad SNMP de esos dispositivo ya que corresponde a estaciones de COCESNA en otros países.

9.7. Consideraciones finales

La configuración de mapas de topología en SW NPM proporciona una visión integral y dinámica de la infraestructura de red, permitiendo a los administradores monitorear tanto la conectividad como el rendimiento de los dispositivos de manera visual e intuitiva. Al implementar prácticas de personalización y actualización automática, se logra una mayor precisión en el monitoreo y una capacidad mejorada para responder ante problemas antes de que afecten al servicio.

El uso de ONA y la integración de métricas específicas en los mapas permite anticipar problemas potenciales y tomar acciones proactivas para garantizar la estabilidad de la red de COCESNA. De este modo, la plataforma SW NPM se convierte en una herramienta fundamental para mantener la calidad y continuidad de los servicios de telecomunicaciones.

CAPÍTULO 10

Uso de la base de datos de SolarWinds Orion

En este capítulo se detalla cómo se utiliza la DB (base de datos) de SolarWinds Orion para almacenar y gestionar la información crítica relacionada con la red. La DB es un componente esencial del sistema de monitoreo, ya que contiene todos los datos recolectados sobre el rendimiento y el estado de los dispositivos, permitiendo a los administradores analizar información histórica y tomar decisiones informadas para el mantenimiento de la infraestructura.

10.1. Guía para realizar un *backup* de la DB de SolarWinds Orion

Para garantizar la continuidad del servicio de monitoreo y la integridad de los datos, es fundamental realizar copias de seguridad de la DB de SolarWinds Orion. En esta sección, se describen dos métodos principales para realizar un *backup:* desde el servidor de Orion y utilizando un administrador universal de DB como DBeaver.

Estos dos métodos aseguran que se puedan realizar copias de seguridad de la DB de manera eficiente, proporcionando opciones tanto para el administrador directamente en el servidor como para aquellos que prefieren usar una herramienta de administración de bases de datos universal como DBeaver.

En la Figura 57 se muestran las configuraciones para establece conexión con la DB de SolarWind Orion.

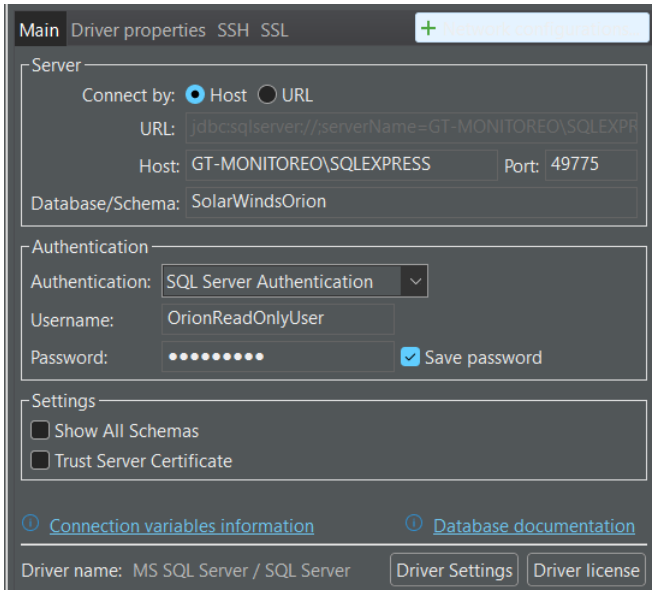


Figura 57: Ejemplo de configuraciones para establecer conexión con la DB Orion

10.1.1. Realizar un *backup* desde el Servidor de Orion

1. Acceder al servidor donde está instalada la base de datos de SolarWinds Orion. La Figura 58 muestra como acceder al servidor por medio de La aplicación Remote Desktop.

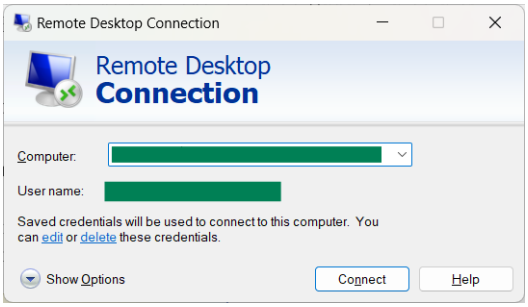


Figura 58: Parámetros para establecer conexión remota con el servidor de la DB de SW, donde "**Computer**" es la IP y "**User name**" el usuario institucional

2. Abrir MS SQL (Microsoft SQL Server) y conectarse al servidor de base de datos. La aplicación tiene el icono que se muestra en la Figura 59

Figura 59: Icono de la aplicación SQL Server Manager Studio

3. Seleccionar la base de datos de SolarWinds Orion en la lista de bases de datos disponibles.

4. Hacer clic derecho sobre la base de datos y seleccionar la opción "Tasks" ->"Back Up...". Ver Figura 60 para el *path*.

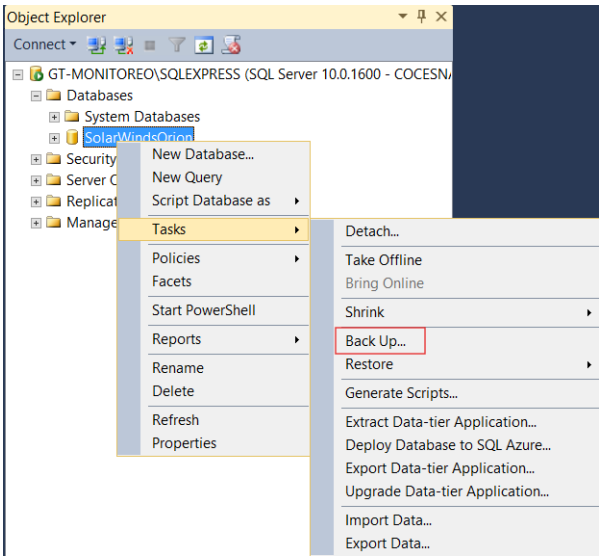


Figura 60: *Path* para realizar BU dentro del *host*

5. En la ventana de backup 61, configurar el tipo de copia de seguridad (completa o diferencial) y especificar la ubicación donde se almacenará el archivo de *backup*.

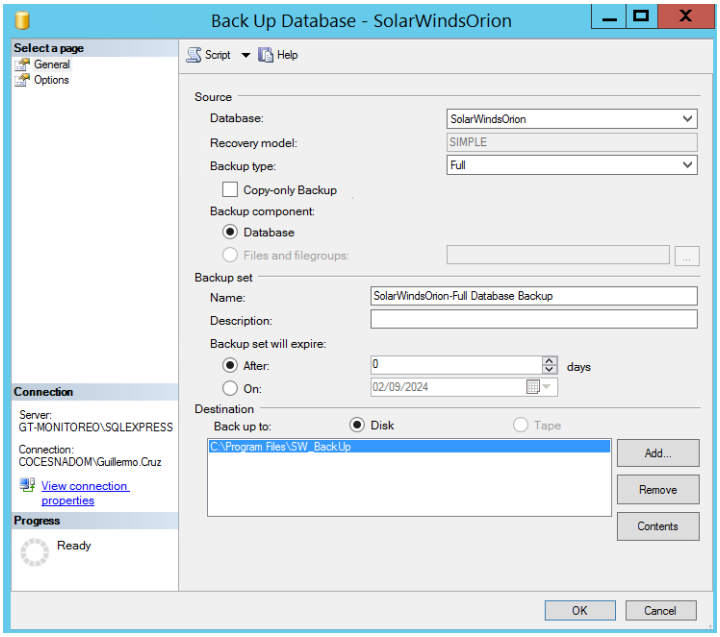


Figura 61: Pestaña Back Up Database

6. Hacer clic en "Ok" para iniciar el proceso de copia de seguridad. Es recomendable verificar que el proceso se haya completado exitosamente. Para esto se puede verificar el tamaño del archivo generado, si este es de un tamaño razonable y esté en el destino correcto se puede considerar que el proceso fue exitoso.

10.1.2. Realizar un *backup* con DBeaver

1. En el panel de conexiones, seleccionar la conexión que corresponde a la base de datos de SolarWinds Orion. Ver Figura 62

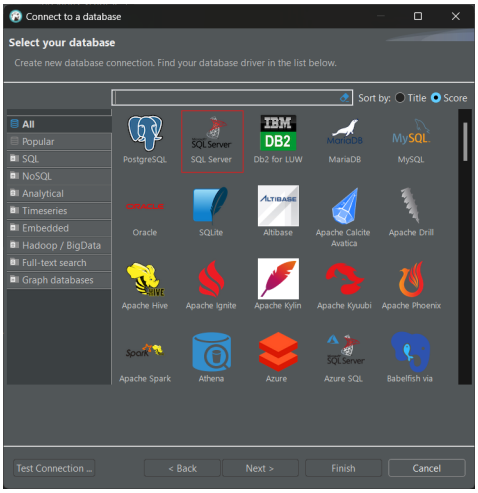


Figura 62: Menú de selección de tipos de DB

2. Abrir DBeaver e iniciar sesión con las credenciales que permiten el acceso al servidor de la base de datos de SolarWinds Orion, Figura 63.

Figura 63: Ingreso de credenciales para poder acceder a la DB

3. Hacer clic derecho sobre la base de datos y seleccionar "Tools" -> "Dump database". Ver Figura 64

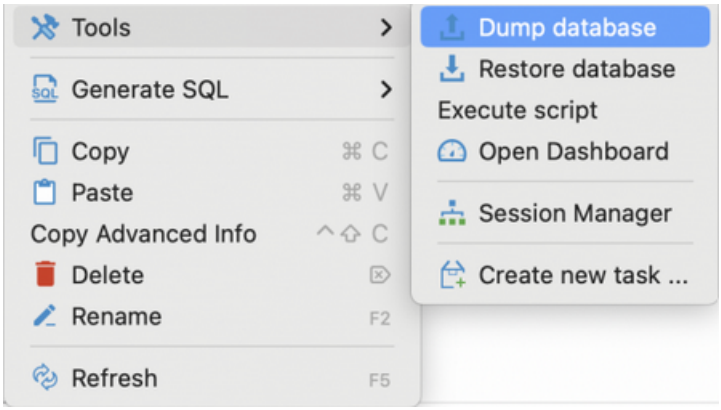


Figura 64: BU utilizando DBeaver

4. Configurar los detalles del *backup*, incluyendo el tipo de *backup* (completo o diferencial) y la ubicación donde se almacenará el archivo resultante.

5. Hacer clic en "Start" para ejecutar la copia de seguridad. Es importante monitorear el proceso y verificar que el archivo de *backup* se haya creado correctamente. Para esto se puede verificar el tamaño del archivo generado, si este es de un tamaño razonable y esté en el destino correcto se puede considerar que el proceso fue exitoso.

10.2. Importancia de los *backups* periódicos

El monitoreo de una infraestructura crítica como la de COCESNA requiere que la DB esté siempre disponible y que los datos se mantengan seguros. Los *backups* regulares permiten:

- **Recuperación rápida ante fallos:** en caso de un fallo en el servidor principal de la DB, los *backups* permiten restaurar el sistema y retomar el monitoreo sin perder datos significativos.
- **Protección contra corrupción de datos:** los *backups* ayudan a proteger contra la corrupción de datos, ya sea por errores humanos o fallos en el hardware, asegurando que siempre exista una versión segura de la información.
- **Cumplimiento de políticas de seguridad:** al trabajar en una infraestructura de telecomunicaciones, es fundamental cumplir con ciertas políticas de seguridad y continuidad del servicio. Las copias de seguridad periódicas forman parte de estas prácticas esenciales para mantener la confiabilidad y disponibilidad del sistema de monitoreo.

10.3. Ejemplos de consultas a la DB de SolarWinds Orion

A continuación, se presentan algunos ejemplos de *querys* que se pueden utilizar para obtener información relevante de la DB de SolarWinds Orion. Estas consultas permiten a los administradores extraer datos específicos sobre el rendimiento y estado de la infraestructura, facilitando así el análisis y la toma de decisiones.

- **Consulta de nodos con mayor utilización de CPU:** Al ejecutar el *query* del Cuadro 3 se obtiene el resultado del Cuadro 4.

```
WITH CPUUsage AS (
    -- CPU maximo ultimas 72 horas
    SELECT
        a.NodeID,
        MAX(a.TimeStamp) AS TimeStamp,
        MAX(a.AvgPercentMemoryUsedMax) AS MaxMemoryUsageLast72Hours,
        b.AvgPercentMemoryUsedMax AS MostRecentMemoryUsage
    FROM
        CPULoad_Statistics a
    CROSS APPLY
        (
            SELECT TOP 1 AvgPercentMemoryUsedMax
            FROM CPULoad_Statistics
            WHERE NodeID = a.NodeID
              AND TimeStamp >= DATEADD(HOUR, -72, GETDATE())
            ORDER BY TimeStamp DESC
        ) b
    WHERE
        a.TimeStamp >= DATEADD(HOUR, -72, GETDATE())
    GROUP BY
        a.NodeID, b.AvgPercentMemoryUsedMax
),
NodeInfo AS (
    -- Nombre, IP, City, Type
    SELECT
        nd.NodeID,
        nd.IP_Address as IP,
        CASE
            WHEN CHARINDEX('.', [SysName], 8) > 0 THEN LEFT([SysName],
                CHARINDEX('.', [SysName]) - 1)
            ELSE [SysName]
        END as Name,
        ncp.City as City,
        ncp.[Type] as Type,
        ns.LastBoot as LastBoot
    FROM
        NodesData nd
    JOIN NodesCustomProperties ncp on
        nd.NodeID = ncp.NodeID
    JOIN NodesStatistics ns on
        nd.NodeID = ns.NodeID
)
SELECT TOP 10
    cs.TimeStamp,
    ni.Name,
    ni.IP,
```

```
    ROUND(cs.MaxMemoryUsageLast72Hours, 0) as MaxCPU72horas,
    ni.City,
    ni.Type
FROM NodeInfo ni
JOIN CPUUsage cs ON ni.NodeID = cs.NodeID
ORDER BY MaxCPU72horas DESC;
```

Cuadro 3: *Query* SQL encargado de recuperar el uso máximo de CPU en las últimas 72 horas para identificar los nodos con mayor demanda de procesamiento, útil para detectar posibles cuellos de botella en el rendimiento

TimeStamp	Name	IP	MaxCPU72h	City	Type
2024-11-07 08:15:02.957	R-GT-TIK-Garex-300-ISR4321	XXX.XXX.XXX.XXX	83	TIK	Router
2024-11-07 08:15:02.957	R-GT-SEB-SUC-01-ISR4331	XXX.XXX.XXX.XXX	70	SEB	Router
2024-11-07 08:15:02.957	R-GT-TIK-SUC-02-4331	XXX.XXX.XXX.XXX	69	TIK	Router
2024-11-07 08:15:02.957	S-GT-CSA-SUC-01-2960	XXX.XXX.XXX.XXX	62	CSA	Switch
2024-11-07 08:15:02.957	S-GT-GEG-ADMIN-9200-FO-01	XXX.XXX.XXX.XXX	61	AUR	Switch
2024-11-07 08:15:02.957	S-GT-PAL-SUC-01-9200-48P	XXX.XXX.XXX.XXX	61	PAL	Switch
2024-11-07 08:15:02.957	S-GT-TIK-TWR-01-2960	XXX.XXX.XXX.XXX	61	TIK	Switch
2024-11-07 08:15:02.957	S-GT-TIK-MET-01-2960	XXX.XXX.XXX.XXX	60	TIK	Switch
2024-11-07 08:15:02.957	S-GT-COM-SEB-SUC-02-9200	XXX.XXX.XXX.XXX	58	SEB	Switch
2024-11-07 08:15:02.957	S-GT-TIK-VOR-01-9200	XXX.XXX.XXX.XXX	54	TIK	Switch

Cuadro 4: Muestra el uso máximo de CPU en las últimas 72 horas para varios dispositivos, con columnas de nombre, IP, uso máximo de CPU, ciudad y tipo de dispositivo. Las triliterales de las ciudades fueron las siguientes: AUR (Aeropuerto La Aurora), CSA (Cerro Santiago), PAL (Cerro Palencia), SEB (Santa Elena Barrillas), TIK (Aeropuerto Internacional Mundo Maya) Las IP fueron aletaradas por razones de seguridad

- **Consulta de Interfaces con Mayor Uso de Ancho de Banda:** Al ejecutar el *query* del Cuadro 5 se obtiene el resultado del Cuadro 6.

```
WITH NodeInfo AS (
    SELECT
        nd.NodeID,
        nd.IP_Address as IP,
        CASE
            WHEN CHARINDEX('.', [SysName], 8) > 0 THEN LEFT([SysName],
                CHARINDEX('.', [SysName]) - 1)
            ELSE [SysName]
        END as Name,
        ncp.City as City,
        ncp.[Type] as Type,
        ns.LastBoot as LastBoot
    FROM
        NodesData nd
    JOIN NodesCustomProperties ncp on
        nd.NodeID = ncp.NodeID
    JOIN NodesStatistics ns on
        nd.NodeID = ns.NodeID
),
LatestInterfaceTraffic AS (
    SELECT
        it.InterfaceID,
        it.NodeID,
        it.[DateTime],
        it.In_Averagebps,
```

```
        it.Out_Averagebps,
        (it.In_Averagebps + it.Out_Averagebps) AS Total_Averagebps,
        ROW_NUMBER() OVER (PARTITION BY it.InterfaceID ORDER BY it.[
            DateTime] DESC) AS rn
    FROM
        InterfaceTraffic it
    WHERE
        it.[DateTime] >= DATEADD(HOUR, -72, GETDATE())
)
SELECT
    TOP 10
    ni.Name,
    ni.IP,
    i.InterfaceName,
    lit.[DateTime],
    ROUND(lit.In_Averagebps/i.InBandwidth*100, 2) as In_Percentbps,
    ROUND(lit.Out_Averagebps/i.OutBandwidth*100, 2) as Out_Percentbps,
    ROUND(lit.Total_Averagebps/(i.InBandwidth+i.OutBandwidth)*100, 2)
        as AvgTotal_Percentbps
FROM
    LatestInterfaceTraffic lit
JOIN NodeInfo ni on
    lit.NodeID = ni.NodeID
JOIN Interfaces i on
    lit.InterfaceID = i.InterfaceID
WHERE
    lit.rn = 1
ORDER BY
    lit.Total_Averagebps DESC;
```

Cuadro 5: *Query* SQL encargado de obtener el uso de ancho de banda en las interfaces de red de los últimos 72 horas, mostrando las interfaces con mayor consumo y ayudando a identificar posibles sobrecargas

Name	IP	IntName	DateTime	Inbound $\%_{bps}$	Outbound $\%_{bps}$	Avgerage $\%_{bps}$
R-GT-AUR-SUC-03-2911	XXX.XXX.XXX.XXX	GE 0/2	2024-11-07 15:14:26.033	1.02	0.62	0.82
R-GT-CSA-SUC-01-3925	XXX.XXX.XXX.XXX	GE 0/1	2024-11-07 15:14:27.437	15.44	7.27	11.35
R-GT-AUR-SUC-01-3925	XXX.XXX.XXX.XXX	GE 0/0	2024-11-07 15:14:20.133	3.33	15.45	9.39
R-GT-AUR-SUC-01-3925	XXX.XXX.XXX.XXX	GE 0/1	2024-11-07 15:14:20.137	0.54	0.05	0.29
R-GT-AUR-SUC-03-2911	XXX.XXX.XXX.XXX	GE 0/0	2024-11-07 15:14:26.023	0.05	0.54	0.29
S-GT-AUR-ACC-01-3750	XXX.XXX.XXX.XXX	GE 1/1/2	2024-11-07 15:14:25.517	0.18	0.38	0.28
S-GT-AUR-ACC-01-3750	XXX.XXX.XXX.XXX	GE 1/0/1	2024-11-07 15:14:25.500	0.07	0.45	0.26
S-GT-GEG-ADMIN-9200	XXX.XXX.XXX.XXX	TGE 0/1/1	2024-11-07 15:13:06.257	0.36	0.16	0.26
R-GT-CSA-SUC-01-3925	XXX.XXX.XXX.XXX	GE 0/2	2024-11-07 15:14:27.450	0.02	0.45	0.24
R-GT-CSA-SUC-03-2911	XXX.XXX.XXX.XXX	GE 0/0	2024-11-07 15:14:27.430	0.45	0.02	0.24

Cuadro 6: Presenta el uso de ancho de banda (entrada, salida y total promedio) para interfaces de red en los últimos 72 horas, con detalles del nombre del dispositivo, IP, nombre de la interfaz y el porcentaje de uso. Las abreviacioens de las interfaces son las siguientes: GE (Gigabit Ethernet), TGE (Ten Gigabit Ethernet). Las IP fueron aletaradas por razones de seguridad

- **Consulta de Nodos con Alertas Activas:** Al ejecutar el *query* del Cuadro 7 se obtiene el resultado del Cuadro 8.

```
SELECT
    TOP 10
    ahv.LastTriggeredDateTime as Time,
    CASE
        WHEN CHARINDEX('.',
```

```
        [SysName],
        8) > 0 THEN LEFT([SysName],
        CHARINDEX('.',
        [SysName]) - 1)
        ELSE [SysName]
    END as Name,
    nd.IP_Address as IP,
    ahv.Name as Alert,
    ncp.City as City,
    ncp.[Type] as Type
FROM
    NodesData nd
JOIN NodesCustomProperties ncp on
    nd.NodeID = ncp.NodeID
JOIN NodesStatistics ns on
    nd.NodeID = ns.NodeID
JOIN AlertHistoryView ahv on
    nd.NodeID = ahv.RelatedNodeId
WHERE
    ahv.EventTypeWord = 'Triggered'
ORDER BY
    ahv.[TimeStamp] DESC;
```

Cuadro 7: *Query* SQL encargado de extraer las alertas activadas recientemente en la red, incluyendo información básica del nodo y tipo de alerta para facilitar el monitoreo de eventos críticos

Time	Name	IP	Alert	City	Type
2024-11-03 19:55:07.477	S-GT-PAL-SUC-01-9200-48P	XXX.XXX.XXX.XXX4	High packet loss	PAL	Switch
2024-11-03 19:55:07.477	R-GT-PBR-01-2901	XXX.XXX.XXX.XXX	High packet loss	IOS	Router
2024-11-03 19:50:06.420	R-GT-CSA-SUC-03-2911	XXX.XXX.XXX.XXX	High packet loss	CSA	Router
2024-11-03 19:50:06.420	S-GT-CSA-SUC-01-2960	XXX.XXX.XXX.XXX	High packet loss	CSA	Switch
2024-11-01 12:31:24.273	R-GT-PBR-01-2901	XXX.XXX.XXX.XXX	Node is down	IOS	Router
2024-11-01 12:19:08.583	R-GT-AUR-SUC-02-3925	XXX.XXX.XXX.XXX	Neighbor is down	AUR	Router
2024-11-01 12:31:24.273	R-GT-PBR-01-2901	XXX.XXX.XXX.XXX	Node is down	IOS	Router
2024-11-03 19:55:07.477	R-GT-PBR-01-2901	XXX.XXX.XXX.XXX	High packet loss	IOS	Router
2024-11-01 12:19:08.583	R-GT-AUR-SUC-02-3925	XXX.XXX.XXX.XXX	Neighbor is down	AUR	Router
2024-11-01 00:09:08.590	S-GT-COM-SEB-SUC-02-9200	XXX.XXX.XXX.XXX	High response time	SEB	Switch

Cuadro 8: Lista las alertas activas recientes, mostrando la fecha y hora de la alerta, el nombre e IP del dispositivo, el tipo de alerta, la ciudad y el tipo de dispositivo. Las triliterales de las ciudades son las siguientes: AUR (Aeropuerto La Aurora), CSA (Cerro Santiago), IOS (Puerto Barrios), PAL (Cerro Palencia) y SEB (Santa Elena Barrillas). Las IP fueron aletaradas por razones de seguridad

10.4. Consideraciones finales

La base de datos de SolarWinds Orion es el núcleo del sistema de monitoreo, y su correcta gestión y respaldo son fundamentales para garantizar la estabilidad y seguridad de la infraestructura. La implementación de un plan de copias de seguridad bien estructurado asegura que los datos estén siempre protegidos y disponibles, proporcionando tranquilidad tanto al equipo de monitoreo como a los responsables de la infraestructura de telecomunicaciones de COCESNA.

El proyecto no contemplo implementar una DB ajena a la de SolarWinds Orion. Si no que se aprovecho la DB de MS SQL (Microsoft SQL Server) que se encontraba instalada.

En resumen, el proceso de *backup* regular y la verificación de estos respaldos son prácticas indispensables para minimizar el riesgo de pérdida de datos y garantizar la continuidad del servicio de monitoreo en todo momento.

CAPÍTULO 11

Desarrollo y Uso de la API *dockerizada* para generación de reportes

En este capítulo se describe el desarrollo y la implementación de una API (*Application Programming Interface*) *dockerizada* que facilita la generación de reportes sobre el estado de la infraestructura de telecomunicaciones de COCESNA. La solución se diseñó para proporcionar un entorno de monitoreo ágil y eficiente, simplificando la recopilación y el análisis de datos críticos de la red.

11.1. Introducción a la API

La necesidad de mejorar el monitoreo y el reporte de los dispositivos de la infraestructura de COCESNA llevó al desarrollo de una API alojada en un contenedor Docker. Este enfoque se seleccionó debido a su capacidad para ofrecer un entorno reproducible, portátil y fácil de desplegar, lo cual asegura la estabilidad y continuidad del servicio. La API fue desarrollada utilizando Flask, un *framework* ligero para Python, que proporciona flexibilidad para gestionar los distintos endpoints y responder a las solicitudes de los usuarios de forma eficiente.

11.2. Arquitectura de la API y componentes involucrados

La arquitectura de la API se compone de varios componentes clave:

- **Flask como *framework*:** la API fue construida utilizando Flask, lo que facilitó la implementación de endpoints ligeros y la gestión de las peticiones HTTP.
- **Docker para *dockerización*:** la API está contenida dentro de un contenedor Docker, lo que facilita su despliegue en cualquier entorno con Docker instalado. Esto proporciona ventajas como la portabilidad y la simplicidad en la actualización del software.
- **Base de datos MS SQL:** la base de datos contiene toda la información de los dispositivos de la infraestructura. La API interactúa con la base de datos mediante consultas SQL para generar reportes personalizados.
- **Docker Compose:** se utilizó Docker Compose para orquestar la API junto con otros servicios, asegurando que todos los componentes necesarios estén desplegados y funcionando de manera coordinada.

La interacción entre estos componentes se muestra en la Figura 65.

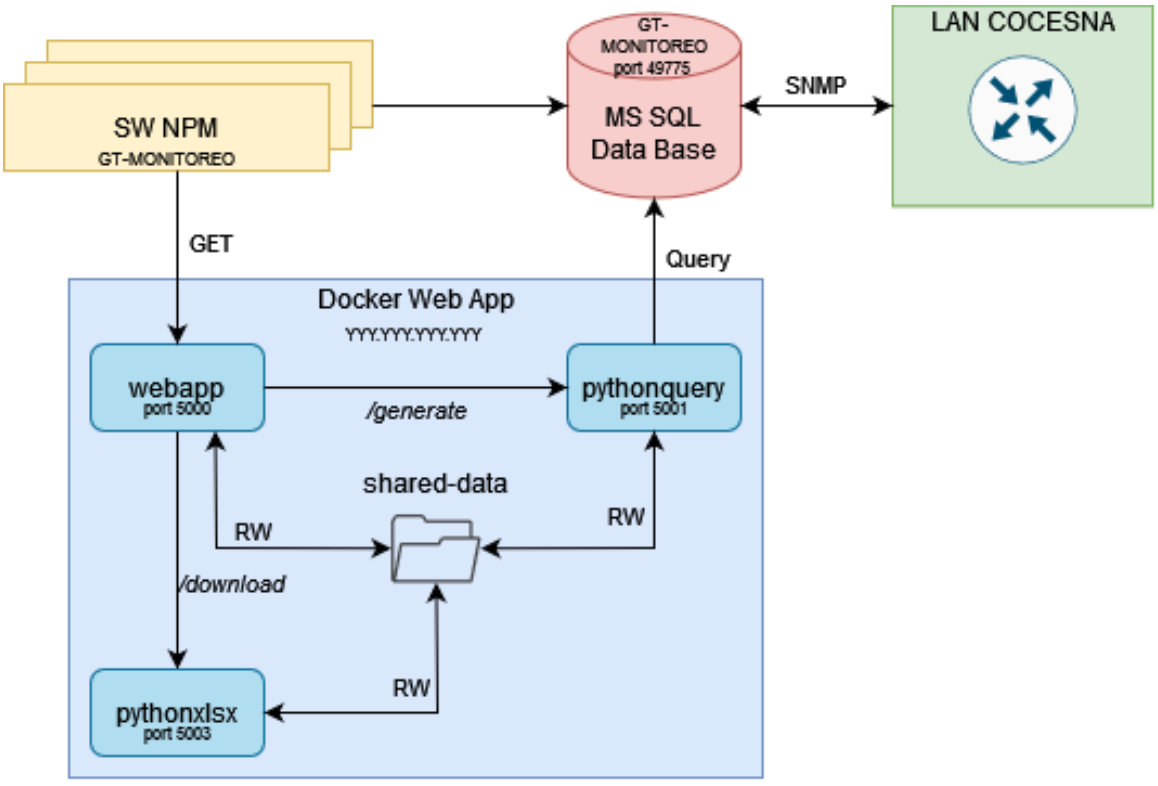


Figura 65: Arquitectura de la API desarrollada demostrando su interacción con la base de datos y otros servicios, así como su despliegue en un entorno *dockerizado*

11.3. Configuración y orquestación con Docker

Uno de los principales beneficios de esta solución es la facilidad de configuración y despliegue gracias a Docker y Docker Compose. El Cuadro 9 define todos los servicios necesarios, permitiendo que la API, el servidor de bases de datos y otros elementos puedan ejecutarse de manera conjunta y coordinada.

1. **Docker Compose:** el Cuadro 9 contiene la configuración de los servicios que se deben desplegar, incluyendo la API y otros contenedores relacionados. Esto permite levantar toda la infraestructura con un solo comando, simplificando el despliegue y asegurando la consistencia entre entornos.
2. **Reproducibilidad:** al *dockerizar* la API, se garantiza que el código funcionará igual en cualquier sistema, siempre que tenga Docker instalado, lo cual elimina problemas comunes de compatibilidad y dependencias.

```
version: '3.9'

services:
  backend:
    build:
      context: ./pythonquery
      dockerfile: Dockerfile
    ports:
      - "49775:49775"  # Expose port for database connection
      - "5001:5001"    # Flask API for internal communication
    volumes:
      - ./shared-data:/usr/src/app/shared-data # Map current directory to /
          app in the container
    networks:
      - app-network

  cli:
    build:
      context: ./pythoncli
      dockerfile: Dockerfile
    ports:
      - "5002:5002"
    volumes:
      - ./shared-data:/usr/src/app/shared-data # Map current directory to /
          app in the container
    networks:
      - app-network

  xlsx:
    build:
      context: ./pythonxlsx
      dockerfile: Dockerfile
    ports:
      - "5003:5003"
    volumes:
      - ./shared-data:/usr/src/app/shared-data # Map current directory to /
          app in the container
    networks:
      - app-network
```

```
37
38    frontend:
39      build:
40        context: ./webapp
41        dockerfile: Dockerfile
42      ports:
43        - "5000:5000"  # Map local port 5000 to container's port 5000
44      volumes:
45        - ./shared-data:/usr/src/app/shared-data # Map current directory to /
              app in the container
46      networks:
47        - app-network
48      environment:
49        FLASK_ENV: development
50
51
52
53  # volumes:
54  #   shared-data:
55
56  networks:
57    app-network:
58      driver: bridge
```

Cuadro 9: Código de Docker encargado de configurar y desplegar servicios en contenedores para ejecutar la aplicación de monitoreo en un entorno aislado y escalable

11.4. Endpoints de la API y funcionalidades

La API proporciona varios *endpoints* que permiten a los usuarios interactuar con la base de datos y generar reportes personalizados. A continuación se describen algunos de los principales endpoints, los cuales pueden ser encontrados en el Cuadro **??**

- ***/download*:** permite la descarga de reportes en formato CSV o XLSX. Los usuarios pueden especificar el tipo de datos que desean descargar, lo cual facilita el análisis posterior.
- ***/generate*:** este endpoint se encarga de la generación de nuevos reportes basados en los datos actuales de la base de datos. El reporte se puede personalizar según ciertos parámetros definidos por el usuario.
- ***/api/filters*:** proporciona la capacidad de aplicar filtros a la información solicitada. Por ejemplo, se pueden aplicar filtros por ciudad o tipo de dispositivo para focalizar el análisis.
- ***/api/table*:** devuelve una tabla de datos directamente de la base de datos, permitiendo la visualización rápida de información en la interfaz de usuario.

Cada uno de estos endpoints se diseñó para proporcionar acceso eficiente a los datos almacenados, facilitando la generación y descarga de reportes específicos y personalizados.

```python
# WebApp/app.py
# ************************************************
# Load libraries
# ************************************************
from flask import Flask, render_template, send_file, redirect, url_for, request, jsonify
import requests
import os
import pandas as pd
app = Flask(__name__, static_folder='static')

# ************************************************
# Local registry
# ************************************************
# Get the directory where the Python script is running
cwd = os.getcwd()

# Path to the local CSV file
csv_node_name = 'node_info.csv'
csv_node_localpath = os.path.join('shared-data', 'csv', csv_node_name)
csv_node_path = os.path.join(cwd, csv_node_localpath)

xslx_path = os.path.join(cwd, 'shared-data', 'xlsx', 'Updated_Formato003.xlsx')

# ************************************************
# Remote registry
# ************************************************
# Define the hostname and port of the PythonQuery service
PYTHONQUERY_HOST = 'http://backend:5001'
PYTHONXLSX_HOST = 'http://xlsx:5003'

# ************************************************
# API
# ************************************************
# Route to display the homepage
@app.route('/')
def index():
    # Simply render the index template; data will be loaded dynamically
    return render_template('index.html')

# Route to download the XLSX
@app.route('/download')
def download_file():
    # Send a POST request to PythonXLSX to generate the XLSX
    response = requests.post(f"{PYTHONXLSX_HOST}/download")

    if response.status_code == 200:
        if os.path.exists(xslx_path):
            return send_file(xslx_path, as_attachment=True, download_name='Reporte.xlsx', mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')
        else:
            return jsonify({"message": "File not found"}), 404
    else:
        return jsonify({"message": "Failed to generate XLSX."}), 500


```

```
55
56  # Route to generate a new CSV (Mocked for now)
57  @app.route('/generate')
58  def generate_csv():
59      # Send a POST request to PythonQuery to generate the CSV
60      response = requests.post(f"{PYTHONQUERY_HOST}/generate")
61
62      if response.status_code == 200:
63          return redirect(url_for('index'))
64      else:
65          return "Failed to generate CSV.", 500
66
67  @app.route('/api/filters')
68  def get_filters():
69      if os.path.exists(csv_node_path):
70          df = pd.read_csv(csv_node_path)
71          types = df['Type'].dropna().unique().tolist()
72          cities = df['City'].dropna().unique().tolist()
73          return jsonify({'types': types, 'cities': cities})
74      else:
75          return jsonify({'types': [], 'cities': []})
76
77  from flask import request, jsonify
78
79  @app.route('/api/table')
80  def get_table_data():
81      if os.path.exists(csv_node_path):
82          df = pd.read_csv(csv_node_path)
83
84          # Apply filters from request parameters
85          type_filter = request.args.get('type')
86          city_filter = request.args.get('city')
87
88          if type_filter:
89              df = df[df['Type'] == type_filter]
90          if city_filter:
91              df = df[df['City'] == city_filter]
92
93          # Ensure the correct column order: NodeID, Name, IP, City, Type, (
                rest of the columns)
94          fixed_columns = ['Name', 'IP', 'City', 'Type']
95          remaining_columns = [col for col in df.columns if col not in
                fixed_columns + ['NodeID']]
96          ordered_columns = fixed_columns + remaining_columns
97
98          df = df[ordered_columns]
99
100         # Convert the filtered DataFrame to JSON
101         return df.to_json(orient='records')
102     else:
103         return jsonify([])  # Return an empty list if no data is available
104
105
106 # New API route to get log data as HTML
107 @app.route('/api/logs')
108 def get_log_data():
109     # Mock log data for demonstration
110     log_content = [
111         "<li>Log entry 1: CSV file generated successfully.</li>",
```

```
        "<li>Log entry 2: User viewed the CSV file.</li>",
        "<li>Log entry 3: User downloaded the CSV file.</li>"
    ]

    # Clean and format log content for dynamic loading
    logs_html = ''.join(log_content).replace('\n', '').strip()

    return logs_html

# Route to view the generation log (without embedding log data)
@app.route('/log')
def view_log():
    # Simply render the log template; data will be loaded dynamically
    return render_template('log.html')

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Cuadro 10: Núcleo de la aplicación Flask; permite descargar archivos, generar CSV y aplicar filtros para el monitoreo de nodos en red

11.5. Generación automática de CSV y reportes XLSX

La API permite la generación automática de archivos CSV (*Comma-Separated Values*) y reportes en formato XLSX, que contienen información clave sobre el estado de los dispositivos de la red.

- **Generador de CSV:** el código de Python del Cuadro 11 es utilizado por la API para conectarse a la base de datos y extraer la información solicitada. A partir de los datos obtenidos, se genera un archivo CSV que luego se puede descargar o convertir a otros formatos.
- **Formato XLSX:** los reportes también se generan en formato XLSX, lo cual permite que los datos sean más fáciles de manejar, visualizar y compartir entre los responsables del monitoreo y gestión de la infraestructura.

```
# pythonquery/Query_executor.py
# ****************************************************
# Load libraries
# ****************************************************
import pandas as pd
import os
import pytds
import json

# ****************************************************
# Local registry
# ****************************************************
# Get the directory where the Python script is running
cwd = os.getcwd()

# Sql path
QueryName = 'Query.sql'
```

```
QueryPath = os.path.join(cwd, 'sql', QueryName)

# Saving files path
data_name = 'node_info.csv'
data_path = os.path.join(cwd, 'shared-data', 'csv', data_name)

# Json path
JsonName = 'secret.json'
JsonPath = os.path.join(cwd, JsonName)
# ***************************************************
# Load DataBase
# ***************************************************
# Function to get database information by name
def get_database_info(db_name, config):
    for db in config["databases"]:
        if db["name"] == db_name:
            return db
    return None

# Load the JSON data from the file
with open(JsonPath) as f:
    config = json.load(f)

# DataBase info
db = get_database_info("SolarWindsOrion", config)
DATABASE = db["name"]
SERVER = db["server"]
PORT = db["port"]
USERNAME = db["username"]
PASSWORD = db["password"]

# Get the Query from file and Load DataBase
with open(QueryPath, 'r') as file:
    Query = file.read()

# Execute Query
with pytds.connect(server=SERVER, database=DATABASE, user=USERNAME, password
    =PASSWORD, port=PORT) as conn:
    print(f'Connection established at: {SERVER}/{DATABASE}:{PORT}')
    with conn.cursor() as cursor:
        cursor.execute(Query)
        results = cursor.fetchall()
    print('Finished Query')

# Save Query on DataFrame
df = pd.DataFrame(results)

# print(df.head())
columns = [
    'NodeID',
    'Name',
    'IP',
    'City',
    'Type',
    'FetchedAt',
    'LastBoot',
    'MemoryUsage',
    'MemoryUsage72h',
    'PowerSupplyStatus',
```

```
    'TemperatureStatus',
    'FanStatus',
    'InterfaceStatus']

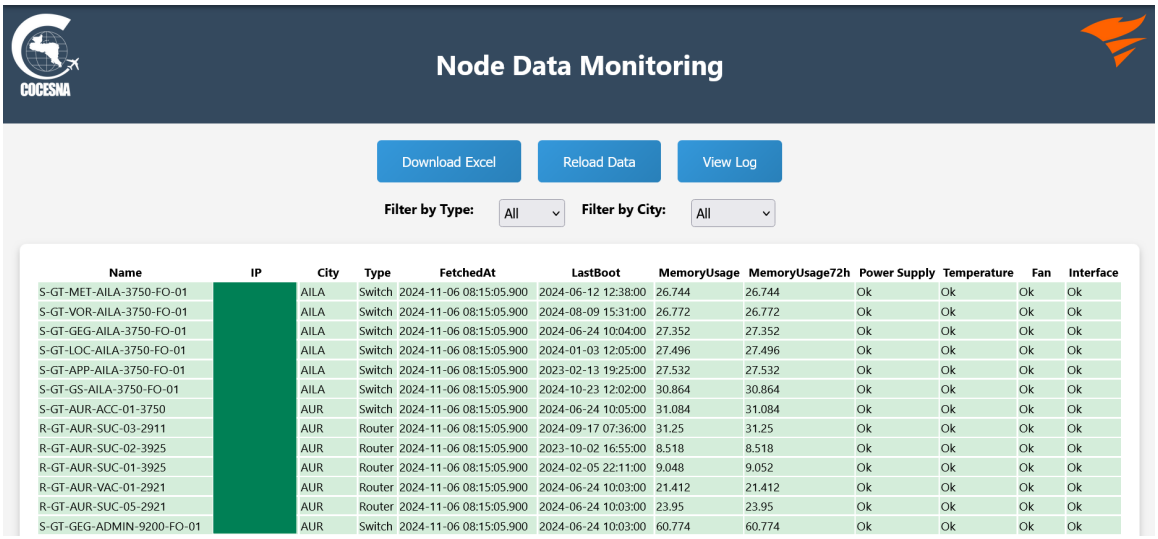
df.columns = columns
df.set_index('NodeID', inplace=True)
df[['MemoryUsage', 'MemoryUsage72h']] = df[['MemoryUsage', 'MemoryUsage72h']].round(3)
# print('Dataframe created')

print("\n----------------------------------------------------------")
print("Final DataFrame:")
print(df.head())
df.to_csv(data_path)
```

Cuadro 11: Código de Python encargado de ejecutar consultas SQL en la base de datos de SolarWinds Orion, exportando datos de nodos a CSV para análisis de infraestructura

11.6. Interfaz de usuario y experiencia del usuario

La API está acompañada de una UI (*User Interface*) que permite una interacción sencilla con las funcionalidades disponibles. La UI se desarrolló utilizando HTML, CSS, y JavaScript, proporcionando una experiencia de usuario intuitiva y fácil de usar. El resultado final de la UI se puede apreciar en la Figura 66.



Name	IP	City	Type	FetchedAt	LastBoot	MemoryUsage	MemoryUsage72h	Power Supply	Temperature	Fan	Interface
S-GT-MET-AILA-3750-FO-01		AILA	Switch	2024-11-06 08:15:05.900	2024-06-12 12:38:00	26.744	26.744	Ok	Ok	Ok	Ok
S-GT-VOR-AILA-3750-FO-01		AILA	Switch	2024-11-06 08:15:05.900	2024-08-09 15:31:00	26.772	26.772	Ok	Ok	Ok	Ok
S-GT-GEG-AILA-3750-FO-01		AILA	Switch	2024-11-06 08:15:05.900	2024-06-24 10:04:00	27.352	27.352	Ok	Ok	Ok	Ok
S-GT-LOC-AILA-3750-FO-01		AILA	Switch	2024-11-06 08:15:05.900	2024-01-03 12:05:00	27.496	27.496	Ok	Ok	Ok	Ok
S-GT-APP-AILA-3750-FO-01		AILA	Switch	2024-11-06 08:15:05.900	2023-02-13 19:25:00	27.532	27.532	Ok	Ok	Ok	Ok
S-GT-GS-AILA-3750-FO-01		AILA	Switch	2024-11-06 08:15:05.900	2024-10-23 12:02:00	30.864	30.864	Ok	Ok	Ok	Ok
S-GT-AUR-ACC-01-3750		AUR	Switch	2024-11-06 08:15:05.900	2024-06-24 10:05:00	31.084	31.084	Ok	Ok	Ok	Ok
R-GT-AUR-SUC-03-2911		AUR	Router	2024-11-06 08:15:05.900	2024-09-17 07:36:00	31.25	31.25	Ok	Ok	Ok	Ok
R-GT-AUR-SUC-02-3925		AUR	Router	2024-11-06 08:15:05.900	2023-10-02 16:55:00	8.518	8.518	Ok	Ok	Ok	Ok
R-GT-AUR-SUC-01-3925		AUR	Router	2024-11-06 08:15:05.900	2024-02-05 22:11:00	9.048	9.052	Ok	Ok	Ok	Ok
R-GT-AUR-VAC-01-2921		AUR	Router	2024-11-06 08:15:05.900	2024-06-24 10:03:00	21.412	21.412	Ok	Ok	Ok	Ok
R-GT-AUR-SUC-05-2921		AUR	Router	2024-11-06 08:15:05.900	2024-06-24 10:03:00	23.95	23.95	Ok	Ok	Ok	Ok
S-GT-GEG-ADMIN-9200-FO-01		AUR	Switch	2024-11-06 08:15:05.900	2024-06-24 10:03:00	60.774	60.774	Ok	Ok	Ok	Ok

Figura 66: *Frontend* de la API para la generación de reportes automaticos

- **Página HTML** (Cuadro 12): Esta página permite a los usuarios acceder a las distintas funciones de la API, como generar reportes o aplicar filtros a los datos disponibles.

```
<!-- WebApp/templates/index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>Node Data</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.
        css') }}">
    <link rel="icon" href="{{ url_for('static', filename='LOGO-COCESNA-
        COLOR-PNG.png') }}">
</head>
<body>
    <div class="header">
        <!-- Logo positioned in the top-left corner -->
        <a href="https://cocesna.org/site/">
            <img src="{{ url_for('static', filename='LOGO-COCESNA-
                BLANCO-PNG.png') }}" class="top-left-logo">
        </a>
        <!-- Logo positioned in the top-right corner -->
        <a href="XXX.XXX.XXX.XXX/Orion"></a>
            <img src="{{ url_for('static', filename='SolarWindsLogo.png
                ') }}" class="top-right-logo">
        </a>

        <h1>Node Data Monitoring</h1>
    </div>
    <!-- Buttons -->
    <div class="button-container">
        <a href="/download">
            <button>Download Excel</button>
        </a>
        <a href="/generate">
            <button>Reload Data</button>
        </a>
        <a href="/log">
            <button>View Log</button>
        </a>
    </div>
    <!-- Filters -->
    <div class="filter-container">
        <label for="typeFilter">Filter by Type:</label>
        <select id="typeFilter">
            <option value="">All</option>
        </select>

        <label for="cityFilter">Filter by City:</label>
        <select id="cityFilter">
            <option value="">All</option>
        </select>
    </div>

    <!-- Table -->
    <div class="data-table-container" id="data-table-container">
        <p style="text-align: center; font-size: 18px;">Loading data...
            </p>
    </div>

```

```
    <div class="footer">
        <p>&copy; 2024 Nicolas Urioste</p>
    </div>

    <script src="{{ url_for('static', filename='script.js') }}"></
        script>
</body>
</html>
```

Cuadro 12: Interfaz principal que permite descargar reportes, recargar datos y aplicar filtros de ubicación y tipo de dispositivo

- **Estilo y diseño** (Cuadro 13): El archivo CSS define el estilo y la apariencia de la interfaz, asegurando que la información sea fácil de leer y que la interfaz tenga una estructura clara.

```
/* WebApp/static/style.css */

/* Basic styling for the web page */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}

/* Header styling */
.header {
    background-color: #34495e;
    color: white;
    padding: 20px;
    text-align: center;
    position: relative;
    height: 100px;
}

.header h1 {
    margin: 0;
    line-height: 100px;
}

/* Logo positioning */
.header img.top-left-logo,
.header img.top-right-logo {
    position: absolute;
    top: 10px;
    width: 80px;
    height: auto;
}

.header img.top-left-logo {
    left: 10px;
}

.header img.top-right-logo {
    right: 10px;
}

```

```
/* Button container styling */
.button-container {
    display: flex;
    justify-content: center;
    margin: 20px 0;
}

.button-container a {
    text-decoration: none;
}

.button-container button {
    background: linear-gradient(135deg, #3498db, #2980b9);
    border: none;
    color: white;
    padding: 15px 30px;
    margin: 0 10px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: transform 0.2s ease-in-out, background 0.3s ease;
}

.button-container button:hover {
    transform: scale(1.1); /* Slight zoom effect */
    background: linear-gradient(135deg, #2980b9, #3498db);
}

/* Filter container styling */
.filter-container {
    display: flex;
    justify-content: center;
    gap: 20px;
    margin: 20px;
}

.filter-container label {
    margin-right: 10px;
    font-weight: bold;
}

.filter-container select {
    padding: 5px;
    font-size: 16px;
}

/* Data table container */
.data-table-container {
    overflow-x: auto;
    margin: 20px;
    background: white;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    padding: 20px;
}

/* Data table styling */
.data {
    width: 100%;
```

```
    border-collapse: separate;  /* Allow space between rows */
    margin: 0% 0;
    font-size: 14px;
}

/* Smooth transition for row color changes */
tr {
    transition: background-color 0.3s ease-in-out;
    animation: fadeIn 0.5s ease-in; /* Fade-in animation */
}

/* Row color styles */
tr.green {
    background-color: #d4edda; /* Light green */
}

tr.yellow {
    background-color: #fff3cd; /* Light yellow */
}

tr.red {
    background-color: #f8d7da; /* Light red */
}

/* Hover effect to enhance visibility */
tr:hover {
    background-color: #e0e0e0;
}

/* Fade-in animation */
@keyframes fadeIn {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}

/* Spinner animation */
.spinner {
    border: 4px solid #f3f3f3;
    border-top: 4px solid #3498db;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    animation: spin 1s linear infinite;
    margin: 20px auto;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

/* Footer styling */
.footer {
    text-align: center;
    padding: 10px;
```

```
    background-color: #34495e;
    color: white;
    position: relative;
    bottom: 0;
    width: 100%;
}
```

Cuadro 13: Estiliza la interfaz web con un diseño claro, efectos de transición y realce visual de datos críticos en tablas

- **JavaScript** (Cuadro 14): Utilizado para mejorar la experiencia del usuario al interactuar con la interfaz, proporcionando funcionalidades como validación de formularios y actualización dinámica de contenido.

```
document.addEventListener('DOMContentLoaded', function () {
    loadFilters();
    loadTableData();

    document.getElementById('typeFilter').addEventListener('change',
        loadTableData);
    document.getElementById('cityFilter').addEventListener('change',
        loadTableData);
});

function loadFilters() {
    fetch('apifilters')
        .then(response = response.json())
        .then(data = {
            populateFilter('typeFilter', data.types);
            populateFilter('cityFilter', data.cities);
        })
        .catch(error = console.error('Error loading filters', error));
}

function populateFilter(filterId, options) {
    const filter = document.getElementById(filterId);
    filter.innerHTML = 'option value=Alloption';

    options.forEach(option = {
        const opt = document.createElement('option');
        opt.value = option;
        opt.textContent = option;
        filter.appendChild(opt);
    });
}

function loadTableData() {
    const type = document.getElementById('typeFilter').value;
    const city = document.getElementById('cityFilter').value;

    const tableContainer = document.getElementById('data-table-
        container');
    tableContainer.innerHTML = 'div class=spinnerdiv';  Spinner while
        loading

    const query = new URLSearchParams({
        type type  '',
        city city  ''
```

```
    }).toString();

    fetch(`apitable${query}`)
        .then(response = response.json())
        .then(data = {
            tableContainer.innerHTML = generateTableHTML(data);
        })
        .catch(error = {
            console.error('Error loading table data', error);
            tableContainer.innerHTML = 'p style=text-align center; font
                -size 18px; color red;Failed to load data. Please try
                again later.p';
        });
}

function generateTableHTML(data) {
    if (data.length === 0) {
        return 'p style=text-align center; font-size 18px;No data
            available for the selected filters.p';
    }

     Define column headers and replace specific words
    const headers = Object.keys(data[0]).map(header = renameHeader(
        header));

    let html = `table class=datatheadtr${headers.map(header = `th${
        header}th`).join('')}trtheadtbody`;

    data.forEach(row = {
        const rowClass = getRowClass(row);
        html += `tr class=${rowClass}${Object.values(row).map(value = `
            td${value}td`).join('')}tr`;
    });

    html += 'tbodytable';
    return html;
}

 Helper function to rename specific headers
function renameHeader(header) {
    const headerMap = {
        'PowerSupplyStatus' 'Power Supply',
        'TemperatureStatus' 'Temperature',
        'FanStatus' 'Fan',
        'InterfaceStatus' 'Interface'
    };
    return headerMap[header]  header;  Use mapped name or original if
        no match
}


function getRowClass(row) {
    let warnings = 0;

    ['MemoryUsage', 'MemoryUsage72h'].forEach(key = {
        const value = parseFloat(row[key]);
        if (!isNaN(value)) {
            if (value = 90) warnings += 1;
            else if (value = 80) warnings += 1;
```

```
93          }
94      });
95
96      ['PowerSupplyStatus', 'TemperatureStatus', 'FanStatus', '
            InterfaceStatus'].forEach(key = {
97          const status = row[key];
98          if (status === 'Bad'  status === 'Warning') warnings += 1;
99      });
100
101     if (warnings  1) return 'red';
102     if (warnings === 1) return 'yellow';
103     return 'green';
104 }
```

Cuadro 14: Gestiona la carga de filtros y datos en la interfaz, resaltando estados críticos para un monitoreo visual eficiente

11.7. Ventajas de utilizar Docker para la API

El uso de Docker para contenerizar la API trajo numerosas ventajas:

- **Portabilidad:** al estar contenida en Docker, la API puede ejecutarse en cualquier sistema que tenga Docker instalado, garantizando consistencia y reduciendo problemas de compatibilidad.
- **Despliegue rápido y sencillo:** docker Compose permite desplegar la API y todos sus componentes asociados con un solo comando, facilitando el proceso de configuración y eliminando errores humanos.
- **Actualización sin interrupciones:** gracias a Docker, se pueden realizar actualizaciones de la API sin interrumpir los servicios en producción, lo cual es crucial para la infraestructura crítica de COCESNA.

11.8. Entorno de pruebas con Rocky Linux

Como parte del desarrollo de la API *dockerizada*, se montó una máquina virtual con una distribución de Rocky Linux que sirvió como ambiente de pruebas para Docker. Este entorno fue fundamental para validar la funcionalidad de la API antes de su despliegue en producción y para identificar los paquetes y configuraciones necesarios en el servidor de COCESNA. La configuración de este entorno se muestra en el Cuadro 15. Algunos aspectos clave de este entorno de pruebas incluyeron:

- **Máquina virtual con Rocky Linux:** se utilizó Rocky Linux debido a su estabilidad y compatibilidad con entornos empresariales. Esta distribución proporcionó un entorno similar al que se utilizaría en producción, facilitando así las pruebas y reduciendo la probabilidad de problemas de compatibilidad al realizar el despliegue final.

- **Pruebas de Docker:** en este ambiente se instaló Docker y se llevaron a cabo diversas pruebas para asegurar que la API se ejecutara correctamente, verificando la contenedorización, la configuración de los volúmenes, y la comunicación entre los contenedores definidos en Docker Compose. Estas pruebas ayudaron a garantizar que la infraestructura necesaria estuviera lista para soportar la API en el entorno de producción de COCESNA.
- **Determinación de paquetes necesarios:** durante las pruebas, se documentaron los paquetes y dependencias adicionales que se requerían instalar para el correcto funcionamiento de Docker y de la API, tales como Docker Engine, Docker Compose, y algunas bibliotecas específicas de Python. Esto permitió una transición más fluida al entorno de producción, asegurando que el equipo de COCESNA contara con una guía clara sobre cómo configurar el servidor para alojar la aplicación.

````
# Rocky Linux Setup Guide

This guide provides the necessary steps to set up a Rocky Linux environment with Docker Engine, VirtualBox and VS Code.

## Changing Hostname

Change the hostname of your system.

```sh
sudo hostnamectl set-hostname api.deployment
```

Check the hostname:

```sh
hostname -f
```

## Updating the System

Update all packages to the latest version.

```sh
sudo yum update -y
```

## SSH Setup

Install and enable the SSH server.

```sh
sudo dnf install -y openssh-server
sudo systemctl start sshd
sudo systemctl enable sshd
```

Check the status of the SSH server

```sh
sudo systemctl status sshd
```

````

````
## Docker Setup

### Docker Installation

Set up the Docker repository

```sh
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Install Docker Engine, containerd, and Docker Compose.

```sh
sudo yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Start Docker

```sh
sudo systemctl start docker
```

Verify the Docker Engine installation by running the hello-world image.

```sh
sudo docker run hello-world
```

### Post-Installation Steps

Create the docker group.

```sh
sudo groupadd docker
```

Add your user to the docker group

```sh
sudo usermod -aG docker $USER
```

Activate the changes

```sh
newgrp docker
sudo system reboot now
```

Verify that the user can run Docker commands without **sudo**

```sh
docker run hello-world
```

### General Docker configuration
````

````
Configure Docker to start on boot with system
```sh
sudo systemctl enable docker.service
sudo systemctl enable containerd.service
```

To disable Docker on startup
```sh
sudo systemctl disable docker.service
sudo systemctl disable containerd.service
```

### Enable NAT for Docker
Load the necessary kernel modules.
```sh
sudo modprobe ip_tables
sudo modprobe iptable_nat
```
Enable and start nftables.
```sh
sudo systemctl enable nftables
sudo systemctl start nftables
```

## VirtualBox Setup
Download and install VirtualBox
```sh
sudo wget https://download.virtualbox.org/virtualbox/rpm/rhel/virtualbox.repo -P /etc/yum.repos.d/
sudo rpm --import https://www.virtualbox.org/download/oracle_vbox.asc
```

Install the EPEL repository and dependencies
```sh
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo dnf install binutils kernel-devel kernel-headers libgomp make patch gcc glibc-headers glibc-devel dkms -y
```

Install VirtualBox
```sh
sudo dnf install VirtualBox-7.0 -y
```

### Additional Steps
Enable the latest nftables.
```sh
sudo systemctl enable nftables
sudo systemctl start nftables
```

## Visual Studio Code Setup
Add the repository.
```sh
sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" | sudo tee /etc/yum.repos.d/vscode.
````
````

```
    repo > /dev/null
```

```

Update the cache and install Visual Studio Code.
```sh
dnf check-update
sudo dnf install code
```

Install KVM
Install KVM and related packages.
```sh
sudo dnf install qemu-kvm libvirt libvirt-daemon virt-install bridge-utils
```
```

Cuadro 15: Comandos de bash utilizados para configurar una VM basada en Rocky Linux 9.X en un entorno que soporte el uso de Docker

11.9. Consideraciones Finales

El desarrollo de la API *dockerizada* para la generación de reportes representa un gran avance en la capacidad de monitoreo y gestión de la infraestructura de telecomunicaciones de COCESNA. La utilización de Docker y Flask, así como la integración con la base de datos existente, permitieron crear una solución eficiente y altamente personalizable. Los reportes generados proporcionan información crítica de manera rápida y accesible, permitiendo a los administradores identificar problemas, analizar el rendimiento de los dispositivos y tomar decisiones informadas para mantener la estabilidad de la red.

CAPÍTULO 12

Conclusiones

- El proceso de implementación de un gestor de red se llevó a cabo de manera satisfactoria, permitiendo optimizar la capacidad de gestión y supervisión de esta misma.
- La integración del protocolo SMTP, junto con un sistema de alarmas en el gestor de red, permitió transmitir alertas en tiempo real y notificar al personal de turno de manera efectiva, mejorando la capacidad de respuesta ante incidentes en un 98.6 %.
- La identificación y etiquetado de los enlaces se realizó de manera precisa y eficiente, lo cual facilitó el diagnóstico de fallas en la red y redujo considerablemente los tiempos de intervención.
- La generación de mapas de ubicación geográfica y topología de la red permitió una visualización clara de la distribución física y lógica, favoreciendo la gestión operativa y la planificación futura de la infraestructura.
- La *dockerización* de una API para la generación automatizada de reportes permitió centralizar y automatizar la recolección de datos, optimizando así la eficiencia operativa al reducir la intervención manual y agilizar la creación de informes consistentes y precisos. Esto resultó en una mejora del 40 % en la capacidad de respuesta y en la calidad de los reportes generados, contribuyendo a una mejor toma de decisiones basada en información confiable.
- La investigación presenta limitaciones relacionadas con la dependencia del protocolo SMTP para las alertas, que puede afectar la fiabilidad y rapidez de las notificaciones, y con la capacidad limitada del gestor de red para adaptarse a un entorno complejo. Esto sugiere explorar tecnologías emergentes que mejoren la eficiencia de las notificaciones y la escalabilidad del sistema.

CAPÍTULO 13

Recomendaciones

- **Capacitación del personal técnico**: proporcionar formación para el personal encargado del sistema de monitoreo, asegurando un conocimiento sólido de las funcionalidades avanzadas de la herramienta y una capacidad de respuesta eficiente ante alertas y problemas imprevistos.
- **Dominio de la base de datos**: fomentar el conocimiento y uso de la estructura de la base de datos para facilitar la creación y optimización de consultas, incrementando la eficiencia del análisis de datos.
- **Ampliación del alcance del monitoreo**: evaluar la posibilidad de extender el monitoreo a otros dispositivos críticos de la red de COCESNA que actualmente no están siendo monitoreados, para aumentar la visibilidad de la infraestructura y reducir riesgos operativos.
- **Actualización del servidor de SW NPM**: actualizar el servidor de SolarWinds NPM a la versión más reciente disponible, aprovechando las mejoras en rendimiento, seguridad y funcionalidades que puedan estar disponibles en las nuevas versiones.
- **Actualizar la base de datos de MSSQL Server**: actualizar la base de datos de MSSQL Server a una versión más reciente y soportada, para garantizar la compatibilidad con las últimas versiones de SolarWinds NPM, soporte con las últimas tecnologías (en especial API) y mejorar la seguridad y el rendimiento del sistema.
- **Actualizar la versión de SW NPM**: actualizar la versión de SolarWinds NPM a la más reciente disponible, para aprovechar las nuevas funcionalidades como monitoreo de Docker, mejoras de rendimiento y correcciones de errores que se hayan implementado en las versiones más recientes.
- **Implementación de redundancia**: implementar mecanismos de redundancia tanto para el sistema de monitoreo como para la base de datos, asegurando la continuidad del servicio en caso de fallas.

- **Consideraciones de ciberseguridad**: implementar segmentación y aislamiento de redes para mejorar la seguridad del sistema, asegurando que los diferentes segmentos operen de forma independiente para reducir la superficie de ataque.
- **Pruebas en máquina virtual antes de trabajar en el servidor**: realizar pruebas en una máquina virtual antes de aplicar cambios al servidor de producción. Documentar los comandos utilizados facilita una implementación segura. Las máquinas virtuales permiten restaurar una imagen previa con mayor facilidad que un servidor físico.
- **Utilización de VMware Workstation Pro**: utilizar VMware Workstation Pro, que está disponible de manera gratuita para estudiantes, ya que ofrece capacidades de snapshots para máquinas virtuales y un mayor control sobre las redes virtualizadas, facilitando el desarrollo y pruebas en entornos aislados.
- **Segmentación y abstracción de contenedores Docker**: implementar estrategias de segmentación y abstracción al utilizar Docker para dividir aplicaciones en múltiples contenedores independientes, mejorando la modularidad y facilitando el mantenimiento y la escalabilidad de las soluciones, al permitir que cada componente se desarrolle, actualice y despliegue de manera aislada.
- **Uso de imágenes de DockerHub basadas en AlpineLinux**: preferir el uso de imágenes de DockerHub basadas en AlpineLinux debido a su menor peso y menor cantidad de paquetes, lo cual reduce la superficie vulnerable a ataques.
- **Generación de mapas en el gestor de red**: al generar mapas en el gestor de red, se recomienda comenzar con los mapas de las ubicaciones específicas y luego proceder al mapa de vista global. Este enfoque facilita la organización de la infraestructura visual.
- **Comunicación efectiva con el personal**: mantener un canal de comunicación continuo y efectivo con el personal destinatario de la aplicación para obtener retroalimentación constante y garantizar que las necesidades sean satisfechas.

CAPÍTULO 14

Bibliografía

[1] GSMA, *Case study: AT&T - future networks*, https://web.archive.org/web/20240412114402/https://www.gsma.com/futurenetworks/wiki/att-energy-efficiency-as-a-service/, 2020.

[2] O. Ergun, *Case Studies: Network Management Insights*, https://orhanergun.net/case-studies-network-management-insights, 2024.

[3] IBM, *What is Network Monitoring*, https://www.ibm.com/topics/network-monitoring,

[4] D. Mauro y K. Schmidt, *Essential SNMP: Help for System and Network Administrators*. O'Reilly Media, 2005, ISBN: 9780596552770. dirección: https://books.google.com.gt/books?id=65_0d25EpB4C.

[5] W. Stallings, *High-speed Networks and Internets: Performance and Quality of Service* (William Stallings books on computer and data communications technology). Prentice Hall, 2002, págs. 11-16, ISBN: 9780130322210. dirección: https://books.google.com.gt/books?id=fP1SAAAAMAAJ.

[6] T. Nadeau y K. Gray, *SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies*. O'Reilly Media, 2013, págs. 70-90, ISBN: 9781449342449. dirección: https://books.google.com.gt/books?id=Bc1qAAAAQBAJ.

[7] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky y S. Uhlig, «Software-Defined Networking: A Comprehensive Survey,» *Proceedings of the IEEE*, vol. 103, págs. 14-76, 2014.

[8] ONF, *Software-Defined Networking: The New Norm for Networks*, https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/, 2012.

[9] M. Birk, G. Choudhury, B. Cortez et al., «Evolving to an SDN-enabled isp backbone: key technologies and applications,» *IEEE Communications Magazine*, vol. 54, n.º 10, págs. 129-135, 2016. DOI: 10.1109/MCOM.2016.7588281.

[10] M. Birk, G. Choudhury, B. Cortez et al., «Evolving to an SDN-enabled isp backbone: key technologies and applications,» *IEEE Communications Magazine*, vol. 54, n.º 10, págs. 129-135, 2016. DOI: 10.1109/MCOM.2016.7588281.

[11] W. Stallings, *SNMP, SNMPv2, and RMON: Practical Network Management*. Addison-Wesley, 1996, ISBN: 9780201634792. dirección: https://books.google.com.gt/books?id=b_pSAAAAMAAJ.

[12] M. Lucas, *SNMP Mastery*. Tilted Windmill Press, 2020, ISBN: 9781642350364. dirección: https://books.google.com.gt/books?id=cM5ZzgEACAAJ.

[13] *RFC 1157, A Simple Network Management Protocol (SNMP)*, https://www.rfc-editor.org/rfc/rfc1157, 1990.

[14] N. pl, *SNMP protocol: Network Basic. AL0-037* (Network Basic). NOITE S.C. dirección: https://books.google.com.gt/books?id=z_1zCwAAQBAJ.

[15] W. Stallings, «SNMP and SNMPv2: the infrastructure for network management,» *IEEE Communications Magazine*, vol. 36, n.º 3, págs. 37-43, 1998. DOI: 10.1109/35.663326.

[16] S. Unión Internacional de Telecomunicaciones (Ginebra, *Identificadores de objeto (OID) y sus autoridades de registro*. UIT, 2012, ISBN: 9789261137830. dirección: https://books.google.com.gt/books?id=8ln7jgEACAAJ.

[17] *Management Information Base for Network Management of TCP/IP-based internets*, https://www.rfc-editor.org/rfc/rfc1156, 1990.

[18] Cisco, *What is network management?* https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-network-management.html, 2023.

[19] K. Douglas y S. Douglas, *PostgreSQL: The Comprehensive Guide to Building, Programming, and Administering PostgresSQL Databases* (Developer's library). Sams Pub., 2006, ISBN: 9780672327568. dirección: https://books.google.com.gt/books?id=ybXCQgAACAAJ.

[20] G. Reese, R. Yarger, T. King y H. Williams, *Managing & Using MySQL: Open Source SQL Databases for Managing Information & Web Sites* (Nutshell handbook). O'Reilly Media, Incorporated, 2002, ISBN: 9780596002114. dirección: https://books.google.com.gt/books?id=kVB1wiF87ooC.

[21] W. Khan, W. Ahmad, B. Luo y E. Ahmed, «SQL Database with physical database tuning technique and NoSQL graph database comparisons,» en *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, págs. 110-116. DOI: 10.1109/ITNEC.2019.8729264.

[22] Microsoft, *What is SQL Server?* https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16, 2024.

[23] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» Tesis doct., University of California, Irvine, 2000.

[24] I. Jacobson, P. Ng, P. McMahon, I. Spence y S. Lidman, *The Essence of Software Engineering: Applying the SEMAT Kernel*. Pearson Education, 2013, ISBN: 9780133153132. dirección: https://books.google.com.gt/books?id=f8fgifs0pO0C.

[25] J. E. Hopcroft y J. D. Ullman, «Set Merging Algorithms,» en *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, ACM, 1972, págs. 251-264. DOI: 10.1145/800297.811532. dirección: https://dl.acm.org/doi/pdf/10.1145/800297.811532.

[26] M. P. Papazoglou y W.-J. van den Heuvel, «Service-Oriented Architectures: Approaches, Technologies and Research Issues,» *The VLDB Journal*, vol. 16, n.º 3, págs. 389-415, 2007. DOI: https://doi.org/10.1007/s00778-007-0044-3.

[27] G. Hohpe y B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* (Addison-Wesley Signature Series (Fowler)). Pearson Education, 2012, ISBN: 9780133065107. dirección: https://books.google.com.gt/books?id=qqB7nrrna_sC.

[28] Microsoft Corporation, *Windows API*, 2021.

[29] S. Malik y D.-H. Kim, «A comparison of RESTful vs. SOAP web services in actuator networks,» en *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017, págs. 753-755. DOI: 10.1109/ICUFN.2017.7993893.

[30] E. Gamma, R. Helm, R. Johnson y J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley professional computing series). Addison-Wesley, 1995, ISBN: 9783827328243. dirección: https://books.google.com.gt/books?id=jUvf7wMUGcUC.

[31] J. Webber, S. Parastatidis e I. Robinson, *REST in Practice: Hypermedia and Systems Architecture* (Theory in practice series). O'Reilly Media, 2010, ISBN: 9781449396923. dirección: https://books.google.com.gt/books?id=1D24-cGQRdsC.

[32] G. Blokdyk, *Private API a Complete Guide - 2020 Edition*. Emereo Pty Limited, 2020, ISBN: 9781867326816. dirección: https://books.google.com.gt/books?id=k8BGzQEACAAJ.

[33] T. Erl, *Service-oriented Architecture: Concepts, Technology, and Design* (Prentice Hall service-oriented computing series from Thomas Erl). Prentice Hall Professional Technical Reference, 2005, ISBN: 9780131858589. dirección: https://books.google.com.gt/books?id=GN1QAAAAMAAJ.

[34] E. G. Nadhan, «Service-Oriented Architecture: Implementation Challenges,» *IT Professional*, vol. 6, n.º 6, págs. 10-12, 2004.

[35] T. Iijima, K. Toumura, H. Kimura, M. Kitani y T. Miyamoto, «Development of Management Interface to Configure Network Equipment,» en *2007 International Symposium on Applications and the Internet Workshops*, 2007, págs. 32-32. DOI: 10.1109/SAINT-W.2007.41.

[36] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari y A. R. B. C. Hussin, «Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation,» *Information Systems*, vol. 91, pág. 101 491, 2020, ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2020.101491. dirección: https://www.sciencedirect.com/science/article/pii/S0306437920300028.

[37] L. Richardson y S. Ruby, *RESTful Web Services*. O'Reilly Media, 2008, ISBN: 9780596554606. dirección: https://books.google.com.gt/books?id=XUaErakHsoAC.

[38] A. Tanenbaum y M. van Steen, *Distributed Systems: Principles and Paradigms*. Pearson Prentice Hall, 2007, ISBN: 9780132392273. dirección: https://books.google.com.gt/books?id=DL8ZAQAAIAAJ.

[39] G. Stattenberger, T. Braun y M. Brunner, «A platform-independent API for quality of service management,» en *2001 IEEE Workshop on High Performance Switching and Routing (IEEE Cat. No.01TH8552)*, 2001, págs. 255-259. DOI: 10.1109/HPSR.2001.923642.

[40] H. Deitel y P. Deitel, *Java How to Program*. Pearson Education Limited, 2013, ISBN: 9781447930167. dirección: https://books.google.com.gt/books?id=pCtTBwAAQBAJ.

[41] L. Bass, P. Clements y R. Kazman, *Software Architecture in Practice: Software Architect Practice_ c3* (SEI Series in Software Engineering). Pearson Education, 2012, ISBN: 9780132942782. dirección: https://books.google.com.gt/books?id=-II73rBDXCYC.

[42] D. Merkel, «Docker: lightweight Linux containers for consistent development and deployment,» *Linux J.*, vol. 2014, n.º 239, 2014, ISSN: 1075-3583.

[43] L. Manases y D. Zinca, «Automation of Network Traffic Monitoring using Docker images of Snort3, Grafana and a custom API,» en *2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 2022, págs. 1-4. DOI: 10.1109/RoEduNet57163.2022.9921063.

[44] Docker, *Get Started with Docker*, https://docs.docker.com/get-started/overview/, 2024.

[45] J. Nickoloff y S. Kuenzli, *Docker in Action, Second Edition*. Manning, 2019, ISBN: 9781638351740. dirección: https://books.google.com.gt/books?id=qzozEAAAQBAJ.

CAPÍTULO 15

Anexos

15.1. Cronograma de actividades

TAREA	INICIO	FIN
FASE 1 - Base de datos		
Estudio de Docker y Microsoft SQL, toma de decisión de la plantilla que se utilizara e interconexión entre las tablas.	11-3-24	31-3-24
Evaluar e identificar los campos que deben ser llenados manualmente e irlos llenando.	1-4-24	7-4-24
Montar un servidor de Ubuntu e integrar la base de datos con un Docker de Microsoft SQL.	8-4-24	**21-4-24**
FASE 2 - *SolarWinds*		
Revisar la base de datos de *SolarWinds* y eliminar los equipos colocados por usuarios previos y agregar los nuevos.	22-4-24	1-5-24
Ingresar los equipos y tener una primera versión de como quedara el gestor de red.	2-5-24	16-5-24
Creación y asignación de grupos y subgrupos para etiquetado de equipos con el fin de facilitar su identificación de procesos internos de *SolarWinds* .	17-5-24	2-6-24
Investigación, creación e implementación de plantilla a utilizar para las distintas ubicaciones y necesidades de los equipos.	3-6-24	**16-6-24**
FASE 3 - Mapa de la red		
Obtener la ubicación geográfica de los equipos y generar un mapa con dicha información el cual represente el estado de dicha ubicación y la conexión que posee con otras.	17-6-24	30-6-24
Crear un mapa de la topología de la red para cada una de las ubicaciones. En este mapa se debe representar tanto el estado del equipo como el de sus interconexiones.	1-7-24	16-8-24
Desplegar los mapas en la biñeta correspondiente dentro del *SolarWinds*	17-8-24	**20-8-24**
FASE 4 - Interfaz web dockerizada de la base de datos		
Estudio de diseño de paginas web y diseñar el front end de la página.	21-8-24	3-9-24
Interconectar el front end con el back end (base de datos de la fase 1)	4-9-24	18-9-24
Buscar bugs en el programa y posteriormente arreglarlos.	19-9-24	22-9-24
Dockerizar la pagina web final y montarla en el servidor de Rocky Linux	23-9-24	**10-10-24**
FASE 5 - Documentación		
Recopilar todos los archivos creados para este proyecto, incluyendo versiones pasadas, e ir ordenando en orden cronológico los cambios y lo que es funcional.	11-10-24	15-10-24
Crear archivos readme que expliquen como funcionan los diversos programas.	16-10-24	20-10-24
Realizar tutoriales explicando la funcionalidad y uso de la base de datos, *SolarWinds* y el mapa de la red.	21-10-24	**11-11-24**

Figura 67: Cronograma de actividades para el desarrollo del proyecto

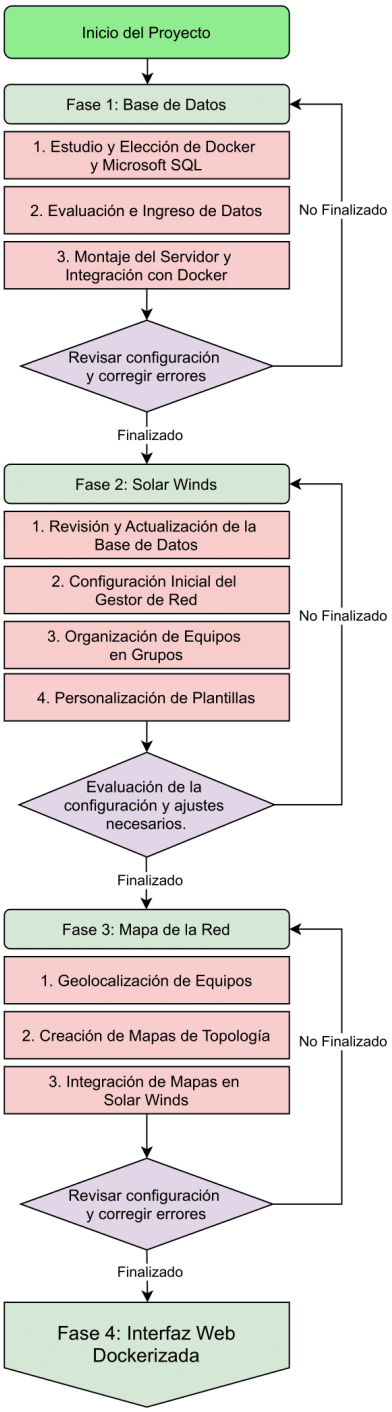


Figura 68: Primera parte del flujo del desarrollo del proyecto

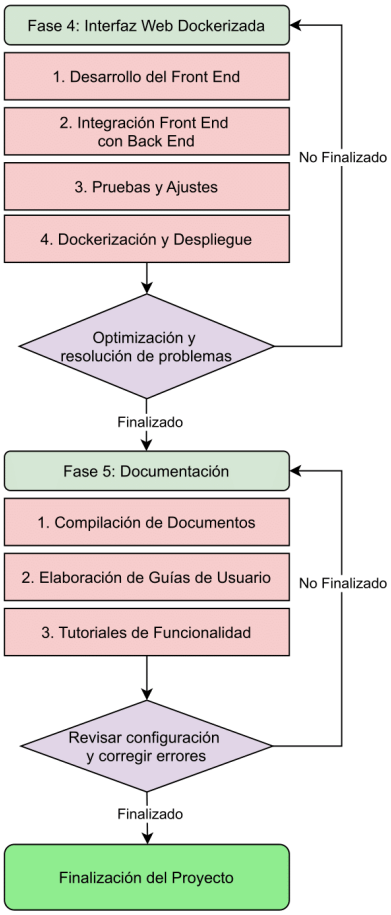


Figura 69: Segunda parte del flujo del desarrollo del proyecto

CAPÍTULO 16

Glosario

Dockerización Proceso de empaquetar una aplicación y sus dependencias en un contenedor Docker, garantizando que se ejecute de manera coherente en cualquier entorno. La dockerización facilita la implementación, escalabilidad y portabilidad de aplicaciones, promoviendo un entorno de desarrollo y producción más eficiente. IX, XI, 8, 14, 73

Endpoint Punto final de comunicación en una red o API, donde un servicio puede ser accedido por aplicaciones o dispositivos. En el contexto de una API, un endpoint representa una URL específica a la cual se pueden realizar peticiones para obtener o enviar datos. 74

Flask Framework minimalista de desarrollo web en Python, utilizado para construir aplicaciones web ligeras y APIs. Flask permite un diseño modular y extensible, ofreciendo herramientas y bibliotecas para gestionar rutas, peticiones y respuestas, sin imponer demasiada estructura. 71

Framework Estructura o plataforma de desarrollo que proporciona un conjunto de herramientas, bibliotecas y convenciones estandarizadas para facilitar la creación y organización de aplicaciones de software. Un framework ayuda a los desarrolladores a optimizar el proceso de desarrollo, al ofrecer una base reutilizable que simplifica tareas comunes. 71

Traps Notificaciones enviadas automáticamente por dispositivos SNMP a un gestor de red cuando ocurre un evento significativo o una falla en la red. 11

AILA (Aeropuerto La Aurora) Anillo de fibra óptica que conecta el Aeropuerto Internacional La Aurora con la red de telecomunicaciones aeronáuticas de COCESNA. 52

API (*Application Programming Interface*) Conjunto de reglas y protocolos que permiten que diferentes aplicaciones o servicios interactúen entre sí, facilitando la automatización y el intercambio de datos. IX, XI, 2, 6, 8, 12, 71

AUR (Aeropuerto La Aurora) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en el Aeropuerto Internacional La Aurora, Guatemala. 53, 67, 69

CLI (*Command Line Interface*) Interfaz de usuario que permite interactuar con el sistema operativo o software a través de comandos escritos. 13, 22, 23, 47

COCESNA (Corporación Centroamericana de Servicios de Navegación Aérea) Organización regional encargada de la gestión del espacio aéreo y la prestación de servicios de navegación aérea en Centroamérica. IX, XI, 1, 71

CSA (Cerro Santiago) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en Cerro Santiago, Jutiapa. 53, 67, 69

CSV (*Comma-Separated Values*) Formato de archivo simple utilizado para almacenar datos en forma de tabla, donde cada línea representa un registro y los campos están separados por comas. Los archivos CSV son ampliamente utilizados para la transferencia de datos entre aplicaciones debido a su simplicidad y compatibilidad. 77

DB (base de datos) Conjunto organizado de información o datos estructurados que se almacenan de manera digital y que pueden ser gestionados, consultados y actualizados de forma eficiente por un sistema de gestión de bases de datos. IX, XI, 5, 8, 12, 14, 60

DBeaver Herramienta universal de administración de bases de datos que permite realizar consultas SQL, conexiones y análisis de datos de diferentes tipos de bases de datos. 60

DevOps Conjunto de prácticas que combina el desarrollo de software (Dev) y la operación de sistemas (Ops) para facilitar el despliegue rápido y eficiente de aplicaciones y servicios. 15

Docker Plataforma de contenedores que permite empaquetar aplicaciones y sus dependencias en un entorno aislado, facilitando su despliegue y ejecución en diferentes sistemas. IX, XI, 2, 8, 14, 71

dynamic query En el contexto de la plataforma SW NPM, permite realizar consultas personalizadas a la base de datos para obtener información específica los nodos y en base a los resultados, incluirlos en los grupos pertenecientes. 29

GE (Gigabit Ethernet) Estándar de red que define la transmisión de datos a velocidades de hasta 1 gigabit por segundo. 68

Gestor de red Plataforma utilizada para configurar, monitorear y gestionar el rendimiento de una red, en este caso implementada mediante SolarWinds NPM. IX, XI, 3, 6

ICMP (*Internet Control Message Protocol*) Protocolo de red utilizado para enviar mensajes de control y error, como los utilizados en herramientas de diagnóstico como "ping". 16

Infraestructura crítica Conjunto de sistemas y activos esenciales para el funcionamiento de una sociedad o empresa, cuya falla puede tener consecuencias severas, como las redes de telecomunicaciones aeronáuticas. 3

UI (*User Interface*) Espacio donde los usuarios interactúan con un software o dispositivo. La interfaz de usuario permite que los usuarios realicen tareas mediante elementos gráficos, botones, menús y otros componentes visuales, facilitando la comunicación entre el usuario y el sistema. 79

IOS (Puerto Barrios) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en Puerto Barrios, Izabal. 54, 69

IP (*Internet Protocol*) Protocolo de comunicación utilizado para identificar y direccionar dispositivos en una red. Cada dispositivo tiene una dirección IP única que permite su localización e intercambio de datos a través de redes como Internet. 5

Kernel Núcleo central de un sistema operativo que actúa como intermediario entre el hardware y el software. El kernel gestiona los recursos del sistema, controla la ejecución de procesos, la memoria y los dispositivos periféricos, proporcionando una interfaz para que las aplicaciones interactúen con el hardware de manera segura y eficiente. 14

Mapa de topología Representación gráfica de los dispositivos y conexiones dentro de una red, que facilita el diagnóstico y monitoreo de su estado. IX, XI, 5, 39

MIB (*Management Information Base*) Base de datos jerárquica utilizada por SNMP que contiene definiciones de las entidades que pueden ser gestionadas en una red. 10

MS SQL (Microsoft SQL Server) Sistema de gestión de bases de datos relacional que almacena y recupera datos a petición de aplicaciones de red, utilizado en este proyecto para gestionar información crítica de monitoreo. IX, XI, 8, 12, 14, 58, 62, 70

SSMS (MS SQL Management Studio) Herramienta gráfica utilizada para gestionar bases de datos Microsoft SQL Server, permitiendo realizar consultas, backups y gestión de la base de datos. 12

NIK (Cerro Niktún) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en cerro Niktún, Petén. 54

Nodo Dispositivo o punto de conexión dentro de una red, como routers, switches o servidores, que puede ser monitoreado o gestionado a través de SNMP o ICMP. IX, 3, 5, 14, 16

OID (*Object Identifier*) Identificadores únicos utilizados por SNMP para representar y gestionar la información de los dispositivos de red. 10

ONA (Orion Network Atlas) Herramienta de SolarWinds utilizada para crear mapas visuales de la topología de la red, facilitando la supervisión y gestión de dispositivos conectados. 39

OF (OpenFlow) Protocolo utilizado en redes SDN que permite al controlador SDN comunicarse con los switches de red, gestionando el flujo de datos. 9

PAL (Cerro Palencia) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en Cerro Palencia, Guatemala. 55, 67, 69

Plano de control Componente de una red encargado de gestionar las decisiones relacionadas con el enrutamiento, la administración de políticas y la configuración de la red. Separa la lógica de control de la red del flujo de datos, permitiendo una gestión centralizada y una mejor optimización de los recursos. 9

Plano de datos Componente de una red responsable de transportar los datos del usuario a través de la infraestructura de red. Ejecuta las decisiones tomadas por el Plano de Control, encaminando el tráfico y aplicando políticas a medida que los paquetes se transmiten por la red. 9

Polling Proceso mediante el cual un gestor de red solicita periódicamente información a los dispositivos de red para monitorear su estado. 11, 18

query Consulta a una base de datos para recuperar información específica. En este proyecto, se utiliza SQL para hacer consultas a la base de datos de SolarWinds Orion. 13, 66–69

Radioenlace Sistema de comunicación inalámbrica que utiliza ondas de radio para transmitir datos entre dos puntos fijos, generalmente en línea de vista directa. Los radioenlaces son comúnmente utilizados para interconectar redes en ubicaciones remotas o para extender la cobertura de la red en áreas donde el cableado es impráctico. x, xii

REU (Retalhuleu) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en Retalhuleu, Guatemala. 17, 55

SDN (*Software Defined Networking*) Arquitectura de redes que separa el plano de control (que decide cómo se envían los datos) del plano de datos (que envía los datos) para facilitar la gestión y configuración de redes de forma centralizada. 9

SEB (Santa Elena Barrillas) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en Santa Elena Barrillas, Guatemala. 43, 56, 67, 69

SLA (*Service Level Agreement*) Contrato o acuerdo entre un proveedor de servicios y un cliente que define el nivel de servicio esperado, incluyendo métricas como disponibilidad, tiempo de respuesta y tiempo de resolución de problemas. ix, xi

SMTP (*Simple Mail Transfer Protocol*) Protocolo de comunicación utilizado para el envío de correos electrónicos a través de redes IP. SMTP es responsable de la transferencia de mensajes de correo desde el cliente de correo saliente hacia el servidor de correo del destinatario. x, xii, 33

SNMP (*Simple Network Management Protocol*) Protocolo estándar utilizado para la super isión y gestión remota de dispositivos de red, como routers y switches, mediante la recopilación de datos y el envío de alertas. ix, xi, 3, 10, 14, 16

SolarWinds Orion Plataforma de monitoreo y gestión de redes que proporciona visibilidad integral de los dispositivos y el tráfico de la red, con la capacidad de enviar alertas y generar reportes. 60

SW NPM (SolarWinds Network Performance Monitor) Herramienta de monitoreo de red utilizada para supervisar y diagnosticar el estado de los dispositivos de red en tiempo real, proporcionando alertas y reportes automatizados. ix, xi, 2, 3, 5, 8, 11, 16, 58

Telecomunicaciones aeronáuticas Redes de comunicación especializadas que proporcionan servicios de soporte para la navegación aérea, facilitando la transmisión de datos entre aeronaves y centros de control de tráfico aéreo. 1, 8

TGE (Ten Gigabit Ethernet) Estándar de red que define la transmisión de datos a velocidades de hasta 10 gigabit por segundo. 68

TIK (Aeropuerto Internacional Mundo Maya) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en Aeropuerto Internacional Mundo Maya, Petén. 56, 67

TWR-AUR (Torre de Control AILA) Estación de la red de telecomunicaciones aeronáuticas de COCESNA ubicada en la Torre de Control del Aeropuerto Internacional La Aurora, Guatemala. 57

XLSX Formato de archivo utilizado por Microsoft Excel para almacenar datos en forma de hojas de cálculo. El formato XLSX permite organizar datos en tablas, realizar cálculos, aplicar formatos y crear gráficos. Es ampliamente utilizado para el análisis y presentación de datos en un entorno estructurado y visualmente organizado. 77