UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA

CARRERA DE INGENIERÍA DEL SOFTWARE

ASIGNATURA
ADMINISTRACIÓN DE REDES DE DATOS

TRABAJO FINAL
Establecer la configuración física y configuración básica de la red, de acuerdo al diagrama y la descripción correspondiente

INTEGRANTES:
Luis Vile Corte
Christian Patricio Crespo Ortega
Pablo Andrés Guartaza

MAYO - 2025

UNIVERSIDAD CATÓLICA DE CUENCA
UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN, E INNOVACIÓN TECNOLÓGICA
CARRERA DE SOFTWARE
MATRIZ CUENCA
RETO FINAL
ADMINISTRACIÓN DE REDES DE DATOS

PARTE I

Este informe detalla la implementación de una infraestructura de red corporativa que incluye conexión a Internet, segmentación por VLANs, configuración de enrutamiento OSPF y servicios de monitoreo SNMP. La red está diseñada para soportar múltiples departamentos (Ventas, Marketing, RRHH y Administración) con segmentación adecuada y acceso controlado a recursos.

La implementación está compuesta por:

- **1 Router ISP**: Proporciona conectividad a Internet
- **3 Routers internos**: R1, R2 y R3 para el enrutamiento interno
- **5 Switches**: S1, S2, S3, S4 y S5 para la conmutación de paquetes y segmentación VLAN
- **5 Computadoras virtuales (VPCs)**: Para los usuarios finales
- **1 Servidor**: Con conectividad directa al ISP, configurado con SNMP para monitoreo

La topología implementa una red jerárquica con separación clara entre el núcleo (routers), distribución (switches principales) y acceso (switches de usuario).

Direccionamiento IP

WAN

- **ISP a Internet**: DHCP (GigabitEthernet3)
- **ISP a R1**: 214.205.5.21/30 (ISP) - 214.205.5.22/30 (R1)
- **Servidor a ISP**: 64.100.1.3/24 - Gateway: 64.100.1.1

LAN

- **VLAN 10 (Ventas)**: 192.168.1.0/26 - Gateway: 192.168.1.1
- **VLAN 20 (Marketing)**: 192.168.1.64/26 - Gateway: 192.168.1.65
- **VLAN 30 (RRHH)**: 192.168.1.128/26 - Gateway: 192.168.1.129
- **VLAN 50 (Admins)**: 172.16.1.0/24 - Gateway: 172.16.1.1
- **VLAN 99 (Nativa)**: 172.16.0.0/24 - Gateway: 172.16.0.1
- **Enlaces entre routers**:
    - R1-R3: 192.168.1.248/30
    - R2-R3: 192.168.1.244/30
    - R1-R2: 192.168.1.240/30
- **Segmentos adicionales en R2**:
    - VLAN 10: 192.168.0.0/25 - Gateway: 192.168.0.1

- VLAN 30: 192.168.0.128/25 - Gateway: 192.168.0.129

Configuraciones Básicas (aplicadas a todos los dispositivos)

- Establecimiento de contraseñas seguras
- Encriptación de contraseñas
- Configuración de mensajes de advertencia (banner MOTD)
- Desactivación de búsqueda de dominio
- Nombres de host significativos

Configuración SSH

- Dominio: red.local/midominio.local
- Generación de claves RSA (módulo 1024)
- SSH versión 2 habilitado
- Autenticación local con usuario "cisco" (privilegio 15)

ISP

- Configurado como punto de conexión a Internet
- Enrutamiento estático hacia redes internas
- Interfaces:
  - G1: 64.100.1.2/24 (para servidor)
  - G2: 214.205.5.21/30 (para R1)
  - G3: DHCP (conexión a Internet)

Router R1

- Actúa como router de borde
- Conectado al ISP y a la red interna
- Configurado con OSPF (Router ID: 1.1.1.1)
- Distribuye ruta por defecto a la red interna
- Configuración SNMP para monitoreo

Router R2

- Configurado con enrutamiento OSPF (Router ID: 2.2.2.2)
- Interfaces troncales con encapsulación 802.1Q
- Subinterfaces para VLANs 10 y 30

Router R3

- Actúa como dispositivo central de enrutamiento
- Configurado con OSPF (Router ID: 3.3.3.3)
- Múltiples subinterfaces para todas las VLANs
- Interfaces troncales con encapsulación 802.1Q

Switches (S1-S5)

- Configuración de VLANs (10, 20, 30, 50, 99)

- Enlaces troncales con VLAN nativa 99
- Puertos de acceso asignados a VLANs específicas
- Configuración de management IP en VLAN 99
- Portfast habilitado en puertos de acceso en S4 y S5

Servidor

- Sistema operativo Linux (Ubuntu)
- Configuración de red mediante Netplan
- Instalación y configuración del servicio SNMP
- Configuración de comunidad SNMP "swlab" para monitoreo

Protocolos y Servicios Implementados

Enrutamiento OSPF

- Configurado en los tres routers internos (R1, R2, R3)
- IDs de Router únicos (1.1.1.1, 2.2.2.2, 3.3.3.3)
- Todas las redes en área 0
- R1 configurado para distribuir ruta por defecto

Seguridad

- SSH versión 2 para acceso remoto seguro
- Encriptación de contraseñas
- Mensajes de advertencia (banner MOTD)
- Autenticación local para SSH

Monitoreo SNMP

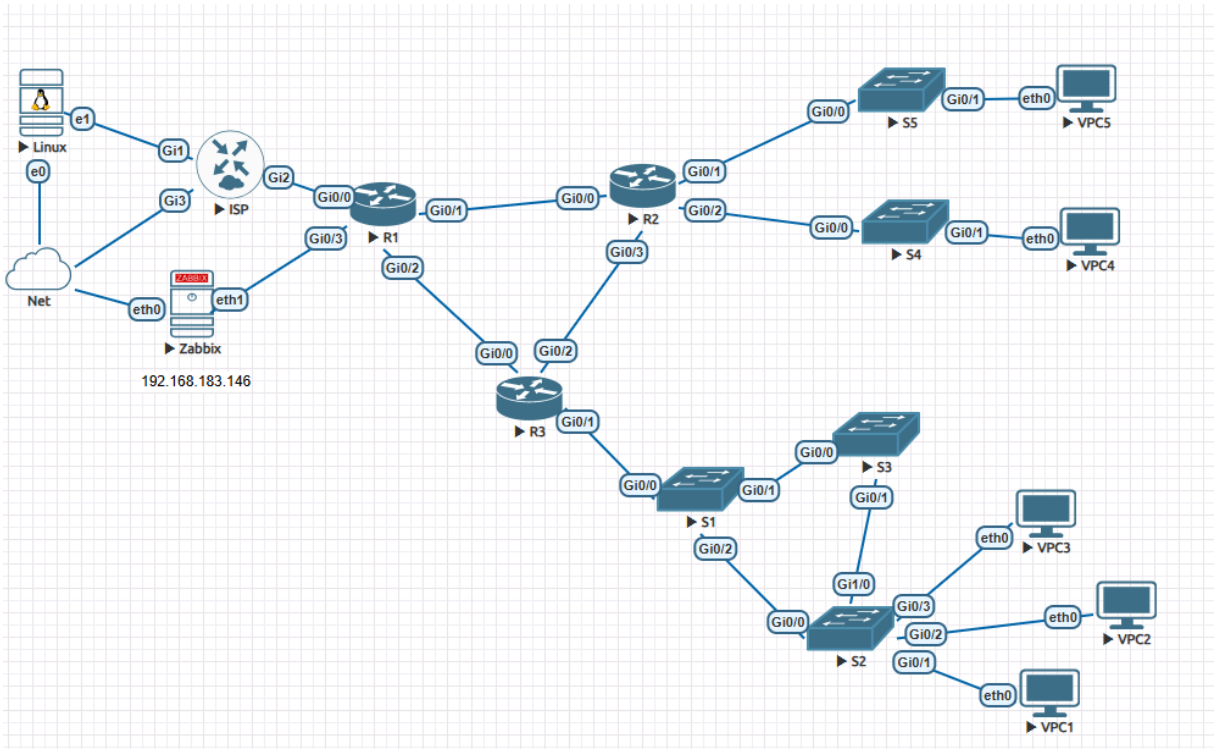
- Configurado en el servidor (64.100.1.3)
- Configurado en R1
- Comunidad "swlab" con acceso de solo lectura
- Usuario SNMPv3 en el servidor para mayor seguridad

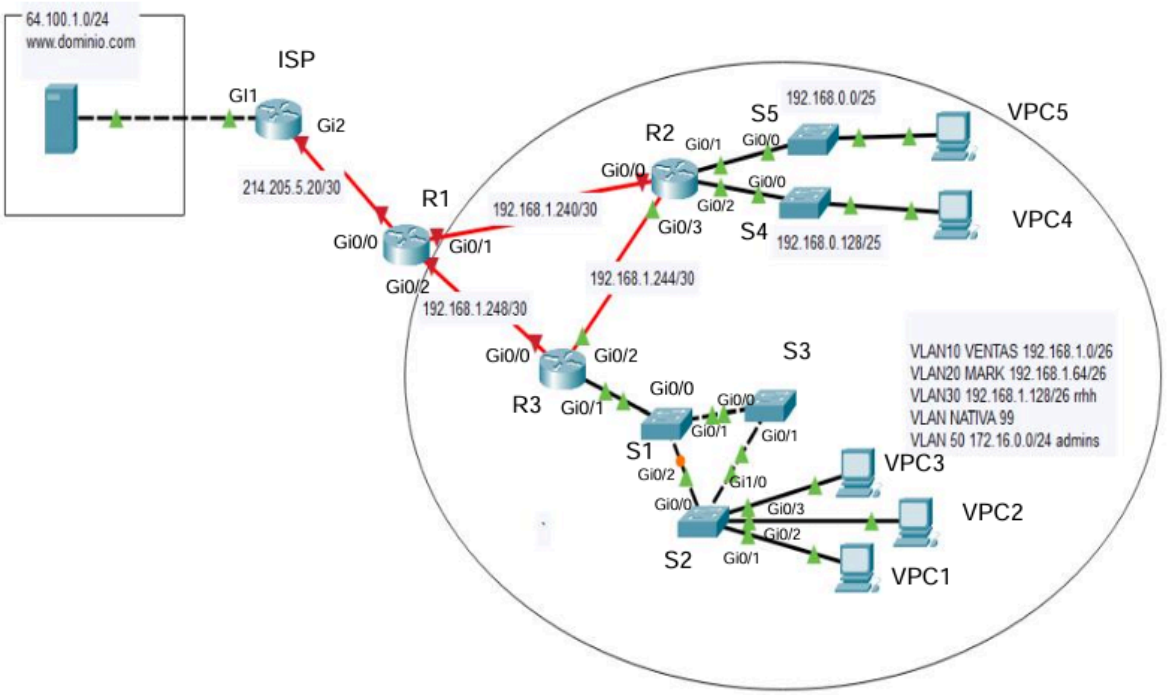
VLANs y Trunking

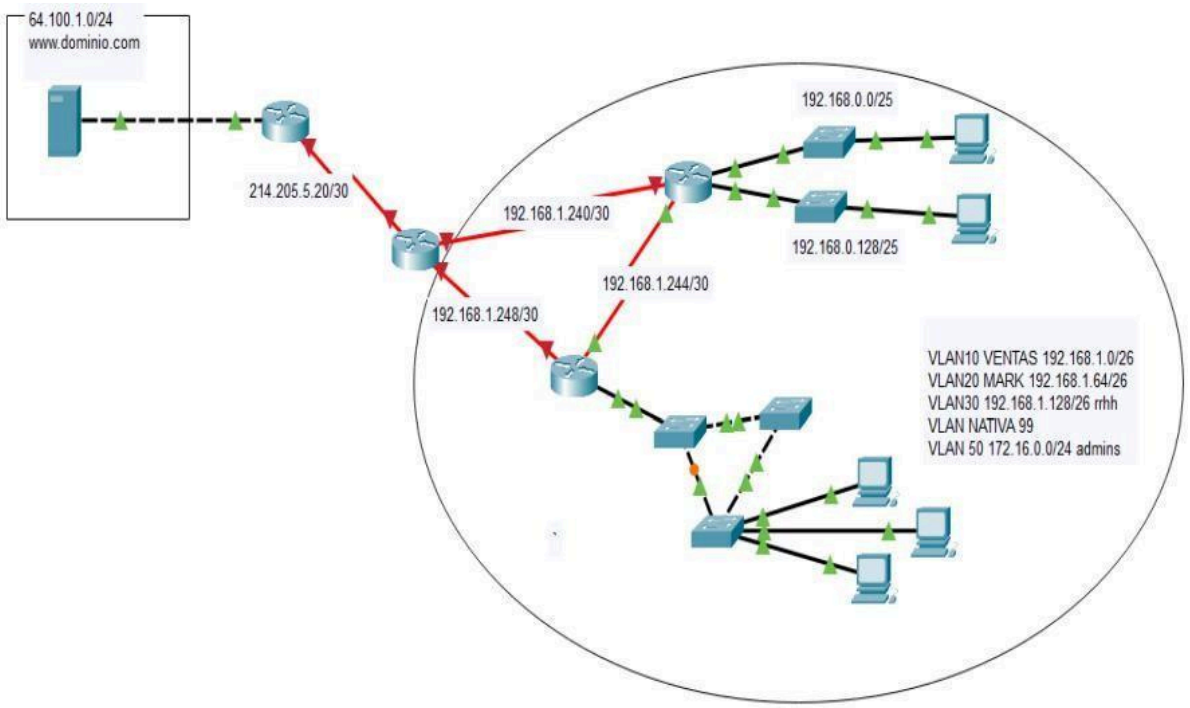
- VLANs por departamento: 10 (Ventas), 20 (Marketing), 30 (RRHH), 50 (Admins)
- VLAN 99 como VLAN nativa para enlaces troncales
- Encapsulación 802.1Q en todos los enlaces troncales

La implementación actual proporciona una infraestructura de red sólida con segmentación adecuada, enrutamiento dinámico, acceso seguro y capacidades básicas de monitoreo. Las configuraciones permiten una comunicación efectiva tanto interna como externa, mientras mantienen un nivel adecuado de seguridad y administración.

La estructura jerárquica implementada facilitará futuras expansiones y modificaciones según evolucionen las necesidades de la organización.


Linux
e1
e0
Gi1
Gi3
ISP
Gi2
Net
eth0
eth1
Zabbix
192.168.183.146
Gi0/0
Gi0/1
Gi0/3
Gi0/2
R1
R2
R3
S1
S2
S3
S4
S5
VPC1
VPC2
VPC3
VPC4
VPC5
Gi1/0


64.100.1.0/24
www.dominio.com
ISP
GI1
Gi2
214.205.5.20/30
R1
Gi0/0
Gi0/1
Gi0/2
192.168.1.240/30
192.168.1.244/30
192.168.1.248/30
R2
R3
S1
S2
S3
S4
S5
192.168.0.0/25
192.168.0.128/25
VPC1
VPC2
VPC3
VPC4
VPC5
VLAN10 VENTAS 192.168.1.0/26
VLAN20 MARK 192.168.1.64/26
VLAN30 192.168.1.128/26 rrhh
VLAN NATIVA 99
VLAN 50 172.16.0.0/24 admins



PARTE II

- Implemente administración de la red mediante SNMP sobre software libre, NTP y SYSLOG y que todos los resultados se puedan visualizar sobre un dashboard con PRTG

Configuración de Servidor - Interfaces de Red (Netplan)

Se ha implementado una configuración dual-homed mediante Netplan en el sistema Linux, lo que permite al servidor funcionar simultáneamente como host interno y como punto de monitoreo de la red externa:

```
yaml
network:
 version: 2
 renderer: networkd
 ethernets:
  eth0:
   dhcp4: yes              # Configuración dinámica para red interna
  eth1:
   dhcp4: no               # Configuración estática para red de monitoreo
   addresses: [64.100.1.3/24]   # Dirección IP del servidor de monitoreo
   gateway4: 64.100.1.1         # Puerta de enlace
   nameservers:
    addresses: [8.8.8.8, 8.8.4.4]  # Servidores DNS de Google
```

Esta configuración permite que:

- La interfaz eth0 obtenga parámetros dinámicamente vía DHCP, adecuada para la red interna.
- La interfaz eth1 mantenga una dirección estática (64.100.1.3) en la red de monitoreo (64.100.1.0/24).

Configuración de Zabbix (Interfaz de Red)

Para asegurar la correcta operación de Zabbix, se ha configurado explícitamente el archivo de configuración de la interfaz eth0 con los siguientes parámetros:

```
DEVICE=eth0

BOOTPROTO=static

IPADDR=198.168.183.146

NETMASK=255.255.255.0

GATEWAY=198.168.183.146

DNS1=8.8.8.8

DNS2=8.8.4.4

ONBOOT=yes
```

Este enfoque garantiza que el servidor Zabbix mantenga conectividad constante tanto para sus funciones de monitoreo como para acceso administrativo.

Sistema de Monitoreo SNMP

Configuración del Agente SNMP

Se ha instalado y configurado el protocolo SNMP en el servidor para permitir la recolección de información de estado del sistema:

```
sudo apt update

sudo apt install snmp snmpd snmp-mibs-downloader -y
```

La configuración del archivo /etc/snmp/snmpd.conf incluye:

```
# Escucha en la IP específica del servidor y puerto estándar 161

agentaddress udp:64.100.1.3:161


# Definición de vistas para control de acceso

view systemonly included .1.3.6.1.2.1.1          # Información del sistema

view systemonly included .1.3.6.1.2.1.25.1       # Recursos de hardware
```

```
view systemonly included .1.3.6.1.2.1.2          # Interfaces de red


# Acceso de solo lectura para equipos específicos
rocommunity swlab 64.100.1.1/32 -V systemonly      # Gateway
rocommunity swlab 64.100.1.2/32 -V systemonly      # ISP


# Configuración de SNMPv3 para mayor seguridad
createUser zabbixUser SHA Admin123 AES Zabbix456
rouser zabbixUser authpriv -V systemonly
```

Esta configuración implementa:

- Escucha únicamente en la dirección IP específica para mayor seguridad
- Control de acceso granular mediante vistas restringidas
- Acceso limitado solo al gateway (64.100.1.1) y al ISP (64.100.1.2)
- Autenticación avanzada mediante SNMPv3 para el usuario Zabbix

Configuración SNMP en Router R1

El equipo R1 también ha sido configurado para reportar su estado vía SNMP:

```
conf t
snmp-server community swlab RO
snmp-server location R1
snmp-server contact admin@ucacue.edu.ec
end
```

Esta configuración permite:

- Monitoreo centralizado del router R1 utilizando la misma comunidad "swlab"
- Información de ubicación y contacto administrativa para identificación rápida en alertas

Sistema de Registro Centralizado (Syslog)

Se ha implementado un sistema centralizado de logs utilizando Rsyslog para consolidar los registros de todos los dispositivos de red:

```
sudo apt install rsyslog -y
```

La configuración en /etc/rsyslog.conf habilita la recepción de mensajes remotos:

```
module(load="imudp")
```

```
input(type="imudp" port="514")
```

Adicionalmente, se han definido reglas específicas para dispositivos de red en /etc/rsyslog.d/network-devices.conf:

```
if $fromhost-ip startswith '192.168.0.' then /var/log/network-devices.log
& stop
```

Este sistema permite:

- Centralización de todos los registros de actividad y eventos
- Segregación de logs por tipo de dispositivo y red
- Base para análisis de correlación de eventos y auditoría de seguridad

Sincronización de Tiempo (NTP)

Se ha implementado Chrony como servidor/cliente NTP para garantizar la sincronización de tiempo precisa en toda la infraestructura:

```
sudo apt install chrony -y
```

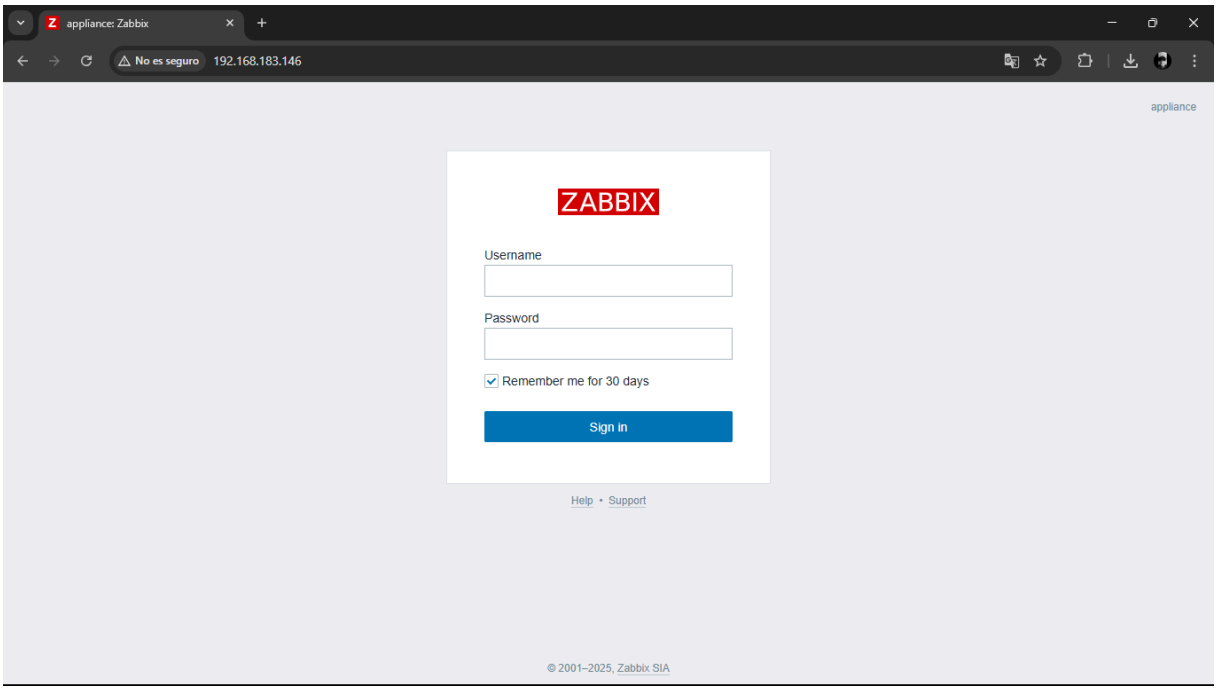
La configuración en /etc/chrony/chrony.conf establece conexión con servidores NTP regionales:

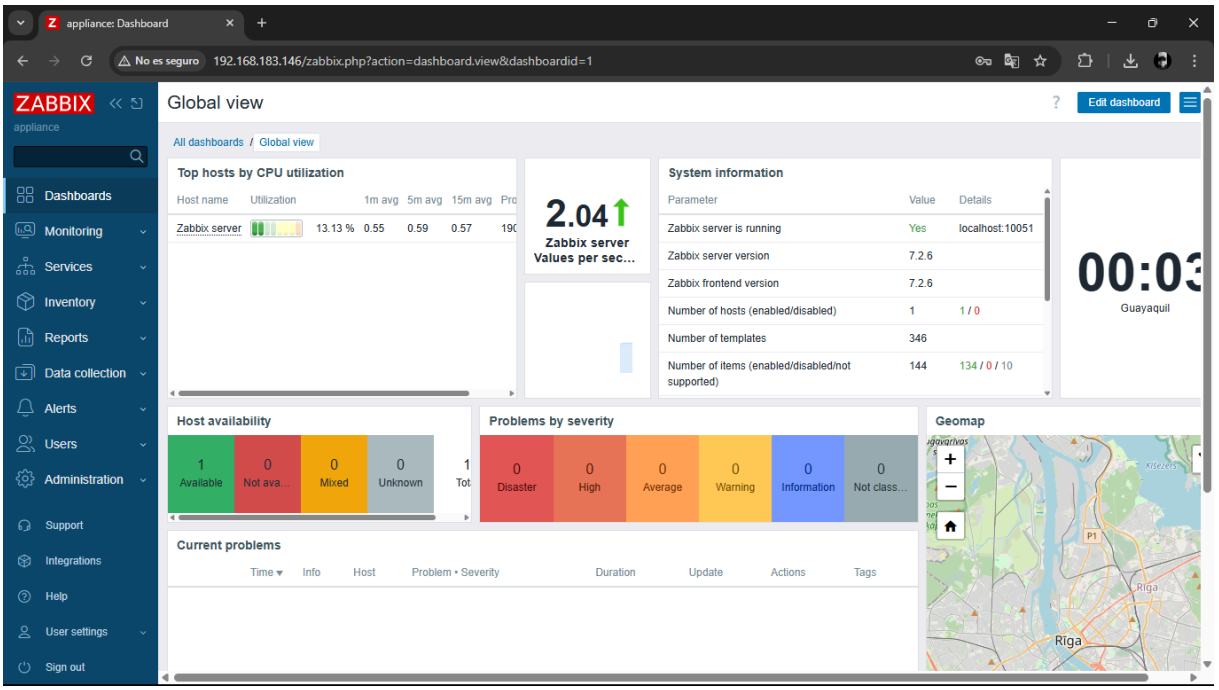
```
server 0.south-america.pool.ntp.org iburst
server 1.south-america.pool.ntp.org iburst
```

La correcta sincronización temporal es crítica para:

- Correlación precisa de eventos entre sistemas
- Validez de registros de auditoría
- Operación adecuada de protocolos sensibles al tiempo

Resultados:





PARTE III

.

Servicios REST sobre dispositivos de networking.

Desarrollo de una solución basada en servicios REST aplicados a dispositivos de red, utilizando el protocolo RESTCONF. La solución permite crear, visualizar y administrar interfaces de red (como

Loopback) de forma remota y eficiente, a través de una aplicación web desarrollada en React.

Este enfoque permite la integración directa con dispositivos que exponen configuraciones y datos operacionales a través de interfaces RESTful, eliminando la necesidad de acceso por consola o CLI tradicionales.

Objetivo

Desarrollar una herramienta web que consuma servicios RESTCONF para interactuar con interfaces de red en dispositivos compatibles, facilitando tareas como:

- Creación y edición de interfaces (nombre, IP, netmask)
- Consulta de todas las interfaces disponibles
- Visualización detallada de configuraciones específicas

Herramientas y Tecnologías

React (JavaScript):Creación del frontend modular e interactivo

Axios:Cliente HTTP para peticiones a la API RESTCONF

RESTCONF:Protocolo para exponer configuraciones y datos de red en dispositivos

YANG Models: Modelado de datos usados por RESTCONF (como ietf-interfaces)

CSS:Estilo visual básico sin frameworks externos

Servidor RESTCONF (Ej. Cisco CSR1000v):Dispositivo o emulador de red compatible con RESTCONF para pruebas

Arquitectura General del Sistema

La solución implementada se basa en una arquitectura de tipo **cliente-servidor**, donde el cliente web se comunica con el servidor RESTCONF implementado en un dispositivo de red. RESTCONF es un protocolo estándar definido por la IETF, diseñado para operar sobre modelos de datos YANG y permitir la manipulación de configuraciones y el monitoreo del estado operativo de equipos de red mediante HTTP y representaciones JSON o XML.

El frontend, desarrollado con React, actúa como consumidor de los recursos RESTCONF expuestos por el servidor del dispositivo, utilizando peticiones HTTP bien definidas (GET, PUT, etc.) para interactuar con las interfaces de red. Esta separación de responsabilidades garantiza una alta cohesión funcional y una clara división entre la capa de presentación y la de infraestructura de red.

Comunicación con Servicios RESTCONF

RESTCONF proporciona una interfaz RESTful para acceder a los modelos de datos definidos en YANG. Esta interfaz permite a las aplicaciones cliente realizar operaciones sobre la configuración y el estado de dispositivos de red mediante rutas bien definidas. En este proyecto, los modelos utilizados se centran principalmente en ietf-interfaces, los cuales estructuran los datos relacionados con interfaces de red como nombres, direcciones IP y máscaras de subred.

Modelo de Datos YANG

YANG (Yet Another Next Generation) es un lenguaje de modelado de datos utilizado para describir la estructura jerárquica de datos que puede gestionar un dispositivo de red. En RESTCONF, los modelos YANG se exponen como recursos RESTful, lo que permite que aplicaciones como la desarrollada en este proyecto puedan interactuar directamente con configuraciones específicas.

El modelo ietf-interfaces es utilizado en esta solución para representar:

- El nombre lógico de la interfaz (name)
- La descripción y tipo de interfaz (description, type)
- Su estado (enabled)
- Las configuraciones IPv4 e IPv6 asociadas

Resultados

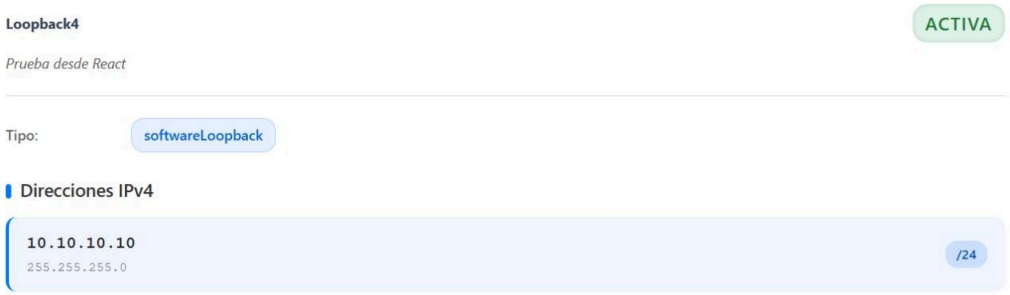
El sistema desarrollado permite una gestión básica de interfaces loopback mediante una interfaz gráfica accesible vía navegador. Se ha logrado:

- Acceso y visualización de interfaces existentes a través de RESTCONF
- Creación y modificación de interfaces utilizando el modelo ietf-interfaces
- Comunicación exitosa entre el cliente React y el servidor RESTCONF simulado

La herramienta ha sido validada sobre entornos de prueba compatibles, como CSR1000v o emuladores de red tipo EVE-NG, demostrando su aplicabilidad en entornos de redes reales.

En conclucion la adopción de RESTCONF como mecanismo de interacción con dispositivos de red representa un avance significativo en la administración moderna de infraestructuras. Permite una integración más fluida con sistemas de orquestación, facilita la automatización de tareas repetitivas y promueve una arquitectura más abierta, escalable y segura.

Este proyecto ejemplifica cómo una interfaz ligera, desarrollada con herramientas modernas como React, puede aprovechar los estándares del IETF para ofrecer una experiencia de usuario intuitiva en la gestión de configuraciones de red, eliminando barreras técnicas tradicionales.




```
Router#show ip int brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       unassigned      YES NVRAM  up                    up
GigabitEthernet2       unassigned      YES NVRAM  up                    up
GigabitEthernet3       192.168.79.140  YES DHCP   up                    up
GigabitEthernet4       unassigned      YES NVRAM  up                    up
Loopback4              10.10.10.10     YES other  up                    up
Loopback9              192.168.20.6    YES other  up                    up
Router#
```

Implemente programabilidad a través de: Scripts en Python que permitan agilitar las tareas de administración

Requisitos para usar el script en Netmiko

1. Topología en EVE-NG:

Configurar una topología con un router principal (R1) y otros routers conectados:

- R1: 192.168.239.135 (Gestionado por DHCP desde Network Management).
- R2: 192.168.2.2.
- R3: 192.168.3.2.
- R4: 192.168.4.2.

2. Configuración obligatoria en el router (R1)

El router debe cumplir los siguientes requisitos:

a. Configuración de SSH

SSH debe estar habilitado para la conexión remota:

- conf t
- hostname R1
- ip domain-name example.com
- crypto key generate rsa
- ip ssh version 2
- username cisco privilege 15 secret cisco
- line vty 0 4
- password cisco
- login local
- transport input ssh
- exit

b. Conexión DHCP al Network Management

La interfaz que conecta al Network Management debe obtener su dirección por DHCP:

- interface GigabitEthernet0/0
- ip address dhcp
- no shutdown

c. Habilitar CDP

El protocolo CDP debe estar activado para que funcione el comando relacionado:

- conf t
- cdp run

d. Dirección IP para administración

Aunque la conexión al Network Management sea DHCP, asegúrate de que R1 tenga acceso a través de su interfaz asignada. Si deseas configurar otra IP estática para pruebas, puedes usar:

- interface GigabitEthernet0/1
- ip address 192.168.239.135 255.255.255.0
- no shutdown

Uso de Netmiko con eve - ng

Netmiko es una biblioteca de Python diseñada para interactuar con dispositivos de red como routers, switches y firewalls mediante SSH. Está construida sobre Paramiko (otra biblioteca para SSH) y simplifica el envío de comandos y la automatización de tareas en equipos de red. **EVE-NG** (Emulated Virtual Environment – Next Generation) es una plataforma para emular laboratorios de redes. Permite crear topologías con routers, switches y otros dispositivos virtuales. Netmiko puede integrarse con EVE-NG para automatizar tareas y gestionar dispositivos emulados.

Uso de netmiko con una topología de eve - ng

Configuración del Dispositivo

```
device = {
  'device_type': 'cisco_ios',
  'ip': '192.168.239.135',
  'username': 'cisco',
  'password': 'cisco',
  'port': 22
```

}

Descripción:

- Define los datos necesarios para conectarse al router:
    - **IP:** Dirección IP de R1.
    - **Credenciales:** Usuario y contraseña configurados en el router.
    - **Puerto:** El puerto 22 para SSH.

Conexión al Router

```
def connect_to_device(device):
  try:
    connection = ConnectHandler(**device)
    print(f"Conexión SSH establecida con éxito a {device['ip']}")
    return connection
  except Exception as e:
    print(f"Error al conectar a {device['ip']}: {e}")
    return None
```

Descripción:

- Usa la biblioteca Netmiko para establecer una conexión SSH.
- Si se conecta con éxito, devuelve la conexión; si no, muestra un error.

Obtener Información del Router

4.1. Mostrar Interfaces Activas

```
def show_ip_interfaces(connection):
  try:
    output = connection.send_command("show ip interface brief")
    return output
  except Exception as e:
    print(f"Error al ejecutar show ip interface brief: {e}")
    return f"Error: {str(e)}"
```

- Envía el comando show ip interface brief para mostrar el estado de las interfaces (nombre, dirección IP, estado).
- La información ayuda a saber qué interfaces están activas y configuradas.

4.2. Mostrar Vecinos CDP

```
def show_cdp_neighbors(connection):
  try:
    output = connection.send_command("show cdp neighbors")
    return output
  except Exception as e:
    print(f"Error al ejecutar show cdp neighbors: {e}")
    return f"Error: {str(e)}"
```

- Envía el comando show cdp neighbors para listar los dispositivos conectados directamente al router R1.
- **Nota:** El protocolo CDP debe estar habilitado para que esto funcione.

Pruebas de Conectividad

```
def ping_routers(connection, routers):
    results = []
    try:
        for router_ip in routers:
            command = f"ping {router_ip}"
            output = connection.send_command(command, delay_factor=2)
            results.append({"ip": router_ip, "result": output})
            time.sleep(1)
        return results
    except Exception as e:
        print(f"Error al ejecutar ping: {e}")
        return [{"ip": "Error", "result": str(e)}]
```

- Permite hacer ping a otros routers (R2, R3 y R4) desde R1.
- Usa el comando ping y guarda los resultados en una lista.

Monitoreo de Recursos

6.1. Uso de CPU

```
def get_cpu_usage(connection):
    try:
        output = connection.send_command("show processes cpu sorted")
        return output
    except Exception as e:
        print(f"Error al obtener el uso de CPU: {e}")
        return f"Error: {str(e)}"
```

- Usa show processes cpu sorted para obtener los procesos que consumen más CPU.

6.2. Uso de Memoria

```
def get_memory_usage(connection):
    try:
        output = connection.send_command("show memory statistics")
        return output
    except Exception as e:
        print(f"Error al obtener el uso de memoria: {e}")
        return f"Error: {str(e)}"
```

- Usa show memory statistics para mostrar estadísticas de memoria: total, usada y disponible

Información del Dispositivo

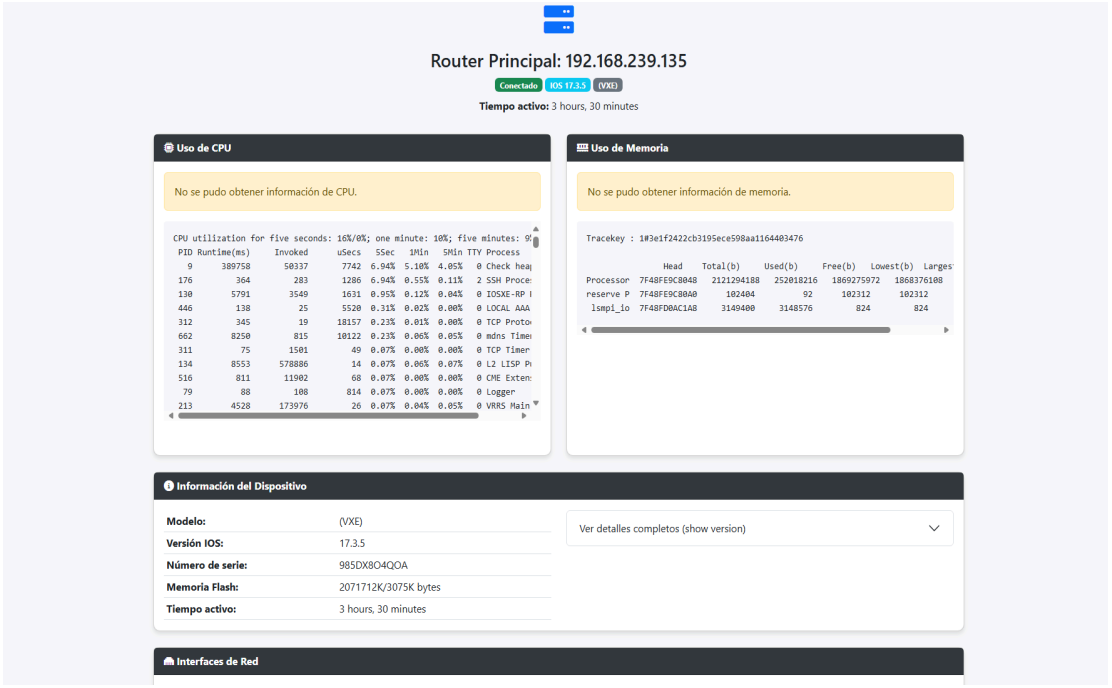
```
def get_device_version(connection):
    try:
        output = connection.send_command("show version")
```

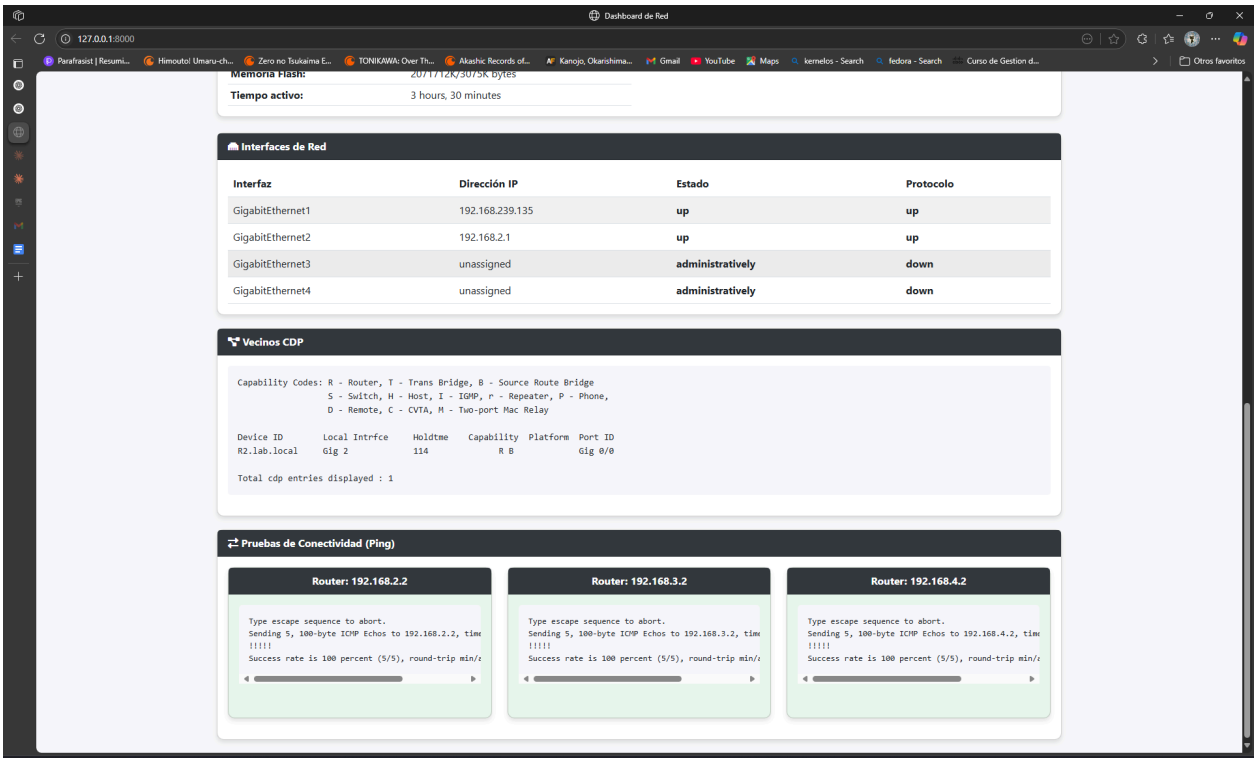
```
    return output
except Exception as e:
    print(f"Error al obtener la versión del dispositivo: {e}")
    return f"Error: {str(e)}"
```

- Usa show version para obtener detalles como:
    - Modelo del dispositivo.
    - Número de serie.
    - Versión del software.
    - Tiempo de actividad (uptime).

El programa es una herramienta práctica para gestionar y supervisar un router Cisco (R1) configurado en un entorno EVE-NG mediante Netmiko. Permite automatizar tareas comunes, como verificar interfaces, vecinos CDP, realizar pruebas de conectividad y monitorear el estado del dispositivo (CPU, memoria, versión del software). Esto reduce significativamente el esfuerzo manual al interactuar con la CLI y mejora la eficiencia al administrar redes. En conjunto, el código demuestra cómo combinar herramientas de red y automatización para simplificar la gestión de dispositivos en un laboratorio o entorno real.

Resultados:




Dashboard de Red
127.0.0.1:8000
Memoria Flash: 207177KB/3075K bytes
Tiempo activo: 3 hours, 30 minutes
Interfaces de Red
Interfaz | Dirección IP | Estado | Protocolo
GigabitEthernet1 | 192.168.239.135 | up | up
GigabitEthernet2 | 192.168.2.1 | up | up
GigabitEthernet3 | unassigned | administratively | down
GigabitEthernet4 | unassigned | administratively | down
Vecinos CDP
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
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
R2.lab.local Gig 2 114 R B Gig 0/0
Total cdp entries displayed : 1
Pruebas de Conectividad (Ping)
Router: 192.168.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.2, tim
!!!!!
Success rate is 100 percent (5/5), round-trip min/
Router: 192.168.3.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.3.2, tim
!!!!!
Success rate is 100 percent (5/5), round-trip min/
Router: 192.168.4.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.4.2, tim
!!!!!
Success rate is 100 percent (5/5), round-trip min/