****

Campus Santa Fe

Implementación de redes de área amplia y servicios distribuidos

Desarrollo de aplicaciones avanzadas de ciencias computacionales

TC3003B.501 y TC3002B.502

Documentación de infraestructura y seguridad de redes

**Assessment**

Benjamín Alejandro Cruz Cervantes - A01747811

Alberto Iván Tamez González - A01026999

Bernardo Alejandro Limón Montes de Oca - A01736575

Fecha de entrega:

Lunes 09 de Junio de 2025

**Infraestructura, Seguridad y Redes**

# 1. Introducción

Este documento describe la infraestructura física, lógica y virtual que sostiene el sistema de recomendación de cultivos. Se explican las configuraciones necesarias para habilitar la comunicación entre los componentes, el control de acceso seguro a los servicios, y la topología de red implementada. Se hace especial énfasis en el uso de redes privadas, balanceo de carga, reglas de *NAT/PAT* y grupos de seguridad, todo esto desplegado en una nube privada utilizando *OpenStack*.

La arquitectura fue implementada para simular un entorno productivo real con políticas de mínima exposición, autenticación fuerte y segmentación lógica por *VLANs*, siguiendo buenas prácticas de seguridad en infraestructura distribuida.

# 2. Infraestructura física y virtual

## 2.1 Topología general

* **Router Principal:** Un dispositivo de equipamiento *Cisco* que funciona como el gateway principal y firewall perimetral de la red.
* **Switch Virtual:** Un switch gestionado por *OpenStack Neutron* que proporciona la conectividad entre las distintas máquinas virtuales del proyecto.
* **Plataforma de Nube Privada:** El despliegue de todos los componentes se realiza sobre la infraestructura de *OpenStack* proporcionada por la institución.

## 2.2 Componentes desplegados en OpenStack

A continuación se describen las instancias activas desplegadas en OpenStack, incluyendo sus nombres, sistema operativo base, dirección IP privada asignada y el servicio que alojan:

| **Nombre de instancia** | **Sistema Operativo** | **Dirección IP** | **Servicio** |
| --- | --- | --- | --- |
| **bab\_back** | *Debian* | *172.28.69.182* | *Backend (FastAPI, puerto 9000)* |
| **bab\_front** | *Debian* | *172.28.69.227* | *Frontend (React, puerto 3000)* |
| **bab\_front\_02** | *Debian* | *172.28.69.127* | *Segunda instancia de frontend (3001)* |
| **bab\_load\_balancer** | *Debian* | *172.28.69.192* | *Load Balancer (NGINX, puerto 443)* |

Estas instancias se comunican internamente a través de la red privada 10.0.69.0/24, y sólo el balanceador de carga posee una *IP flotante pública* para exponer los servicios hacia el exterior.

## 2.3 Asignación de IPs y comunicación

* Se utilizaron direcciones IP privadas dentro de la *VLAN 69* (10.0.69.0/24)
* Sólo el load balancer tiene una IP flotante pública (10.49.12.48)
* Se aplicó *NAT/PAT* para exponer el backend al exterior de forma segura: https://10.49.12.46:9943/

# 3. Arquitectura de red y configuración

## 3.1 VLANs y capa lógica

A diferencia de una configuración tradicional con múltiples *VLANs*, la arquitectura de este proyecto opera sobre una **única red virtual plana**. Esta red es gestionada por el switch virtual de *OpenStack*, lo que permite que todas las instancias se comuniquen directamente entre sí sin necesidad de configurar enrutamiento entre *VLANs*.

El segmento de red asignado al proyecto es **172.28.69.0/24**. Todas las instancias, incluyendo el balanceador de carga (172.28.69.192), los servidores frontend (172.28.69.227 y 172.28.69.127) y el backend (172.28.69.182), residen en esta misma subred, simplificando la conectividad interna.

## 3.2 Traducción de direcciones (PAT)

Para exponer el backend en HTTPS sobre la red institucional:

interface Vlan69  
 ip nat inside  
  
interface GigabitEthernet0/0  
 ip nat outside  
  
ip nat inside source static tcp 10.0.69.30 443 interface GigabitEthernet0/0 9443



Esto redirige el tráfico del puerto *9443* en la *IP pública* al *443* del backend.

## 3.3 Red en OpenStack

* Red privada: vlan69-net con subred 10.0.69.0/24
* Gateway: 10.0.69.1 (*SVI* del switch)
* No se utilizó un router virtual en *OpenStack*, todo el tráfico es manejado por el switch físico.

# 4. Seguridad del sistema

## 4.1 Load Balancer como único punto de entrada

* *NGINX* instalado en bab-lb-01
* Certificados *SSL* configurados manualmente
* Redirección segura al backend:

# Define el grupo de servidores web (front-end) para balanceo de carga.

# Puertos actualizados según tu indicación.

upstream frontend\_servers {

server 172.28.69.227:3000; # Instancia bab\_front

server 172.28.69.127:3001; # Instancia bab\_front\_02

}

server {

listen 443 ssl;

server\_name 172.28.69.192;

access\_log /var/log/nginx/backend\_access.log;

ssl\_certificate /etc/nginx/ssl/selfsigned.crt;

ssl\_certificate\_key /etc/nginx/ssl/selfsigned.key;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers HIGH:!aNULL:!MD5;

# --- Rutas de la aplicación (Locations) ---

# 1. Peticiones al Frontend (todo lo que no sea /api/)

location / {

proxy\_pass http://frontend\_servers;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

# 2. Peticiones a la API (todo bajo /api/)

location /api/ {

proxy\_pass http://172.28.69.182:9000;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

Reinicio del servicio:

sudo systemctl restart nginx



## 4.2 Grupos de seguridad en OpenStack

### Grupo: Load Balancer (SG-lb)

- TCP 80, 443 desde 0.0.0.0/0  
- TCP 22 desde IPs institucionales

### Grupo: Backend y AI (SG-app)

- TCP 9000, 1337 desde IP del load balancer (10.0.69.50)

### Grupo: Base de datos (SG-db)

- TCP 5432 localmente desde IP del backend (10.0.69.30)

## 4.3 Seguridad entre servicios

Validación de API Key entre backend y microservicio:

## @app.middleware("http") async def check\_api\_key(request: Request, call\_next): if request.headers.get("x-api-key") != os.getenv("INTERNAL\_API\_KEY"): raise HTTPException(status\_code=403) return await call\_next(request)

## 4.4 Gestión de secretos y acceso SSH

* Todas las instancias fueron configuradas con llaves públicas *SSH*.
* Contraseñas deshabilitadas.
* Variables de entorno almacenadas en archivos .env:

DATABASE\_URL= postgresql://postgres:bbva321@172.28.69.160:5432/fastapi\_auth  
INTERNAL\_API\_KEY=clave-secreta  
WORKOS\_SECRET=otro-secreto

# 

# 5. Validación de conectividad

Verificación desde la red institucional:

curl -k https://10.49.12.48:9443/docs  
ping 10.0.69.1  
nc -zv 10.0.69.30 9000

Comprobación del estado de NGINX:

sudo nginx -t  
sudo systemctl status nginx

# 6. Conclusión

La infraestructura diseñada replica un entorno profesional controlado, implementando políticas de segmentación, mínimo privilegio y visibilidad reducida. Esto se logró a través de VLANs configuradas específicamente para aislar los servicios, el uso de grupos de seguridad en OpenStack con permisos estrictos, y una arquitectura de red centrada en el uso de un único punto de entrada a través del Load Balancer.

El sistema cuenta con protección de servicios mediante cabeceras (como la API Key en el microservicio de IA), comunicación cifrada con NGINX y certificados SSL, así como autenticación gestionada externamente para evitar el almacenamiento de contraseñas.

Además, la arquitectura de red emplea NAT y PAT configurados en un switch físico de Capa 3, integrando el tráfico interno con la red institucional sin comprometer los servicios sensibles.

El despliegue sobre OpenStack permitió automatizar la provisión de instancias, la asignación de redes y el uso de IPs flotantes públicas, facilitando un entorno replicable y escalable con una inversión mínima en infraestructura física adicional.