

---

# Señas Chapinas: Traductor de LENSEGUA Infraestructura de red

---

Jose Miguel Gonzalez y Gonzalez





UNIVERSIDAD DEL VALLE DE GUATEMALA  
Facultad de Ingeniería



**Señas Chapinas: Traductor de LENSEGUÁ  
Infraestructura de red**

Trabajo de graduación en modalidad megaproyecto tecnológico  
presentado por  
Jose Miguel Gonzalez y Gonzalez  
Para optar al grado académico de Licenciado en Ingeniería en Ciencias  
de la Computación y Tecnologías de la Información

Guatemala, noviembre del 2024



Vo.Bo.:

(f) \_\_\_\_\_  
Miguel Novella Linares

Tribunal Examinador:

(f) \_\_\_\_\_  
Miguel Novella Linares

(f) \_\_\_\_\_  
Erick Francisco Marroquin Rodriguez

(f) \_\_\_\_\_  
Jorge Andres Yass Coy

Fecha de aprobación: Guatemala, 28 de noviembre de 2024.



---

## Prefacio

---

El proyecto *Señas Chapinas: Traductor de Lensegua* nació de la necesidad de reducir las barreras de comunicación que enfrenta la comunidad sorda en Guatemala, quienes utilizan la lengua de señas guatemalteca, Lensegua, como principal medio de expresión. Esta iniciativa tuvo el propósito de ofrecer una solución tecnológica que permita traducir Lensegua a texto y voz en tiempo real, ayudando a la comunidad sorda a participar en la sociedad de una manera más equitativa y autónoma.

Mi contribución al proyecto se centra en el desarrollo de la infraestructura de red y la arquitectura de backend, componentes esenciales para asegurar la estabilidad, seguridad, y eficiencia de la plataforma. Esta arquitectura se concibe como el núcleo técnico del sistema, integrando y facilitando la comunicación entre todos los módulos de la aplicación, desde el procesamiento de lenguaje natural hasta la visión por computadora. El backend no solo habilita el funcionamiento del traductor de señas, sino que también provee la base para escalar el proyecto en el futuro, adaptándose a las necesidades crecientes de la comunidad y del equipo de desarrollo.

Quiero expresar mi sincero agradecimiento al Ing. Miguel Novella, quien me brindó apoyo y orientación en cada etapa de esta compleja infraestructura. También agradezco a la Universidad del Valle de Guatemala, así como a En-Señas Guatemala y ASEDES, por los recursos y respaldo necesario para alcanzar los objetivos planteados. Su apoyo hizo posible que los sistemas de comunicación y procesamiento de datos del proyecto tomaran forma en condiciones óptimas y en un entorno seguro.

A mi familia y amigos, gracias por su inagotable apoyo y por ser mi motivación constante a lo largo de este camino. Este proyecto no habría sido posible sin su confianza y aliento inquebrantable.



---

## Agradecimientos

---

A mis seres queridos, mis padres por apoyarme en este camino y por creer en mi, a mi pareja por siempre darme alientos cuando mas lo necesitaba ayudarme y apoyarme en los momentos donde mas lo requeria.

Ingeniero Miguel Novella por estar pendiente y apoyarme como asesor, como docente y como profesional.

Ingeniero Erick por apoyarme en el levantamiento de forticlient en el sistema al inicio del proyecto y su apoyo en el intento de acceso al port forward.

Agradezco también a la Universidad del Valle de Guatemala y a las organizaciones En-Señas Guatemala y ASEDES por proporcionar el respaldo y los recursos necesarios para llevar a cabo esta investigación.

A T1 por mostrarme que hay que mantenerse fuerte y constante, que las batallas mas duras se pueden lograr.



---

## Índice

---

<b>Prefacio</b>	<b>v</b>
<b>Agradecimientos</b>	<b>vii</b>
<b>Lista de Figuras</b>	<b>xiii</b>
<b>Lista de Cuadros</b>	<b>xv</b>
<b>Resumen</b>	<b>xvii</b>
<b>1. Introducción</b>	<b>1</b>
<b>2. Objetivos</b>	<b>3</b>
2.1. Objetivo General . . . . .	3
2.2. Objetivos Específicos . . . . .	3
<b>3. Justificación</b>	<b>5</b>
<b>4. Marco Teórico</b>	<b>8</b>
4.1. Lensegua . . . . .	8
4.1.1. ¿Qué es Lensegua? . . . . .	8
4.1.2. Historia de Lensegua . . . . .	8
4.2. Sistema Operativo: Linux Server . . . . .	9
4.2.1. ¿Qué es Linux? . . . . .	9
4.2.2. Historia de Linux . . . . .	9
4.2.3. Linux Server . . . . .	9
4.3. Virtualización . . . . .	9
4.4. Multipass . . . . .	10
4.5. VPN . . . . .	10
4.5.1. ¿Qué es una VPN? . . . . .	10
4.5.2. ¿Para qué sirven las VPNs? . . . . .	10
4.6. Arquitectura de base de datos . . . . .	10
4.6.1. Bases de datos relacionales . . . . .	10
4.6.2. PostgreSQL . . . . .	11
4.6.3. Arquitectura de base de datos . . . . .	11
4.7. APIs . . . . .	11
4.7.1. ¿Qué es una API? . . . . .	11
4.7.2. ¿Cómo usar una API? . . . . .	11

4.7.3. Implementar APIs a bases de datos . . . . .	11
4.7.4. Implementar APIs para recepción y devolución de datos . . . . .	12
4.7.5. ¿Qué es Crontab? . . . . .	12
4.8. Bases de diseño y filosofía de DARPA . . . . .	12
4.8.1. Costo-efectividad y control . . . . .	12
4.8.2. Flexibilidad y adaptabilidad . . . . .	12
4.8.3. Seguridad y privacidad . . . . .	12
4.9. Pruebas de eficiencia y de extremo a extremo en el servidor mediante APIs . . . . .	13
4.9.1. ¿Qué es una prueba de eficiencia? . . . . .	13
4.9.2. ¿Qué es una prueba de extremo a extremo (E2E)? . . . . .	13
4.9.3. Impacto de la eficiencia y de las pruebas de extremo a extremo en el servidor . . . . .	13
4.10. Herramientas para pruebas de eficiencia y de extremo a extremo . . . . .	13
4.10.1. Monitoreo del sistema . . . . .	14
4.10.2. Pruebas de carga y end to end . . . . .	14
4.11. Trabajo de campo . . . . .	15
<b>5. Antecedentes</b>	<b>17</b>
<b>6. Alcance</b>	<b>19</b>
6.1. Consideraciones adicionales . . . . .	20
<b>7. Metodología</b>	<b>22</b>
7.1. Levantamiento de un sistema operativo y configuración de VPN . . . . .	22
7.2. Desarrollo de la arquitectura de base de datos . . . . .	23
7.3. Implementación de API's . . . . .	26
7.3.1. Flujo de trabajo para sistema de APIs . . . . .	28
7.3.2. Diagrama de Flujo de la Arquitectura de Sistemas . . . . .	28
7.3.3. Implementación de Flask con Gunicorn y NGINX . . . . .	30
7.3.4. Implementación de SQLAlchemy para manejo de base de datos y entidades . . . . .	33
7.3.5. Implementación de módulo API para bases de datos . . . . .	33
7.3.6. API para integración de modelos de procesamiento . . . . .	34
7.3.7. Implementación de Crontab . . . . .	35
7.4. Virtualización del servidor para múltiples modelos . . . . .	36
7.5. Implementación de pruebas de eficiencia . . . . .	38
7.6. Pruebas de Carga . . . . .	38
7.6.1. Objetivo . . . . .	38
7.6.2. Procedimiento . . . . .	39
7.7. Pruebas de Extremo a Extremo (E2E) . . . . .	42
7.7.1. Objetivo . . . . .	42
7.7.2. Procedimiento . . . . .	42
7.8. Implementación de pruebas CVE para seguridad . . . . .	42
7.9. Pruebas de Seguridad con Lynis . . . . .	43
7.9.1. Objetivo . . . . .	43
7.9.2. Procedimiento . . . . .	44
7.10. Mejoras de Seguridad en /etc/sysctl.conf . . . . .	44
7.11. Monitoreo Continuo de Seguridad con ClamAV . . . . .	44
7.11.1. Instalación y Configuración . . . . .	44
7.11.2. Monitoreo y Ajustes Adicionales . . . . .	44
<b>8. Resultados</b>	<b>46</b>
8.1. Redireccionamiento de puertos para acceso SSH Externo . . . . .	46
8.2. Resultados de la Prueba de Carga . . . . .	47
8.3. Resultados de la Prueba E2E . . . . .	50
8.4. Resultados de la Prueba de Seguridad con Lynis . . . . .	51

<b>9. Análisis de Resultados</b>	<b>55</b>
<b>10. Conclusiones</b>	<b>58</b>
<b>11. Recomendaciones</b>	<b>60</b>
Referencias . . . . .	63
<b>Bibliografía</b>	<b>64</b>
<b>Anexos</b>	<b>65</b>
Carta de Solicitud de Acceso a la VPN . . . . .	65



---

## Lista de Figuras

---

3.1. Aplicación: Señas Chapinas. . . . .	6
7.1. LevantadoOpenVPN. . . . .	23
7.2. ConexionSSH. . . . .	23
7.3. Diagrama entidad relación. . . . .	26
7.4. Diagrama de flujo. . . . .	30
7.5. Flujo de trabajo completo del sistema de APIs. . . . .	31
7.6. Estado de las máquinas virtuales dentro del sistema . . . . .	38
7.7. Resultados de la prueba de carga con 50 usuarios concurrentes. . . . .	40
7.8. Resultados de la prueba de carga con 100 usuarios concurrentes. . . . .	40
7.9. Resultados de la prueba de carga con 200 usuarios concurrentes. . . . .	40
7.10. Resultados de la prueba de carga con 300 usuarios concurrentes. . . . .	40
7.11. Resultados de la prueba de carga con 400 usuarios concurrentes. . . . .	40
7.12. Resultados de la prueba de carga con 500 usuarios concurrentes. . . . .	41
7.13. Resultados de la prueba de carga con 600 usuarios concurrentes. . . . .	41
7.14. Resultados de la prueba de carga con 700 usuarios concurrentes. . . . .	41
7.15. Resultados de la prueba de carga con 800 usuarios concurrentes. . . . .	41
7.16. Resultados de la prueba de carga con 900 usuarios concurrentes. . . . .	41
7.17. Resultados de la prueba de carga con 1100 usuarios concurrentes. . . . .	42
8.1. Métricas de Conexiones y Solicitudes: Requests/s y Conexiones Actuales . . . . .	47
8.2. Códigos de Respuesta HTTP . . . . .	48
8.3. Tiempo de Respuesta y Tiempo de Respuesta Upstream . . . . .	48
8.4. Métricas de CPU, Memoria y Carga del Sistema . . . . .	49
8.5. Tráfico de Red (Network I/O) . . . . .	50
8.6. Resultados de la prueba E2E, mostrando un 100 % de éxito en todas las operaciones realizadas. . . . .	51
8.7. Resultado de la primera prueba de seguridad con Lynis mostrando un índice de robustez de 58. . . . .	51
8.8. Resultado de la segunda prueba de seguridad con Lynis mostrando un índice de robustez de 60. . . . .	52
8.9. Resultado de la tercera prueba de seguridad con Lynis mostrando un índice de robustez de 62. . . . .	52
8.10. Resumen del estado del servidor. . . . .	53



---

## Lista de Cuadros

---

7.1.	Tabla: user . . . . .	24
7.2.	Tabla: video . . . . .	25
7.3.	Tabla: traducción . . . . .	25
7.4.	Tabla: dictionary . . . . .	25
7.5.	Resumen de Rutas de API: <code>user_routes</code> . . . . .	26
7.6.	Resumen de Rutas de API: <code>video_routes</code> . . . . .	27
7.7.	Resumen de Rutas de API: <code>traduction_routes</code> . . . . .	27
7.8.	Resumen de Rutas de API: <code>dictionary_routes</code> . . . . .	27
7.9.	Resumen de Rutas de API: <code>profile_routes</code> . . . . .	27
7.10.	Resumen de Rutas de API: <code>mail_routes</code> . . . . .	28
7.11.	Resumen del sistema . . . . .	37
7.12.	Distribución de recursos entre las máquinas virtuales y el host . . . . .	37



---

## Resumen

---

En este proyecto se desarrolló e implementó un servidor seguro y eficiente, capaz de administrar datos y soportar modelos avanzados de inteligencia artificial y visión por computadora. El principal objetivo fue crear una instancia de desarrollo que permitiera realizar despliegues, pruebas y ejecuciones de modelos de IA a través de un ambiente accesible para usuarios externos mediante APIs.

Para cumplir con estos objetivos, se configuraron máquinas virtuales, abriendo puertos para acceso remoto y desarrollando scripts de despliegue que facilitan la instalación y actualización de los modelos de IA. La infraestructura del servidor fue optimizada para manejar eficientemente grandes volúmenes de datos, con una arquitectura basada en Nginx y Gunicorn, respaldada por SQLAlchemy para la gestión de base de datos.

Las pruebas de carga y seguridad realizadas confirmaron la capacidad del sistema para operar bajo condiciones de alta demanda. Además, la evaluación de seguridad mediante Lynis mostró que el servidor cumple con estándares elevados, superando el objetivo de seguridad planteado. Este proyecto concluye que el servidor es adecuado para un entorno de producción, ofreciendo una plataforma estable y segura para el despliegue y gestión de aplicaciones de IA.



# CAPÍTULO 1

---

## Introducción

---

El objetivo central de este proyecto fue el desarrollo de una aplicación móvil que utilizó tecnología avanzada para la traducción en tiempo real de lengua de señas a texto y voz, facilitando así la comunicación para personas con discapacidad auditiva. Esta aplicación se valió de visión por computadora para detectar gestos y señas de manera efectiva y de tecnologías de inteligencia artificial para convertir estas señas a lenguaje natural, asegurando una interacción fluida y comprensible.

Adicionalmente, se diseñó una infraestructura de red dedicada que permitió no solo la comunicación efectiva entre la aplicación y el servidor central, sino también la gestión eficiente del flujo de datos. Esta infraestructura fue clave para recibir, procesar y responder a los videos enviados por los usuarios, implementando procesos que iban desde la interpretación de imagen hasta el procesamiento por modelos de inteligencia artificial, culminando con la entrega de la traducción al dispositivo del usuario.

Con el objetivo de hacer un proyecto accesible y que tuviera la oportunidad de crecer en el futuro, se buscó que la arquitectura de red y el servidor fueran seguros, eficientes y que tuvieran la capacidad de escalar. Esto se logró con el uso de APIs robustas y de máquinas virtuales que sirvieron para administrar el uso de recursos de la mejor manera posible.



# CAPÍTULO 2

---

## Objetivos

---

### 2.1. Objetivo General

- Desarrollar y desplegar un servidor que administre datos de forma segura y eficiente, optimizando el uso de recursos computacionales para soportar modelos de inteligencia artificial avanzados.
- Generar una instancia de desarrollo que posea las herramientas y facilidades para poder levantar, instalar, probar y ejecutar modelos de inteligencia artificial y visión por computadora.

### 2.2. Objetivos Específicos

- Crear un ambiente virtual que pueda ser accedido por usuarios externos, que les permita realizar deploys, pruebas y ejecuciones de diferentes modelos de inteligencias artificiales y de visión por computadora a través del uso de API's.
- Optimizar el proceso de recepción, almacenamiento y procesamiento de videos para múltiples usuarios, priorizando la eficiencia y reduciendo costos computacionales, medido a través de mejoras en tiempo de procesamiento, uso de recursos y costos operativos.
- Fortalecer la seguridad para reducir el riesgo de robos de datos y ataques, buscando tener un valor de 4.0 en la escala de CVE (Common Vulnerabilities and Exposures) en el servidor.



## CAPÍTULO 3

---

### Justificación

---

En Guatemala, según datos del Conadi (Consejo Nacional para la Atención de las Personas con Discapacidad) recogidos hasta el mes de marzo del año 2021, existían entre 240,000 y 250,000 personas que utilizaban la lengua de señas como su principal herramienta de comunicación (Consejo Nacional para la Atención de las Personas con Discapacidad, 2021). Sin embargo, el uso de esta lengua generaba barreras comunicativas entre una persona oyente y una persona sorda. Estas barreras limitaban la comunicación diaria de estas personas, restringiendo sus oportunidades e independencia.

Esta aplicación tuvo un enfoque especial en las frases de uso diario, preguntas, consultas y dudas que podían surgir en un día cotidiano, ya que estas personas enfrentaban muchos problemas para comunicarse en la realización de tareas cotidianas. Aunque estas tareas podían considerarse triviales para otros, para este sector representaban un reto significativo. Además, se buscó fomentar una mayor autonomía para estas personas, lo que les permitió una inserción más rápida, dinámica y amplia dentro del mundo laboral, facilitando así la obtención de una variedad más amplia de empleos.

También se destacaron las dificultades y barreras de lenguaje que podían ocasionar problemas más serios en situaciones delicadas y de alto estrés. Por ejemplo, en casos en los que las personas que usaban la lengua de señas enfrentaban grandes dificultades para escribir o comunicarse de cualquier otra forma que no fuera mediante esta lengua. Esto podía ocurrir en emergencias médicas, donde la obtención de información básica del paciente era imperativa, o en situaciones como un robo o la necesidad de pedir indicaciones o información bajo estrés, como al interactuar con un agente de policía. Estas son solo algunas de las muchas situaciones en las que las personas que necesitaban la lengua de señas podían encontrarse en una seria desventaja o en riesgo, lo que hacía que una comunicación óptima y la posibilidad de hacer y responder preguntas fueran de gran importancia.

Un aspecto fundamental de este proyecto fue la integración de los diferentes componentes tecnológicos para que trabajaran como una unidad cohesiva en lugar de entidades separadas. Esta integración tuvo una relevancia crítica, ya que permitió desarrollar un sistema que conecta las capacidades de los modelos de inteligencia artificial, la funcionalidad de la aplicación móvil y la infraestructura del servidor. La aplicación tenía como objetivo principal permitir a los usuarios grabar videos de personas utilizando LENSEGUÁ, enviar textos en español LENSEGUÁ, y proporcionar herramientas accesibles directamente en sus manos. Para lograr esto, fue indispensable contar con modelos sólidos para el procesamiento de datos, una aplicación intuitiva que sirviera como puente entre los usuarios y la tecnología, y un servidor robusto que facilitara la comunicación entre todos los componentes. Este enfoque integral no solo garantizó la efectividad del sistema, sino que también

aseguró su escalabilidad y capacidad para atender las necesidades reales de la comunidad sorda.

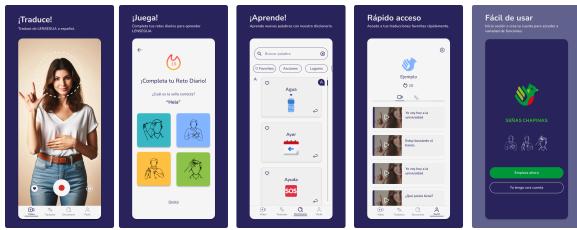


Figura 3.1: Aplicación: Señas Chapinas.



# CAPÍTULO 4

---

Marco Teórico

---

## 4.1. Lensegua

### 4.1.1. ¿Qué es Lensegua?

En Guatemala, el Lenguaje de Señas Guatemalteco (Lensegua) se oficializó el 18 de febrero de 2020 con el Decreto 3-2020. Este decreto regula la aprobación, desarrollo, uso y fomento de Lensegua, asegurando su reconocimiento y promoción en el país.

"Lengua de señas de Guatemala. Es la lengua oficial para poder comunicarse en lengua de señas en Guatemala (según Decreto 3-2020)"(Lensegua, s.f.).

### 4.1.2. Historia de Lensegua

En Guatemala, la comunidad sorda comenzó a desarrollar sus propios signos y métodos de comunicación, adaptando movimientos, posiciones y representaciones de letras a sus necesidades culturales y sociales. Este proceso fue orgánico, impulsado por la interacción diaria y la necesidad de comunicación efectiva dentro de la comunidad.

El reconocimiento oficial de Lensegua llegó el 18 de febrero de 2020, con la promulgación del Decreto 3-2020. Este decreto estableció la regulación para la aprobación, desarrollo, uso y fomento de la Lengua de Señas de Guatemala. Con esta legislación, el gobierno guatemalteco no solo reconoció la importancia de Lensegua, sino que también se comprometió a promover su enseñanza y uso en todo el país, asegurando que la comunidad sorda tenga acceso a una forma de comunicación efectiva y respetada. La implementación de Lensegua ha sido un hito significativo para la inclusión y el reconocimiento de los derechos de las personas sordas en Guatemala, marcando un paso crucial hacia una sociedad más inclusiva y equitativa (Congreso de Guatemala, 2021).

## 4.2. Sistema Operativo: Linux Server

### 4.2.1. ¿Qué es Linux?

Linux es un sistema operativo de código abierto basado en Unix que se ha establecido como una opción poderosa y versátil para una variedad de plataformas, desde servidores y supercomputadoras hasta dispositivos móviles y electrodomésticos. Fue creado inicialmente por Linus Torvalds en 1991. El núcleo de Linux, conocido como el kernel, gestiona las interacciones del software con el hardware y es vital para la regulación de los recursos del sistema. A lo largo de los años, Linux ha ganado una popularidad masiva, en parte debido a su naturaleza de código abierto, que permite a los usuarios modificar y mejorar su software según necesiten. Además, su alta configurabilidad, estabilidad y robustez en seguridad lo hacen preferido en entornos empresariales y académicos. La comunidad de Linux es una de sus mayores fortalezas, contribuyendo constantemente con una amplia gama de distribuciones adaptadas a diferentes usos y preferencias (Negus, 2020a).

### 4.2.2. Historia de Linux

Linux es un sistema operativo de código abierto creado en 1991 por Linus Torvalds, un estudiante de la Universidad de Helsinki. Inspirado en Unix, Torvalds deseaba ofrecer una alternativa gratuita y accesible, por lo que liberó Linux bajo la Licencia Pública General de GNU para permitir su uso, modificación y redistribución libre. Inicialmente adoptado por entusiastas tecnológicos y desarrolladores, Linux ha evolucionado hasta ser ampliamente utilizado en servidores, dispositivos móviles y supercomputadoras. Destacado por su estabilidad, seguridad y flexibilidad, es ahora uno de los sistemas operativos más predominantes a nivel mundial, con múltiples distribuciones como Ubuntu, Fedora y Arch que facilitan su uso en una gran variedad de entornos (GeeksforGeeks, 2024).

### 4.2.3. Linux Server

Un servidor Linux se compone fundamentalmente de Linux, una familia de sistemas operativos de software libre y código abierto que se desarrollan alrededor del kernel de Linux. Originalmente creado como una versión alternativa y libre del sistema operativo MINIX, los servidores Linux se han popularizado gracias a su estabilidad, seguridad y flexibilidad. Estas características no solo los diferencian de sus contrapartes propietarias, sino que también mantienen bajos los costos de configuración y mantenimiento. Además, ofrecen una flexibilidad aumentada en la configuración, operación y mantenimiento de un servidor. Un sistema operativo de servidor Linux proporciona la interfaz central para la gestión de usuarios e implementa diversos servicios de seguridad y administración, todos cruciales para operar en una arquitectura cliente-servidor (Red Hat, s.f.).

## 4.3. Virtualización

La virtualización, en el contexto de la computación, implica la creación de versiones virtuales de recursos que originalmente eran físicos, como sistemas operativos y hardware de red. Este proceso permite a un solo servidor físico alojar múltiples máquinas virtuales (VMs), cada una con su propio sistema operativo y aplicaciones, operando de manera independiente pero compartiendo los recursos subyacentes del hardware físico (Techopedia, 2022).

El propósito principal de la virtualización fue maximizar la eficiencia del uso de recursos al permitir que múltiples sistemas operativos y aplicaciones se ejecutaran en un solo servidor físico. Esto no solo reduce los costos de hardware, sino que también aumenta la flexibilidad y la escalabilidad.

de los sistemas de TI. Las VMs pueden migrar entre servidores con mínimo tiempo de inactividad, lo que facilita el mantenimiento y la gestión del sistema (Negus, 2020b).

## 4.4. Multipass

Multipass es una herramienta ligera de virtualización diseñada para facilitar la creación y gestión de máquinas virtuales, enfocada especialmente en entornos de desarrollo y pruebas rápidas. Al integrarse de manera nativa con Ubuntu, Multipass permite desplegar y administrar múltiples instancias de Linux de manera eficiente, actuando como una mini-nube local en tu propio equipo. Multipass ha sido desarrollado con la simplicidad en mente, proporcionando una experiencia optimizada para usuarios que necesitan acceder rápidamente a entornos virtuales consistentes sin la complejidad de configuraciones avanzadas (Multipass, s.f.).

## 4.5. VPN

### 4.5.1. ¿Qué es una VPN?

Una VPN (red privada virtual) establece una conexión segura y privada entre dispositivos a través de Internet. Este tipo de red es fundamental para la transmisión de datos de manera segura y anónima a través de infraestructuras públicas. El proceso implica enmascarar las direcciones IP de los usuarios y cifrar la información transmitida, asegurando que solo las partes autorizadas puedan acceder y leer estos datos (Amazon Web Services, s.f.b).

### 4.5.2. ¿Para qué sirven las VPNs?

Las VPNs, o redes privadas virtuales, son esenciales para permitir conexiones remotas seguras a servidores y recursos de red. OpenVPN, un destacado software de VPN de código abierto, facilitó el establecimiento de redes locales virtuales entre computadoras dispersas geográficamente, permitiendo a los usuarios acceder y operar servidores y datos como si estuvieran físicamente presentes en la misma red local. Esto es crucial para empresas y organizaciones con equipos distribuidos, asegurando que la comunicación y el acceso a recursos críticos sean seguros y eficientes (Servicio al Cliente de NordVPN, s.f.).

## 4.6. Arquitectura de base de datos

### 4.6.1. Bases de datos relacionales

Una base de datos relacional es una estructura que organiza la información en filas y columnas dentro de tablas, facilitando el almacenamiento, la búsqueda y la gestión de datos de manera eficiente. Las tablas se enlazan entre sí mediante claves primarias y claves foráneas, creando relaciones que reflejan cómo se conectan los datos en diferentes tablas. Este sistema permite a los usuarios emplear consultas SQL para cruzar información de diversas tablas, proporcionando una herramienta poderosa para analizar y resumir el rendimiento empresarial (IBM, s.f.b).

#### 4.6.2. PostgreSQL

PostgreSQL es una mejora del sistema de gestión Postgres, un prototipo avanzado de DBMS de próxima generación. Mientras conserva el modelo de datos potente y los tipos de datos ricos de Postgres, PostgreSQL ha reemplazado el lenguaje de consulta POSTQUEL por un subconjunto extendido de SQL, convirtiéndolo en una opción gratuita y de código abierto para el manejo de bases de datos. Desarrollado inicialmente en la Universidad de California en Berkeley, PostgreSQL ha evolucionado significativamente gracias a la contribución de un equipo de desarrolladores a nivel mundial (Momjian, 2000).

#### 4.6.3. Arquitectura de base de datos

La arquitectura de datos se refiere al diseño estructural y a la metodología empleada para organizar y gestionar los recursos de datos en una organización o sistema. Este marco establece las políticas, normas y procedimientos para la recopilación, almacenamiento, manejo y uso de los datos con el fin de asegurar su accesibilidad, consistencia, integridad y seguridad. Una arquitectura de datos eficaz apoya la estrategia de información de una empresa, facilitando la integración y la operación de sistemas de TI, mejorando el soporte para las decisiones basadas en datos y optimizando el rendimiento de las aplicaciones (IBM, s.f.a).

### 4.7. APIs

#### 4.7.1. ¿Qué es una API?

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre sí. Funciona como un puente que permite la interacción entre diferentes piezas de software de manera estandarizada. Por ejemplo, una aplicación de pronóstico del tiempo puede recibir datos de un servicio meteorológico utilizando una API para comunicarse con el sistema de dicho servicio (Amazon Web Services, s.f.a).

#### 4.7.2. ¿Cómo usar una API?

Usar una API a nivel básico implica interactuar con un conjunto de definiciones y protocolos que permiten la comunicación entre diferentes componentes de software. Primero, es esencial entender los métodos estándar que la API ofrece, como obtener, crear, actualizar o eliminar recursos. Cada interacción sigue un patrón predefinido, facilitando así la integración y uso de la API en proyectos (HowToGeek, 2021).

#### 4.7.3. Implementar APIs a bases de datos

Al desarrollar una aplicación que necesite interactuar con una base de datos mediante una API, puedes realizar operaciones como crear, recuperar, actualizar o eliminar registros. Estas operaciones se definen claramente con ciertos parámetros de entrada esperados y formatos de respuesta, lo que facilita la integración de sistemas complejos (HowToGeek, s.f.a).

#### 4.7.4. Implementar APIs para recepción y devolución de datos

El uso de una API para enviar y recibir información implica interacciones estructuradas mediante solicitudes y respuestas definidas. Un ejemplo común es usar un método HTTP POST para enviar datos a la API, que luego devuelve una respuesta indicando éxito o fallo (HowToGeek, s.f.b).

#### 4.7.5. ¿Qué es Crontab?

‘Crontab’ es una herramienta en sistemas Unix/Linux que permite la programación de tareas automatizadas que se ejecutan a intervalos específicos. Estas tareas, conocidas como cron jobs, son útiles para ejecutar scripts y comandos de forma recurrente sin intervención manual, lo cual es ideal para tareas de mantenimiento o procesamiento automatizado, como el reinicio de rachas en nuestra aplicación (Hostinger, 2024).

### 4.8. Bases de diseño y filosofía de DARPA

Entre los conceptos fundamentales aplicados en el diseño de la arquitectura del sistema, se adoptaron varios principios expuestos en el documento "The Design Philosophy of the DARPA Internet Protocols" (Clark, 1988). A continuación, se destacan los siguientes conceptos clave:

#### 4.8.1. Costo-efectividad y control

La arquitectura de Internet, según lo planteado por DARPA, enfatiza la necesidad de manejar los recursos de manera eficaz bajo control administrativo distribuido. Esto justifica la decisión de utilizar servidores propios en lugar de servicios en la nube para este proyecto, reduciendo dependencias externas y costos operativos. Al mantener un control directo sobre la infraestructura, se puede gestionar de manera más eficiente el uso de los recursos, optimizando la relación costo-beneficio sin sacrificar el rendimiento.

#### 4.8.2. Flexibilidad y adaptabilidad

La filosofía de diseño también subraya la importancia de adaptarse a diferentes entornos y requisitos técnicos. En este proyecto, la elección de Ubuntu como sistema operativo refleja un compromiso con un sistema flexible y robusto, ya que permite una personalización extensa. Aunque inicialmente se consideró el uso de RAID para la gestión del almacenamiento, se decidió desinstalarlo y desconfigurar debido a las limitaciones específicas del entorno de este proyecto. Esto asegura que la infraestructura sea adaptable a las necesidades actuales, manteniendo un enfoque simplificado en la gestión de recursos.

#### 4.8.3. Seguridad y privacidad

Siguiendo los principios de diseño de DARPA, la seguridad es una prioridad fundamental. La integridad y seguridad de la comunicación entre los sistemas es vital para el éxito del proyecto. En este sentido, la implementación de una VPN segura mediante OpenVPN garantiza la protección de los datos transmitidos, asegurando que el acceso remoto y la transferencia de información se realicen de manera segura y confiable, reduciendo el riesgo de vulnerabilidades y ataques.

## 4.9. Pruebas de eficiencia y de extremo a extremo en el servidor mediante APIs

### 4.9.1. ¿Qué es una prueba de eficiencia?

Las pruebas de eficiencia miden cómo el servidor responde bajo condiciones de carga habitual, evaluando aspectos como el tiempo de respuesta, la utilización de recursos (CPU, memoria y disco), y la capacidad del sistema para gestionar solicitudes simultáneas sin afectar negativamente el rendimiento. Estas pruebas permiten identificar oportunidades de optimización y ajustar la infraestructura para garantizar un rendimiento óptimo en condiciones normales de operación (Parasoft, 2024).

### 4.9.2. ¿Qué es una prueba de extremo a extremo (E2E)?

Las pruebas de extremo a extremo (E2E) verifican el funcionamiento integral de la aplicación, evaluando cómo interactúan todos sus componentes en un flujo completo desde el punto de vista del usuario. Estas pruebas son esenciales para asegurarse de que cada parte de la aplicación, incluidas las APIs, funcione como se espera y que todos los puntos de interacción respondan adecuadamente dentro del sistema.

- **Objetivo:** Confirmar que todas las partes de la aplicación operan de manera integrada, verificando que los flujos de trabajo completos (como registro, autenticación, envío de datos y consulta de información) se ejecuten sin errores.
- **Resultado esperado:** Que la aplicación realice todas las operaciones y flujos clave sin problemas, asegurando que cada funcionalidad principal esté correctamente integrada y funcione en conjunto.

### 4.9.3. Impacto de la eficiencia y de las pruebas de extremo a extremo en el servidor

- **Eficiencia:** Un servidor eficiente maximiza el uso de los recursos disponibles, lo que permite reducir los costos operativos y mantener un rendimiento constante. Esto es esencial para asegurar tiempos de respuesta rápidos y garantizar que el sistema pueda manejar múltiples solicitudes concurrentes sin una degradación del servicio.
- **Pruebas E2E:** Las pruebas de extremo a extremo permiten validar el flujo completo de operaciones, asegurando que cada componente del sistema funcione correctamente y en sincronía. Estas pruebas ayudan a identificar cualquier posible problema de integración entre APIs y otros módulos de la aplicación, lo que es esencial para una experiencia de usuario fluida y un sistema robusto.

## 4.10. Herramientas para pruebas de eficiencia y de extremo a extremo

Para realizar las pruebas de eficiencia y extremo a extremo, se emplearon las siguientes herramientas:

- **Python:** Se utilizó para desarrollar scripts personalizados que permiten simular múltiples usuarios y escenarios de carga en la aplicación, evaluando el tiempo de respuesta, el rendimiento y el uso de recursos en situaciones de carga típica y bajo prueba de integración completa (Python Software Foundation, 2024).
- **NGINX Amplify:** Herramienta de monitoreo utilizada para supervisar el uso de recursos en tiempo real, como CPU, memoria y tráfico de red. NGINX Amplify proporciona métricas detalladas y facilita el análisis de rendimiento durante las pruebas, identificando posibles áreas de optimización y problemas de integración (NGINX, Inc., 2024).

#### 4.10.1. Monitoreo del sistema

- **htop:** Proporciona una vista en tiempo real del uso de CPU, memoria y procesos en ejecución.
- **dstat:** Genera estadísticas combinadas de CPU, disco, memoria y red.
- **iostat:** Muestra el uso de disco por cada proceso, facilitando la identificación de procesos que consumen muchos recursos.
- **Prometheus:** Sistema de monitorización para métricas a gran escala.
- **Prometheus-Exporter:** Herramienta para monitorear las solicitudes entrantes al servidor.
- **Grafana:** Plataforma de visualización de métricas, utilizada para analizar los resultados de Prometheus (DigitalOcean, 2021).

#### 4.10.2. Pruebas de carga y end to end

Las pruebas de carga se centran en medir la respuesta del sistema bajo condiciones de uso previstas, simulando múltiples usuarios y solicitudes simultáneas. Estas pruebas permiten evaluar el rendimiento, tiempos de respuesta y estabilidad del sistema cuando procesa diversas acciones en paralelo (LoadView, 2024).

- **Objetivo:** Evaluar cómo responde el sistema bajo una carga esperada de usuarios o solicitudes simultáneas, enfocándose en el rendimiento y la estabilidad mientras se procesan múltiples acciones.
- **Resultado esperado:** Que el sistema mantenga tiempos de respuesta y niveles de rendimiento aceptables bajo condiciones de uso normales (es decir, el número máximo de usuarios esperados en condiciones típicas).
- **Prueba:** La prueba de carga implementada simula a varios usuarios ejecutando operaciones simultáneas, como autenticación, envío de videos y marcación de favoritos. Se registra el tiempo de respuesta y el estado de cada acción, evaluando así la capacidad de la aplicación para manejar un volumen esperado de usuarios sin degradar su rendimiento.

Las pruebas de Extremo a Extremo (E2E) se orientan a evaluar el flujo completo de la aplicación desde la perspectiva del usuario. A diferencia de las pruebas de carga, estas pruebas verifican que todos los componentes del sistema funcionen conjuntamente como se espera en un caso de uso completo.

- **Objetivo:** Confirmar que todas las partes de la aplicación funcionan de manera integrada y que los componentes interactúan correctamente en un flujo de uso completo.

- **Resultado esperado:** Que la aplicación ejecute todas las operaciones y flujos clave sin errores (por ejemplo, registro, autenticación, envío y consulta de datos), asegurando que las funcionalidades principales se integren de manera adecuada.
- **Prueba:** La prueba E2E implementada cubre un caso de uso completo, desde la creación de un usuario hasta la interacción con las principales funciones de la aplicación, como autenticación, carga de videos, traducción y manipulación de favoritos. Esta prueba garantiza que cada parte del sistema funcione en conjunto en el flujo real de un usuario.

## 4.11. Trabajo de campo

Durante el proyecto, se tomó la decisión de no utilizar servicios en la nube como AWS o Azure debido al alto costo de sus equipos y servicios. En su lugar, se optó por utilizar los servidores disponibles en la UVG, específicamente el modelo DELL R740 ubicado en Jack's Cave. Este equipo, que no poseía un sistema operativo preinstalado, fue configurado con Ubuntu Server 22.08, elegido por su versatilidad y disponibilidad gratuita. Este sistema operativo permitió la implementación de múltiples herramientas, incluyendo VPNs mediante OpenVPN siguiendo los pasos proporcionados por DigitalOcean (DigitalOcean, s.f.), lo que facilitaron el acceso remoto seguro al servidor.

El proceso de instalación incluyó la interacción con el controlador de RAID del servidor, el cual estaba gestionado por el controlador DELL PERC. Aunque RAID permite gestionar múltiples discos duros de manera eficiente, en este caso específico, donde solo se dispone de un disco duro de 1.92 TB, se decidió no utilizar el controlador RAID, ya que no ofrecía ventajas significativas para el sistema. En su lugar, el disco duro fue gestionado directamente, implementando mejoras manuales que ofrecían un rendimiento adecuado sin la complejidad adicional de RAID (Dell, s.f.).

Tras la instalación del sistema operativo, se procedió a configurar OpenVPN para gestionar el acceso remoto al servidor. Los pasos seguidos incluyeron: 1. Generación y configuración de los materiales criptográficos necesarios, como claves y certificados, para garantizar una comunicación segura entre el servidor y los clientes. 2. Ajuste de la configuración de red de OpenVPN para la correcta distribución del tráfico de datos. 3. Configuración de las reglas de firewall necesarias para permitir el tráfico a través de la VPN sin comprometer la seguridad del sistema. 4. Inicio del servicio de OpenVPN y realización de pruebas para asegurar que las conexiones remotas funcionen correctamente (HowToGeek, 2022).



## CAPÍTULO 5

---

### Antecedentes

---

No existen antecedentes específicos en la Universidad sobre proyectos previos que integren de forma directa la arquitectura universitaria con el fin de acceder a un servidor dedicado para pruebas y desarrollo de modelos de inteligencia artificial y visión por computadora. Este proyecto representa el primer esfuerzo en crear un ambiente accesible y seguro para el despliegue de modelos avanzados, permitiendo el acceso externo mediante una infraestructura de servidores segura y controlada.

Sin embargo, se puede mencionar como antecedente cercano el proyecto titulado *Plataforma para el registro, visualización y difusión de competencias educativas para la mejora en el proceso de reclutamiento laboral de los miembros del departamento de computación de la UVG*, desarrollado por Paul De Jesús Belches Flores-Gómez y David Uriel Soto Alvarez. Este proyecto tiene una orientación hacia el uso de APIs para la centralización y gestión de competencias educativas en la Universidad, mejorando el acceso a la información y facilitando su uso en contextos administrativos y de reclutamiento laboral.

A nivel administrativo, el proyecto de Belches y Soto estableció un precedente en el uso de tecnologías API para centralizar y organizar datos en la Universidad, sentando una base técnica que influyó en el planteamiento de este proyecto al buscar una integración efectiva de APIs en el entorno universitario.

Por otro lado, el profesor Miguel Novella jugó un rol importante en apoyar los esfuerzos para establecer un acceso seguro a la arquitectura universitaria, un proceso que involucró la colaboración con el equipo de IT de la Universidad. Su experiencia y respaldo fueron esenciales en la gestión de permisos y en el acceso a la infraestructura universitaria, sentando las bases necesarias para la implementación exitosa del presente proyecto.



# CAPÍTULO 6

---

## Alcance

---

Este proyecto se desarrollará dentro de los límites de la infraestructura universitaria disponible, cumpliendo con los objetivos propuestos y permitiendo la ejecución de todas las funciones del servidor bajo las restricciones del entorno técnico de la Universidad. Los alcances abarcan tanto la capacidad de respuesta como la conexión segura para los usuarios de la aplicación, quienes deberán acceder al servidor a través de la VPN institucional FortiClient o desde la red local en el área conocida como la Jack's Cave, donde se encuentra físicamente ubicado el equipo.

Dada la restricción en la configuración de red y la falta de soporte para port-forwarding al exterior de la Universidad, se define que las conexiones externas no podrán realizarse sin acceso previo a la VPN universitaria o sin una conexión local directa al servidor. Esto introduce una capa adicional en el proceso de conexión, lo cual puede tener un impacto en la eficiencia de respuesta del sistema debido a la latencia introducida por la VPN.

Además, el proyecto está limitado al uso del equipo proporcionado en las instalaciones de la Universidad. Dado que no fue posible realizar expansiones o modificaciones en el hardware disponible, la infraestructura actual impone un límite en la escalabilidad del sistema. Este proyecto, por lo tanto, se ha diseñado cuidadosamente para optimizar el rendimiento dentro de los recursos de procesamiento, almacenamiento y red que el equipo de la Universidad permite.

Las pruebas de seguridad realizadas en este proyecto se enfocaron exclusivamente a nivel de software debido a que el servidor utilizado formaba parte de la infraestructura de la universidad, la cual contaba con políticas y configuraciones de seguridad preestablecidas que no estuvieron bajo nuestro control directo. Estas condiciones limitaron las mejoras adicionales que pudimos implementar a nivel de hardware o configuraciones específicas. Sin embargo, el enfoque en software fue válido, ya que según el NIST Cybersecurity Framework (National Institute of Standards and Technology, 2024), la mayoría de las vulnerabilidades críticas en servidores están relacionadas con configuraciones y software, no con el hardware. Además, este diseño aseguró que el sistema pudiera ser trasladado a otros servidores en el futuro, manteniendo la robustez de las medidas implementadas sin depender de un entorno físico específico. Herramientas como Lynis fueron diseñadas para evaluar y garantizar un nivel adecuado de seguridad en sistemas que podían operar en entornos diversos, haciendo este enfoque flexible y alineado con estándares modernos de seguridad.

## 6.1. Consideraciones adicionales

1. **Seguridad y restricciones de acceso:** Las pruebas de acceso remoto estarán sujetas a las políticas de seguridad de la Universidad, limitándose al entorno universitario, lo cual asegura un control estricto sobre los datos y recursos.
2. **Pruebas de carga y estrés limitadas al entorno interno:** Las evaluaciones de eficiencia y respuesta se basarán en simulaciones dentro del sistema de la Universidad, dado que la infraestructura no se encuentra abierta al público externo. Esto limita las pruebas a las condiciones internas del entorno universitario, lo cual deberá tenerse en cuenta para futuras escalas de uso.
3. **Conformidad con los recursos de red de la Universidad:** Debido a la dependencia de la red de la Universidad, el rendimiento del servidor está condicionado a la calidad de la conexión local y a la estabilidad de la VPN, factores externos que pueden influir en la velocidad y confiabilidad del sistema.
4. **Limitación en las pruebas de carga por duración del video:** Debido a la naturaleza del modelo y el material disponible, las pruebas de carga se realizaron exclusivamente con videos de una duración de 2 segundos.

Estas precisiones establecen claramente el alcance dentro de los recursos disponibles y el entorno controlado de la Universidad, asegurando que el proyecto funcione de manera efectiva bajo las condiciones actuales y anticipando limitaciones para posibles expansiones o cambios futuros.



# CAPÍTULO 7

---

## Metodología

---

Se necesitan 4 pasos para poder cumplir los objetivos y crear una infraestructura de red funcional, que permita la ejecución de múltiples consultas, manejo de datos y el desarrollo de la aplicación.

1. Levantado de un sistema operativo y una VPN que permita el acceso a los servidores, permitiendo un mejor ambiente de trabajo.
2. Desarrollar arquitectura de base de datos para gestionar los datos, asegurando la seguridad de los mismos.
3. Implementar un módulo de APIs para bases de datos que permita consultas e inserciones.
4. APIs para acceder a los diferentes modelos para uso interno y externo, que permitan el envío y recepción de información entre los módulos. Al igual que la arquitectura de red y de sistema para que el proceso sea lo más eficiente posible.

### 7.1. Levantamiento de un sistema operativo y configuración de VPN

Para garantizar un ambiente de trabajo seguro y eficiente, se ha implementado Ubuntu 22.08 como sistema operativo en un servidor Dell R740. Se eliminó el controlador RAID para liberar un disco duro de 1.92T, adecuándolo para las necesidades del proyecto. Además, se configuró una VPN utilizando OpenVPN, lo cual permite un acceso directo y seguro a los servidores. La configuración de la VPN se complementó con ajustes en los firewalls institucionales otorgados por la universidad, facilitando el acceso remoto y el manejo del servidor desde múltiples dispositivos sin comprometer la seguridad. Esta configuración permite una administración centralizada y segura del servidor, asegurando que solo usuarios autorizados puedan acceder a los recursos críticos del sistema.

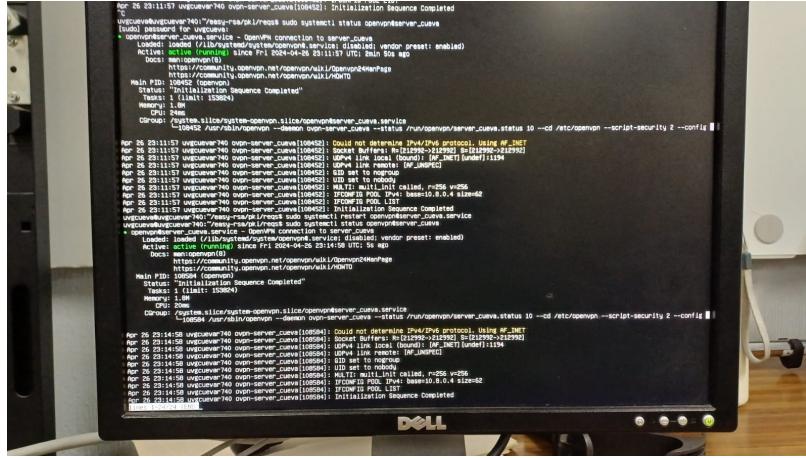


Figura 7.1: LevantandoOpenVPN.

Como se puede observar en la Figura 7.1, la máquina virtual ha sido levantada correctamente.

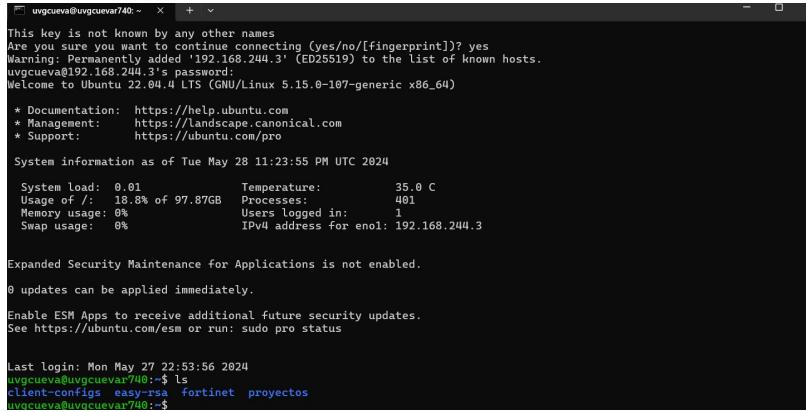


Figura 7.2: ConexionSSH.

Además, en la Figura 7.2 se muestra la conexión generada con la VPN mediante SSH.

El proceso requiere conectarse a través de la infraestructura de red de la universidad y la VLAN específica donde se encuentra el rack y el servidor. Para lograr esto, se utiliza FortiClient VPN, que proporciona una conexión segura y cifrada directamente a la VLAN donde se encuentra el servidor. Para acceder a la VPN, se necesita un archivo de configuración .ovpn que se utiliza con OpenVPN. Una vez conectado a la VPN, se puede acceder al servidor de manera segura mediante SSH. Esto se realiza abriendo una terminal y utilizando el comando ssh uvrgcueva@192.168.244.3 con las credenciales correspondientes. Este método asegura que la administración remota de las máquinas virtuales se realice de manera eficiente y segura, permitiendo a los usuarios realizar tareas administrativas y de mantenimiento sin estar físicamente presentes en el sitio.

## 7.2. Desarrollo de la arquitectura de base de datos

Se ha seleccionado PostgreSQL para gestionar la base de datos relacional del proyecto debido a su robustez y eficacia en el manejo de grandes volúmenes de datos, incluyendo datos de usuarios,

videos y otros tipos de información. PostgreSQL ofrece avanzadas características de seguridad que son esenciales para proteger los datos contra accesos no autorizados. Su arquitectura permite realizar operaciones complejas de base de datos de manera eficiente, asegurando la integridad y la privacidad de los datos gestionados. En la arquitectura se hace especial hincapié en la importancia de diseñar con flexibilidad, utilizando estándares que permitan la interacción entre diferentes sistemas y tecnologías.

Para implementar PostgreSQL en nuestro proyecto de forma efectiva y segura, seguiremos una metodología estructurada que abarcará desde la planificación inicial hasta la integración con otros sistemas. Comenzaremos evaluando los requisitos de hardware y software para seleccionar la versión óptima de PostgreSQL y preparar el entorno del sistema operativo. Una vez instalado PostgreSQL, procederemos a configurar los parámetros de rendimiento y seguridad, como la memoria, conexiones permitidas, y configuraciones de cifrado, además de establecer un protocolo regular de mantenimiento que incluirá actualizaciones y parches.

La seguridad será una prioridad, implementando roles y permisos detallados para controlar el acceso a los datos y asegurar todas las comunicaciones mediante SSL/TLS. Paralelamente, desarrollaremos y configuraremos procesos de ETL para automatizar la extracción, transformación y carga de datos, asegurando la integridad y utilidad de la información gestionada. Además, diseñaremos y desarrollaremos APIs que permitan una interacción segura y eficiente con otras aplicaciones, implementando mecanismos robustos de autenticación y autorización.

Finalmente, estableceremos sistemas de monitoreo para supervisar constantemente el rendimiento y la seguridad de la base de datos y las APIs, permitiendo ajustes continuos para optimizar la configuración según las necesidades cambiantes del proyecto. Este enfoque metodológico asegura no solo la implementación técnica efectiva de PostgreSQL, sino también el mantenimiento de un sistema seguro, eficiente y adaptable a largo plazo.

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único del usuario.
mail	String(120)	Unique, Not Null	Correo electrónico del usuario.
password	String(120)	Not Null	Contraseña del usuario.
streak	Integer	Default = 0	Puntuación o racha del usuario.
quetzalito	String	Nullable	Color de la imagen de perfil.
confirmed	Boolean	Default=False	Indica si el usuario ha confirmado su cuenta.
last_streak_update	DateTime	Default=datetime.utcnow	Indica la última hora que se agregó un streak.

Tabla 7.1: Tabla: user

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único del video.
id_user	Integer	ForeignKey(user.id), Not Null	Referencia al usuario que subió el video.
traduction_esp	String(255)	Nullable	Traducción del video al español.
sentence_lenseguia	String(255)	Not Null	Oración en lengua de señas guatemalteca.
video	String(255)	Not Null	Ruta del archivo de video.
prev_image	String(255)	Nullable	Imagen previa del video.
is_favorite	Boolean	Default=False	Indica si el video es favorito del usuario.

Tabla 7.2: Tabla: video

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único de la traducción.
id_user	Integer	ForeignKey(user.id), Not Null	Referencia al usuario que hizo la traducción.
sentence_lenseguia	String(255)	Not Null	Oración en lengua de señas guatemalteca.
traduction_esp	String(255)	Nullable	Traducción al español.
is_favorite	Boolean	Default=False	Indica si la traducción es favorita.

Tabla 7.3: Tabla: traducción

Nombre del campo	Tipo de dato	Restricciones	Descripción
id	Integer	Primary Key	Identificador único del diccionario.
id_user	Integer	ForeignKey(user.id), Not Null	Referencia al usuario.
id_word	String(255)	Not Null	Palabra o identificador del término en el diccionario.

Tabla 7.4: Tabla: dictionary

El diagrama de entidad-relación presentado en la Figura 7.3 muestra la estructura y las relaciones entre las tablas diseñadas para el proyecto.

En este diagrama se observan las entidades principales del sistema: `user`, `video`, `traducción` y `dictionary`. Cada una de estas entidades incluye atributos que definen su propósito dentro del sistema. Por ejemplo, la tabla `user` almacena información esencial del usuario como su correo, contraseña y estado de confirmación, mientras que la tabla `video` registra información de los videos asociados a los usuarios, incluyendo traducciones y metadatos adicionales.

Las relaciones entre las entidades destacan cómo interactúan entre sí. Por ejemplo, la tabla `video` está vinculada a la tabla `user` mediante una relación de uno a muchos (`1:N`), lo que indica que cada usuario puede tener múltiples videos asociados. Asimismo, la tabla `traducción` permite almacenar traducciones específicas realizadas por los usuarios, vinculándose también a través de una relación de `1:N` con `user`.

Este diseño garantiza la flexibilidad y escalabilidad del sistema, permitiendo el manejo eficiente de los datos necesarios para cumplir con los objetivos del proyecto.

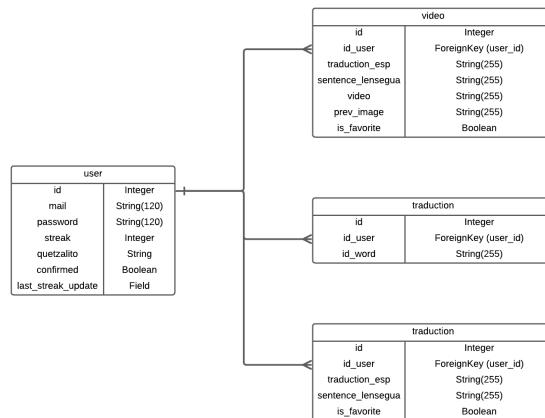


Figura 7.3: Diagrama entidad relación.

### 7.3. Implementacion de API's

Para facilitar las consultas y las inserciones de datos, se utilizará Flask como framework principal debido a su ligereza y capacidad para manejar peticiones de manera eficiente. Este módulo API permitirá un manejo flexible de diferentes formatos de datos, incluyendo videos y texto. La estructura del módulo está diseñada para optimizar el rendimiento del sistema, minimizando el uso de recursos y maximizando la velocidad de respuesta, lo que es crucial para procesar y responder a las interacciones de los usuarios en tiempo real.

Nombre	TIPO	Entrada	Salida
login	POST	email, password	id_user
signup	POST	email, password, quetzalito	id_user
forgot_password	POST	email	200-OK
change_password	POST	id_user, new_password	id_user, 200-OK

Tabla 7.5: Resumen de Rutas de API: `user_routes`

Nombre	TIPO	Entrada	Salida
send_video	POST	id_user, video	id_video, traduction_lensegua, traduction_esp
report_video	POST	id_user, id_video, report_message, report_img (png)	200-OK
fav_video	POST	id_user, id_video, prev_video (png)	200-OK
remove_video	DELETE	id_video	200-OK
remove_fav_video	POST	id_user, id_video	200-OK
download_video	POST	route_video	ruta_video
download_images	POST	route_image	ruta_image

Tabla 7.6: Resumen de Rutas de API: `video_routes`

Nombre	TIPO	Entrada	Salida
send_traduction	POST	id_user, sentence_lensegua	id_sentence, traduction_esp
fav_traduction	POST	id_user, id_sentence	200-OK
remove_traduction	DELETE	id_sentence	200-OK
remove_fav_traduction	POST	id_user, id_sentence	200-OK

Tabla 7.7: Resumen de Rutas de API: `traduction_routes`

Nombre	TIPO	Entrada	Salida
add_dictionary	POST	id_user, id_word	200-OK
remove_dictionary	DELETE	id_user, id_word	200-OK
get_dictionary	POST	id_user	palabras (json)

Tabla 7.8: Resumen de Rutas de API: `dictionary_routes`

Nombre	TIPO	Entrada	Salida
get_user_info	POST	id_user	email, streak, quetzalito, videos_fav (json), traductions_fav (json)
get_video	POST	id_user, id_video	video (mp4)
get_image	POST	id_user, id_video	image (png)
delete_user	DELETE	id_user	200-OK
add_streak	POST	id_user	200-OK
remove_streak	POST	id_user	200-OK

Tabla 7.9: Resumen de Rutas de API: `profile_routes`

Nombre	TIPO	Entrada	Salida
confirm	POST	email	200-OK

Tabla 7.10: Resumen de Rutas de API: `mail_routes`

### 7.3.1. Flujo de trabajo para sistema de apis

La infraestructura del sistema sigue un flujo bien definido en el manejo de solicitudes y respuestas, diseñado para garantizar un equilibrio entre eficiencia y robustez.

1. **Cliente envía una solicitud:** El punto de partida se da cuando un cliente realiza una solicitud HTTP al servidor, por ejemplo, `mi-ip:4242/example`. Este podría ser un navegador web, una aplicación móvil, o cualquier sistema que consuma las APIs que estamos desarrollando. En este primer punto, la solicitud se envía al servidor en la ruta definida.
2. **Nginx recibe la solicitud:** Nginx actúa como la primera línea de defensa y como un proxy inverso. Es el encargado de recibir todas las solicitudes entrantes y decidir cómo deben ser manejadas. Nginx no solo se encarga de redirigir las solicitudes, sino que también es clave en el manejo de conexiones concurrentes, balanceo de carga y la administración de recursos estáticos si es necesario. Gracias a Nginx, el sistema puede gestionar eficientemente un alto volumen de solicitudes simultáneas, *distribuyéndolas* entre los diferentes trabajadores de Gunicorn.
3. **Nginx reenvía la solicitud a Gunicorn:** Una vez que Nginx recibe la solicitud, la reenvía a Gunicorn utilizando un socket Unix o TCP, según la configuración definida. Gunicorn es un servidor de aplicaciones WSGI diseñado específicamente para ejecutar aplicaciones Flask en entornos de producción. En este paso, Gunicorn toma la solicitud y la pasa al núcleo de la aplicación Flask.
4. **Gunicorn procesa la solicitud:** Al recibir la solicitud de Nginx, Gunicorn la envía a la aplicación Flask, encargada de manejar la lógica de negocio asociada a cada *endpoint*. Por ejemplo, si la solicitud se realizó a la ruta `/example`, Flask ejecutará la función asociada a esa ruta. Esta función puede incluir lógica para consultar una base de datos, realizar cálculos, o incluso comunicarse con otros servicios.
5. **Flask genera y envía la respuesta:** Una vez que Flask ha procesado la solicitud, genera una respuesta adecuada. Esta respuesta puede estar en formato JSON, HTML, o cualquier otro formato según lo requerido por el cliente. Posteriormente, Flask devuelve la respuesta a Gunicorn, que se encargará de llevarla al siguiente paso en el flujo.
6. **Gunicorn reenvía la respuesta a Nginx:** Al recibir la respuesta de Flask, Gunicorn la reenvía a Nginx, que tiene la responsabilidad de empaquetarla y prepararla para ser entregada de vuelta al cliente.
7. **Nginx devuelve la respuesta al cliente:** Finalmente, Nginx envía la respuesta al cliente que hizo la solicitud. El cliente recibe la respuesta HTTP, que puede ser procesada y presentada de acuerdo a sus necesidades. Este paso cierra el ciclo de la solicitud, asegurando que el sistema haya gestionado de manera eficiente tanto la entrada como la salida de información.

### 7.3.2. Diagrama de Flujo de la Arquitectura de Solicitudes

En esta sección se presenta el diagrama de flujo de la arquitectura del proyecto (Figura 7.4), el cual ilustra cómo se gestionan las solicitudes en el sistema. Nuestra estructura sigue el principio de "dividir

y conquistar", separando las múltiples solicitudes en secciones que tienen objetivos específicos y bien definidos.

Las solicitudes se agrupan en diferentes módulos, cada uno manejado por un archivo dedicado:

- **profile\_routes.py:** Gestiona operaciones relacionadas con la información del perfil del usuario, como obtener datos personales, videos y traducciones favoritas, y la gestión de la racha (*streak*).
- **user\_routes.py:** Maneja operaciones de autenticación y registro, como iniciar sesión, registrarse, y cambiar contraseñas.
- **video\_routes.py:** Se ocupa de la carga, gestión y eliminación de videos, así como de marcar o desmarcar favoritos y manejar imágenes de previsualización.
- **traduction\_routes.py:** Administra las traducciones de frases en Lensegua al español, permitiendo marcar traducciones como favoritas o eliminarlas.
- **dictionary\_routes.py:** Permite agregar, eliminar y consultar palabras en el diccionario de favoritos del usuario.
- **mail\_routes.py:** Facilita el envío de correos electrónicos, como en el caso de restablecimiento de contraseñas.

Todos estos módulos giran en torno al uso del archivo **models.py**, el cual actúa como el eje central del sistema. Este archivo define las entidades y modelos utilizados para interactuar con la base de datos, permitiendo que todas las secciones puedan acceder y realizar operaciones en ella de manera eficiente y centralizada.

Esta división modular asegura una arquitectura organizada, fácil de mantener y escalable, donde cada módulo se encarga de tareas específicas, mientras que el modelo central proporciona una única fuente de verdad para la interacción con los datos.

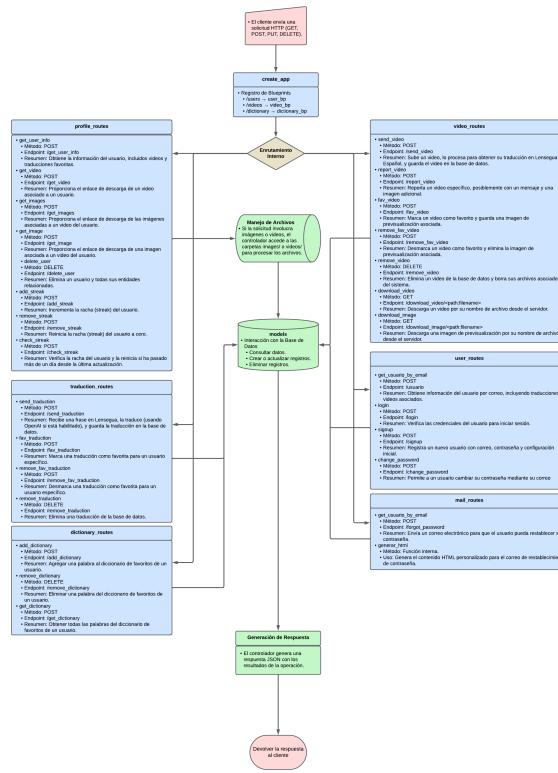


Figura 7.4: Diagrama de flujo.

### 7.3.3. Implementación de Flask con Gunicorn y NGinx

## Componentes Principales

- a) **Flask**: Framework de Python utilizado para construir la lógica de la aplicación.
  - b) **Gunicorn**: Servidor WSGI que ejecuta la aplicación Flask en un entorno de producción.
  - c) **Nginx**: Proxy inverso que maneja las solicitudes del cliente, distribuye la carga, y mejora la seguridad.

### Flujo de trabajo del sistema de APIs:

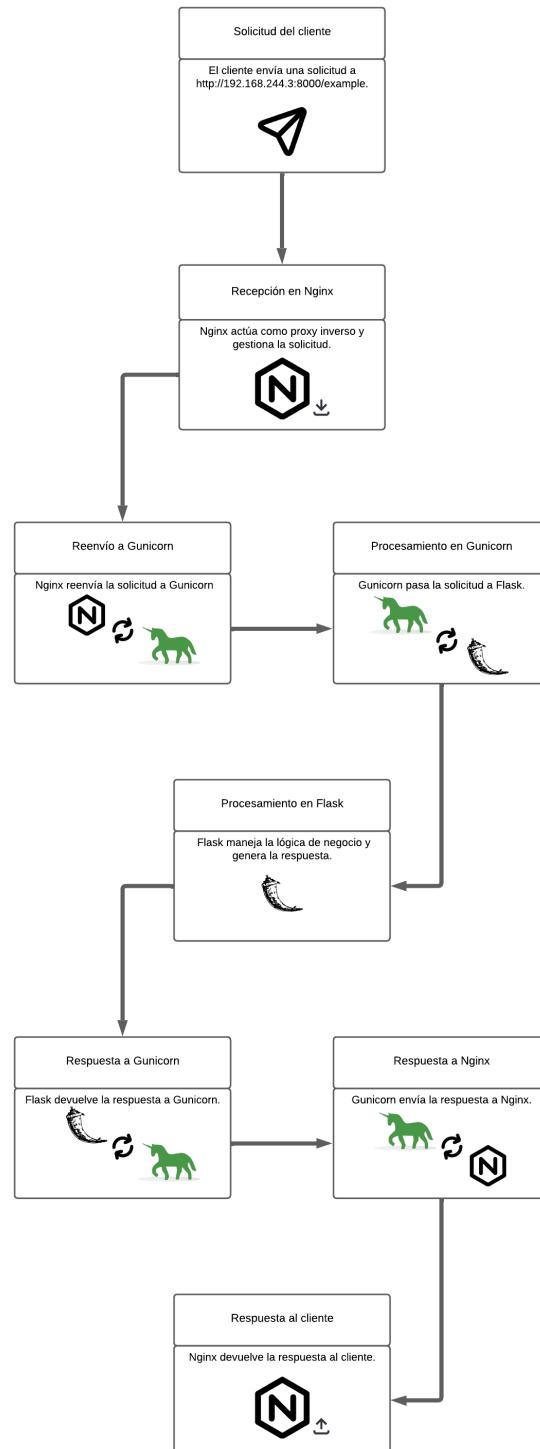


Figura 7.5: Flujo de trabajo completo del sistema de APIs.

## Configuración general

### a. Configuración de Flask

La aplicación Flask fue configurada con soporte para migraciones de bases de datos, utilizando SQLAlchemy y Flask-Migrate. Esta configuración permite la manipulación sencilla de esquemas de bases de datos a medida que se añaden nuevas funcionalidades al sistema.

### b. Configuración de Gunicorn

- I. Gunicorn se seleccionó como el servidor WSGI debido a su capacidad para manejar múltiples solicitudes concurrentes mediante el uso de workers. Gunicorn se ejecuta vinculándose a todas las interfaces de red en un puerto específico, lo que permite que la aplicación esté disponible en la red local o en internet.
- II. Para gestionar múltiples procesos de Gunicorn, se configuró su ejecución como un servicio del sistema. Esto asegura que Gunicorn inicie automáticamente con el sistema y facilite la gestión del servidor en producción.

### c. Configuración de Nginx

- I. Nginx se configuró como un proxy inverso para manejar la entrada de solicitudes del cliente y distribuirlas a Gunicorn. Este paso es crucial para manejar eficientemente la concurrencia y asegurar que el tráfico de red no sobrecargue el servidor de aplicaciones.

### d. Archivo de configuración de Nginx:

Listing 7.1: Archivo de configuración de Nginx.

```

1 server {
2     listen 4242;
3     server_name 192.168.244.3;
4
5     location / {
6         proxy_pass http://unix:/srv/web-apps/api-central/api-central
7             .sock;
8         include proxy_params;
9         proxy_redirect off;
10    }
}

```

### Explicación del archivo de configuración de Nginx:

- I. **listen 4242**: Especifica el puerto en el que Nginx escuchará las solicitudes.
- II. **server\_name 192.168.244.3**: Define la dirección IP del servidor.
- III. **proxy\_pass**: Redirige las solicitudes recibidas a Gunicorn mediante un socket Unix, lo que mejora el rendimiento y la seguridad al evitar conexiones TCP adicionales.
- IV. **include proxy\_params**: Incluye parámetros comunes para proxies inversos en Nginx.

- v. **proxy\_redirect off**: Desactiva la redirección automática de respuestas, permitiendo un control más preciso sobre el flujo de la solicitud y respuesta.

### 7.3.4. Implementación de SQLAlchemy para manejo de base de datos y entidades

#### 1. Definición de Entidades

Las entidades en SQLAlchemy se definen como clases que heredan de `db.Model`. Cada clase representa una tabla en la base de datos, y sus atributos corresponden a las columnas de la tabla. Por ejemplo, una entidad `Usuario` se define para representar la tabla `usuarios` con columnas como `id`, `correo` y `contraseña`. Estas clases se definen en un archivo específico, como `models.py`, para mantener el código organizado y escalable.

#### 2. Relación entre Entidades

SQLAlchemy permite definir relaciones entre entidades usando claves foráneas (*ForeignKey*). Por ejemplo, en la entidad `Video`, existe una relación con la entidad `Usuario` mediante la columna `usuario_id`, que referencia al `id` de la tabla `usuarios`. Este tipo de relaciones permiten manejar estructuras de datos más complejas y realizar consultas entre tablas relacionadas de forma sencilla.

#### 3. Migraciones de Base de Datos

Flask-Migrate se utiliza para aplicar cambios en el esquema de la base de datos de manera controlada. Una vez que las entidades están definidas, Flask-Migrate permite generar y aplicar migraciones para reflejar estos modelos en la base de datos. Este proceso se realiza en tres pasos:

- a. **Inicialización de las Migraciones**: Se configura Flask-Migrate para gestionar la base de datos.
- b. **Creación de Migraciones**: Cada vez que se añade o modifica una entidad, se genera una nueva migración.
- c. **Aplicación de Migraciones**: Las migraciones se aplican a la base de datos, creando o modificando las tablas según sea necesario.

#### 4. Operaciones CRUD

Con SQLAlchemy, las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) se manejan fácilmente mediante el uso de sesiones de base de datos. Estas sesiones permiten realizar consultas, insertar nuevos registros, actualizar datos existentes y eliminar registros, todo de una manera declarativa y fluida.

### 7.3.5. Implementación de módulo API para bases de datos

Para facilitar las consultas y las inserciones de datos, se utilizará Flask como framework principal debido a su ligereza y capacidad para manejar peticiones de manera eficiente. Este módulo API permitirá un manejo flexible de diferentes formatos de datos, incluyendo videos y texto. La estructura

del módulo está diseñada para optimizar el rendimiento del sistema, minimizando el uso de recursos y maximizando la velocidad de respuesta, lo que es crucial para procesar y responder a las interacciones de los usuarios en tiempo real.

#### **Módulo: Procesamiento de lenguaje**

- **Nombre de la API:** API\_Procesamiento\_de\_Lenguaje
- **Función de la API:** Procesar texto y generar respuestas basadas en el modelo
- **Entrada:** XML con texto
- **Salida Esperada:** Respuesta en XML del modelo GPT-4

#### **Módulo: Llama**

- **Nombre de la API:** API\_Llama
- **Función de la API:** Analizar y responder consultas con el modelo Llama
- **Entrada:** XML con consulta
- **Salida Esperada:** Respuesta en XML del modelo Llama

#### **Módulo: Visión por computadora**

- **Nombre de la API:** API\_Vision
- **Función de la API:** Analizar imágenes y videos, y detectar objetos
- **Entrada:** XML con imagen/video
- **Salida Esperada:** Respuesta en XML con análisis de visión por computadora

#### **Módulo: HyperVisor**

- **Nombre de la API:** API\_HyperVisor
- **Función de la API:** Gestionar y monitorizar las máquinas virtuales
- **Entrada:** XML con comandos
- **Salida Esperada:** Respuesta en XML con estado de VMs

### **7.3.6. API para integración de modelos de procesamiento**

Se desarrollará una API específica para facilitar la interacción entre el módulo de visión por computadora y los modelos de lenguaje natural. Esta API asegurará que la información fluya de manera eficiente desde la recepción de videos hasta la entrega de resultados en texto o voz, pasando por la consulta de bases de datos. La arquitectura de red y del servidor será diseñada para manejar altas demandas de solicitudes sin degradar la calidad del servicio, utilizando técnicas de balanceo de carga y optimización de recursos para garantizar un proceso robusto y confiable.

Se diseñará un entorno virtualizado utilizando Multipass sobre un servidor con Ubuntu 22.08 ya instalado, lo que facilitará la segregación de los módulos de visión por computadora, inteligencia

artificial con GPT-4 y el modelo de IA Llama. Cada uno de estos módulos operará dentro de su propia máquina virtual, configurada específicamente para satisfacer sus requisitos operativos y de procesamiento.

Para iniciar el proceso, se configurarán máquinas virtuales individuales para cada módulo bajo el hypervisor Multipass en Ubuntu 22.08. Esta configuración implica la asignación de los recursos necesarios, como CPU dedicada, memoria RAM adecuada y espacio de almacenamiento suficiente, de acuerdo con las demandas específicas de cada aplicación. Además, se establecerán redes virtuales que permitan una comunicación eficaz y segura entre las máquinas virtuales y los sistemas de bases de datos externos.

### 7.3.7. Implementación de Crontab

Para garantizar que los usuarios mantengan una racha activa y que se reinicie automáticamente si ha pasado más de un día sin actividad, se ha implementado un proceso automatizado utilizando ‘crontab’ en el servidor Linux. Esta tarea se ejecuta todos los días a las 3:00 a.m. y ejecuta un script de Python que revisa la última hora de actualización de la racha de cada usuario y la reinicia si ha pasado más de 24 horas desde la última actualización.

#### Configuración del Cron Job

Para configurar la tarea programada que se ejecuta a las 3:00 a.m. todos los días, se debe agregar la siguiente línea al archivo de crontab del usuario adecuado:

```
1 0 3 * * *
```

Listing 7.2: Frecuencia de ejecución

Indica la frecuencia de ejecución (todos los días a las 3:00 a.m.).

```
1 /usr/bin/python3
```

Listing 7.3: Ruta al intérprete de Python

Es la ruta al intérprete de Python.

```
1 /ruta/al/script/reset_streaks.py
```

Listing 7.4: Ruta al script de Python

Es la ruta completa al script de Python que contiene la lógica de reinicio de rachas.

```
1 >> /ruta/al/log/reset_streaks.log 2>&1
```

Listing 7.5: Redirección de salida y errores

Redirige la salida y los errores del script a un archivo de log para facilitar la revisión de su ejecución.

El script de Python que se ejecuta está diseñado para realizar las siguientes tareas:

1. Crear una aplicación Flask y un contexto de aplicación para que el script pueda interactuar con la base de datos.
2. Obtener todos los usuarios de la base de datos y revisar su última hora de actualización de racha.

3. Calcular la diferencia de tiempo entre la hora actual y la última actualización. Si ha pasado más de 24 horas, reiniciar la racha del usuario y actualizar la hora de la última modificación.
4. Registrar la acción en la base de datos y mostrar un mensaje de confirmación en la consola.

El código del script es el siguiente:

```

1  from app import create_app, db
2  from app.models import User
3  from datetime import datetime, timedelta
4
5  # Crear la aplicación Flask y contexto
6  app = create_app()
7  app.app_context().push()
8
9  def reset_streaks():
10     # Obtener todos los usuarios
11     users = User.query.all()
12     for usuario in users:
13         if usuario.last_streak_update:
14             # Calcular la diferencia de tiempo desde la última
15             # actualización
16             delta = datetime.utcnow() - usuario.last_streak_update
17             if delta > timedelta(days=1):
18                 # Ha pasado más de un día, reiniciar el streak
19                 usuario.streak = 0
20                 usuario.last_streak_update = datetime.utcnow()
21                 db.session.commit()
22                 print(f"Streak reset for user {usuario.mail}")
23
24 if __name__ == "__main__":
25     reset_streaks()

```

Listing 7.6: Script de restablecimiento de streaks en Flask

### Importancia de la Implementación

El uso de ‘crontab’ para esta tarea es fundamental para mantener la consistencia y la precisión en el control de rachas de los usuarios sin intervención manual. Esto garantiza que las rachas de los usuarios se gestionen de manera automatizada, promoviendo un comportamiento constante y ofreciendo una mejor experiencia de usuario. Además, el uso de ‘crontab’ permite que la tarea se ejecute en horarios de baja actividad del servidor, minimizando el impacto en el rendimiento del sistema.

## 7.4. Virtualización del servidor para múltiples modelos

### Implementación de arquitectura con máquinas virtuales

Para implementar esta arquitectura, se crearán un total de tres máquinas virtuales adicionales al servidor principal que actuará como HOST. Cada máquina virtual será dedicada a uno de los módulos: visión por computadora, inteligencia artificial con GPT-4 y el modelo de IA Llama.

Utilizando Multipass sobre Ubuntu 22.08, cada máquina virtual se configurará meticulosamente para cumplir con los requisitos específicos de procesamiento, asignando CPU dedicadas, memoria RAM adecuada y espacio de almacenamiento suficiente. Además, se establecerán redes virtuales para asegurar una comunicación eficiente y segura entre las máquinas virtuales y los sistemas de bases de datos. Esta configuración garantizará que cada módulo opere de manera independiente y óptima, permitiendo un procesamiento robusto y confiable bajo altas demandas de solicitudes sin comprometer la calidad del servicio.

### Resumen del Sistema:

Memoria RAM	Espacio en disco	CPU
Total: 125 GiB	/dev/mapper/ubuntu-vg-ubuntu-lv: 98G total, 19G usado, 75G disponible (20 % uso)	Arquitectura: x86_64
Usada: 50 MiB	/dev/sda2: 2.0G total, 253M usado, 1.6G disponible (14 % uso)	CPUs: 32 (Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz)
Libre: 75 GiB	/dev/sda1: 1.1G total, 6.1M usado, 1.1G disponible (1 % uso)	Núcleos por CPU: 16
Disponible: 123 GiB		Hilos por núcleo: 2
Swap Total: 8.0 GiB		Virtualización: VT-x
Swap Usada: 0B		

Tabla 7.11: Resumen del sistema

El uso de técnicas de balanceo de carga y la optimización de recursos serán esenciales para garantizar que el sistema pueda manejar altos volúmenes de solicitudes sin comprometer la calidad del servicio. Cada máquina virtual se configurará con las dependencias y bibliotecas necesarias, garantizando que los módulos de visión por computadora y los modelos de IA operen de manera óptima y confiable dentro de este entorno virtualizado controlado.

Parte de este balanceo implica la distribución de recursos entre las máquinas virtuales. Para esto, se dividieron los recursos entre las máquinas de la siguiente manera:

Máquina Virtual 1 (VM1)	RAM: 32 GiB	CPUs: 8
Máquina Virtual 2 (VM2)	RAM: 32 GiB	CPUs: 8
Máquina Virtual 3 (VM3)	RAM: 32 GiB	CPUs: 8
Host (host de VM)	RAM: 30 GiB (restante)	CPUs: 8 (restante)

Tabla 7.12: Distribución de recursos entre las máquinas virtuales y el host

Name	State	IPv4	Image
VM1	Running	10.47.92.160	Ubuntu 22.04 LTS
VM2	Running	10.47.92.70	Ubuntu 22.04 LTS
VM3	Running	10.47.92.195	Ubuntu 22.04 LTS

Figura 7.6: Estado de las máquinas virtuales dentro del sistema

## 7.5. Implementación de pruebas de eficiencia

### 1. Instalación de Herramientas de Monitoreo:

- 1.a. Instalar el agente NGINX Amplify para monitorear el uso de recursos (CPU, memoria, disco y red) en tiempo real.
- 1.b. Adicionalmente, instalar herramientas complementarias como `htop`, `dstat`, `iostop`, `nload` y `sysdig` para monitoreo detallado y comparación de métricas.

### 2. Preparación del Entorno de Pruebas:

- 2.a. Desarrollar un script en Python que realice solicitudes a las APIs, utilizando librerías como `requests` para solicitudes HTTP y `psutil` para monitoreo de recursos.
- 2.b. Configurar el sistema de logging para registrar métricas en un archivo de log y enviar datos al agente NGINX Amplify.

### 3. Monitoreo Continuo:

- 3.a. Monitorizar los recursos del sistema en tiempo real utilizando NGINX Amplify y las herramientas complementarias instaladas.
- 3.b. Registrar métricas clave como el uso de CPU, memoria, disco y red durante las pruebas.
- 3.c. Configurar alertas en NGINX Amplify para notificaciones automáticas sobre uso excesivo de recursos.

### 4. Análisis de Resultados:

- 4.a. Revisar los logs generados durante las pruebas para evaluar el tiempo de respuesta y la utilización de recursos.
- 4.b. Utilizar el panel de NGINX Amplify para analizar métricas históricas y tendencias de rendimiento.
- 4.c. Identificar cuellos de botella y áreas de mejora mediante las métricas y gráficos proporcionados por NGINX Amplify.

### 5. Optimización y Repetición de Pruebas:

- 5.a. Implementar ajustes necesarios en la configuración del servidor y las aplicaciones basados en los resultados del análisis.
- 5.b. Repetir las pruebas para verificar las mejoras en el rendimiento del sistema.
- 5.c. Continuar el ciclo de pruebas y optimización hasta alcanzar un rendimiento óptimo, utilizando NGINX Amplify para validar las mejoras.

## 7.6. Pruebas de Carga

### 7.6.1. Objetivo

El objetivo de las pruebas de carga es evaluar el rendimiento del sistema bajo condiciones de carga esperadas, simulando múltiples usuarios y solicitudes concurrentes para determinar la capacidad del

sistema. Estas pruebas permiten identificar posibles áreas de mejora en la infraestructura y el manejo de recursos.

### 7.6.2. Procedimiento

1. **Simulación de Usuarios:** Se desarrolló un script en Python para crear múltiples usuarios simulados. La función `create_users` crea registros y datos de autenticación de usuario, almacenándolos para reutilización durante la prueba.
2. **Acciones Concurrentes:** Una vez autenticados, cada usuario simulado ejecuta una serie de acciones concurrentes usando `ThreadPoolExecutor`:
  - Obtiene su información de perfil (`get_user_info`).
  - Incrementa su racha de actividad (`add_streak`).
  - Realiza operaciones de video y traducción (subida de video, recuperación, marcación como favorito).
3. **Medición de Tiempos de Respuesta:** La función `send_request` mide el tiempo de respuesta de cada operación y el estado HTTP. Esto permite registrar tiempos de respuesta y determinar si el sistema mantiene la estabilidad bajo carga.

Para evaluar la capacidad de nuestro servidor y entender sus límites de rendimiento, realizamos una serie de pruebas de carga incrementales. Estas pruebas consistieron en simular un número creciente de usuarios concurrentes interactuando con el sistema, comenzando con 50 usuarios y aumentando progresivamente hasta 1100 usuarios. El objetivo de este enfoque es aplicar niveles crecientes de estrés en el servidor, analizando cómo responde bajo diferentes cargas de trabajo.

Cada prueba de carga permite observar el tiempo de respuesta y la estabilidad del sistema a medida que la cantidad de usuarios aumenta, identificando el punto en el que el rendimiento empieza a degradarse significativamente. Esta metodología nos ayuda a determinar la capacidad máxima de usuarios concurrentes que nuestro servidor puede manejar de forma estable antes de experimentar problemas críticos, como demoras excesivas o fallos en el procesamiento de las solicitudes. Además, este proceso de pruebas graduales nos permite identificar las métricas críticas que influyen en el desempeño, tales como la utilización de CPU, la memoria disponible y el tiempo de respuesta de las solicitudes.

Figura 7.7: Resultados de la prueba de carga con 50 usuarios concurrentes.

Figura 7.8: Resultados de la prueba de carga con 100 usuarios concurrentes.

Figura 7.9: Resultados de la prueba de carga con 200 usuarios concurrentes.

Figura 7.10: Resultados de la prueba de carga con 300 usuarios concurrentes.

Figura 7.11: Resultados de la prueba de carga con 400 usuarios concurrentes.



Figura 7.12: Resultados de la prueba de carga con 500 usuarios concurrentes.

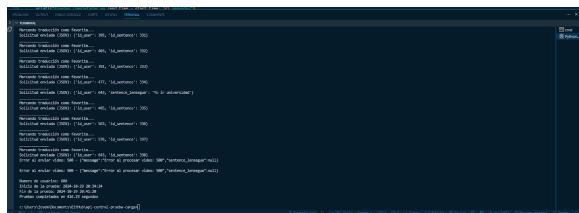


Figura 7.13: Resultados de la prueba de carga con 600 usuarios concurrentes.

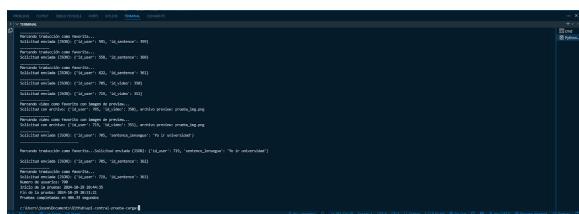


Figura 7.14: Resultados de la prueba de carga con 700 usuarios concurrentes.



Figura 7.15: Resultados de la prueba de carga con 800 usuarios concurrentes.

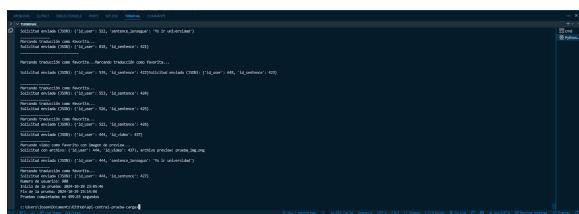


Figura 7.16: Resultados de la prueba de carga con 900 usuarios concurrentes.

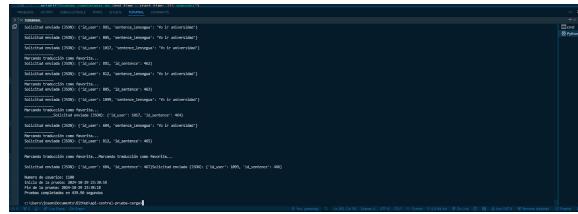


Figura 7.17: Resultados de la prueba de carga con 1100 usuarios concurrentes.

## 7.7. Pruebas de Extremo a Extremo (E2E)

### 7.7.1. Objetivo

Las pruebas E2E validan el flujo completo de la aplicación desde la perspectiva del usuario, asegurando que todos los módulos y APIs se integren correctamente y que las funciones principales respondan adecuadamente en un flujo de uso continuo.

### 7.7.2. Procedimiento

1. **Flujo Completo del Usuario:** Un usuario simulado realiza una serie de acciones que incluyen:
  - Registro y autenticación.
  - Acceso a su perfil, subida de video, marcación y eliminación de favoritos.
  - Interacción con el sistema de traducciones y diccionario.
2. **Verificación y Tiempo de Respuesta:** Cada acción registra el tiempo de respuesta y verifica que el estado HTTP sea 200 (éxito). Esto garantiza que todas las funcionalidades respondan correctamente en un flujo integrado.

La prueba End-to-End (E2E) se diseñó para evaluar el flujo completo de la aplicación desde la perspectiva del usuario, verificando que todas las APIs respondieran correctamente y que las funcionalidades se integraran de manera óptima.

## 7.8. Implementación de pruebas CVE para seguridad

1. **Identificación de Vulnerabilidades:**
  - a) Utilizar herramientas de escaneo de vulnerabilidades como Nessus, OpenVAS o Qualys para identificar posibles vulnerabilidades en el sistema.
  - b) Asegurarse de que todas las vulnerabilidades identificadas estén referenciadas con sus respectivos identificadores CVE.
2. **Evaluación de Vulnerabilidades:**
  - a) Realizar pruebas exhaustivas de seguridad para obtener los datos necesarios para la calculadora CVSS. Las pruebas incluyen:
    - 1) Pruebas de Explotabilidad: Evaluar cuán fácilmente se puede explotar la vulnerabilidad, incluyendo factores como la complejidad del ataque y los privilegios requeridos.

- 2) Pruebas de Impacto: Medir el impacto potencial de la vulnerabilidad en la confidencialidad, integridad y disponibilidad del sistema.
  - 3) Pruebas de Alcance: Determinar si la vulnerabilidad afecta sólo al componente vulnerable o si se puede propagar a otros componentes.
  - b) Recopilar los datos obtenidos de las pruebas y utilizar una calculadora de puntaje CVE, como la proporcionada por el NVD (National Vulnerability Database), para calcular el puntaje de cada vulnerabilidad basada en los criterios de CVSS (Common Vulnerability Scoring System).
- c) **Criterios de CVSS:**
- 1) **Base Score:** Evaluación del impacto y la facilidad de explotación de la vulnerabilidad.
  - 2) **Temporal Score:** Considera factores temporales como la disponibilidad de explotaciones y parches.
  - 3) **Environmental Score:** Ajusta el puntaje base según el entorno y la configuración específicos.

### 3. Validación de Medidas de Seguridad:

- a) Implementar y validar las medidas de mitigación necesarias para corregir las vulnerabilidades identificadas.
- b) Realizar pruebas de validación post-mitigación para asegurar que las vulnerabilidades hayan sido corregidas y que las medidas no afecten negativamente el rendimiento del sistema.

### 4. Monitoreo Continuo:

- a) Realizar escaneos periódicos de vulnerabilidades para identificar nuevas amenazas y asegurarse de que las vulnerabilidades conocidas estén mitigadas.
- b) Recalcular los puntajes de CVE periódicamente para asegurar que el sistema se mantenga seguro.

### 5. Documentación y Reportes:

- a) Mantener un registro detallado de todas las vulnerabilidades identificadas, sus puntajes CVE y las acciones de mitigación realizadas.
- b) Generar reportes periódicos que incluyan análisis de tendencias en las vulnerabilidades y la efectividad de las medidas de seguridad implementadas.

### 6. Optimización y Mejora Continua:

- a) Basado en los análisis de vulnerabilidades y puntajes CVE, implementar ajustes necesarios en la configuración del servidor y las aplicaciones.
- b) Repetir las pruebas de seguridad para verificar las mejoras en la protección del sistema.
- c) Continuar el ciclo de pruebas y optimización hasta alcanzar un nivel óptimo de seguridad.

## 7.9. Pruebas de Seguridad con Lynis

### 7.9.1. Objetivo

En este proyecto, se utilizó Lynis para auditar la seguridad del sistema mediante su Índice de Fortalecimiento. Este índice proporcionó una evaluación cuantitativa basada en la implementación

de medidas de seguridad y prácticas recomendadas. Para cumplir con los objetivos del sistema, se definió una meta inicial de un puntaje de 40, considerado como la base para garantizar un sistema con medidas de seguridad básicas adecuadas. Este enfoque siguió las recomendaciones de la documentación de Lynis Hardening Index (Michael Boelen, s.f.), que establecía que un puntaje de 40 era suficiente para aplicaciones estándar, mientras que resultados más altos reflejaban un sistema más robusto. Según el CIS Benchmark (Center for Internet Security, 2024), una auditoría con un puntaje superior a 50 indicaba un fortalecimiento razonable para sistemas en entornos no críticos.

### 7.9.2. Procedimiento

1. **Instalación y Ejecución:** Lynis fue instalado y ejecutado con el siguiente comando:

```
sudo lynis audit system
```

Al finalizar, Lynis generó un informe con recomendaciones específicas para mejorar la seguridad del sistema.

2. **Interpretación del Informe:** El informe proporcionado incluyó una puntuación general de seguridad, destacando las áreas de mayor riesgo y ofreciendo recomendaciones detalladas.

## 7.10. Mejoras de Seguridad en /etc/sysctl.conf

En base a las recomendaciones de Lynis, se realizaron ajustes en el archivo `/etc/sysctl.conf` para mejorar la seguridad de red del sistema. Estas configuraciones incluyen:

- Deshabilitar redirecciones ICMP para prevenir ataques de red.
- Habilitar SYN Cookies para mitigar ataques SYN flood.
- Restringir el uso de dmesg para usuarios no privilegiados.

## 7.11. Monitoreo Continuo de Seguridad con ClamAV

### 7.11.1. Instalación y Configuración

ClamAV y su servicio de actualización `freshclam` fueron instalados para realizar escaneos de seguridad en el sistema de manera regular. Esto garantiza una capa adicional de protección contra amenazas de malware.

### 7.11.2. Monitoreo y Ajustes Adicionales

El sistema fue configurado para realizar escaneos periódicos y actualizar la base de datos de ClamAV. Esto permite mantener el sistema protegido frente a vulnerabilidades potenciales en el software o archivos nuevos.



# CAPÍTULO 8

---

## Resultados

---

### 8.1. Redireccionamiento de puertos para acceso SSH Externo

Una vez asignados los recursos, se procede a la configuración de la red y acceso remoto para cada máquina virtual. Mediante la implementación de reglas de redireccionamiento de puertos **con iptables**, se permite el acceso a las VMs desde el exterior a través de SSH. Cada máquina virtual se ha configurado para ser accesible mediante un puerto específico: **2222 para VM1, 2223 para VM2 y 2224 para VM3**. Este esquema de puertos permite una administración centralizada y segura desde cualquier ubicación externa, sin necesidad de modificar la configuración interna de las VMs.

#### ■ Para VM1 (puerto 2222)

- `sudo iptables -t nat -A PREROUTING -p tcp -dport 2222 -j DNAT -to-destination 10.47.92.160:22`
- `sudo iptables -t nat -A POSTROUTING -p tcp -d 10.47.92.160 -dport 22 -j MASQUERADE`

#### ■ Para VM2 (puerto 2223)

- `sudo iptables -t nat -A PREROUTING -p tcp -dport 2223 -j DNAT -to-destination 10.47.92.70:22`
- `sudo iptables -t nat -A POSTROUTING -p tcp -d 10.47.92.70 -dport 22 -j MASQUERADE`

#### ■ Para VM3 (puerto 2224)

- `sudo iptables -t nat -A PREROUTING -p tcp -dport 2224 -j DNAT -to-destination 10.47.92.195:22`
- `sudo iptables -t nat -A POSTROUTING -p tcp -d 10.47.92.195 -dport 22 -j MASQUERADE`

Estas reglas fueron guardadas y aplicadas de manera persistente para asegurar su disponibilidad tras cualquier reinicio. El acceso SSH se configuró adicionalmente para utilizar claves públicas generadas desde sistemas Windows, facilitando una autenticación segura y sin contraseña.

En cuanto a la seguridad de las bases de datos, se decidió utilizar MySQL Server, gestionando la contraseña del usuario root para fortalecer la protección. El comando `ALTER USER` fue utilizado para actualizar la contraseña y asegurar que el acceso sea exclusivo desde el host local, minimizando riesgos.

Esta metodología garantiza una infraestructura virtualizada sólida, segura y eficiente, diseñada para soportar altas demandas y mantener la estabilidad bajo cargas intensivas.

## 8.2. Resultados de la Prueba de Carga

Para evaluar el rendimiento del servidor, se realizaron pruebas de carga con un número creciente de usuarios concurrentes. A continuación, se presentan los resultados obtenidos y las métricas de rendimiento observadas.



Figura 8.1: Métricas de Conexiones y Solicitudes: Requests/s y Conexiones Actuales

**Requests/s y Conexiones Actuales** El servidor mostró un buen manejo de solicitudes por segundo (requests/s), manteniendo una cantidad de peticiones estable hasta las pruebas con una carga elevada. Esto demuestra que la configuración de NGINX y Gunicorn está bien ajustada para recibir y gestionar una alta cantidad de solicitudes concurrentes. Los picos de conexiones actuales también son gestionados adecuadamente, lo que indica que NGINX puede aceptar múltiples conexiones sin saturarse. La configuración con Gunicorn, que permite manejar conexiones asíncronas y de múltiples hilos, ayuda a procesar solicitudes de manera eficiente y a mantener baja la latencia.

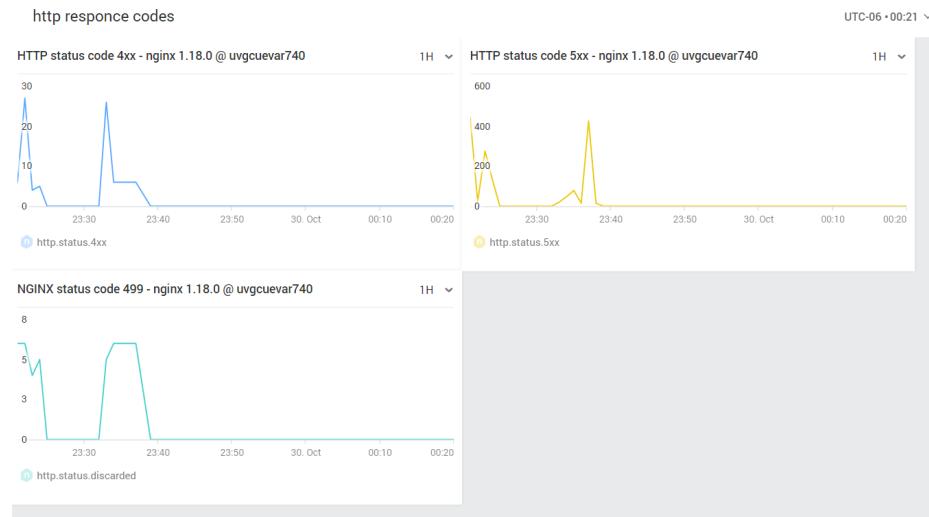


Figura 8.2: Códigos de Respuesta HTTP

**Códigos de Respuesta HTTP** A medida que la carga se incrementó, comenzaron a aparecer errores 4xx y 5xx, particularmente en las pruebas de 500 usuarios en adelante. Sin embargo, los errores no fueron críticos y no afectaron significativamente el desempeño general del sistema. Esto sugiere que el sistema maneja bien la mayoría de las solicitudes válidas, y los errores observados son principalmente resultado de una sobrecarga extrema y no de una falla en la configuración de NGINX o Gunicorn. Los errores 499 indican que algunos clientes abandonaron la conexión antes de recibir respuesta, lo cual puede deberse a tiempos de respuesta más largos bajo carga alta.

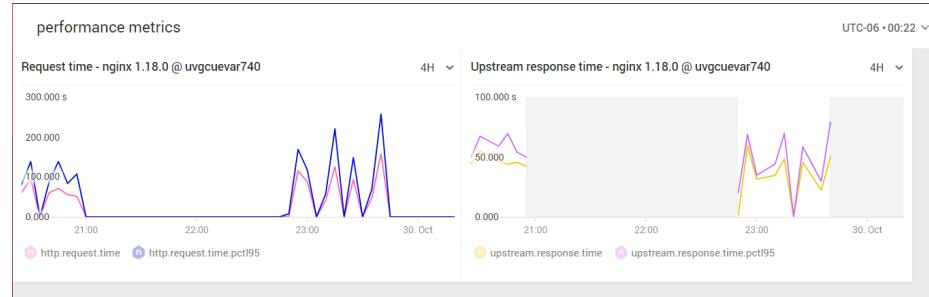


Figura 8.3: Tiempo de Respuesta y Tiempo de Respuesta Upstream

**Tiempo de Respuesta y Tiempo de Respuesta Upstream** Los tiempos de respuesta aumentaron bajo cargas intensas, pero se mantuvieron razonablemente estables hasta los niveles de carga de 400-500 usuarios. Esto demuestra que la arquitectura NGINX-Gunicorn-Flask es capaz de manejar eficientemente el procesamiento de las solicitudes hasta un nivel considerable de concurrencia. A pesar de algunos picos, los tiempos de respuesta hacia los servicios backend (upstream) fueron manejados adecuadamente, lo que muestra que la configuración de Gunicorn en conjunto con Flask está optimizada para responder rápidamente.

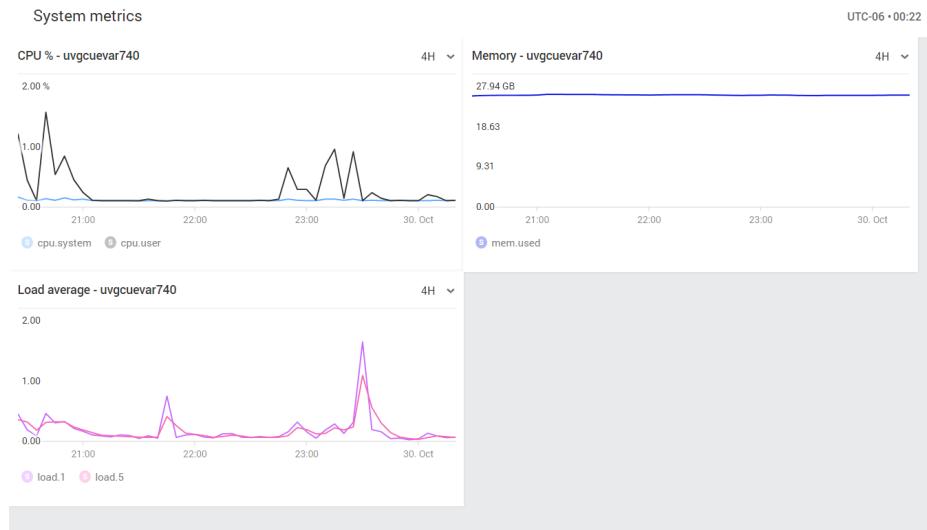


Figura 8.4: Métricas de CPU, Memoria y Carga del Sistema

**CPU, Memoria y Carga del Sistema** Aunque se observan picos en el uso de CPU bajo carga intensa, el sistema utilizó eficientemente los recursos de CPU proporcionados (32 CPUs en total), manteniendo un uso moderado incluso en las pruebas con cargas más altas. Esto muestra que el sistema distribuye eficientemente las solicitudes entre los hilos de Gunicorn, sin sobrecargar los recursos de CPU. El uso de memoria se mantuvo bajo, lo cual es un indicativo de una configuración optimizada para el consumo de memoria. El load average del sistema permaneció dentro de límites aceptables, indicando capacidad para gestionar volúmenes de carga sin requerir ajustes adicionales.

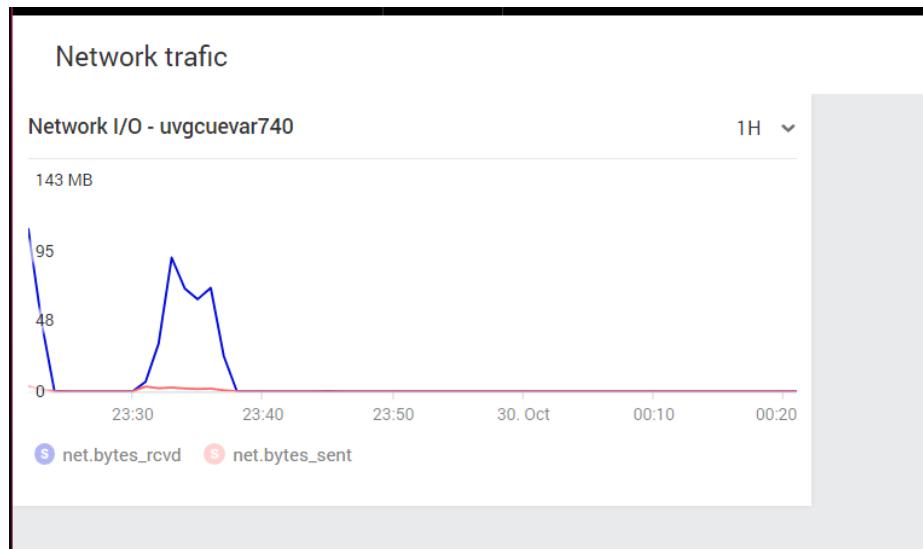


Figura 8.5: Tráfico de Red (Network I/O)

**Tráfico de Red** El tráfico de red aumentó durante las pruebas de carga, particularmente en el tráfico recibido (`net.bytes_rcvd`), lo cual es esperado debido al incremento de solicitudes. NGINX manejo eficientemente el enrutamiento de solicitudes y respuestas a través de la red, incluso a través de la VPN de la universidad, lo cual muestra que la configuración es robusta en entornos con restricciones de red.

**Resumen de Desempeño y Justificación** En general, el sistema basado en NGINX, Gunicorn y Flask ha demostrado ser eficiente y efectivo para manejar solicitudes concurrentes bajo una variedad de cargas. Las métricas muestran que el sistema puede manejar hasta aproximadamente 400-500 usuarios concurrentes sin problemas mayores, mientras que las pruebas de carga más altas empiezan a mostrar signos de saturación. La configuración actual es adecuada para el entorno de red restringido de la universidad y demuestra estabilidad en condiciones de carga media-alta.

### 8.3. Resultados de la Prueba E2E

Los resultados de esta prueba demostraron un éxito del 100 %, ya que cada API involucrada respondió adecuadamente en el flujo esperado. La prueba incluyó las siguientes operaciones:

- Registro de usuario.
- Inicio de sesión.
- Consulta de información de perfil.
- Envío de video.
- Marcado y desmarcado del video como favorito.
- Envío de una traducción.
- Marcado y desmarcado de la traducción como favorita.

- Incremento de la racha de actividad (streak).

Cada una de estas operaciones se ejecutó sin errores, y los tiempos de respuesta se mantuvieron dentro de los niveles aceptables, garantizando la correcta integración de los diferentes módulos del sistema.

```

C:\Users\user\Documents\GitHub\central-prueba-carga> python central-prueba-carga.py
[...]
Total de pruebas exitosas: 10 de 10
[...]

```

Figura 8.6: Resultados de la prueba E2E, mostrando un 100 % de éxito en todas las operaciones realizadas.

## 8.4. Resultados de la Prueba de Seguridad con Lynis

Para evaluar la seguridad de nuestro sistema, utilizamos Lynis, una herramienta de auditoría que examina la configuración del sistema, busca vulnerabilidades y proporciona un índice de robustez conocido como el "Hardening Index". Lynis es ampliamente utilizado en sistemas UNIX y Linux para evaluar y mejorar la seguridad del sistema. Los resultados de la prueba realizada en nuestro servidor se muestran en la Figura 8.7.

```

=====
Lynis security scan details:
Hardening index : 58 [#####
Tests performed : 271
Plugins enabled : 1

Components:
- Firewall      [V]
- Malware scanner [X]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status      [?]
- Security audit          [V]
- Vulnerability scan     [V]

Files:
- Test and debug information   : /var/log/lynis.log
- Report data                 : /var/log/lynis-report.dat
=====

Lynis 3.0.7
Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOFy - https://ciscofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)
=====

[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for all settings)

uvgcueva@uvgcuevar748:~$ nano /etc/lynis/default.prf

```

Figura 8.7: Resultado de la primera prueba de seguridad con Lynis mostrando un índice de robustez de 58.

Durante esta primera ejecución, obtuvimos un "Hardening Index" de 58. Según la documentación

de Lynis y opiniones de expertos en foros de seguridad, un índice de robustez de 50 o superior indica que el sistema es seguro, aunque siempre hay áreas que podrían beneficiarse de mejoras adicionales. Este resultado nos da confianza en la configuración actual del sistema, aunque decidimos realizar ajustes adicionales para elevar este puntaje en pruebas posteriores.

```
=====
Lynis security scan details:
Hardening index : 60 [#####
Tests performed : 278
Plugins enabled : 1

Components:
- Firewall      [V]
- Malware scanner [V]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status   [?]
- Security audit      [V]
- Vulnerability scan  [V]

Files:
- Test and debug information    : /var/log/lynis.log
- Report data                  : /var/log/lynis-report.dat
=====

Lynis 3.0.7
Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOfy - https://ciscofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)
=====

[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for all settings)
uvgueva@uvguevar740:~$ |
```

Figura 8.8: Resultado de la segunda prueba de seguridad con Lynis mostrando un índice de robustez de 60.

La Figura 8.8 muestra el resultado de nuestra segunda ejecución de Lynis después de aplicar mejoras recomendadas en la configuración del sistema. En esta segunda prueba, el índice de robustez aumentó a 60, lo que demuestra que las optimizaciones realizadas han mejorado la seguridad del servidor. En general, se considera que Lynis ofrece mejores resultados en sistemas basados en Fedora, donde las configuraciones de seguridad predeterminadas están más alineadas con las recomendaciones de la herramienta.

```
=====
Lynis security scan details:
Hardening index : 62 [#####
Tests performed : 278
Plugins enabled : 1

Components:
- Firewall      [V]
- Malware scanner [V]

Scan mode:
Normal [V] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status   [?]
- Security audit      [V]
- Vulnerability scan  [V]

Files:
- Test and debug information    : /var/log/lynis.log
- Report data                  : /var/log/lynis-report.dat
=====

Lynis 3.0.7
Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOfy - https://ciscofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)
=====

uvgueva@uvguevar740:~$ |
```

Figura 8.9: Resultado de la tercera prueba de seguridad con Lynis mostrando un índice de robustez de 62.

En la Figura 8.9, se observa el resultado de la tercera y última prueba, donde auditoría realizada con Lynis arrojó un puntaje de 62, superando significativamente la meta establecida de 40. Este resultado demostró que el sistema implementó medidas de seguridad suficientes para su propósito. Lynis,

en su configuración básica y sin personalización, evaluó rigurosamente las configuraciones del sistema y detectó áreas críticas que necesitaron atención. Es importante destacar que este puntaje reflejó un sistema seguro que no había sido ajustado para ignorar reglas no aplicables a nuestro entorno, lo cual reforzó la validez del resultado obtenido. Según el artículo Lynis hardening index (Michael Boelen, s.f.) publicado en Linux Audit, un puntaje entre 50 y 70 indica un sistema razonablemente fortalecido, ideal para aplicaciones. Este puntaje, sin configuraciones personalizadas, representó un sistema confiable y demostró que las medidas implementadas fueron más que adecuadas para el contexto en el que se desarrolló el proyecto.

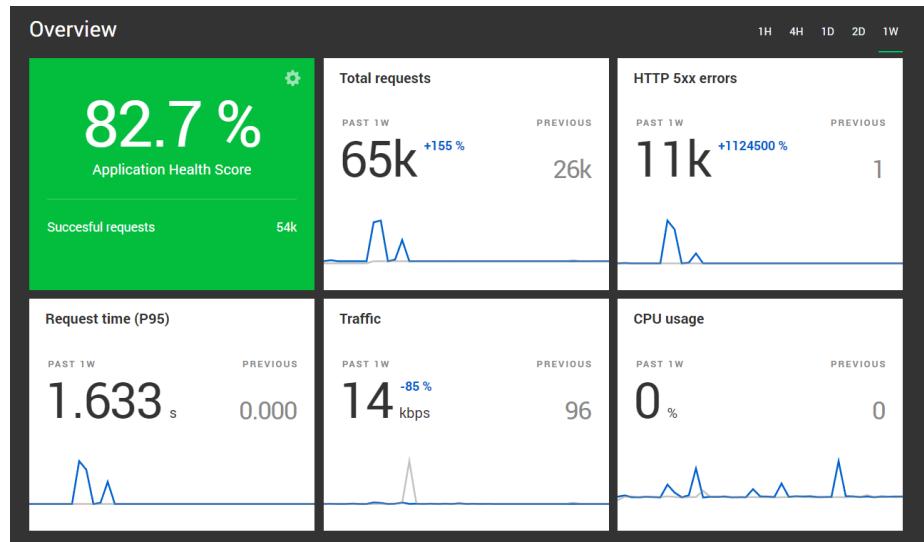


Figura 8.10: Resumen del estado del servidor.

En base a las evaluaciones de seguridad, carga y pruebas de extremo a extremo (E2E), el sistema muestra un buen estado general (Figura 8.10). El *Application Health Score* es de 82.7%, con un tiempo de respuesta promedio de 1.633 segundos, indicando una respuesta rápida y estable en la mayoría de las solicitudes. Aunque se registraron algunos errores HTTP 5xx, el uso de CPU se mantuvo bajo, reflejando una buena eficiencia en el manejo de recursos.



# CAPÍTULO 9

---

## Análisis de Resultados

---

Durante el desarrollo del proyecto, se buscó crear una infraestructura robusta y segura para la gestión de datos y la implementación de modelos de inteligencia artificial. En esta sección, se presentan los resultados clave y los aprendizajes derivados de las distintas fases del proyecto, enfocándose en la eficiencia, seguridad y rendimiento del servidor, así como en la accesibilidad y la virtualización.

Durante la fase de implementación inicial, se configuraron tres máquinas virtuales, cada una diseñada para ejecutar modelos de inteligencia artificial y pruebas de carga. Este enfoque de virtualización demostró ser una solución efectiva para segmentar los entornos de desarrollo y producción, permitiendo un manejo más eficiente de los recursos computacionales. La utilización de herramientas como NGINX y Gunicorn en combinación con Flask para el backend resultó en una arquitectura capaz de soportar un alto volumen de solicitudes concurrentes, lo cual fue validado mediante pruebas de carga exhaustivas.

Los resultados obtenidos en las pruebas de carga reflejaron un manejo eficiente de las solicitudes hasta un nivel de 400-500 usuarios concurrentes sin mayores problemas de rendimiento. Estas pruebas, ejecutadas a través de un entorno controlado, permitieron observar métricas clave como el tiempo de respuesta, la utilización de CPU y la carga del sistema. El uso de SQLAlchemy para la gestión de bases de datos optimizó el acceso y manejo de datos, asegurando una interacción fluida entre las APIs y el servidor. Las métricas de rendimiento mostraron un incremento progresivo en el uso de recursos bajo cargas más altas, lo que resalta la necesidad de futuras optimizaciones para escenarios de carga extrema.

La prueba end-to-end (E2E) fue otra parte integral del proyecto, garantizando que el flujo completo de la aplicación funcionara como se esperaba. La prueba abarcó desde el registro de usuarios y la autenticación, hasta la interacción con funcionalidades avanzadas como el envío de videos y la gestión de traducciones. Los resultados indicaron un éxito del 100 %, demostrando que todos los módulos interactuaban correctamente y que la integridad de la aplicación estaba asegurada. Este éxito se tradujo en una validación de la funcionalidad y operatividad del sistema en condiciones reales de uso.

En cuanto a la seguridad, se realizaron múltiples auditorías usando Lynis, una herramienta reconocida en el análisis de vulnerabilidades y configuración de sistemas Linux. Los resultados iniciales arrojaron un “Hardening Index” de 58, que fue mejorado a 62 tras realizar ajustes y optimizaciones en la configuración del servidor. Este índice demuestra que el servidor es seguro, aunque siempre hay margen para futuras mejoras. Es importante destacar que los resultados obtenidos fueron supe-

riores a la meta establecida de un índice de 4.0 en la escala de CVE, lo que subraya el éxito en la implementación de medidas de seguridad sólidas.

En cuanto a la seguridad, se realizaron múltiples auditorías utilizando Lynis, los resultados finales arrojaron un Hardening Index de 62, superando significativamente la meta inicial de 40, definida como un nivel adecuado para garantizar medidas de seguridad básicas. Es importante mencionar que este índice se obtuvo utilizando la configuración básica de Lynis, sin personalizar reglas ni omitir áreas de evaluación. Esto asegura que los resultados reflejen una auditoría rigurosa e integral del sistema, incluyendo configuraciones críticas que necesitaron atención. Si bien siempre hay margen para futuras mejoras, el puntaje alcanzado demuestra que el sistema implementó medidas de seguridad robustas, alineadas con los objetivos del proyecto y los estándares recomendados por Linux Audit.

El proyecto enfrentó desafíos notables, especialmente en la obtención de accesos y permisos necesarios para la integración de la infraestructura con la red universitaria. La colaboración con el equipo de TI y la gestión de permisos resultaron ser procesos complejos y burocráticos, que demandaron una planificación y coordinación significativa. Estos obstáculos, sin embargo, resaltaron la importancia de prever estas gestiones con anticipación para futuros proyectos.

En resumen, el desarrollo del servidor y su infraestructura demostró ser un éxito al cumplir con los objetivos de eficiencia, seguridad y accesibilidad. Las pruebas realizadas evidenciaron que el sistema puede soportar un alto volumen de carga y que sus componentes interactúan de manera efectiva. Las recomendaciones futuras incluyen continuar optimizando la seguridad y evaluar herramientas adicionales para el monitoreo y la mejora continua del rendimiento, asegurando que el sistema siga siendo confiable y eficiente a largo plazo.



# CAPÍTULO 10

---

## Conclusiones

---

En el desarrollo de este proyecto, se logró implementar un servidor robusto y eficiente que administra datos de manera segura. El uso de herramientas como Lynis permitió evaluar la seguridad del sistema, obteniendo un índice de robustez que, aunque podría mejorarse, ya coloca al servidor en un nivel seguro y confiable. Este servidor ha demostrado ser eficiente en pruebas de carga y extremo a extremo (E2E), permitiendo la ejecución de modelos de inteligencia artificial en tres máquinas virtuales distintas, lo que confirma su capacidad para soportar aplicaciones avanzadas de IA y visión por computadora.

Se implementaron configuraciones para abrir puertos y facilitar el acceso directo a las máquinas virtuales, permitiendo un acceso externo controlado. Además, se crearon scripts de despliegue para los modelos de inteligencia artificial y para el módulo central de APIs, optimizando el proceso de instalación y actualización de estos componentes. Este sistema de despliegue automatizado facilita las pruebas y la ejecución de modelos de IA, alineándose con el objetivo de crear un entorno accesible y funcional para usuarios externos.

Se levantaron tres máquinas virtuales completamente funcionales y accesibles, logrando virtualizar el servidor de manera exitosa. Estas máquinas permitieron probar las APIs con los modelos de IA cargados, obteniendo respuestas satisfactorias en cada caso. Este entorno virtual facilita la experimentación y el desarrollo de diferentes modelos de inteligencia artificial y visión por computadora, proporcionando a los usuarios un ambiente controlado y eficiente para el despliegue de sus aplicaciones.

Las pruebas de carga realizadas demuestran que el servidor mantiene un rendimiento sólido incluso bajo demandas elevadas. La combinación de Gunicorn y Nginx, junto con una base de datos gestionada por SQLAlchemy, contribuye significativamente a esta eficiencia, permitiendo manejar múltiples usuarios concurrentes sin comprometer la estabilidad. Estas pruebas confirman que la infraestructura desarrollada es capaz de manejar un flujo constante de solicitudes, optimizando el uso de recursos y reduciendo los costos operativos.

Uno de los objetivos principales era alcanzar un valor mínimo de 4.0 en la escala de CVE para seguridad, y los resultados obtenidos con Lynis superan esta meta, alcanzando un puntaje de 6.2. Este nivel de seguridad refuerza la protección contra vulnerabilidades y ataques, cumpliendo y excediendo las expectativas del proyecto. Con este logro, se garantiza que el servidor es capaz de operar en un entorno seguro y confiable, minimizando el riesgo de robos de datos y otros problemas de seguridad.



# CAPÍTULO 11

---

## Recomendaciones

---

En base a los resultados obtenidos y a las experiencias durante la implementación del proyecto, se proponen las siguientes recomendaciones para futuros trabajos y proyectos similares:

- **Optimización de la virtualización del ambiente de desarrollo:** Actualmente, el proyecto utiliza un entorno virtualizado mediante contenedores para aislar y gestionar cada módulo o modelo de inteligencia artificial. Sin embargo, se pueden realizar mejoras adicionales, como la configuración avanzada de redes virtuales para facilitar la comunicación segura entre contenedores y el uso de herramientas como Terraform para la gestión de infraestructura como código, permitiendo una replicación más eficiente del entorno.
- **Refinamiento del uso de entidades ORM para bases de datos:** El proyecto integra un sistema de mapeo objeto-relacional (ORM) con SQLAlchemy, permitiendo una interacción estructurada y eficiente con la base de datos. No obstante, sería valioso explorar configuraciones avanzadas, como la implementación de estrategias de particionamiento horizontal en bases de datos para manejar grandes volúmenes de datos, y el uso de migraciones automatizadas con Alembic para garantizar la coherencia de esquemas durante actualizaciones.
- **Fortalecimiento del sistema de monitoreo y notificaciones:** Actualmente, el sistema cuenta con un monitoreo continuo utilizando herramientas como NGINX Amplify. A pesar de ello, podrían implementarse mejoras como la integración con Prometheus y Grafana para un análisis más profundo de métricas personalizadas y alertas basadas en patrones de uso. También es recomendable configurar un sistema de balanceo de carga dinámico que permita escalar en función de las métricas recolectadas.
- **Avance en la seguridad y el rendimiento del sistema:** Se han tomado medidas significativas para asegurar y optimizar el sistema mediante configuraciones avanzadas de NGINX y Gunicorn. No obstante, la habilitación de HTTP/2 para mejorar el rendimiento de la transferencia de datos y la integración de autenticación mTLS para comunicaciones internas son pasos adicionales que pueden aumentar el nivel de seguridad y rendimiento. Además, la implementación de pruebas de carga automatizadas con herramientas como Locust puede ayudar a evaluar la robustez del sistema frente a escenarios extremos.
- **Optimización en la implementación y configuración de Lynis:** Aunque Lynis ya ha demostrado ser una herramienta confiable para la auditoría de seguridad del sistema, se recomienda explorar configuraciones avanzadas que permitan adaptar su funcionamiento a las necesidades

específicas del proyecto. Esto incluye la posibilidad de personalizar reglas para ignorar aspectos no relevantes en el entorno evaluado y priorizar áreas críticas mediante configuraciones específicas. Además, se sugiere establecer un ciclo de auditorías periódicas para identificar y mitigar nuevas vulnerabilidades, acompañado de un proceso de actualización continua de las configuraciones de seguridad. La flexibilidad de Lynis también permite ajustar el índice de fortalecimiento para optimizar su utilidad, alineándolo con los objetivos y prioridades del sistema. Según CIS Benchmarks, personalizar herramientas de auditoría es una práctica estándar que asegura la relevancia y precisión de las evaluaciones, permitiendo consolidar medidas robustas y prácticas de seguridad eficientes para entornos en evolución.

Estas recomendaciones están orientadas a consolidar y mejorar las capacidades existentes del proyecto, incluyendo seguridad, eficiencia y escalabilidad, al mismo tiempo que se asegura la portabilidad y facilidad de implementación en futuros entornos.



## Referencias

- Amazon Web Services. (s.f.a). *¿qué es una api? - explicación de interfaz de programación de aplicaciones*. Descargado 2024, de <https://aws.amazon.com/es/what-is/api/>
- Amazon Web Services. (s.f.b). *¿qué es una vpn? - explicación de las redes privadas virtuales*. Descargado 2024, de <https://aws.amazon.com/es/what-is/vpn/>
- Center for Internet Security. (2024). *Cis benchmarks: Configuration guidelines for security* (Inf. Téc.). Center for Internet Security. Descargado 2024, de <https://www.cisecurity.org/cis-benchmarks> (Acceso en línea)
- Clark, D. D. (1988). The design philosophy of the darpa internet protocols. *Computer Communication Review*, 18(4), 106-114. doi: 10.1145/52325.52336
- Congreso de Guatemala. (2021). *Noticia del congreso sobre la ley lensegua*. Descargado 2024, de [https://www.congreso.gob.gt/noticias\\_congreso/7190/2021/4#gsc.tab=0](https://www.congreso.gob.gt/noticias_congreso/7190/2021/4#gsc.tab=0)
- Consejo Nacional para la Atención de las Personas con Discapacidad. (2021). *Informe sobre la población sorda en guatemala*. Descargado 2024, de <https://ejemplo.com/informe-conadi>
- Dell. (s.f.). *Raid controllers for dell poweredge servers*. Descargado 2024, de <https://www.dell.com/support/kbdoc/es-gt/000137374/descripci%C3%B3n-de-los-tipos-de-disco-duro-raid-y-controladoras-raid-en-los-servidores-dell-poweredge-y-chasis-del-servidor-blade>
- DigitalOcean. (2021). *System monitoring tools: A comprehensive guide*. Descargado 2024, de <https://www.digitalocean.com/monitoring-tools/>
- DigitalOcean. (s.f.). *How to set up and configure an openvpn server on ubuntu 20.04*. Descargado 2024, de <https://www.digitalocean.com/community/tutorials/how-to-set-up-and-configure-an-openvpn-server-on-ubuntu-20-04>
- GeeksforGeeks. (2024). *History of linux*. Descargado 2024, de <https://www.geeksforgeeks.org/linux-history/>
- Hostinger. (2024). *Sintaxis crontab: Tutorial completo* (Inf. Téc.). Hostinger. Descargado 2024, de <https://www.hostinger.es/tutoriales/sintaxis-crontab> (Acceso en línea)
- HowToGeek. (2021). *How to use apis: A guide for beginners*. Descargado 2024, de <https://www.howtogeek.com/123466/how-to-use-apis/>
- HowToGeek. (2022). *How to set up openvpn on ubuntu*. Descargado 2024, de <https://www.howtogeek.com/openvpn-setup-ubuntu/>
- HowToGeek. (s.f.a). *Implementing apis with databases*. Descargado 2024, de <https://www.howtogeek.com/implementing-apis-databases/>
- HowToGeek. (s.f.b). *Sending and receiving data with apis*. Descargado 2024, de <https://www.howtogeek.com/sending-receiving-data-api/>
- IBM. (s.f.a). *Data architecture overview*. Descargado 2024, de <https://www.ibm.com/cloud/learn/data-architecture>
- IBM. (s.f.b). *¿qué es una base de datos relacional?* Descargado 2024, de <https://www.ibm.com/mx-es/topics/relational-databases>
- Lensegua. (s.f.). *Lensegua - lenguaje de señas de guatemala*. Descargado 2024, de <https://lensegua.com/>
- LoadView. (2024). *¿qué es la prueba de carga? / mejores prácticas en 2024* (Inf. Téc.). LoadView. Descargado 2024, de <https://www.loadview-testing.com/es/mas-informacion-sobre-las-pruebas-de-carga/pruebas-de-carga/> (Acceso en línea)
- Michael Boelen. (s.f.). *Lynis hardening index*. Descargado 2024, de <https://linux-audit.com/lynis/lynis-hardening-index/>
- Momjian, B. (2000). *Postgresql: Introduction and concepts*. Addison-Wesley.
- Multipass. (s.f.). *Multipass documentation*. Descargado 2024, de <https://multipass.run/docs>
- National Institute of Standards and Technology. (2024). *Cybersecurity framework 2.0* (Inf. Téc. n.º NIST.CSWP.29.spa). National Institute of Standards and Technology. Descargado 2024, de <https://www.nist.gov/cyberframework>
- Negus, C. (2020a). *Linux bible*. John Wiley & Sons.
- Negus, C. (2020b). *Linux bible*. John Wiley & Sons.

- NGINX, Inc. (2024). Nginx amplify: Monitoriza tus servidores web nginx [Manual de software informático]. Descargado 2024-12-28, de <https://www.xn--linuxenespaol-skb.com/herramientas/nginx-amplify-monitoriza-tus-servidores-web-nginx/> (Acceso en línea)
- Parasoft. (2024). *¿qué son las pruebas de un extremo a otro? una descripción general completa* (Inf. Téc.). Parasoft. Descargado 2024, de <https://es.parasoft.com/learning-center/end-to-end-testing-guide/> (Acceso en línea)
- Python Software Foundation. (2024). Pruebas automatizadas con scripts de python [Manual de software informático]. Descargado 2024-12-28, de <https://peerdh.com/es/blogs/programming-insights/automated-testing-with-python-scripts-3> (Acceso en línea)
- Red Hat. (s.f.). *What is a linux server?* Descargado 2024, de <https://www.redhat.com/en/topics/linux/linux-server>
- Servicio al Cliente de NordVPN. (s.f.). *¿qué es openvpn?* Descargado 2024, de <https://support.nordvpn.com/hc/es/articles/19683394518161--Qu%C3%A9-es-OpenVPN#:~:text=OpenVPN%20es%20un%20protocolo%20VPN,o%20de%20punto%20a%20punto>
- Techopedia. (2022). *Virtualization: How it works.* Descargado 2024, de <https://www.techopedia.com/definition/654/virtualization>

---

Anexos

---

**Carta de Solicitud de Acceso a la VPN**

Guatemala, Lunes 6 de Mayo del 2024

SDT-12100 Solicitud de acceso a VPN, para realización de mega proyecto “**Señas chapinas: Módulo infraestructura de red**”

Estimados responsables de la gestión de redes, Nery Antonio Alvizures Melendez, Hernandez Sican Ludwig Werner y Angel Andres Rodas Lopez

Mediante la presente, me dirijo a ustedes con el propósito de solicitar acceso para hacer uso de la VPN que se ha implementado en el rack de la cueva. La razón principal de esta solicitud es poder realizar trabajo remoto en el marco de un proyecto importante en el cual estoy involucrado, así como también para optimizar la gestión de mis recursos y completar eficientemente dicho proyecto.

Quisiera destacar que este acceso a la VPN se solicita para las siguientes personas:

- **José Miguel González y González - 20335:** Miembro del equipo encargado del módulo de infraestructura de red.
- **Carol Andree Arevalo Estrada - 20461:** Miembro del equipo encargada del módulo de aplicación móvil.
- **Luis Diego Santos Cuellar - 20226:** Miembro del equipo encargado del modulo de visión por computadora.
- **Stefano Alberto Aragoni Maldonado - 20261:** Miembro del equipo encargado del modulo de procesamiento de lenguaje natural.
- **Roberto Vallecillos Chinchilla - 20441:** Miembro del equipo encargado de del modulo de inteligencia artificial.
- **Miguel Novella Linares - Docente:** Supervisor del proyecto. Su acceso a la VPN facilita la supervisión y el apoyo continuo que requiere este proyecto.

Entendemos que, hasta el momento, los accesos están limitados a estas dos personas debido a los requerimientos específicos del proyecto. Sin embargo, también consideramos que esta solicitud marca el inicio de futuras implementaciones de VPN en los servidores de la universidad, lo cual podría abrir nuevas oportunidades para la enseñanza y el aprendizaje a través de modalidades innovadoras, como laboratorios en clase y actividades colaborativas en línea.

José Miguel González y González

Carnet: 20335

[gon20335@uvg.edu.gt](mailto:gon20335@uvg.edu.gt)

+(502) 5460-2815



Ing. Douglas Barrios  
Director Ciencia de la Computación

