UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA
CARRERA DE COMPUTACIÓN

TEMA:

Desarrollo de un Prototipo de Framework que integre escaneo activo, inteligencia de exploit-DB y análisis de contexto que priorice y optimice la gestión de vulnerabilidades en servidores virtualizados LINUX a ser aplicado en una empresa de telecomunicaciones de la ciudad de Guayaquil.

AUTORES:
Baque Ortiz, Lizbeth Melissa
Ruiz Panezo, James Jordy

Proyecto de tecnología de información previo a la obtención del título de INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

TUTOR:
Yong Yong, Byron Severo

Guayaquil, Ecuador
27 de febrero de 2026

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA
CARRERA DE COMPUTACIÓN

CERTIFICACIÓN

Certificamos que el presente Proyecto de tecnología de información fue realizado en su totalidad por **Baque Ortiz, Lizbeth Melissa & Ruiz Panezo, James Jordy**, como requerimiento para la obtención del título de **Ingeniero en Ciencias de la Computación**.

TUTOR

f._____

Yong Yong, Byron Severo

Guayaquil, a los 27 días del mes de febrero del año 2026

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA
CARRERA DE COMPUTACIÓN

DECLARACIÓN DE RESPONSABILIDAD

Nosotros, **Baque Ortiz, Lizbeth Melissa**
Ruiz Panezo, James Jordy

DECLARAMOS QUE:

El proyecto de tecnología de información, **Desarrollo de un Prototipo de Framework que integre escaneo activo, inteligencia de exploit-DB y análisis de contexto que priorice y optimice la gestión de vulnerabilidades en servidores virtualizados LINUX a ser aplicado en una empresa de telecomunicaciones de la ciudad de Guayaquil**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referenciaso bibliografías. Consecuentemente este trabajo es de nuestra total autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 27 días del mes de febrero del año 2026

LOS AUTORES

f._____
Baque Ortiz, Lizbeth Melissa

f._____
Ruiz Panezo, James Jordy

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA
CARRERA DE COMPUTACIÓN

AUTORIZACIÓN

Nosotros, **Baque Ortiz, Lizbeth Melissa**
Ruiz Panezo, James Jordy

Autorizamos a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del proyecto de tecnología de información, **Desarrollo de un Prototipo de Framework que integre escaneo activo, inteligencia de exploit-DB y análisis de contexto que priorice y optimice la gestión de vulnerabilidades en servidores virtualizados LINUX a ser aplicado en una empresa de telecomunicaciones de la ciudad de Guayaquil**., cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y total autoría.

Guayaquil, a los 27 días del mes de febrero del año 2026

LOS AUTORES:

f._____
Baque Ortiz, Lizbeth Melissa

f._____
Ruiz Panezo, James Jordy

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA
CARRERA DE COMPUTACIÓN

REPORTE ANTIPLAGIO

TUTOR

Byron Yong Yong
鄺勇

Firmado digitalmente por BYRON SEVERO YONG YONG
Fecha: 2026.02.06 23:34:06 -05'00'
Versión de Adobe Acrobat Reader: 2025.00 1.21151

f._____

Yong Yong Byron Severo

AGRADECIMIENTO

En primer lugar, estamos profundamente agradecidos con Dios por habernos brindado la fortaleza, la claridad mental y la salud necesarias para superar este desafío académico y culminar con éxito esta etapa crucial de nuestras vidas. Queremos expresar nuestro profundo agradecimiento a nuestras familias, que, con su apoyo constante, comprensión y paciencia lograron que todo fuera posible ante ese anhelo necesario para continuar y culminar esta etapa muy importante en nuestras vidas.

De igual manera, extendemos nuestro sincero reconocimiento a nuestro tutor de tesis, Ing. Byron Yong Yong, por su acompañamiento permanente y sus valiosos consejos, los cuales fueron fundamental para una mejor orientación en el desarrollo de este proyecto y fortalecer nuestra preparación como futuros Ingenieros en Ciencias de la Computación.

Agradecemos de manera especial a la empresa auspiciante, que nos brindó la oportunidad de trabajar en su entorno tecnológico para la validación de este prototipo.

Asimismo, a nuestros compañeros, con quienes compartimos incontables horas de estudio y aprendizaje colectivo, haciendo de este una experiencia más enriquecedora a nivel académica y personal.

Finalmente, expresamos nuestro agradecimiento a los docentes de la carrera, quienes compartieron sus conocimientos y experiencias, para nuestro desarrollo académico y profesional.

DEDICATORIA

Dedico este trabajo a mis amados padres, quienes, con su sacrificio y amor me permitieron cumplir este logro fruto de su siembra en mí. A mi abuelita Mirna, quien sostuvo mi mano con ternura al inicio de este camino, aunque hoy me sonríes desde el cielo, este es tu legado. A mi abuelito Tiburcio, por ser mi refugio, mi fortaleza, y el que me enseñó a seguir adelante con valentía.

Y finalmente, extiendo esta dedicatoria a mi más fiel y noble compañero de cuatro patas, Lucas. Gracias por esas largas madrugadas de estudio donde tu compañía incondicional me recordaron que nunca estaba sola. Fuiste mi terapia silenciosa, mi alegría en el estrés y el abrazo peludo que reiniciaba mi mundo. Gracias por lealtad infinita.

Lizbeth Melissa Baque Ortiz.

Primero que nada agradezco a Dios, quien es el arquitecto de mi destino y con su mano amorosa me ha guiado en cada paso que he dado; muchas gracias por la vida que me has dado y por permitirme estar en este momento de mi vida, rodeado de mis seres queridos y con la satisfacción de haber logrado el más grande de mis sueños en lo profesional.

A mi madre, mi motor y ejemplo de lucha; te dedico este momento madre, porque estuviste desde el principio hasta el final del proceso. Este título de ingeniero es tanto mío como tuyo, porque tú fuiste el aliento en las noches que pasé estudiando y la guía cuando no sabía hacia dónde ir. Gracias por darme la vida; gracias por todo.

A mi abuela, quien con cariño y sabiduría forjó mi carácter; muchas gracias por los grandes valores y principios que me enseñaste para tener un camino trazado. Tu legado se queda en cada uno de mis logros personales y profesionales. Muchas gracias a ustedes dos por creer en mí antes de que yo mismo lo hiciera.

James Jordy Ruiz Panezo

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA
CARRERA DE COMPUTACIÓN

TRIBUNAL DE SUSTENTACIÓN

f._____

ING. ANA CAMACHO CORONEL, Mgs

DIRECTOR DE CARRERA

f._____

ING. ISMAEL SOSA RENDÓN, Mgs

DOCENTE DE LA CARRERA

f._____

ING. EDISON TOALA QUIMI, Mgs

OPONENTE

ÍNDICE GENERAL

INDICE DE IMÁGENES

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

Resumen

La propuesta de nuestro proyecto de titulación se basó en el desarrollo de un framework para la gestión y priorización de vulnerabilidades en los servidores virtualizados Linux, empleado para una empresa de telecomunicaciones de la ciudad de Guayaquil. Debido al progresivo avance de la dependencia tecnológica, se desea robustecer la seguridad para los activos críticos de la organización como su continuidad operativa.

La investigación se enfocó en un análisis descriptivo y aplicado, la cual se realizó bajo el proceso actual de la gestión de vulnerabilidades mediante una metodología mixta, la cual implicó un análisis cualitativo para comprender el contexto operacional y un análisis cuantitativo para observar las métricas de los indicadores de avance. Para el desarrollo tecnológico se aplica una metodología de prototipado que permite el control ordenado de las etapas diseño, implementación, prueba y validación del prototipo.

El estudio inicia con la revisión de antecedentes y conceptos claves en seguridad informática, orientado en la gestión de vulnerabilidades en entornos virtualizados Linux. Se examinan las limitaciones del enfoque actual y se identifican las razones vinculadas a la falta de automatización y priorización contextual adecuada; Se desarrolla una propuesta para un sistema que combina escaneo activo, inteligencia sobre amenazas y análisis contextual con el fin de clasificar las vulnerabilidades de acuerdo con su criticidad y su importancia para la empresa.

El framework propuesto facilita la elaboración de reportes confiables y efectivos, mejora la supervisión operativa de los técnicos y optimiza la utilización de recursos en el campo de la ciberseguridad.

El prototipo presentado resulta para mejorar el manejo de vulnerabilidades en los servidores virtualizados de la compañía de telecomunicaciones, ofreciendo ventajas operativas y estratégicas. Este estudio fundamenta de manera consistente futuras investigaciones y avances en la protección digital empresarial.

Palabras Clave: *virtualización, ciberseguridad, framework, telecomunicaciones.*

Abstract

Our degree project proposal was based on the development of a framework for managing and prioritizing vulnerabilities in virtualized Linux servers, used by a telecommunications company in the city of Guayaquil. Due to the progressive advance of technological dependence, there is a desire to strengthen security for the organization's critical assets, such as its operational continuity.

The research focused on a descriptive and applied analysis, which was carried out under the current vulnerability management process using a mixed methodology, which involved a qualitative analysis to understand the operational context and a quantitative analysis to observe the metrics of the progress indicators. For technological development, a prototyping methodology is applied that allows for the orderly control of the design, implementation, testing, and validation stages of the prototype.

The research begins by reviewing essential background and core concepts in information security, centering on vulnerability management within virtualized Linux environments. It identifies major limitations in the existing method—particularly the lack of automation and proper context-based prioritization—and explains the underlying causes. From there, the study proposes a system that brings together active scanning, threat intelligence integration, and contextual assessment to classify vulnerabilities based on their true criticality and business relevance.

The proposed framework supports the production of dependable, practical reports, gives technicians clearer operational oversight, and helps make more efficient use of cybersecurity resources overall.

In the end, the prototype developed here improves how vulnerabilities are handled on the company's virtualized servers, providing tangible operational gains as well as strategic benefits for the long term. This work lays a solid foundation for continued research and practical advances in enterprise digital protection.

***Keywords**: virtualization, cybersecurity, framework, telecommunications.*

Introducción

En la era digital, las organizaciones dependen de la conectividad y la gestión de grandes volúmenes de datos para su crecimiento y competitividad. No obstante, esta evolución trae consigo un incremento notable en las amenazas cibernéticas que comprometen la integridad, la privacidad y la accesibilidad de la información; las fallas en seguridad cibernética están clasificadas entre los cinco mayores riesgos a nivel mundial, según el Informe de Riesgos Globales del Foro Económico Mundial de 2021, lo que resalta la seriedad y urgencia de la cuestión.

En el contexto teórico para la implementación de sistemas seguros para la protección de datos es muy relevante para una organización; sin embargo, no se cumple con este tipo de sistemas donde sus consecuencias es el aumento de vulnerabilidades frente a ciberataques.

Al considerar este asunto es relevante comprender el desarrollo de alternativas tecnológicas que no solo eviten la divulgación de datos no autorizados, sino que ayuden a la continuidad operativa de la organización y con ello evitar consecuencias que podrían ser pérdidas económicas, desprestigio de la compañía como posibles responsabilidades legales que serían pertinentes para la entidad.

El aporte de este trabajo es significativo para la protección y seguridad de la información en los servidores virtualizados Linux, en el contexto de las telecomunicaciones. Para esto, se revisaron antecedentes e ideas claves del área de ciberseguridad.

También se pudo analizar a profundidad aquellas amenazas y vulnerabilidades que suelen ser más comunes en la organización, resultado de los desafíos de la virtualización Linux. Esta propuesta propone mejorar en la seguridad en ambientes virtuales, como la optimización en la operatividad de la compañía.

CAPÍTULO I: EL PROBLEMA

En la actualidad, las compañías de telecomunicaciones, por ejemplo, SETEL (Servicios de Telecomunicaciones Setel S.A.), se encuentran inmersos en entorno tecnológicos altamente complejos y de forma dinámica, donde cada de sus gestiones son eficientes y seguras en cada componente de sus activos digitales porque con ello se garantiza la continuidad del negocio y la calidad del servicio que se ofrece.

No obstante, el proceso de la gestión y la priorización de cada vulnerabilidad en los entornos de servidores virtualizados Linux presenta deficiencias significativas para la obtención de reportes confiables y oportunos en cada actividad, así como el control de las actividades operativas de cada técnico que se encarga dependiendo de su trabajo de campo.

Esta falta de automatización que existe y la estandarización de estos procesos llevan una dependencia de criterios tanto manuales como la generación de retrasos y posibles riesgos de seguridad que se deben mitigar.

ANÁLISIS DEL PROBLEMA.

Ubicación del Problema en un Contexto

El problema se ubica en una empresa de telecomunicaciones de la ciudad de Guayaquil que depende de servidores virtualizados Linux como parte de su tecnología fundamental. La situación que se comprende es la de reportes inconsistentes y tardíos sobre los estados de las vulnerabilidades, ausencia de una priorización definida de los riesgos y los problemas en la supervisión y regulación de las medidas correctivas llevadas a cabo por los técnicos del departamento.

Los factores subyacentes identificadas en la práctica incluyen la ausencia de una herramienta integrada que pueda combinar el escaneo activo, los análisis contextuales e inteligencia de amenazas por la alta dependencia del factor humano para evaluar y gestionar estas vulnerabilidades.

Este contexto se sitúa en el periodo actual, caracterizada por un aumento de riesgos y un avance continuo de la tecnología.

Las variables principales que dan forma al problema son: la efectividad en la gestión de vulnerabilidades, la automatización de tareas, el acceso a datos

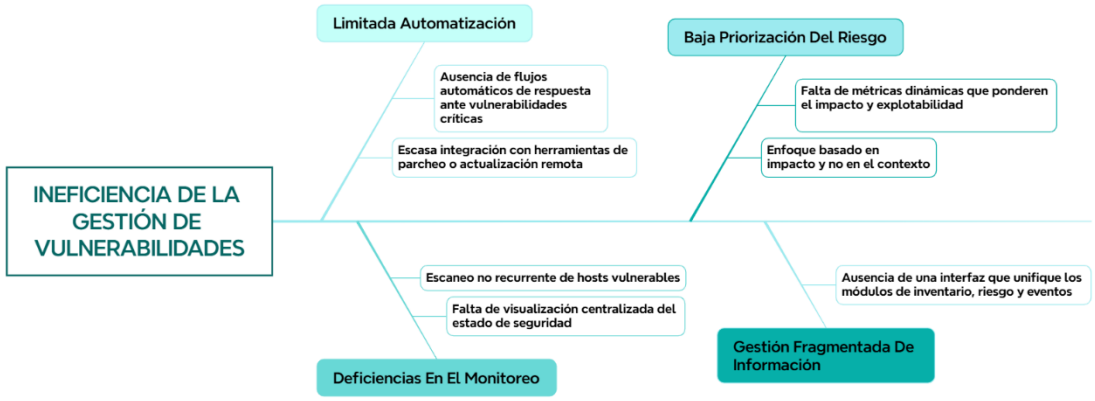
recientes, la precisión en la valoración de riesgos, y la supervisión operativa del equipo técnico.

Causas y Consecuencias del Problema

Entre las causas principales se encuentra la operatividad manual y poco sistematizada del procedimiento de priorización, la dispersión de la información y la limitación de las herramientas tecnológicas adecuadas para una gestión eficiente. La combinación de estos elementos genera consecuencias críticas, entre los que se incluye la divulgación prolongada de las vulnerabilidades de alto riesgo, retrasos en la implementación de parches y soluciones, así como un impacto directo en la continuidad operativa y la reputación de la compañía.

A partir de estos elementos, en la siguiente figura se sintetizan las principales causas que originan la ineficiencia en la gestión de vulnerabilidades, así como la forma en que se relacionan entre sí. Este esquema permite visualizar de manera estructurada cómo contribuyen al problema identificado en la organización.

Ilustración 1
Diagrama de Ishikawa



Nota: Resumen gráfico de la problemática.

Delimitación del Problema

Para delimitar el problema se consideraron cuatro criterios, los cuales se describen a continuación:

Tabla 1
Criterios de Limitación de Problema

#	Criterios	Detalles
1	Campo	Tecnologías de la Información y Seguridad Informática.

#	Criterios	Detalles
2	Área	Gestión de vulnerabilidades y control de operaciones en servidores virtualizados.
3	Aspecto	Automatización y priorización de riesgos en entornos Linux.
4	Tema	Optimización de la gestión operativa y seguridad en servidores virtualizados Linux mediante el desarrollo de un framework integrado en una empresa de telecomunicaciones.

Formulación del Problema

¿Cómo se puede mejorar el proceso de administración y priorización de vulnerabilidades en servidores virtualizados Linux, para obtener reportes precisos y manejar adecuadamente las actividades de los técnicos de campo en una empresa de telecomunicaciones en Guayaquil, teniendo en cuenta la automatización de tareas y el análisis contextual de riesgos?

Evaluación del Problema

De los diez criterios definidos para la evaluación del problema, se seleccionó siete, los cuales se indican en la tabla siguiente:

Tabla 2
Criterios de Evaluación del Problema

#	Criterios	Contexto
1	Delimitado	El problema se encuentra definido dentro del manejo y la clasificación de vulnerabilidades en servidores Linux virtualizados, en una empresa de telecomunicaciones situada en Guayaquil, estableciendo claramente los alcances de tiempo y espacio.
2	Claro	La definición del problema permite comprender con precisión las limitaciones actuales en la generación de reportes y el control operativo de técnicos
3	Relevante	Abordar el inconveniente es esencial para avanzar en la administración de la ciberseguridad, favoreciendo la mejora de recursos y la protección de activos críticos, elementos fundamentales en la industria de las telecomunicaciones.
4	Evidente	Como se indica en la Figura **# 1** las causas que se indican son claras, no requieren mayor detalle para poder definir el problema.
5	Concreto	El problema se presenta de manera precisa y específica en relación con la ineficiencia en la generación de informes y control de actividades técnicas, evitando ambigüedades y generalizaciones.
6	Factible	La empresa dispone de los recursos humanos y tecnológicos requeridos para aplicar soluciones en marcos integrados que optimicen la administración de vulnerabilidades y operaciones.
7	Identifica productos esperados	Se pretende crear un software que facilite la automatización y la priorización de vulnerabilidades, aumentando la precisión de los informes y optimización de la supervisión operativa de los técnicos.

OBJETIVOS

Objetivo General

Desarrollar un prototipo funcional de un framework que integre herramientas de escaneo activo, obtención de inteligencia de exploit-DB y análisis contextual, orientado a la priorización y optimización de la gestión de vulnerabilidades basadas en versionamiento en servidores virtualizados Linux, para su aplicación en una empresa de telecomunicaciones en la ciudad de Guayaquil.

Objetivos Específicos

1. Identificar y evaluar las herramientas más adecuadas de escaneo activo y fuentes de inteligencia de exploits[1], enfocadas en vulnerabilidades basadas en versionamiento para entornos Linux, garantizando su compatibilidad y efectividad para el caso de estudio.
2. Construir el prototipo mediante el uso de las herramientas seleccionadas para la detección y análisis en los servidores virtualizados de la empresa de telecomunicaciones.
3. Integrar el prototipo en una arquitectura tecnológica funcional que incluya componentes de backend para el procesamiento de datos y análisis, y un frontend intuitivo y eficiente para la visualización y gestión de vulnerabilidades, facilitando la toma de decisiones operativas.

ALCANCES DEL PROBLEMA

El proyecto se centra en la identificación, análisis y priorización de vulnerabilidades basadas en versionamiento dentro de servidores virtualizados Linux pertenecientes a una empresa de telecomunicaciones de la ciudad de Guayaquil. Lo que implica evaluar las vulnerabilidades a partir de las versiones específicas de software y componentes presentes en el entorno virtualizado, tales como librerías y servicios levantados en los puertos de tal forma que contribuya a una gestión más precisa, contextualizada y eficiente de los riesgos.

[1] Código, secuencia de datos o técnica que aprovecha las vulnerabilidades o fallas de un sistema informático para acceder a recursos de TI de manera no autorizada.

El desarrollo del prototipo abarcará todo el ciclo del proceso: desde el escaneo activo inicial hasta la priorización y la visualización de las vulnerabilidades detectadas. De esta forma se garantiza que los resultados estén realmente alineados con la operación diaria de la empresa y con las necesidades técnicas concretas de quien auspicia el proyecto.

El framework propuesto incorporará herramientas de escaneo activo, fuentes de inteligencia sobre exploits y un módulo específico de análisis contextual que permita priorizar las vulnerabilidades según su nivel real de criticidad y su importancia para el negocio. La idea central es lograr un modelo de priorización dinámico, que se adapte automáticamente cuando cambien las versiones del software, se modifiquen las configuraciones de red o se actualicen las políticas internas de la empresa en cuanto a aplicación de parches.

En cuanto al alcance, se incluye el diseño y la construcción del prototipo, su integración dentro de una arquitectura tecnológica que contenga los componentes necesarios para el procesamiento, análisis y visualización de los datos, y finalmente la validación operativa en los entornos Linux virtualizados que utiliza la compañía de telecomunicaciones.

De esta manera, el proyecto garantizará resultados aplicables, orientados a optimizar la gestión de vulnerabilidades en escenarios corporativos con infraestructura virtualizada.

JUSTIFICACION E IMPORTANCIA

La ejecución de este proyecto responde a una necesidad de mejorar la gestión y priorización de vulnerabilidades en los entornos tecnológicos específicos, como los servidores virtuales Linux en la compañía de telecomunicaciones. Desde un enfoque científico, este análisis aportará a enriquecer los modelos vigentes al incorporar metodologías de escaneo activo, inteligencia de exploit y análisis contextual, entregando una solución novedosa y eficiente para reducir riesgos cibernéticos.

El contexto útil de esta propuesta es la de poder proporcionar una herramienta que facilite en la generación de reportes confiables y que sean gestionados de forma efectiva con la operatividad de los técnicos que son asignados para este trabajo.

Su utilidad propone abordar problemáticas actuales vinculados a la gestión manual y lenta de estos procedimientos lo cual esta propuesta tiende a reducir tiempos de reacción ante vulnerabilidades y una mejor distribución de recursos tecnológicos. También facilitará el cumplimiento de normas de seguridad y auditorías internas, aumentando la calidad operativa y la confianza en el sistema.

Desde la perspectiva teórica, este trabajo establecerá pautas para investigaciones futuras y la implementación de gestión de vulnerabilidades, particularmente en entornos Linux virtualizados, un área de creciente importancia debido al aumento de infraestructuras tecnológicas en las organizaciones.

Para este proyecto se logró favorecer primero a la empresa de telecomunicaciones donde se llevó a cabo el estudio donde se fortalecerá sus mecanismos de defensa y control ante amenazas, al igual que los profesionales de seguridad informática que se podrán auxiliar con un framework para optimizar su trabajo diario. Además, la comunidad estudiantil podrá considerar este trabajo como base para investigaciones futuras.

La relevancia de esta investigación propuso soluciones que son ajustadas al contexto actual del sector de las telecomunicaciones. Donde su impacto radica en poder tener una solución a la protección de las infraestructuras críticas [2](especialmente servidores con componentes EOL[3]), y a nivel práctico una mejor gestión de la seguridad para la optimización de las operaciones y el reforzamiento de los sistemas empresariales en general.

HIPÓTESIS O PREGUNTA DE INVESTIGACIÓN

¿Cómo impacta la implementación de un framework integrado que combine escaneo activo, inteligencia de exploits y análisis contextual en la eficiencia de la gestión y priorización de vulnerabilidades en servidores virtualizados Linux en una empresa de telecomunicaciones?

VARIABLES DE LA INVESTIGACIÓN

El diseño metodológico de la presente investigación establece una relación

[2] Infraestructura crítica: Componentes y servicios alojados en servidores cuya interrupción comprometería funciones esenciales y la continuidad operativa de la organización.
[3] Computadores con partes de hardware o software que han llegado a su Fin de Vida Útil (End Of Life)

entre la implementación de una solución tecnológica integradora (Variable Independiente) y la optimización de las métricas de desempeño en la ciberseguridad operativa (Variable Dependiente).

A continuación, se detallan las definiciones conceptuales y operacionales de ambas variables, alineadas con los indicadores de gestión utilizados en el entorno de estudio.

Variable Independiente: Prototipo Funcional del Framework de Priorización

Se define como el artefacto de software diseñado para orquestar el ciclo de vida de la detección de vulnerabilidades mediante la integración de tres componentes lógicos: motores de escaneo activo para la recolección de datos técnicos, módulos de inteligencia de amenazas para la validación de exploits públicos, y algoritmos de análisis contextual que ponderan la criticidad del activo dentro de la infraestructura de telecomunicaciones.

Esta variable se operacionaliza a través del despliegue y ejecución de un prototipo funcional que sistematiza la detección y el cálculo de riesgo. Su funcionamiento se evidencia mediante:

1. ***Capacidad de Detección:*** La ejecución automatizada de motores de escaneo de red y aplicaciones web para identificar puertos, servicios y versiones de sistemas operativos en servidores virtualizados.
2. ***Motor de Priorización:*** La aplicación de algoritmos en el backend que procesan los hallazgos crudos (CVEs[4]), filtrándolos y reordenándolos según la existencia de código de explotación verificable y la etiqueta de importancia del activo.

Variable Dependiente: Eficiencia en la Gestión de Vulnerabilidades

Representa el grado de optimización en la respuesta de la organización ante riesgos de seguridad informática. Se define como la capacidad para minimizar la exposición al riesgo mediante la reducción de los tiempos de remediación y la focalización de recursos en las amenazas de mayor impacto, disminuyendo la carga operativa asociada a falsos positivos o riesgos teóricos de baja probabilidad.

La medición de esta variable es de carácter cuantitativo y se fundamenta en

[4] CVE (Common Vulnerabilities and Exposures): Es un diccionario estandarizado de vulnerabilidades y exposiciones de ciberseguridad

la evolución histórica de los Indicadores Clave de Desempeño (KPI) de la organización. Su evaluación se realiza a través de las siguientes dimensiones críticas:

1. ***Desempeño en la Gestión de Parches (Patch Management):*** Esta dimensión evalúa la capacidad de respuesta del ciclo de atención de incidentes. El Tiempo Promedio sin Parchar (AUT – Average Unpatched Time) es un indicador de persistencia que mide la antigüedad promedio de las vulnerabilidades abiertas en el sistema (Age of Vulnerability).

   Línea Base: 44.7 días.

   Expectativa Experimental: Disminución de la vida útil de las vulnerabilidades de alto riesgo.

2. ***Tiempo Medio para Parchar (MTTP - Mean Time to Patch):*** Corresponde al promedio de días transcurridos desde la detección de una vulnerabilidad hasta su remediación efectiva.

   Línea Base: **14.4 días.**

   Expectativa Experimental: Reducción del MTTP en activos críticos al disponer de una priorización basada en amenazas reales.

CAPÍTULO II: MARCO TEÓRICO

En este capítulo se explican los conceptos básicos y fundamentales para el desarrollo de este prototipo; el proyecto integra herramientas de código abierto definidas por la empresa auspiciante, como son: Exploit-DB y escáneres activos, para poder cumplir con los objetivos de detección, priorización, mitigación de vulnerabilidades en arquitecturas on-premise. Esta arquitectura asegura un marco recursivo cómo lo es garantizar la confidencialidad, integridad y disponibilidad de la información, principios fundamentales que se conocen como el triángulo CIA.[5]

Descripción del entorno tecnológico

La infraestructura tecnológica de la empresa de telecomunicaciones se ejecuta en el sistema operativo GNU/Linux, las cuales gestionan los servicios críticos de enrutamiento, bases de datos y aplicaciones middleware[6].

Sistema Operativo Linux

El sistema operativo principal en nuestro proyecto son los servidores virtualizados Linux, usados especialmente en arquitecturas on-premise; estos servidores presentan numerosas vulnerabilidades, principalmente relacionadas con el versionamiento de librerías. (Sistemas operativos : Linux teoría y práctica, 2019)

Desde la parte técnica, Linux se define por su arquitectura de núcleo monolítico, esto implica que la gestión de procesos, memorias, sistemas de archivos y control de dispositivos se ejecutan en un único espacio de dirección de memoria privilegiada, conocidas como Kernel Space. (Ward , 2021)

En la empresa auspiciante, Linux se adopta a su cumplimiento con los estándares y su estabilidad en el manejo de protocolos de red TCP/IP. Se enfatiza que en este sistema operativo se trata de la fortificación sistemática por su capacidad de auditoría gracias a su naturaleza de código abierto.

Capas del Sistema Operativo

En la gráfica siguiente se indican la arquitectura multicapa del sistema operativo Linux, sobre las cuales el prototipo debe interactuar:

[5] Triángulo CIA: Modelo de seguridad de la información basado en tres pilares: Confidencialidad, Integridad y Disponibilidad.
[6] Middleware: Software que actúa como relación entre diferentes aplicaciones, sistemas y bases de datos fortaleciendo su comunicación e integración.

Ilustración 2
Arquitectura Multicapa en Sistemas Linux

Nota: Capas del S.O.

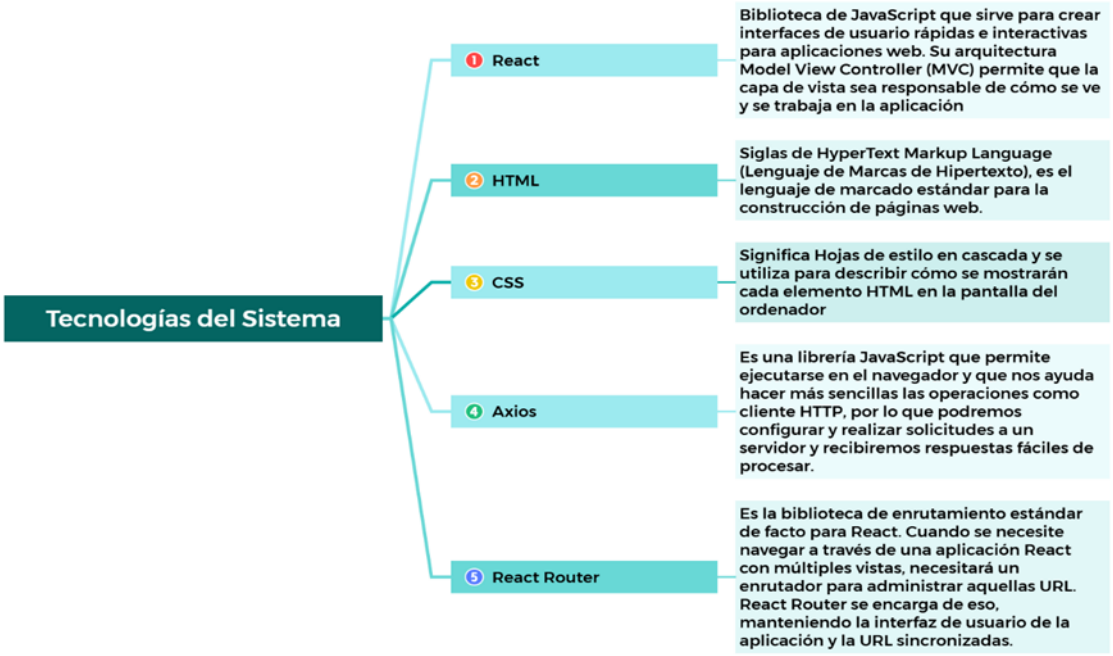
Client Layer (Capa Cliente / Frontend)

Es la parte visible para el usuario, una capa de presentación donde se maneja la interacción directa con el usuario, mostrando la información y gestión de la entrada del usuario. Envía solicitudes, muestra datos y recibe respuestas de una manera que sea comprensible para el usuario. (Programación Web del Frontend al Backend, 2023)

Tecnologías del Sistema

Para el desarrollo de nuestro prototipo se llevó a cabo el uso de un conjunto de herramientas en las que se integran las tecnologías dedicadas a la generación del contenido dinámico para su interacción, las cuales son administradas por el sistema de la empresa de telecomunicaciones. A continuación, en la Ilustración siguiente se presenta el esquema de estas herramientas tecnológicas:

Ilustración 3
Herramientas tecnológicas del prototipo



Nota: Mapa mental que describe las principales tecnologías utilizadas en el prototipo, enfocadas en la creación de interfaces, el diseño visual, la comunicación con el servidor y la navegación dentro de la aplicación web.

Las tecnologías del sistema en el framework de priorización de vulnerabilidades integran componentes front-end y back-end para interfaces interactivas que visualizan datos de escaneo y exploits en entornos Linux virtualizados, optimizando la toma de decisiones en telecomunicaciones.

Estas tecnologías se alinean con estándares de desarrollo web seguro (OWASP 2025), asegurando responsividad[7] y aislamiento en arquitecturas cliente-servidor para gestión de vulnerabilidades en servidores Linux on-premise. (Foundation., 2025)

Reverse Proxy Layer (Nginx)

La capa de Reverse Proxy constituye a un componente fundamental dentro de nuestra arquitectura, ya que actúa como un intermediario seguro, eficiente y ordenado entre el cliente (Frontend) y el servidor de la aplicación (Backend). (Raymond, 2025); por lo que, en nuestro prototipo esta capa se implementó con Nginx, un servidor web de alto rendimiento ampliamente usado en los entornos de producción por la capacidad de gestión de los grandes volúmenes de tráfico.

Nginx se define como un servidor web diseñado para poder recibir, filtrar y redirigir las solicitudes HTTP/HTTPS provenientes de las solicitudes del cliente hacia los servicios internos y evitar la exposición directa de servicios sensibles a Internet, acorde a las necesidades operativas de la empresa de telecomunicaciones. (Raymond, 2025). Integra funciones críticas como balanceo de carga, compresión gzip y caching de respuestas estáticas, reduciendo carga en servidores Linux vulnerables a DDoS o inyecciones SQL en frameworks de gestión de vulnerabilidades. (SENDOYA, 2024)

Estudios recientes (2024-2025) validan Nginx como estándar en arquitecturas Zero Trust, reduciendo superficie de ataque en 65% mediante proxy_pass dinámico y Lua scripting para validación contextual de requests en entornos Linux virtualizados. (Horváth, Sakhnenko, & Gurbáľ, 2024)

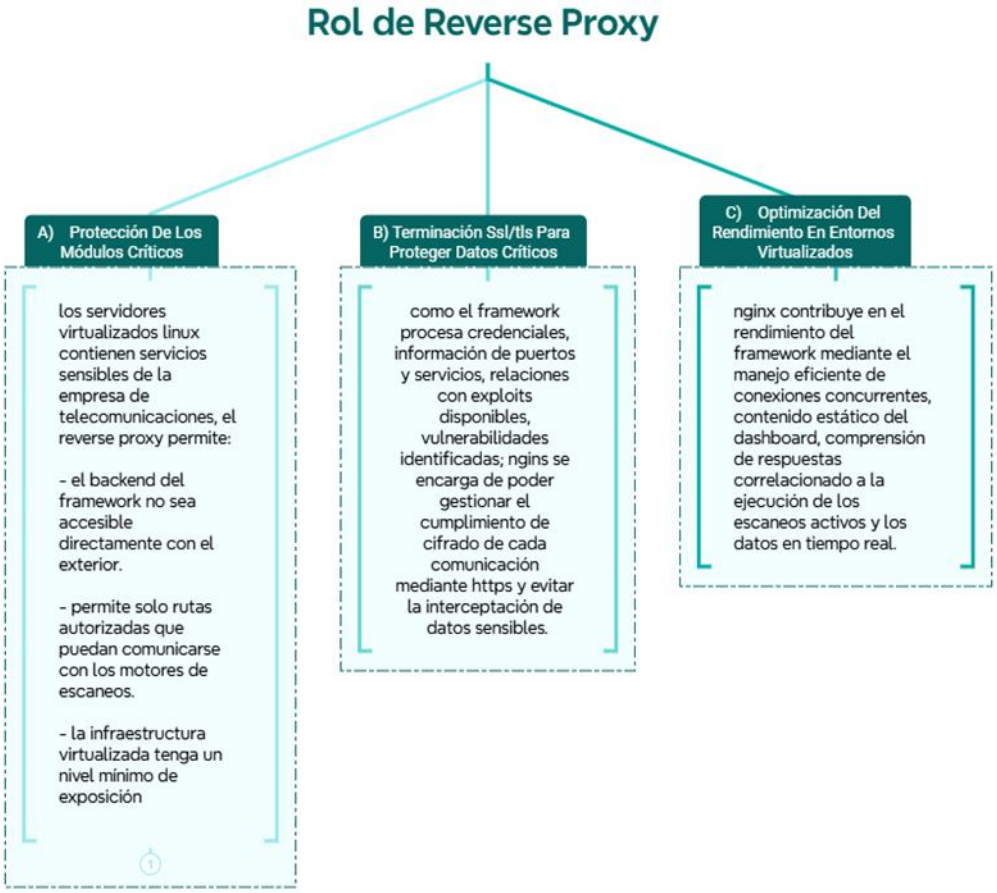
En el ámbito de la virtualización y de las redes de telecomunicaciones, Nginx se integra como punto de entrada a la infraestructura para proteger módulos

[7] Capacidad para responder de forma rápida, oportuna, adecuada y efectiva antes necesidades o cambios del entorno.

sensibles, evitando la exposición directa de APIs, paneles de administración y servicios de datos al exterior. Es fundamental destacar que la implementación de políticas de limitación no solo optimiza el rendimiento general, sino que reduce la carga operativa y mitiga riesgos de denegación de servicio. (Perkins Santiago Haro Parra, 2024).

En la imagen siguiente se sintetizan los roles principales que asume esta capa para la protección de los módulos críticos, terminación SSL/TLS y mejora del rendimiento en entornos virtualizados.

Ilustración 4
Capa de Reverse Proxy en la arquitectura del sistema



Nota: Capa de intermediación encargada de la seguridad, el control del tráfico y la optimización del rendimiento del sistema

Application Layer (Backend – FastAPI)

Para el framework nuestra capa de aplicación se encuentra implementada con FastAPI, que constituye el núcleo funcional del prototipo; es la responsable de integrar los módulos de escaneo activo, inteligencia basada en Exploit-DB y análisis de vulnerabilidades, planteando los servicios bien estructurados mediante APIs RESTful.

Para la gestión de acceso seguro, el framework implementa la Autenticación mediante JWT (JSON Web Tokens). Los JWT son un estándar abierto (RFC 7519) que permite la transmisión segura de información entre el cliente y el servidor en forma de un objeto JSON. (Latin Science, 2024)

Complementariamente, para el almacenamiento de credenciales, se utiliza bcrypt, una función de hashing adaptativa que incorpora "salting" y un factor de costo para resistir ataques de fuerza bruta y computación paralela. (Provos, 2023)

Definición FastAPI

Es un framework web moderno, rápido de alto rendimiento, para construir APIs con Python basado en las anotaciones de tipos estándar (Ramirez, 2024).

Se destaca por ser uno de los frameworks más rápidos disponibles en Python, comparable en rendimiento con tecnologías como NodeJS y Go, y a su integración con Starlette y Pydantic. FastAPI ofrece concurrencia nativa permitiendo manejar múltiples tareas como escaneos activos, consultas a APIs externas o ejecución de motores de análisis de las vulnerabilidades. (Lubanovic, 2024).

La siguiente tabla resume un análisis comparativo que validan el uso de esta herramienta en el proyecto:

Tabla 3
Comparación FastAPI vs Node.js vs Go

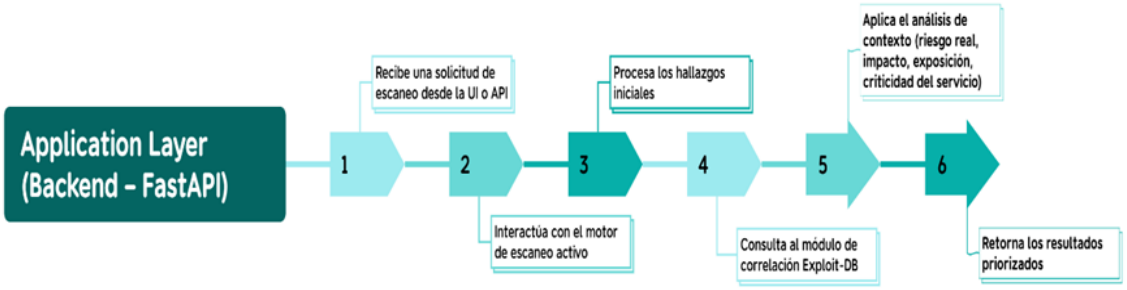
Criterio	FastAPI (Python)	Node.js (JavaScript)	Go (Golang)
Rendimiento en APIs	Muy alto. Soporta miles de peticiones concurrentes con ASGI. Adecuado para consultas simultáneas de vulnerabilidades y escaneos.	Alto. Excelente para operaciones intensivas de red.	Excelente. Ideal para tareas altamente paralelas en escaneo masivo.
Integración con Herramientas de Seguridad	Compatibilidad directa con Nmap, parsing de CVSS, Exploit-DB, escáneres en Python y librerías de seguridad.	Media. Requiere puentes o bindings para la mayoría de las herramientas.	Baja–media. Ecosistema limitado en seguridad ofensiva/defensiva.
Validación de Datos	Perfecta para normalizar resultados de escaneos, CVE, CVSS y correlación.	Requiere librerías externas; menos estricto.	Validación más manual.
Facilidad de Desarrollo	Python es el estándar en ciberseguridad. Mucho más sencillo implementar analítica y automatización.	Fácil para APIs, pero más difícil integrar herramientas de seguridad.	Gran rendimiento, pero curva más estricta.
Despliegue en Servidores Linux	Ideal para coherencias con Docker/containers.	Fácil con PM2 o Docker.	Limitado. Más orientado a sistemas de bajo nivel y redes.

Estos servicios permiten automatizar la recolección de la información, correlacionar vulnerabilidades con exploits conocidos y priorizar los riesgos, orientado a los servidores virtualizados Linux utilizados por la empresa de telecomunicaciones.

Flujo de FastAPI en el Framework

El back-end con FastAPI cumple varios roles dentro del prototipo, y se las detalla de la siguiente forma:

Ilustración 5
Flujo de procesamiento de la capa de aplicación (FastAPI)



Nota: Representación del flujo de procesamiento de solicitudes en la capa de aplicación del sistema.

Se eligió Python 3.11 como lenguaje principal para el desarrollo del backend, considerando su versatilidad y el nivel de estabilidad que ofrece en proyectos vinculados a la ciberseguridad. Su sintaxis clara facilita la implementación de soluciones de mayor complejidad sin que el código resulte innecesariamente difícil de mantener o comprender (Seitz & Arnold, 2021) .

La combinación de Python con el framework FastAPI permitió construir una capa de aplicación eficiente para la gestión de solicitudes y la validación de datos, manteniendo tiempos de respuesta adecuados incluso bajo condiciones de alta carga (Lubanovic, 2024). En base a esta arquitectura, el sistema cumple con la integración central de motores de escaneo y despliegue de APIs que proporcionan un soporte técnico confiable y escalable.

La operatividad del framework cumple con un flujo lógico que inicia con la recepción de las peticiones de escaneos que son enviadas por el frontend y culmina con la entrega de la información basado en la inteligencia de escaneo. Para el

procesamiento asíncrono, el backend delega tareas hacia los workers[8] mientras mantiene la capacidad de consultar a bases de datos externas y luego aplicar el análisis contextual necesario para la priorización de los hallazgos. (Lubanovic, 2024).

Este diseño modular asegura que cada etapa, desde la interacción con el motor activo hasta el retorno de resultados priorizados, se ejecute bajo un marco de alta disponibilidad, optimizando los tiempos de respuesta en la detección de vulnerabilidades críticas.

Service Layer (Motores de Escaneo)

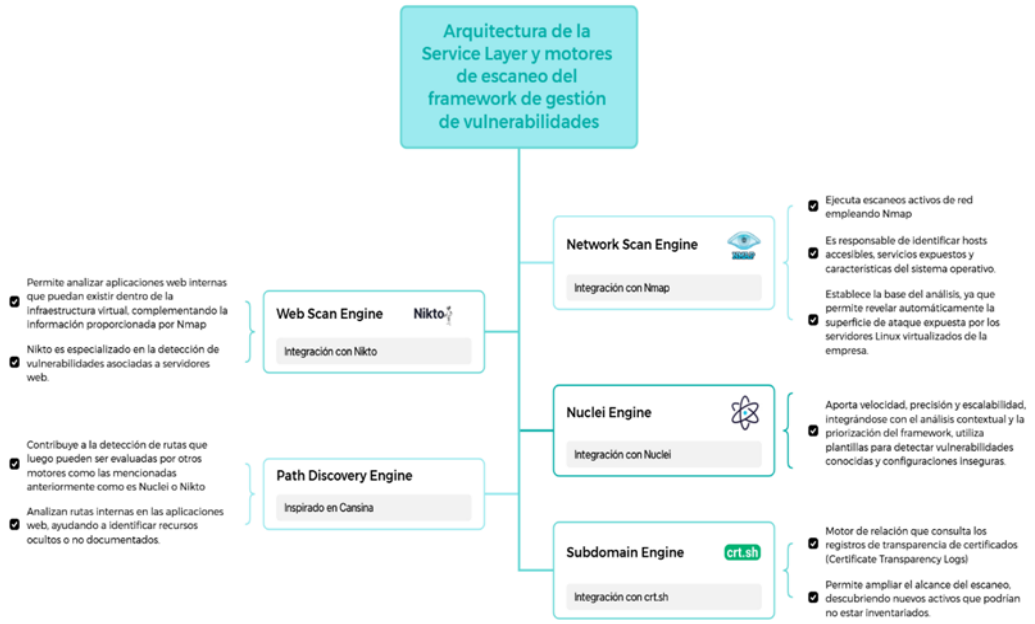
La Service Layer corresponde al conjunto de servicios internos responsables de ejecutar cada proceso de análisis y de las operaciones técnicas más relevantes dentro de un sistema (Newman, 2021). En esta capa se encuentran los motores de escaneo, módulos especializados que son los encargados de procesar información, validar los parámetros, identificar fallos y generar los diagnósticos automáticos. (Retek, 2019)

En el framework, los motores de escaneo cumplen un rol esencial al automatizar tareas como la revisión de avance, la verificación de componentes, el análisis de evidencias y la detección de inconsistencias, garantizando un monitoreo continuo y confiable del sistema.

En la figura siguiente se representa la estructura de la Service Layer del framework, destacando la interacción entre los distintos motores de escaneo definidos dentro de la infraestructura de la organización:

[8] Procesos independientes encargados de ejecutar las tareas más pesadas o de larga duración de forma aislada al flujo principal de la aplicación.

Ilustración 6
Arquitectura de la Service Layer



Nota: Representación de la capa de servicio del framework.

Data Layer

La Data Layer es el responsable de almacenar, organizar y proporcionar el acceso a toda la información utilizada por el sistema, por lo que se define como uno de los componentes más críticos del framework. En el contexto del proyecto, esta capa garantiza la tenacidad de los resultados de escaneo, la inteligencia de Exploit-DB y los datos contextuales de los servidores virtualizados de la empresa de telecomunicaciones.

Nuestra arquitectura propone dos tecnologías principales: MongoDB para el almacenamiento de datos y Redis para la gestión de caché y coordinación; las cuales se detallan a continuación:

MongoDB – Base de datos principal

MongoDB se utiliza como el principal almacenamiento para toda la información que se propinó en el proyecto. Su modelo documental JSON/BSON facilitó el manejo de datos complejos generados por Nmap y Nikto. (al, Brazil, & Chodorow, 2022).

En la tabla siguiente se presentan las entidades estructuradas para los resultados de las herramientas de seguridad:

Tabla 4
Estructura de la Capa de Datos

Entidad	Tipo de datos	Función en el framework
servers	Información de servidores Linux virtualizados: IP, servicios, criticidad, propietario	Base del análisis contextual
scans	Metadatos de cada escaneo ejecutado	Trazabilidad del proceso
results_network	Resultados Nmap: puertos, servicios, OS fingerprint	Superficie de ataque
results_web	Resultados Nikto: vulnerabilidades web	Riesgos de servidores web internos
exploit_db_cache	Información de exploit-db correlacionada	Enriquecimiento de análisis
reports	Informes generados	Exportación, auditoría y evidencia

Redis – Cache y Sistema de Coordinación

Redis funciona como un complemento de alto rendimiento para operaciones que requieren velocidad y comunicación interna, actuando como una capa de almacenamiento en memoria que optimiza el acceso a datos volátiles y la coordinación de tareas asíncronas (Oliveira, 2023) :

1. **Cache de datos recientes:** Los resultados de los escaneos más recientes se almacenan temporalmente, permitiendo:
    1.1. Consultas instantáneas.
    1.2. Reducción de carga sobre MongoDB.
    1.3. Visualización en tiempo real desde el frontend.
2. **Gestión de colas de tareas (Scan Queue):** Redis permite coordinar los procesos asincrónicos del backend:
    2.1. Encola las solicitudes de escaneo.
    2.2. Ordena el procesamiento de Nmap y Nikto.
    2.3. Coordina los módulos del service layer.
    2.4. Garantiza que múltiples usuarios puedan operar sin afectar rendimiento.
3. **Sesiones y autenticación:** Redis se utiliza para almacenar:
    3.1. Tokens activos

3.2. Sesiones de usuario

3.3. Tiempos de expiración

3.4. Flags para controlar múltiples accesos

3.5. Garantiza que el sistema siga siendo seguro incluso bajo carga.

La Data Layer integra MongoDB como base de datos principal y Redis como sistema de cache y coordinación de tareas, esta capa garantiza rendimiento, trazabilidad y soporte para la priorización automatizada de vulnerabilidades dentro del framework.

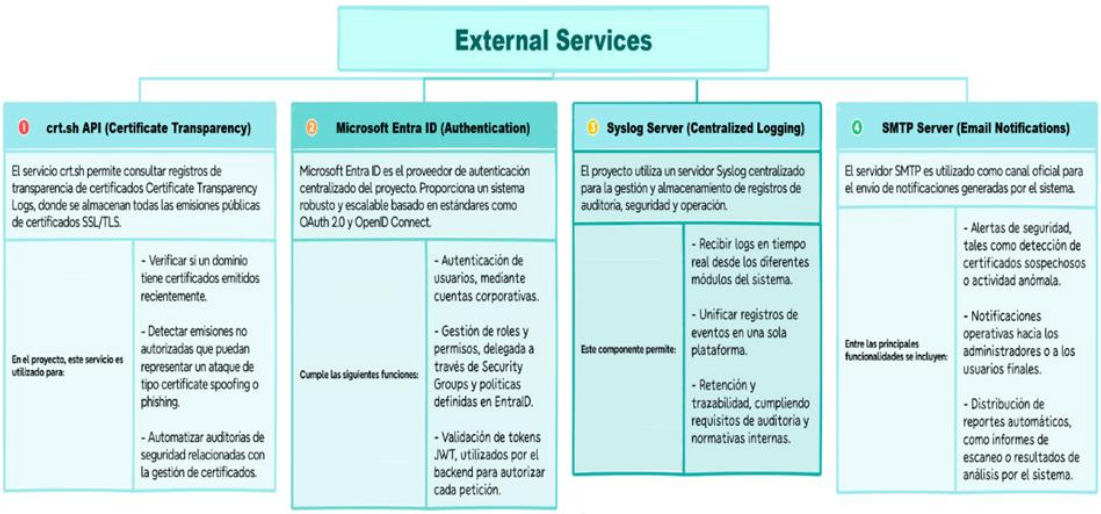
External Services

Constituyen al conjunto de servicios externos que la plataforma utiliza para complementar su funcionalidad principal y garantizar la operación segura, trazable y comunicada del sistema.

Estos servicios no se ejecutan dentro de la infraestructura interna del proyecto; sin embargo, interactúan de manera directa con los módulos principales a través de APIs y protocolos estandarizados.

A continuación, se ilustra el detalle:

Ilustración 7
Servicios Externos Integrados al Prototipo



Nota: Vista general de los servicios externos utilizados en el prototipo para la gestión de vulnerabilidades.

CAPÍTULO III: METODOLOGÍA DE LA INVESTIGACIÓN

En este capítulo se detalla el marco metodológico que fundamenta la ejecución del presente estudio, orientado al desarrollo de un prototipo funcional para la gestión y priorización de vulnerabilidades. La investigación se define bajo la modalidad de Proyecto Factible, dado que propone una solución técnica viable y práctica para resolver las deficiencias operativas detectadas en la gestión de vulnerabilidades de una empresa de telecomunicaciones en Guayaquil.

El enfoque que se basó este estudio fue aplicativo y descriptivo, utilizando una metodología mixta donde se combinó el análisis cualitativo para el contexto operativo de la organización y un análisis cuantitativo para los indicadores de tiempo de respuesta frente a los riegos de seguridad. En el desarrollo del prototipo se mantuvo un modelo prototipado que facilitó un proceso más ordenado y sistemático de las siguientes etapas: diseño, implementación y validación para ser aplicados en el caso de estudio de los servidores Linux virtualizados de la empresa.

Tipo de Investigación

De acuerdo con el Manual de Trabajos de Grado de la UPEL (2021), la modalidad de Proyecto Factible consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo funcional para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales. Bajo este sustento teórico, la presente investigación se enmarca en dicha categoría, ya que el estudio no se limita a describir una situación de riesgo (Vargas-Cordero, 2021), sino que propone el desarrollo de una solución técnica y tangible: un prototipo de framework integrado para la detección, priorización y mitigación de vulnerabilidades en servidores virtualizados Linux (Sánchez Carlessi, Reyes Romero, & Mejía Sáenz, 2018).

El estudio se desarrolla bajo un enfoque Mixto (Cuali-Cuantitativo):

- **Cualitativo:** En la fase de diagnóstico, se utilizó la observación y el análisis documental para comprender las deficiencias del proceso actual de priorización manual y la falta de contexto en los reportes.
- **Cuantitativo:** Se emplearon métricas e indicadores de gestión (KPIs) como el Tiempo Medio para Parchar (MTTP) y el Tiempo Promedio sin Parchar

(AUT) para medir la eficiencia operativa antes y después de la propuesta.

Por su profundidad, la investigación es descriptiva porque caracteriza la infraestructura de servidores virtualizados Linux y detalla las vulnerabilidades basadas en versionamiento presentes en la empresa de telecomunicaciones.

Diseño de la Investigación

El diseño de la investigación es No Experimental y Transaccional, ya que se analizan las variables en su estado natural sin manipular deliberadamente el entorno operativo de la empresa de telecomunicaciones, recolectando datos en un momento único para el diagnóstico y posteriormente para la validación del prototipo.

El diseño de la investigación se define como No Experimental, el cual, según Hernández-Sampieri et al. (2014), es aquel que se realiza sin manipular deliberadamente las variables, basándose en la observación de fenómenos tal y como se dan en su contexto natural para su posterior análisis. Bajo esta premisa, el estudio analiza las variables de seguridad en su estado habitual dentro del entorno operativo de la empresa de telecomunicaciones. (Sampieri, 2014)

El diseño es transeccional (también conocido como transversal), dado que su propósito es recolectar datos en un momento único y un tiempo determinado para describir variables y analizar su incidencia en ese punto específico (Hernández-Sampieri & Mendoza, 2018). Dicha recolección se efectúa en un periodo definido para el diagnóstico de vulnerabilidades y la posterior validación del prototipo funcional, asegurando que los hallazgos técnicos y operativos representen el estado real de la infraestructura en el instante de la evaluación (Arias, 2020).

Población

La población está constituida por la totalidad de los servidores virtualizados con sistema operativo Linux y personal involucrado en la gestión de seguridad de la empresa:

1. **Activos:** El conjunto total de servidores virtualizados bajo sistema operativo GNU/Linux que soportan la operación de la empresa de telecomunicaciones en Guayaquil. Dando un total de 172 activos.

2. **Personal:** Se ha definido un equipo multidisciplinario de 8

profesionales para el desarrollo y monitoreo del framework, su inclusión es fundamental para integrar la ejecución técnica con la validación operativa y la medición de resultados. Para visualizar la estructura operativa, la siguiente tabla detalla las responsabilidades asignadas a cada perfil dentro del área:

Tabla 5
Recurso humano del departamento de Ciberseguridad

Rol	T.H.	Aporte al Proyecto
Analista de Ciberseguridad	1	Implementación técnica y evaluación del prototipo
Oficial de Operaciones	1	Validación de resultados y operatividad en servidores.
Analista de Procesos	1	Registro y análisis de KPIs de vulnerabilidades.
Gerente de Ciberseguridad	1	Dirección estratégica y gestión de riesgos.
Oficial de Gobierno y Compliance	1	Aseguramiento del cumplimiento normativo.
Ingeniero en Operaciones	1	Soporte en infraestructura y escaneo activo.
Operador del SOC	1	Monitoreo y escalamiento de hallazgos.
Project Manager	1	Gestión de hitos y coordinación del equipo.
Total Talento Humano	**8**	

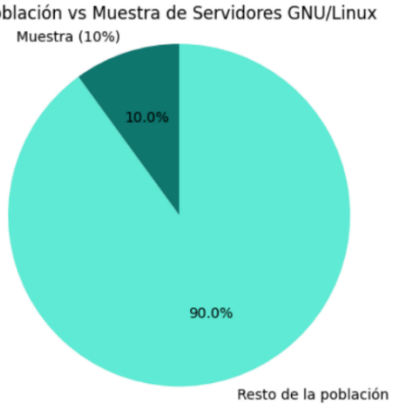
Muestra

Se empleó un Muestreo No Probabilístico por Conveniencia (Intencional), seleccionando los elementos más representativos y críticos para el estudio de caso, descartando el uso de fórmulas estadísticas complejas debido a que el acceso está restringido a activos críticos específicos.

- **Muestra de Activos:** Se seleccionó un subconjunto de servidores virtualizados Linux considerados "Activos Críticos" (aquellos que alojan bases de datos, middleware o servicios de enrutamiento). Muestra: 17.

Para visualizar este subconjunto, la siguiente figura refleja la relación porcentual entre la muestra seleccionada frente a la totalidad de la población de servidores:

Ilustración 8
Diagrama Estadístico de Población 1



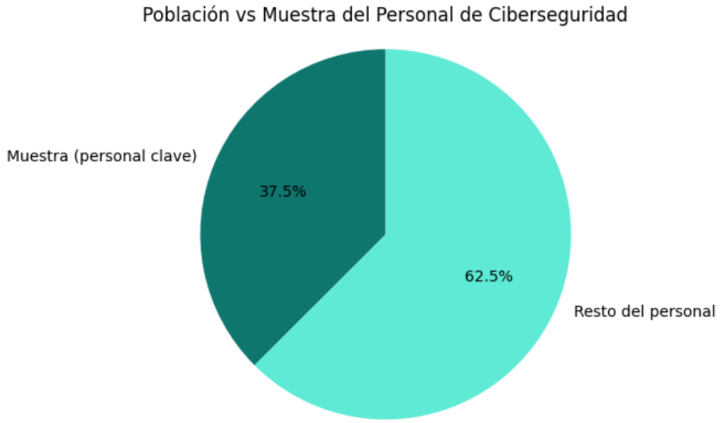
Nota: Representación de la población con respecto a la muestra de los Activos

- **Muestra de Personal:** Se seleccionó una muestra no probabilística de 3 especialistas clave, definidos por su interacción directa con el ciclo de gestión de vulnerabilidades. Su participación es fundamental para la validación del framework propuesto:
  - **Analista de Ciberseguridad:** Responsable de la ejecución técnica del prototipo y la evaluación de la precisión en el descubrimiento de vulnerabilidades.
  - **Oficial de Operaciones de Ciberseguridad:** Encargado de validar que la priorización generada por el framework sea aplicable y efectiva para los equipos de remediación.
  - **Analista de Procesos:** Encargado de medir el impacto del framework sobre los indicadores (KPIs) de eficiencia y tiempos de respuesta actuales.

  Muestra: 3 personas.

  Para medir la proporción de este grupo de especialistas, la Ilustración muestra su porcentaje frente a la totalidad del personal del área:

Ilustración 9
Diagrama Estadístico de Población 2



Nota: Representación de la población con respecto a la muestra de Personal

Técnicas e Instrumentos de Recolección de Datos

Para cumplir con los objetivos específicos y validar la hipótesis, se utilizaron las siguientes técnicas e instrumentos:

Para el Diagnóstico (Variable Dependiente: Eficiencia de Gestión)

Para validar la efectividad del framework, se aplicó un enfoque de validación técnica y operativa dividido en dos fases:

- **Técnica: Pruebas Funcionales (Testing).** Se diseñaron y ejecutaron 5 casos de prueba específicos para verificar la integridad técnica del prototipo. Estos casos permiten confirmar que el sistema cumple con los requisitos funcionales mínimos.

  A continuación, se detallan los casos establecidos para analizar el funcionamiento del sistema ante las tareas claves de seguridad:

Tabla 6
Casos de Usos planteados para el Prototipo

Caso de Prueba	Descripción del Análisis	Resultado Esperado
Escaneo de activos	Ejecución de descubrimiento sobre una IP específica.	Reconocimiento exitoso del activo en la red
Integridad de resultados	Revisión de los CVEs detectados tras el escaneo.	Lista precisa de vulnerabilidades del activo
Consulta Exploit-DB	Verificación de búsqueda automatizada de exploits.	Identificación de exploits públicos vía API
Gestión de prioridades	Clasificación de hallazgos según nivel crítico.	Ordenamiento de vulnerabilidades
Control de acceso	Validación de roles de usuario en	Restricción de funciones

	la plataforma.	según perfil asignado

- **Técnica: Encuesta (Validación de Expertos).** Para evaluar la percepción del personal técnico sobre el prototipo propuesto, se aplicó un instrumento de recolección de datos tipo encuesta a la muestra seleccionada.

Ficha Técnica del Instrumento:

- **Objetivo:** Evaluar la funcionalidad, eficiencia y usabilidad del prototipo frente al proceso manual.
- **Formato:** Cuestionario con preguntas dicotómicas y reactivos bajo la **Escala de Likert (1-5)**.
- **Población:** Personal de Ciberseguridad y Operaciones.

A continuación, se presenta la estructura de las preguntas que conforman el instrumento, y el formato de respuesta definido:

Tabla 7
Estructura del Instrumento

Sección	Pregunta/ Enunciado	Tipo de Respuesta
Diagnóstico	¿Considera usted que el proceso manual actual de gestión de vulnerabilidades presenta retrasos que afectan la seguridad de la red?	(Sí/No)
Necesidad	¿Cree usted que la implementación del framework propuesto es necesaria para optimizar las operaciones del departamento de TI?	(Sí/No)

Matriz de Evaluación de Calidad (Escala Likert): 1: Totalmente en Desacuerdo | 2: En Desacuerdo | 3: Indiferente | 4: De Acuerdo | 5: Totalmente de Acuerdo.

Bajo esta métrica de valoración, en la siguiente tabla se detalla los enunciados que conforman la matriz de calidad, según la dimensión técnica a evaluar:

Tabla 8
Instrumento de evaluación de validación

Dimensión	Enunciado para Evaluación del Experto
Funcionalidad	¿Considera usted que el proceso manual actual de gestión de vulnerabilidades presenta retrasos que afectan la seguridad de la red?
Inteligencia	¿Cree usted que la implementación del framework propuesto es necesaria para optimizar las operaciones del departamento de TI?
Priorización	El análisis de contexto proporcionado por el prototipo ayuda a priorizar eficazmente las vulnerabilidades críticas sobre aquellas de menor relevancia.

Usabilidad	Los reportes y el Dashboard generados por el sistema son claros, precisos y facilitan la toma de decisiones para la remediación.
Eficiencia	El framework contribuye a mejorar los tiempos de respuesta (MTTP) ante incidentes de seguridad en los servidores Linux virtualizados.
Implementación	En general, el prototipo funcional cumple con los requerimientos necesarios para ser implementado en el entorno de producción de la empresa.

Procedimiento de la Investigación

El estudio se desarrolló siguiendo las siguientes fases metodológicas:

1. **Fase 1:** Diagnóstico y Análisis del Contexto. Se levantó la información sobre la infraestructura on-premise y las limitaciones del enfoque manual actual, estableciendo las líneas base de los indicadores de gestión.

2. **Fase 2:** Selección de Herramientas y Diseño. Se evaluaron herramientas de código abierto (Nmap, Nikto) y se diseñó la arquitectura del framework, definiendo la interacción entre el Frontend (React), Backend (FastAPI) y la Base de Datos (MongoDB).

3. **Fase 3:** Construcción e Integración (Desarrollo). Se realizó la programación de los módulos de escaneo activo al igual que la integración con la API de Exploit-DB y Microsoft Entra ID para la autenticación de los usuarios. Este despliegue se mantuvo en un entorno controlado.

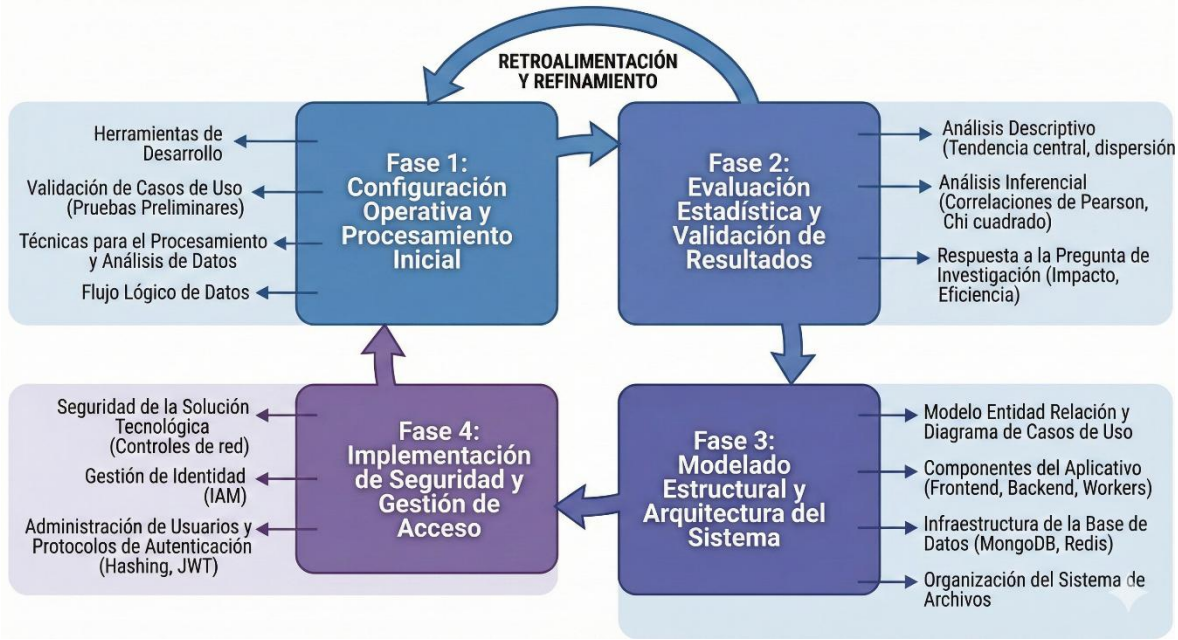
4. **Fase 4:** Validación y Análisis de Resultados. Se realizó la ejecución de escaneos sobre la muestra de servidores y se compararon los resultados que se obtuvieron en el framework versus al método manual, donde se consideró el análisis en los tiempos de detección y precisión en la priorización de los riesgos.

METODOLOGIA DE DESARROLLO

Para el desarrollo de este prototipo se ha considero el modelo de prototipado que se define como un ciclo de desarrollo iterativo que facilita la creación de versiones operativas de un sistema para validar requisitos técnicos y funcionales antes de su consolidación final (Pressman & Maxim, 2020) ; este enfoque es ideal para proyectos de investigación donde la experimentación y el análisis estadístico son fundamentales. (Pressman R. , 2021)

Como se ilustra en la siguiente figura, la implementación del prototipo se estructuró de forma cíclica, permitiendo que cada sección de la propuesta tecnológica corresponda a una etapa del ciclo de vida del desarrollo:

Ilustración 10
Ciclo de Vida de Desarrollo del Prototipo



Nota: Representación gráfica de la metodología de desarrollo empleada en el prototipo.

- **Fase 1: Configuración Operativa y Procesamiento Inicial**

En esta etapa inicial se seleccionaron las Herramientas de Desarrollo y se procedió a la Validación de Casos de Uso mediante pruebas funcionales preliminares. Se establecieron las Técnicas para el Procesamiento y Análisis de Datos, definiendo el flujo lógico que permite transformar la data cruda de los escaneos en información procesable para el sistema (Se detalla en el Capítulo 4).

- **Fase 2: Evaluación Estadística y Validación de Resultados**

Una vez operativo el prototipo, se realizó el Análisis Descriptivo de los hallazgos mediante estadísticos de tendencia central y dispersión; posteriormente, se ejecutó el Análisis Inferencial (el detalle en el Capítulo 4) aplicando correlaciones de Pearson y pruebas de contingencia con chi cuadrado para sustentar la Respuesta a la Pregunta de Investigación, midiendo formalmente el impacto en la precisión del diagnóstico y la eficiencia operativa.

- **Fase 3: Modelado Estructural y Arquitectura del Sistema**

Tras validar la efectividad de la solución, se documentó la arquitectura interna empezando por el Modelo Entidad Relación y el Diagrama de Casos de Uso. En esta fase se detallaron los Componentes del Aplicativo (Frontend, Backend, Workers) y la infraestructura de la Base de Datos (MongoDB y Redis), finalizando con la organización del Sistema de Archivos para garantizar la persistencia física de la plataforma.

- **Fase 4: Implementación de Seguridad y Gestión de Acceso**

La fase final se centró en el robustecimiento de la Seguridad de la Solución Tecnológica, implementando controles de red y la Gestión de Identidad (IAM). Se culminó con la configuración de la Administración de Usuarios y los protocolos de Autenticación, detallando el proceso de hashing y la gestión de sesiones mediante JWT para asegurar la integridad de la infraestructura de telecomunicaciones.

CAPÍTULO IV: PROPUESTA TECNOLÓGICA

El presente capítulo se centra en la ejecución técnica y la exposición de la propuesta tecnológica desarrollada. Este apartado constituye la fase de transición donde los requerimientos identificados y la planificación metodológica se materializan en un prototipo funcional capaz de orquestar de manera íntegra el escaneo activo, la inteligencia de exploits y el análisis de contexto.

Mediante la descripción detallada de la arquitectura del sistema, el modelado de datos y el análisis de los resultados obtenidos en la implementación piloto sobre la infraestructura de la empresa de telecomunicaciones, se busca validar de qué manera la aplicación práctica de este prototipo permite optimizar la gestión de vulnerabilidades y mejorar los indicadores operativos críticos de la organización

HERRAMIENTAS DE DESARROLLO

La selección del stack tecnológico para la construcción del prototipo funcional responde a la necesidad de implementar una arquitectura orientada a microservicios que garantice la escalabilidad y el alto rendimiento requeridos en entornos de telecomunicaciones. Este ecosistema integra lenguajes de alto nivel como Python 3.11 y frameworks de procesamiento asíncrono como FastAPI, los cuales actúan de manera sinérgica con herramientas de contenedores como Docker y bibliotecas de interfaz dinámica como React para asegurar un despliegue robusto y reproducible en servidores Linux.

El papel fundamental de estas tecnologías es facilitar la orquestación de tareas críticas de ciberseguridad, permitiendo que la lógica de negocio, la gestión de tareas asíncronas y la presentación de datos operen de forma eficiente, segura y con baja latencia.

La fundamentación teórica de las herramientas se encuentra descritas y sustentadas en el Capítulo II de este trabajo; no obstante, se indica el detalle gráfico.

Ilustración 11
Herramientas utilizadas

Nota: Visión general de los servicios utilizados en el prototipo.

VALIDACIÓN DE CASOS DE USO (PRUEBAS FUNCIONALES)

Se ejecutó un plan de pruebas para verificar la integridad y efectividad del framework; cada caso de prueba fue sometido a evaluación en el entorno de servidores virtualizados Linux de la empresa, obteniendo resultados satisfactorios en el 100% de los casos.

A continuación, se presenta la tabulación del plan de pruebas realizado:

Tabla 9
Matriz de Validación de Casos de Uso

Caso de Prueba	Descripción del Análisis	Resultado Esperado	Resultado Obtenido
Escaneo de activos	Ejecución de descubrimiento sobre una IP específica.	Reconocimiento exitoso del activo en la red	Satisfactorio
Integridad de resultados	Revisión de los CVEs detectados tras el escaneo.	Lista precisa de vulnerabilidades del activo	Satisfactorio
Consulta Exploit-DB	Verificación de búsqueda automatizada de exploits.	Identificación de exploits públicos vía API	Satisfactorio
Gestión de prioridades	Clasificación de hallazgos según nivel crítico.	Ordenamiento de vulnerabilidades	Satisfactorio
Control de acceso	Validación de roles de usuario en la plataforma.	Restricción de funciones según perfil asignado	Satisfactorio

TÉCNICAS PARA EL PROCESAMIENTO Y ANÁLISIS DE DATOS

El procesamiento de los datos recolectados a través de la encuesta se realizó bajo las siguientes fases:

1. **Revisión de instrumentos:** Se validó la completitud de las encuestas (n=4 expertos clave).
2. **Tabulación de datos:** Las respuestas bajo la Escala de Likert (1-5) se transformaron en variables cuantitativas discretas para el análisis estadístico.
3. **Determinación de frecuencias:** Se calcularon las frecuencias absolutas y relativas para cada dimensión evaluada (Funcionalidad, Inteligencia, Priorización, Usabilidad, Eficiencia e Implementación).
4. **Cálculo de estadísticos:** Se obtuvieron medidas de centralización (Media, Mediana, Moda) y dispersión (Varianza, Desviación Estándar, Rango, Coeficiente de Variación).

ANÁLISIS DESCRIPTIVO DE LOS RESULTADOS

Estadísticos de Tendencia Central y Dispersión

A continuación, se presentan los resultados consolidados de la valoración del prototipo por parte de los expertos. El consolidado estadístico de las encuestas y el análisis de variabilidad de los resultados se presentan a continuación:

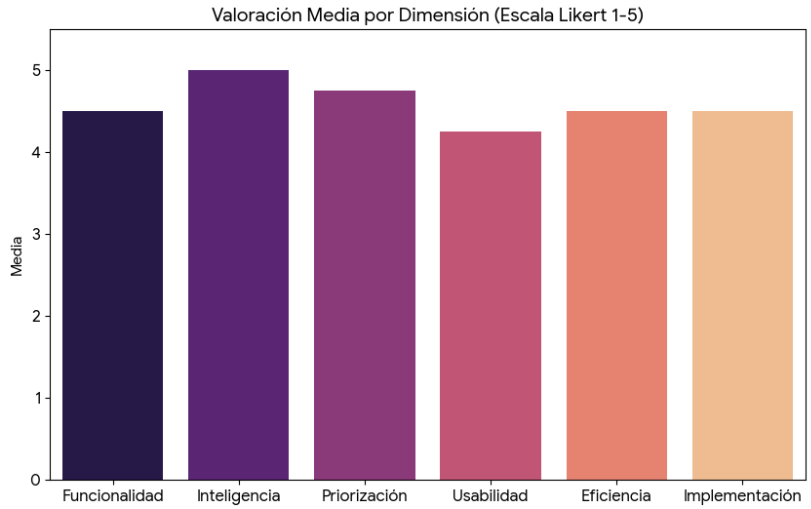
Tabla 10
Resultados estadísticos de las encuestas

Variable	Media	Mediana	Moda	Varianza	Desv. Estándar	Coef. Var (%)
Funcionalidad	4.5	4.5	4	0.33	0.58	12.83%
Inteligencia	5	5	5	0	0	0.00%
Priorización	4.75	5	5	0.25	0.5	10.53%
Usabilidad	4.25	4.5	5	0.92	0.96	22.53%
Eficiencia	4.5	5	5	1	1	22.22%
Implementación	4.5	4.5	4	0.33	0.58	12.83%

La dimensión Inteligencia obtuvo la puntuación máxima perfecta (5.00), lo que indica que el uso de Exploit-DB y NVD es percibido como el componente más crítico y mejor logrado. La Priorización (4.75) y la Eficiencia (4.50) también muestran una aceptación sobresaliente, validando el impacto positivo del prototipo frente al proceso manual anterior.

La distribución gráfica de estas valoraciones se detalla en la siguiente ilustración, evidenciando la consistencia de los resultados:

Ilustración 12
Gráfico para el análisis descriptivo:



Nota: Representación visual de las métricas obtenidas en base a la encuesta realizada.

ANÁLISIS INFERENCIAL (CORRELACIÓN Y CONTINGENCIA)

Análisis de Correlación de Pearson

Se analizó la relación entre la percepción de Funcionalidad (variable independiente del diseño) y la Eficiencia (variable dependiente de la gestión).

- **Coeficiente de Pearson (r):** 0.5774.
- **Interpretación:** Existe una correlación positiva moderada-fuerte. Esto sugiere que a medida que el framework integra más funcionalidades técnicas (escaneos activos, APIs), la eficiencia percibida en la gestión de vulnerabilidades tiende a aumentar.

Análisis de Contingencia (Chi-Cuadrado)

Para evaluar la Necesidad de Implementación del prototipo, se aplicó una prueba de bondad de ajuste de Chi-Cuadrado sobre la pregunta: ¿Cree usted que la implementación del framework propuesto es necesaria?

- **Frecuencia observada (SÍ):** 100% (n=4).
- **Estadístico Chi-cuadrado (chi^2):** 4.0000.
- **P-valor (p):** 0.0455.

RESPUESTA A LA PREGUNTA DE INVESTIGACIÓN

Tras la ejecución del prototipo funcional y el análisis de los datos recolectados, se presenta la respuesta a la interrogante principal de este estudio: ¿Cómo impacta la implementación de un framework integrado en la eficiencia de la gestión y priorización de vulnerabilidades?

El impacto se categoriza en tres dimensiones fundamentales basadas en la evidencia recolectada:

Impacto en la Precisión del Diagnóstico e Inteligencia

La implementación del prototipo genera un impacto positivo crítico en la identificación de riesgos reales. Esto se evidencia cuantitativamente en la dimensión de Inteligencia, que obtuvo la valoración máxima de 5.0/5.0 por parte de los expertos. Al integrar fuentes como Exploit-DB y NVD, el prototipo permite que el equipo de la empresa de telecomunicaciones pase de una gestión basada en severidad teórica a una basada en explotabilidad real, eliminando el "ruido

operativo" de vulnerabilidades que no poseen un código de ataque público.

Impacto en la Eficiencia Operativa y Tiempos de Respuesta

Los resultados del análisis de correlación de Pearson (r = 0.5774) confirman que existe una relación positiva directa entre las capacidades técnicas del prototipo de framework y la eficiencia percibida en la gestión. Esto responde a la pregunta de investigación demostrando que la integración de escaneo activo y análisis de contexto:

- **Optimiza el triaje:** Reduce el tiempo de análisis manual al jerarquizar automáticamente los hallazgos en la colección vulnerability_management.
- **Focaliza la remediación:** Permite concentrar los recursos técnicos en el 10% de los activos críticos intervenidos, buscando una futura convergencia que reduzca el MTTP (actualmente en 14.4 días) y el AUT (44.7 días) en la infraestructura global.

Impacto en la Toma de Decisiones Estratégicas

Estadísticamente, la necesidad de este cambio de paradigma es significativa, con un valor de p = 0.0455 en la prueba de Chi-cuadrado. Esto indica que, para la empresa auspiciante, el prototipo no es solo una mejora incremental, sino una solución necesaria para resolver la inconsistencia y tardanza en los reportes detectada en el diagnóstico inicial; el análisis de contexto permite que la priorización sea coherente con la realidad de los servidores virtualizados Linux, asegurando la continuidad del negocio y la protección de activos críticos de la red.

En síntesis, la implementación del prototipo integrado impacta de manera favorable y significativa en la gestión de vulnerabilidades; transforma un proceso manual, fragmentado y dependiente del factor humano en un modelo operativo automatizado y contextualizado que maximiza la capacidad de defensa de la organización frente a amenazas cibernéticas vigentes.

MODELO ENTIDAD RELACIÓN

Para garantizar la flexibilidad requerida por los distintos formatos de reportes de escaneo (XML de Nmap, texto de Nikto, JSON de Exploit-DB), se diseñó un modelo de datos documental utilizando MongoDB.

A diferencia de los modelos relacionales rígidos, este enfoque permite el

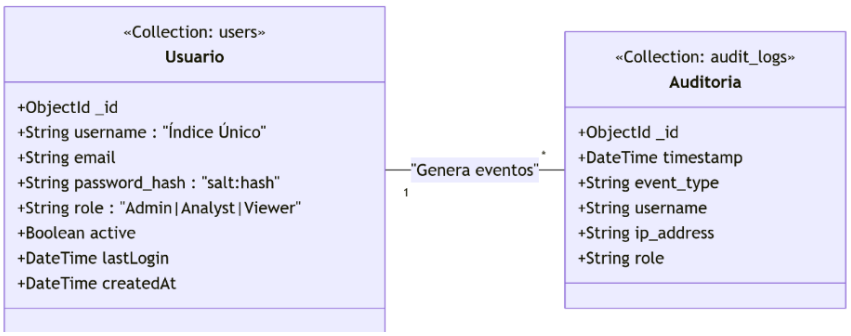
almacenamiento de objetos complejos y anidados, facilitando la normalización de datos heterogéneos en una estructura unificada. Para el prototipo se definieron 4 colecciones primarias:

1. Colección users
2. Colecciones scans y scan_results
3. Colección vulnerability_management
4. Colección Auxiliares

A continuación, se describen las colecciones principales y sus esquemas lógicos:

Colección users: Gestiona el acceso y la seguridad del sistema para los 8 usuarios del departamento. Ver imagen siguiente:

Ilustración 13
Esquema de la Colección de Usuarios y Auditoría

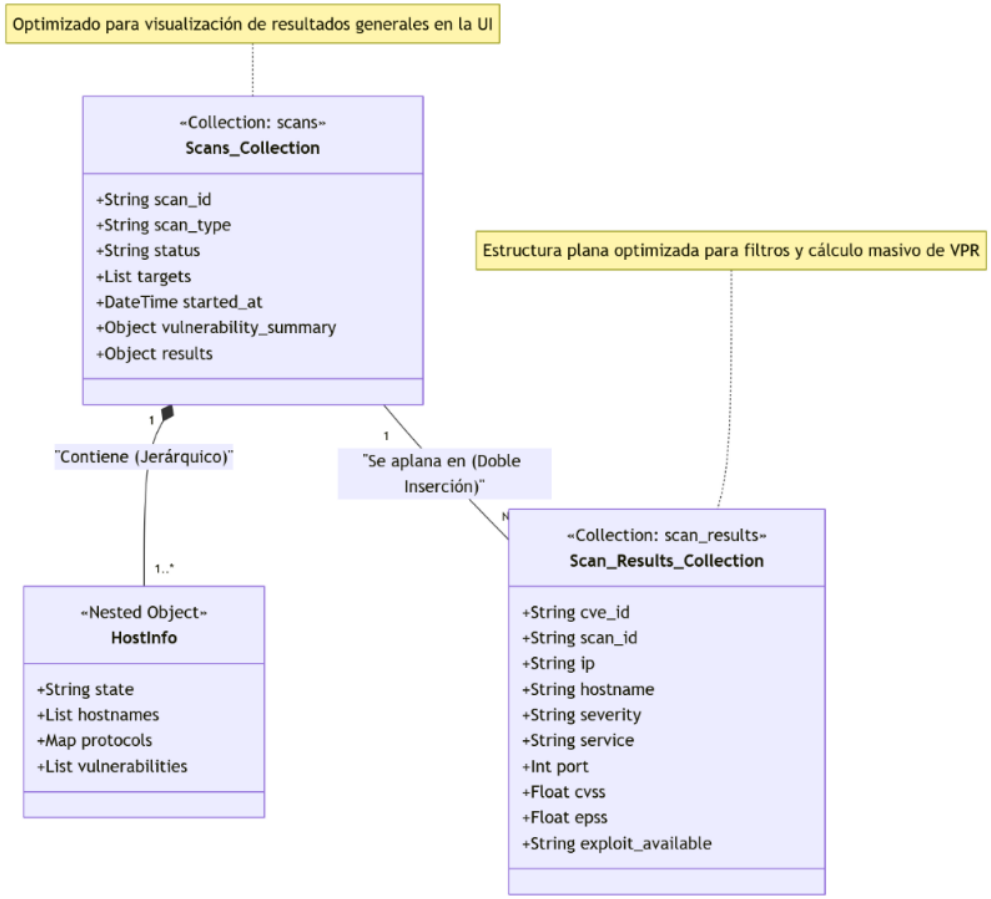


Nota: Diagrama de Entidad Relación de Usuarios y Auditoría.

- **Campos principales:** username, email, password_hash (cifrado con bcrypt), role (Admin, Analyst, Viewer)
- **Propósito:** Controlar el ciclo de vida de las cuentas y la autorización mediante roles (RBAC).

Colecciones scans y scan_results: para optimizar el rendimiento y la integridad de la información, el framework implementa un patrón arquitectural de doble inserción, vinculando las colecciones scans y scan_results mediante un identificador único denominado scan_id; este diseño permite separar la gestión operativa del almacenamiento masivo de datos técnicos, facilitando un procesamiento ágil de la información. La estructura interna de este modelo de almacenamiento y la jerarquía de sus atributos se observan en el siguiente gráfico:

Ilustración 14
Diagrama de Doble Inserción



Nota: Esquema de doble inserción del escaneo e inteligencia de Exploit-DB.

- **Colección scans (Gestión y Metadatos):** Este componente cumple con almacenar la configuración y el estado de cada tarea de escaneo. Sus campos principales incluyen el '*scan_id'*, para el tipo de escaneo '*scan_type*: Network, Web, Vuln', el estado actual '*status*: running, completed, failed', los objetivos definidos '*targets*', el identificador del usuario '*user_id*' y las marcas de tiempo de inicio y finalización.
- **Colección scan_results (Repositorio de Hallazgos):** Esta colección contiene aquellos hallazgos técnicos que son detectados durante el proceso de escaneo. Se desarrolla bajo un esquema flexible para integrar las salidas de múltiples motores, como el formato XML de Nmap o el JSON de Exploit-DB. Su estructura se basa en un objeto, denominado *results*, el cual agrupa datos críticos como el estado de los hosts, los puertos abiertos y servicios expuestos. Adicionalmente, incluye campos de vulnerabilidades específicas

como el identificador CVE, la severidad, el servicio afectado, el *output* técnico y la confirmación existente de un exploit disponible.

Esta implementación unificada permitió que el sistema se ejecute como un repositorio central de datos planos, la cual será fundamental para las siguientes fases del flujo de trabajo del framework como el análisis y priorización de riesgos.

Colección vulnerability_management: Esta colección es la parte esencial de la entidad crítica para un análisis estratégico del framework, ya que permite la transición del dato hacia una priorización orientada al negocio.

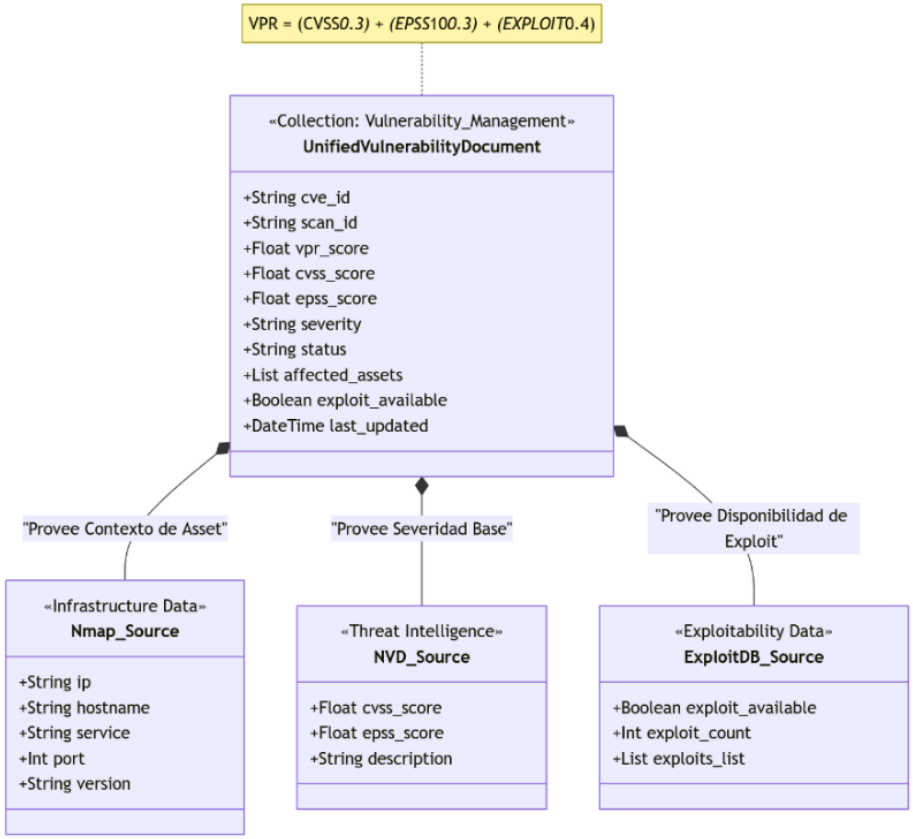
- **Campos principales:** cve_id, vpr_score (puntuación usando el sistema de priorización), cvss_score, epss_score (probabilidad de explotación), exploit_available (booleano), affected_assets y status de remediación.
- **Propósito:** Consolidar el análisis VPR mediante la correlación de las vulnerabilidades con el contexto de los activos y la inteligencia de amenazas.

Colecciones Auxiliares:

- **api_keys:** Almacena de forma segura las llaves para servicios externos como NVD y Shodan.
- **audit_logs:** Registra cada acción realizada (usuario, acción, recurso, IP) para garantizar la trazabilidad y el cumplimiento normativo.

La arquitectura de datos que fundamenta el algoritmo VPR y la interconexión de sus variables se observa en el siguiente gráfico:

Ilustración 15
Esquema de Datos implementado en el VPR



Nota: Correlación entre severidad base, explotabilidad e identificación del activo.

DIAGRAMA DE CASOS DE USO

La interacción con el prototipo funcional se define a través de tres actores principales identificados en la metodología, cada uno con privilegios y responsabilidades delimitadas para garantizar la seguridad de la información y la eficiencia operativa en la empresa de telecomunicaciones.

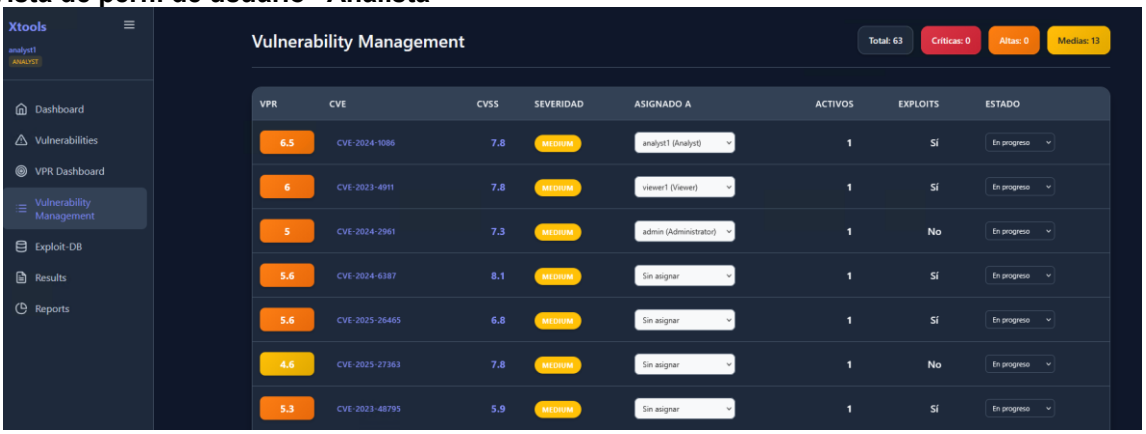
- **Analista de Ciberseguridad (Analyst):** Es el actor principal encargado de la ejecución técnica y el análisis de riesgos. Sus casos de uso incluyen:
    - **Ejecución de Escaneos Activos:** Ejecuta las tareas de descubrimiento de red mediante Nmap y escaneos de aplicaciones web con Nikto.
    - **Análisis y Priorización VPR:** Observa la matriz de riesgos; en ella, el sistema enlaza de manera automatiza los hallazgos de Nmap con inteligencia de Exploit-DB y scores EPSS.
    - **Gestión del Estado de Vulnerabilidades:** Actualiza el estatus de

remediación de los hallazgos directamente en la pestaña de "Vulnerability Management"

- **Inteligencia de Amenazas:** Realiza búsquedas manuales de exploits en la base de datos de go-ExploitDB para validar la viabilidad de ataques sobre activos específicos.

A continuación, se muestra una representación gráfica de la forma en que estos actores interactúan dentro del sistema, ilustrando cómo los hallazgos técnicos se convierten en inteligencia operativa mediante el control estatal y la visibilidad de métricas críticas.

Ilustración 16
Vista de perfil de usuario "Analista"

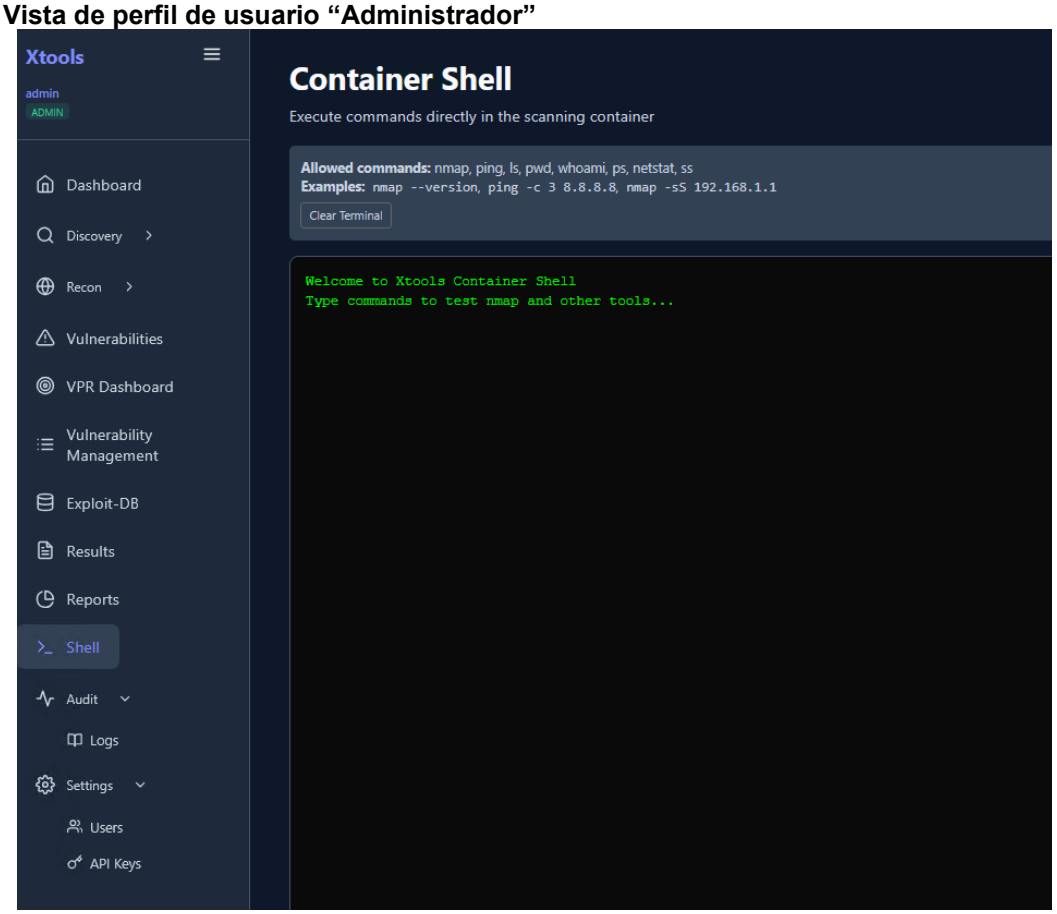


Nota: Visualización de la consola de gestión y asignación de activos para el analista.

- **Oficial de Operaciones (Administrator):** Posee control total sobre el sistema y asegura la integridad de la plataforma. Sus casos de uso incluyen:
  - **Gestión Integral de Usuarios:** Administra el ciclo de vida de las cuentas, incluyendo la creación, actualización y desactivación de usuarios en la base de datos.
  - **Control de Acceso Basado en Roles (RBAC):** Asigna y modifica los perfiles de acceso restringiendo las funciones críticas según el cargo del personal.
  - **Configuración de Inteligencia y APIs:** Gestiona las Api Keys usadas para la sincronización con fuentes externas como NVD y Exploit-DB.
  - **Supervisión de Auditoría:** Revisa los logs de auditoría garantizando la trazabilidad de todas las acciones ejecutadas dentro del sistema.

En la siguiente ilustración se muestra el Container Shell o consola de comandos del sistema, una herramienta de acceso restringido y exclusivo para el perfil de Administrador; este componente permite la ejecución directa de las funciones de gestión, configuración y supervisión técnica mencionadas anteriormente, facilitando el control operativo sobre los servicios y contenedores que componen la plataforma.

Ilustración 17
Vista de perfil de usuario "Administrador"



Nota: Visualización de la consola de administración con acceso total a los módulos.

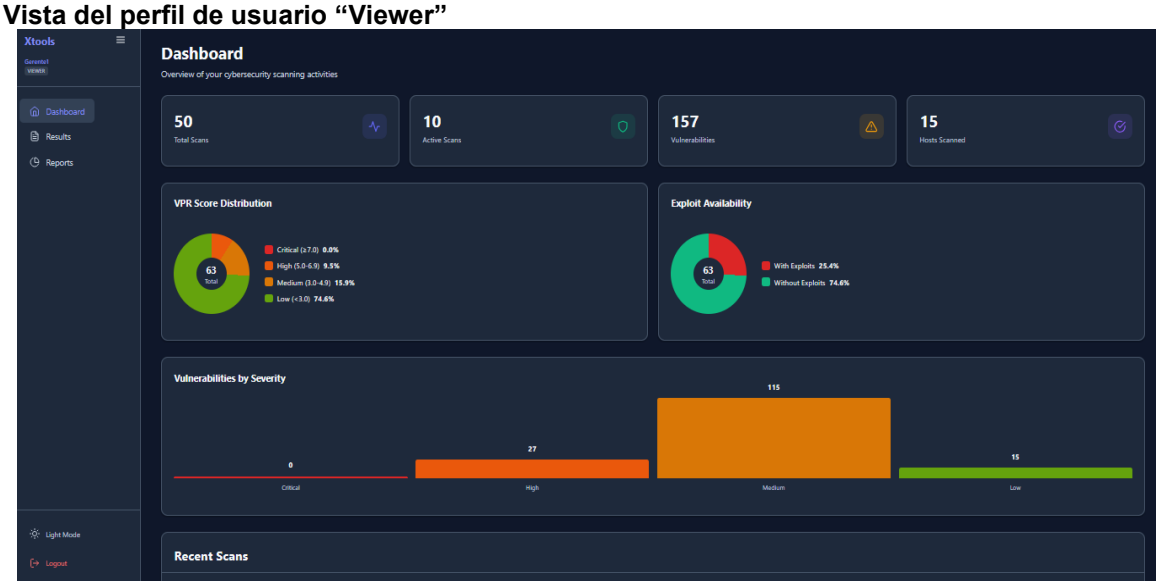
- **Gerente/Auditor (Viewer):** Representa el rol de supervisión con acceso limitado para proteger la configuración del sistema. Sus casos de uso incluyen:
    - **Visualización de Dashboard:** Accede al panel de resumen donde se presentan las métricas de densidad de vulnerabilidades, porcentajes de activos afectados y KPIs de gestión en tiempo real.
    - **Consulta de Resultados Históricos:** Revisa el historial de escaneos

completados para analizar la evolución de la postura de seguridad de la red.

- **Acceso de Solo Lectura a Reportes:** Consulta y visualiza los informes de vulnerabilidades generados por los analistas sin capacidad de modificar estados o configuraciones.

En la siguiente ilustración se detalla la interfaz de visualización a la que tiene acceso el rol de Gerente/Auditor, diseñada específicamente para la supervisión de alto nivel sin capacidad de intervención técnica.

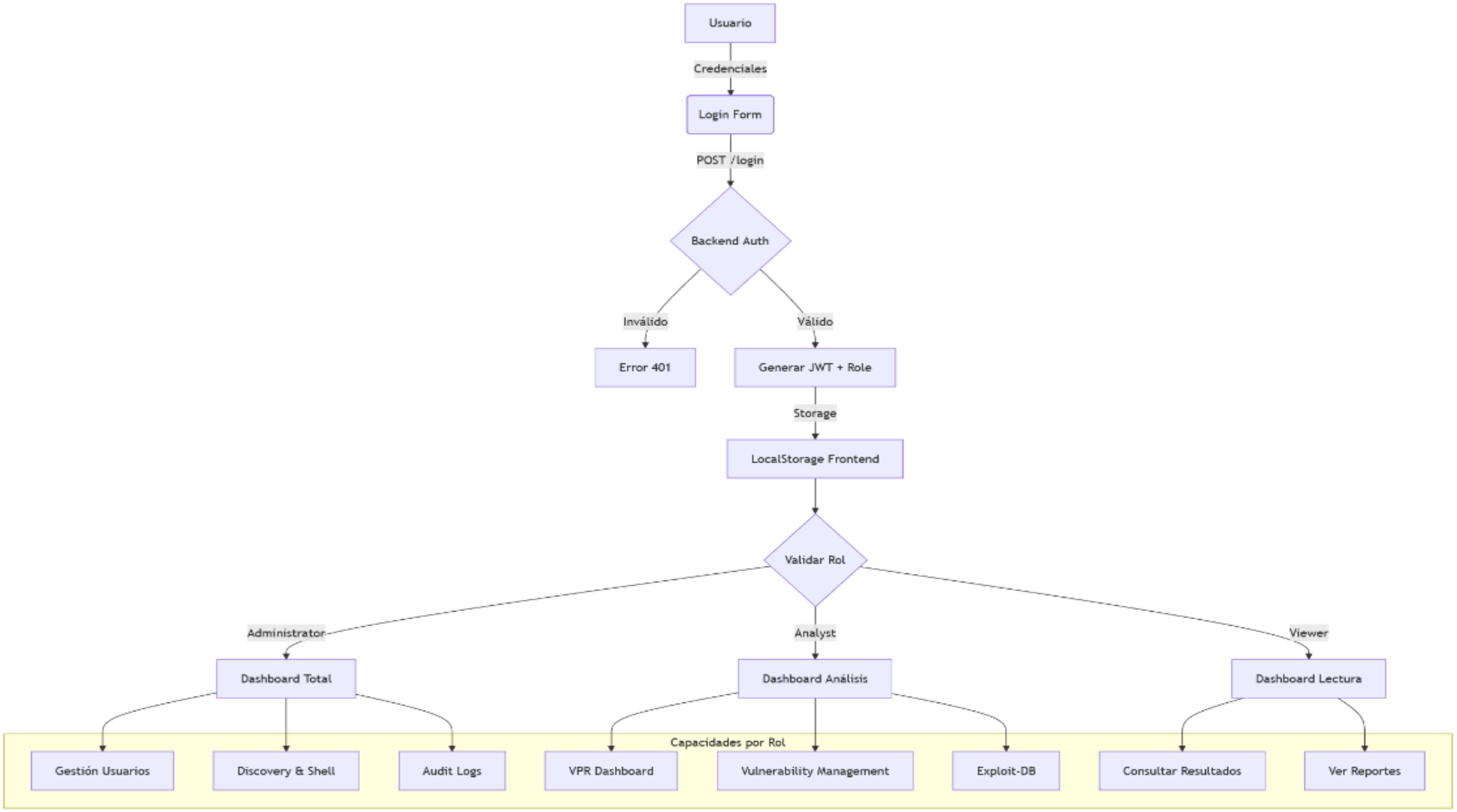
Ilustración 18
Vista del perfil de usuario "Viewer"



Nota: Visualización de indicadores clave y distribución de criticidad para observadores.

La ilustración siguiente sintetiza el flujo sistémico que abarca desde la captura de credenciales y su validación criptográfica en el backend, hasta la generación de tokens de sesión y la subsiguiente validación de privilegios. Esto permite observar cómo la arquitectura implementada garantiza una transición segura y exclusiva según el perfil de acceso (Administrador, Analista o Visualizador) asignado bajo el modelo RBAC.

Ilustración 19
Diagrama de Flujo de Interacción por Rol



Nota: Representación gráfica de la jerarquía de acceso y navegación por perfiles del sistema.

COMPONENTES DEL APLICATIVO

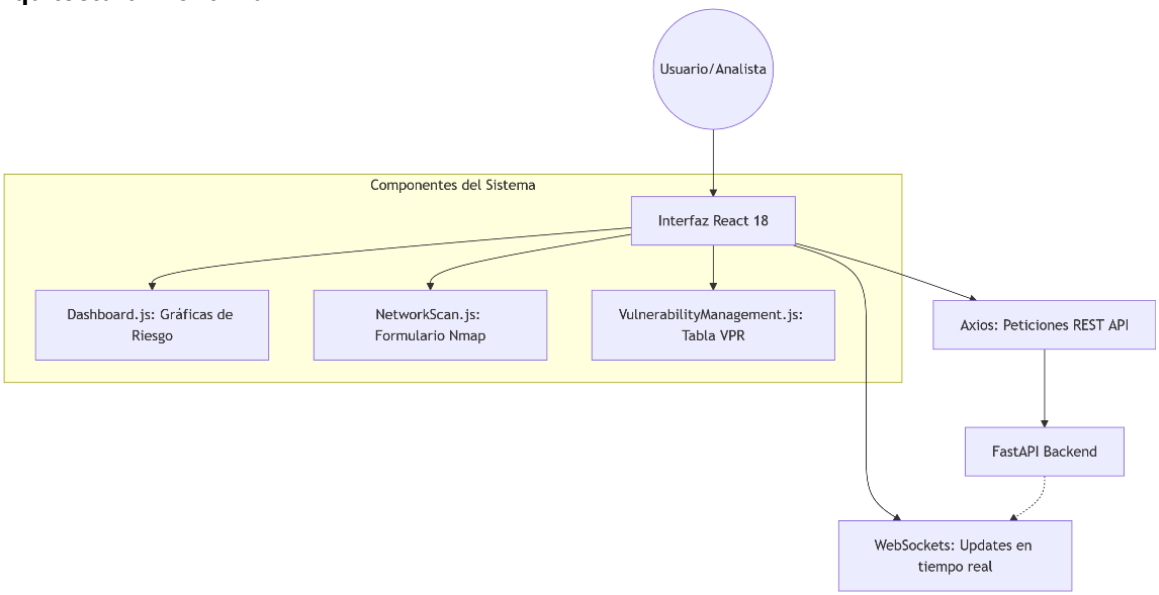
La arquitectura del prototipo funcional se fundamenta en un diseño modular desacoplado, estructurado en capas que interactúan de forma asíncrona para garantizar la escalabilidad y el rendimiento en la red de la empresa de telecomunicaciones; este diseño permite separar las responsabilidades de presentación, lógica de negocio y persistencia de datos.

Capa de Presentación (Frontend)

Desarrollada bajo la biblioteca React 18, esta capa constituye la interfaz de interacción principal para los analistas de ciberseguridad; su arquitectura se basa en componentes reutilizables y estados dinámicos para ofrecer una experiencia de usuario fluida.

En la siguiente ilustración se detalla la arquitectura técnica de la capa de presentación, estructurada para maximizar la interactividad y la respuesta técnica del sistema

Ilustración 20
Arquitectura FrontEnd



Nota: Esquema de modularización de la interfaz en React 18.

- **Módulos de Visualización:** Utiliza la librería Recharts para la representación gráfica de métricas de vulnerabilidades y dashboards de estado en tiempo real.
- **Comunicación Asíncrona:** Implementa Axios para el consumo de la API

REST del backend y la gestión de solicitudes HTTP.

- **Interconectividad en Tiempo Real:** Se integra con WebSockets mediante el hook useWebSocket.js, permitiendo recibir actualizaciones instantáneas sobre el progreso de los escaneos sin necesidad de recargar la interfaz.

Capa de Lógica de Negocio (Backend)

La inteligencia del sistema reside en su capacidad para transformar datos crudos de vulnerabilidades en métricas de prioridad accionables para el equipo de seguridad. Esta transición se materializa mediante una rutina de procesamiento que pondera variables estáticas y dinámicas para determinar el impacto real de un hallazgo en la red corporativa.

En la gráfica se presenta la implementación en Python del algoritmo VPR Score, el cual constituye la rutina central de análisis del backend.

Ilustración 21
Código: Algoritmo VPR

```python
def calculate_vpr_score(cvss: float, epss: float, has_exploit: bool, exploit_count: int) -> flo
    """Calculate VPR score based on CVSS, EPSS, and exploit availability"""
    # Base score from CVSS (40% weight)
    cvss_component = cvss * 0.4

    # EPSS component (40% weight) - convert 0-1 to 0-10 scale
    epss_component = epss * 10 * 0.4

    # Exploit availability bonus (20% weight)
    exploit_bonus = 0
    if has_exploit:
        exploit_bonus = min(2.0, exploit_count * 0.5) * 0.2 * 10  # Max 2 points

    vpr = cvss_component + epss_component + exploit_bonus
    return round(min(10.0, vpr), 1)
```

Nota: Elaboración de la lógica de pesos para el cálculo de las Vulnerabilidades.

La función ***calculate_vpr_score*** ejecuta una ponderación diseñada para equilibrar la severidad teórica con la probabilidad real de explotación:

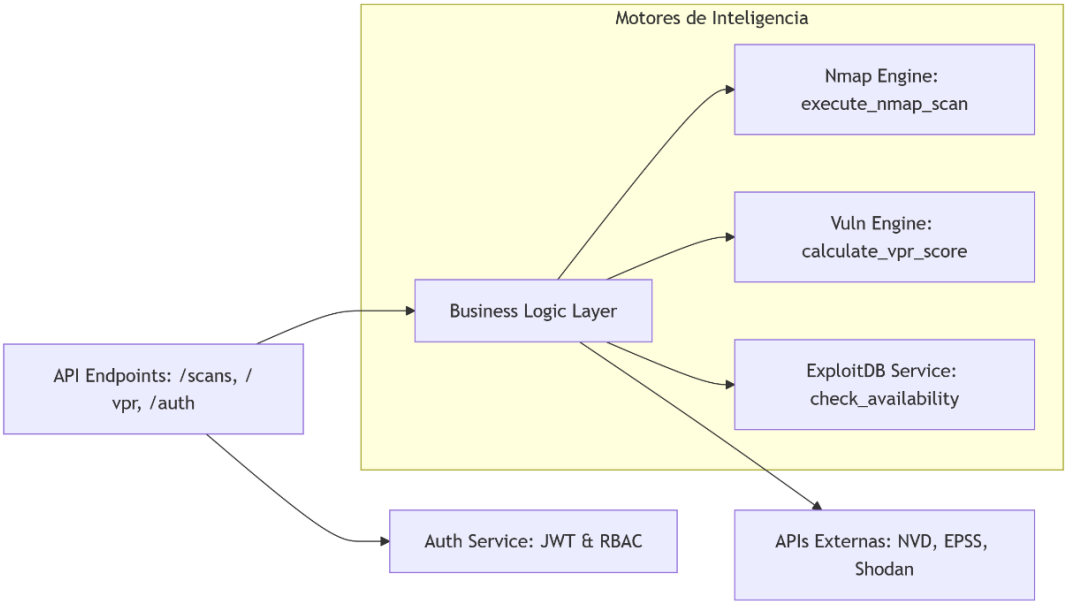
- **Severidad (40%):** Utiliza el puntaje CVSS base para establecer el impacto potencial inicial.
- **Probabilidad de Explotación (40%):** Integra el EPSS, lo cual se escaló a una base de 10 para normalizar la probabilidad de explotabilidad.
- **Disponibilidad de Exploits (20%):** Verifica la existencia de algún exploit asociado al CVE.

- **Normalización:** Se limita el resultado a un máximo de 10.0 y se aproxima a un decimal, para asegurar una escala de priorización consistente.

Esta capa se constituye como el núcleo del sistema, para la gestión de seguridad y el análisis de la información. Su diseño permite coordinar de forma eficiente una respuesta ágil ante las peticiones de la interfaz de usuario.

La siguiente ilustración detalla la distribución de estos módulos y su integración operativa.

Ilustración 22
Arquitectura Backend



Nota: Coordinación de los motores de escaneo, inteligencia e integración de las APIs externas.

- **Motores de Escaneo:** Incluye servicios como 'scan_engine.py' para integrarse con Nmap y 'vuln_scan_engine.py' para la detección de vulnerabilidades CVE con NVD.
- **Módulo de Inteligencia VPR:** Procesa el algoritmo de priorización, consultando las bases locales de Exploit-DB y calculando los scores de riesgo real.
- **Seguridad y Autenticación:** Gestiona la validación de tokens JWT y la autorización basada en roles (RBAC) para proteger los endpoints de la API.

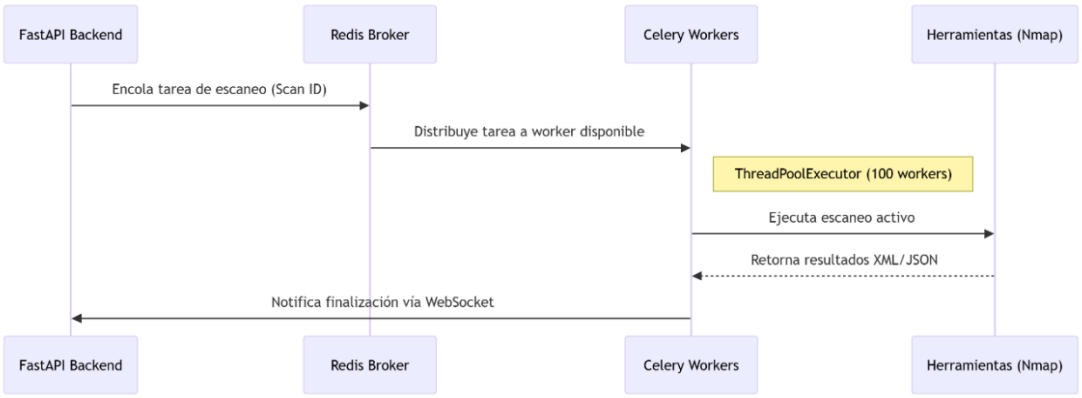
Capa de Servicios (Workers y Asincronía)

Para garantizar la estabilidad del sistema durante inspecciones masivas, se implementó una arquitectura de procesamiento asíncrono que delega las tareas

técnicas a trabajadores independientes; este diseño evita el bloqueo del backend y permite una gestión concurrente de los activos, asegurando la fluidez de la interfaz de usuario mientras se ejecutan procesos en segundo plano.

Para visualizar la gestión de tareas en segundo plano, el siguiente diagrama esquematiza el recorrido técnico de una solicitud, desde su recepción en el backend hasta su ejecución distribuida:

Ilustración 23
Flujo lógico del procesamiento de scans.



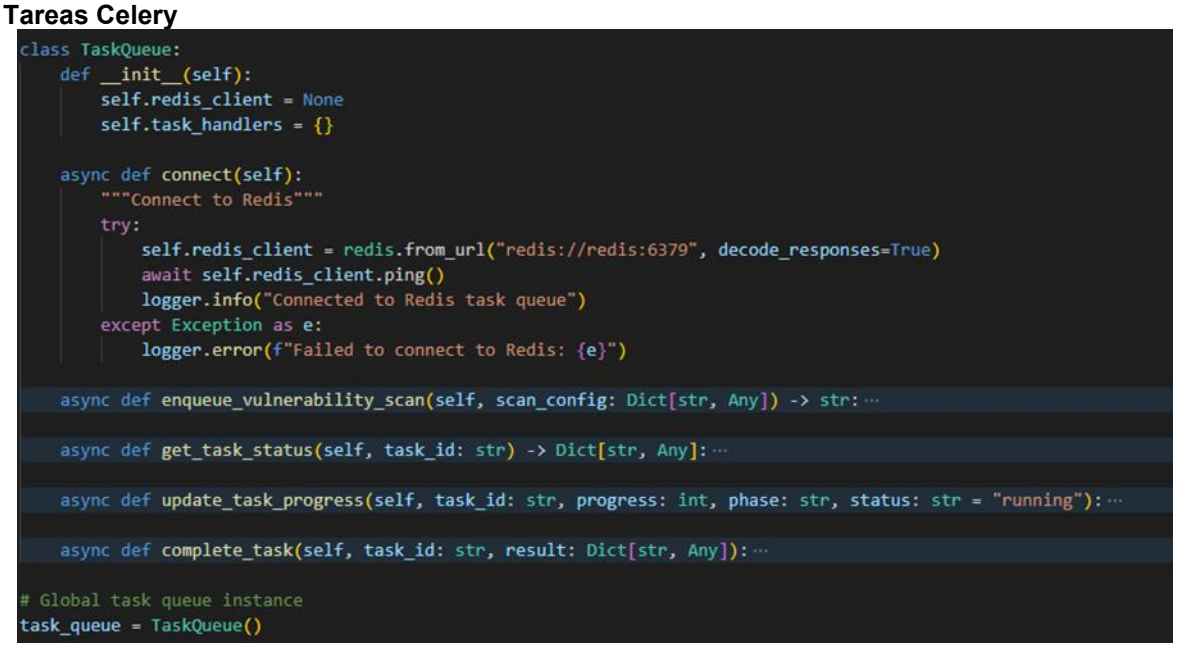
Nota: Notificación de resultados mediante los workers.

- **Gestión de Colas:** Utiliza Redis como broker de mensajes para encolar y distribuir las peticiones de escaneo iniciadas por los usuarios desde el frontend.
- **Concurrencia Controlada:** Emplea un *ThreadPoolExecutor* configurado para manejar hasta 100 trabajadores simultáneos, asegurando que los escaneos sobre una red se ejecuten de forma paralela, sin afectar la respuesta del servidor.

La gestión de tareas intensivas se desacopla del flujo principal mediante una arquitectura de colas asíncronas, garantizando que el sistema permanezca reactivo durante los diagnósticos de red; esta capa de servicios utiliza un broker para la gestión de colas logrando orquestar la ejecución de los motores técnicos en segundo plano.

A continuación, se ilustra la implementación de la clase encargada de gestionar esta cola de tareas.

Ilustración 24
Tareas Celery



```python
class TaskQueue:
    def __init__(self):
        self.redis_client = None
        self.task_handlers = {}

    async def connect(self):
        """Connect to Redis"""
        try:
            self.redis_client = redis.from_url("redis://redis:6379", decode_responses=True)
            await self.redis_client.ping()
            logger.info("Connected to Redis task queue")
        except Exception as e:
            logger.error(f"Failed to connect to Redis: {e}")

    async def enqueue_vulnerability_scan(self, scan_config: Dict[str, Any]) -> str: ···

    async def get_task_status(self, task_id: str) -> Dict[str, Any]: ···

    async def update_task_progress(self, task_id: str, progress: int, phase: str, status: str = "running"): ···

    async def complete_task(self, task_id: str, result: Dict[str, Any]): ···

# Global task queue instance
task_queue = TaskQueue()
```

Nota: Implementación del cliente Redis, usado para la gestión de los estados en escaneos.

- **enqueue_vulnerability_scan**: Introduce una nueva solicitud de análisis a la cola de tareas, lo que permite que la carga sea procesada de forma independiente por los workers.
- **get_task_status**: Permite conseguir el *status* del proceso al recuperar los metadatos y el estado actual de una tarea específica mediante su identificador único.
- **update_task_progress**: Proporciona actualizaciones en tiempo real, informando los cambios de fase, avances parciales y estados de ejecución de una tarea activa.
- **complete_task**: Finaliza formalmente el flujo de trabajo, persistiendo el resultado final y liberando los recursos de procesamiento asignados.

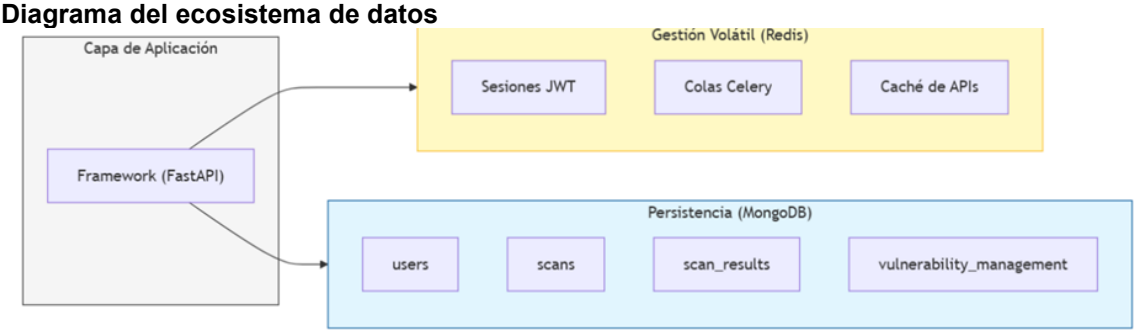
Capa de Persistencia (Almacenamiento)

Para optimizar la integridad y disponibilidad de la información, se implementó una infraestructura de persistencia híbrida que separa el almacenamiento documental del procesamiento en memoria. Esta configuración permite gestionar datos de seguridad heterogéneos mediante esquemas flexibles, garantizando la agilidad operativa en tareas asíncronas y sesiones de usuario.

El siguiente esquema detalla la organización de las colecciones y los

módulos de gestión inmediata del prototipo.

Ilustración 25
Diagrama del ecosistema de datos



Nota: Segregación entre persistencia documental y gestión volátil.

- **Persistencia Documental (MongoDB):** Almacena de forma permanente los metadatos de los escaneos, los hallazgos técnicos normalizados y la base de conocimientos de exploits. Su modelo NoSQL facilita el manejo de reportes con estructuras de datos variables.
- **Persistencia Volátil y Caché (Redis):** Además de actuar como broker de Celery, gestiona el almacenamiento temporal de resultados en curso y la expiración de sesiones de usuario, mejorando significativamente la velocidad de carga de los dashboards.

BASE DE DATOS

La infraestructura de persistencia se fundamenta en un modelo híbrido que combina el almacenamiento documental de MongoDB (versión 6.0+) con la velocidad de procesamiento en memoria de Redis; este diseño responde a la necesidad de gestionar datos de seguridad no estructurados provenientes de múltiples motores de escaneo, permitiendo una escalabilidad horizontal eficiente.

Estructura de Persistencia Documental (MongoDB)

La selección de MongoDB como base de datos se debe por su capacidad para manejar esquemas dinámicos, lo cual es esencial para integrar resultados heterogéneos de herramientas como Nmap y Nikto en un modelo de datos unificado. La definición técnica y la estructura de sus campos críticos se detallan en la siguiente tabla:

Tabla 11
Descripción de las Colecciones en MongoDB

Colección	Propósito Técnico	Atributos Críticos
users	Gestión de identidad y control de acceso basado en roles (RBAC).	username, password_hash (bcrypt), role (Admin, Analyst, Viewer).
scans	Almacenamiento de metadatos operativos y configuración de tareas de escaneo.	scan_id, scan_type, status (running, completed, failed), targets.
scan_results	Repositorio de hallazgos técnicos crudos para análisis posterior.	results (objeto anidado), hosts, ports (Nmap), vulnerabilities (Nikto).
Vulnerability _management	Entidad para el análisis VPR para la posterior gestión de vulnerabilidades basadas en NVD y Exploit-DB.	vpr_score, cvss_score, epss_score, exploit_available.
api_keys	Almacenamiento y cifrado de llaves para servicios de inteligencia externos.	service_name, api_key (encriptada).
audit_logs	Registro y trazabilidad de las peticiones para auditoría y control interno.	user_id, action, resource, ip_address, timestamp.

Optimización e Indexación

Para garantizar el rendimiento de las consultas y la integridad de los datos en la infraestructura de telecomunicaciones, se implementaron estrategias de indexación avanzada como:

Tabla 12
Índices y Optimización de Consultas

Campo Indexado	Tipo de Índice	Justificación Técnica
cve_id + scan_id	Compuesto Único	Evita la duplicidad de vulnerabilidades CVE en un mismo escaneo dentro de vulnerability_management.
scan_id	Simple	Acelera el filtrado de resultados masivos en la colección scan_results para el dashboard.
username	Único	Optimiza los tiempos de respuesta durante el proceso de autenticación de usuarios.
host_ip	Simple	Permite la correlación rápida de hallazgos técnicos con activos específicos de la red.

Gestión de Caché y Broker de Mensajería (Redis)

Redis se utiliza como una capa de persistencia volátil para soportar la lógica asíncrona y la seguridad de la sesión. Las funciones críticas soportadas por esta capa de memoria y su contribución al rendimiento general se describen a continuación:

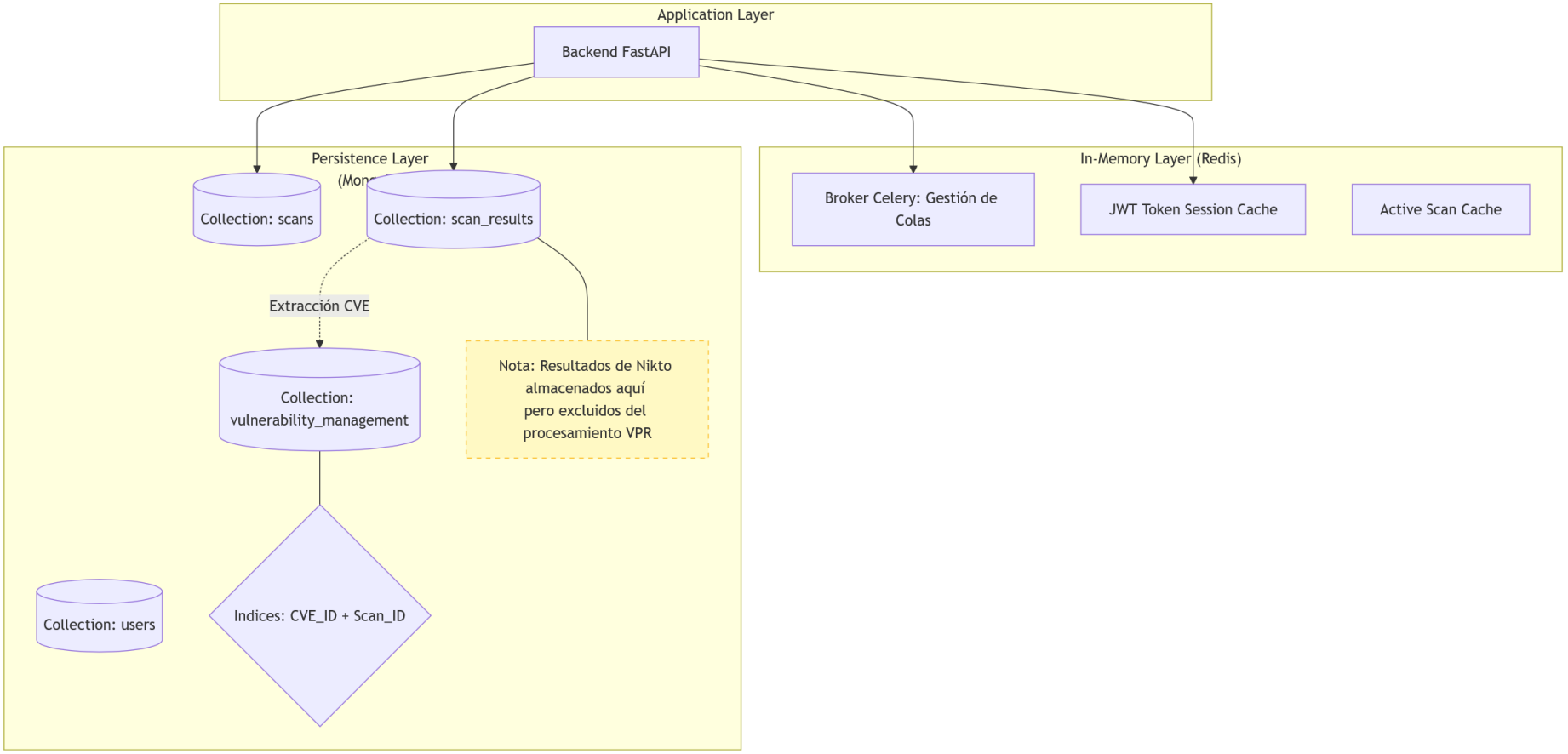
Tabla 13
Funcionalidades de la Capa de Memoria con Redis

Funcionalidad	Descripción	Beneficio Operativo
Broker Celery	Gestión de colas para tareas de escaneo de red y web	Permite la ejecución concurrente de hasta 100 request sin bloquear el backend.
Sesiones JWT	Almacenamiento temporal y gestión de expiración de los tokens otorgados para el acceso al framework.	Controla la vigencia de la sesión para mitigar riesgos de inicios de sesión no autorizados por tokens filtrados.
Caché de Escaneo	Almacenamiento temporal de resultados en estado running.	Mejora la velocidad de respuesta del dashboard al evitar consultas directas a MongoDB en tareas activas.

La estructura adopta un modelo híbrido, es decir, que combina la agilidad en las transacciones y la flexibilidad de los modelos no relacionales que se encuentran en el entorno. Esto permite que el flujo de información, desde la captura del dato hasta su consolidación en el módulo de gestión, se realice de una forma más eficiente mediante el trabajo separado entre MondoDB y Redis.

En el siguiente gráfico se detalla cómo se organiza el despliegue de estas capas de almacenamiento y memoria, manteniendo la lógica de indexación y el tratamiento que recibe cada hallazgo técnico dentro del prototipo.

Ilustración 26
Arquitectura de Datos



Nota: Interfaz entre la capa de aplicación, persistencia y gestión de memoria.

SISTEMA DE ARCHIVOS

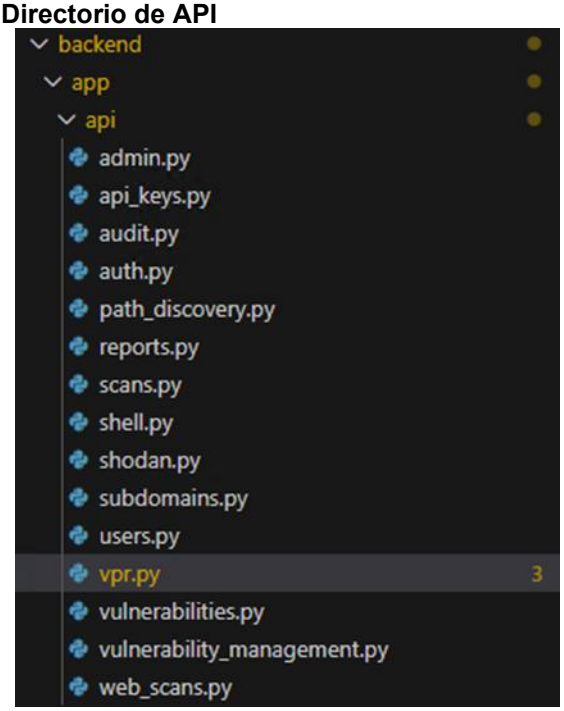
El sistema de archivos del prototipo se ha estructurado siguiendo patrones de diseño modulares para separar la lógica de negocio, la interfaz de usuario y la gestión de recursos persistentes. Esta organización garantiza que el framework sea mantenible y que los activos digitales generados se gestionen de forma segura.

Organización del Código Fuente (Estructura Lógica)

La solución se divide en dos grandes directorios raíz que separan el backend (procesamiento) del frontend (visualización):

- **Backend (/app):** Organizado bajo el framework FastAPI, segmenta las responsabilidades en:
  - **/api:** Contiene los controladores que definen los endpoints REST para escaneos, autenticación y gestión de vulnerabilidades. A continuación, se presenta la organización de los archivos que habilitan esta funcionalidad especificada dentro del prototipo.

Ilustración 27
Directorio de API



Nota: Organización modular de los controladores de las API de Backend

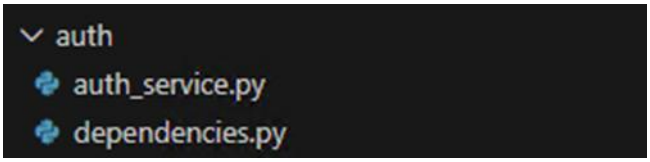
  - **/services:** Aloja los motores de lógica, como el motor de Nmap, el motor de cálculo VPR y los servicios de integración con Exploit-DB como se presenta en la siguiente captura del esquema de servicios.

Ilustración 28
Directorio de Services

Nota: Capa de servicios, motores de escaneo y auditorias.

- **/models:** Define los esquemas de datos (Pydantic) para la validación de entrada y salida.
- **/auth:** Gestiona la lógica de tokens JWT y la seguridad de acceso. En la siguiente ilustración se visualiza el esquema de archivos de autenticación.

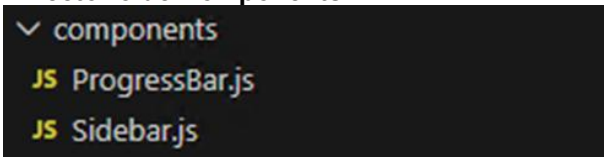
Ilustración 29
Directorio de auth



Nota: Controladores para la generación y verificación de JWT (JSON Web Token)

- **Frontend (/frontend/src):** Estructurado en React para una gestión eficiente de la interfaz:
  - **/components:** Elementos reutilizables como la barra lateral y barras de progreso. Los elementos principales del frontend se observan en la siguiente imagen.

Ilustración 30
Directorio de Components

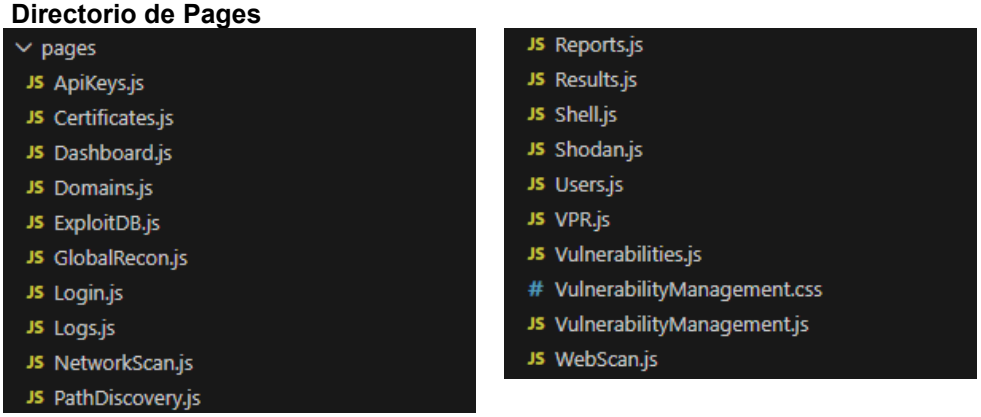


Nota: Segregación de los componentes principales para la consistencia de la interfaz

.

- **/pages:** Vistas principales para el Dashboard, escaneos de red y el módulo de gestión de vulnerabilidades. Tal como se observa en la representación gráfica del sistema de archivos, cada componente ha sido segmentado según su responsabilidad técnica.

Ilustración 31
Directorio de Pages



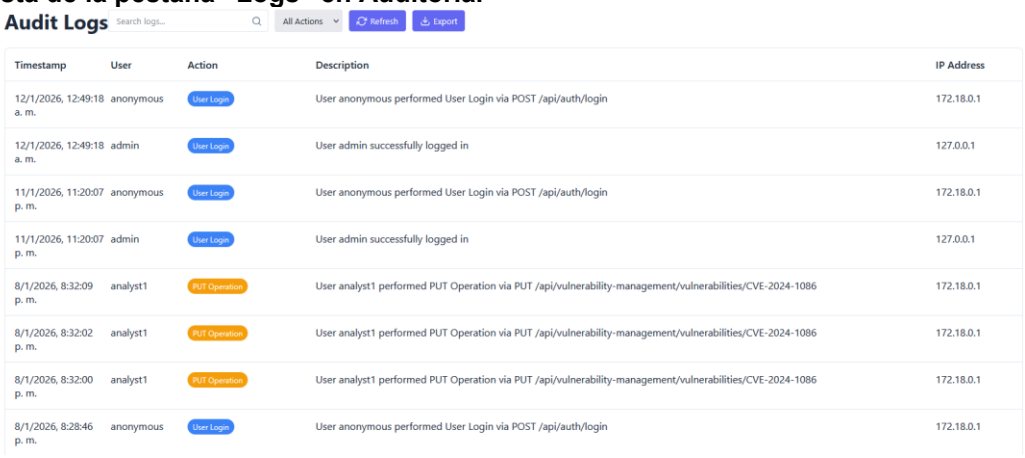
Nota: Vistas principales para la gestión operativa y administrativa del prototipo.

Gestión de Archivos Persistentes y Almacenamiento Físico

Además del código, el aplicativo gestiona el almacenamiento de recursos generados durante la operación técnica:

- **Logs de Auditoría:** Se almacenan en el directorio /var/log/xtools/; estos archivos registran cada acción del usuario para garantizar la trazabilidad y están configurados para una rotación diaria, integrándose con el servidor Syslog del contenedor para poder ser enviados a un centralizador de logs (Colector). Para asegurar la transparencia operativa del framework, el diseño del módulo de logs se despliega tal como se muestra en la ilustración adjunta.

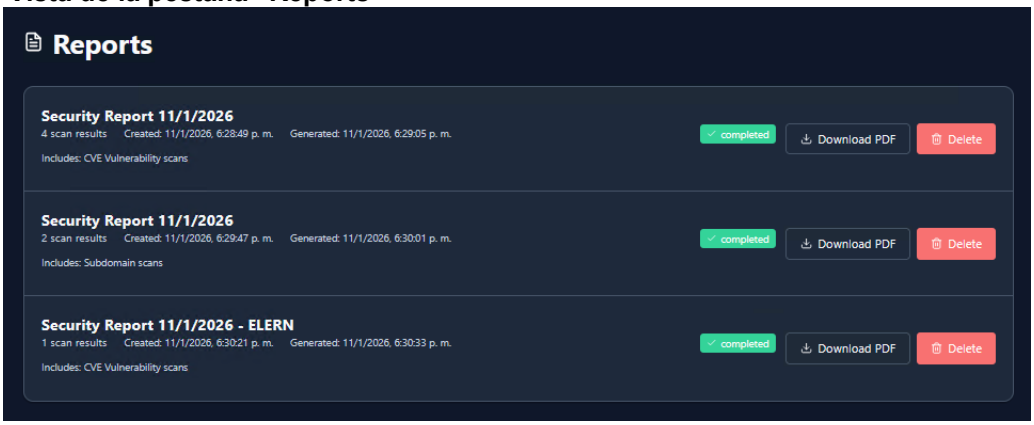
Ilustración 32
Vista de la pestaña “Logs” en Auditoría.



Nota: Consola de monitoreo de peticiones y autenticaciones.

- **Reportes de Seguridad:** Los informes generados en formatos PDF y CSV se guardan temporalmente en /app/reports/. La capacidad del sistema para generar y exportar informes técnicos detallados se ilustra gráficamente a continuación.

Ilustración 33
Vista de la pestaña “Reports”



Nota: Interfaz para la descarga de reportes en formato PDF.

La integridad de la solución tecnológica no solo depende de su capacidad de procesamiento, sino de una organización estructural que garantice la mantenibilidad y la segregación eficiente de sus componentes; esta disposición jerárquica permite que la lógica del backend, el dinamismo del frontend y los repositorios de auditoría coexistan bajo una arquitectura de directorios diseñada para la escalabilidad y la trazabilidad física de los datos.

En la siguiente ilustración se expone la composición vertical del sistema de archivos, detallando la ubicación de los módulos críticos y la gestión de recursos

persistentes que sustentan la plataforma.

Ilustración 34
Diagrama de Estructura de Archivos

Nota: Esquema de distribución modular para el mantenimiento y escalabilidad del prototipo.

SEGURIDAD DE LA SOLUCIÓN TECNOLÓGICA

Para proteger el prototipo y los datos sensibles de la infraestructura crítica dentro de la red corporativa, se implementó una estrategia de seguridad en profundidad. Esta arquitectura asegura que el sistema sea resistente ante intentos de acceso no autorizado y garantiza la confidencialidad de los hallazgos técnicos.

Controles de Seguridad de Red y Aplicación

Se aplicaron mecanismos de protección en las capas de transporte y aplicación para mitigar riesgos de interceptación de datos y manipulación externa. Para detallar estas medidas defensivas, la siguiente tabla presenta las tecnologías de cifrado y aislamiento aplicadas en el prototipo:

Tabla 14
Controles de Protección de Comunicaciones y Secretos

Control de Seguridad	Implementación Técnica	Propósito Operativo
Cifrado en Tránsito	TLS 1.3 mediante Reverse Proxy Nginx.	Asegura que la comunicación entre el navegador y el servidor sea cifrada, mitigando ataques de Man-in-the-Middle.
Protección de Secretos	Inyección de variables de entorno en tiempo de ejecución.	Evita el almacenamiento de credenciales (DB) y API Keys (NVD, Shodan) de forma explícita (hardcoded) en el código fuente.
Aislamiento de Servicios	Despliegue mediante contenedores Docker aislados.	Restringe el alcance de posibles compromisos técnicos, separando los procesos del frontend, backend y BD.
Gestión de Logs	Registro de auditoría en /var/log/xtools/.	Almacena registros detallados de cada acción realizada por los usuarios para garantizar la trazabilidad y el cumplimiento normativo.

Gestión de Identidad y Control de Acceso (IAM)

El sistema utiliza un modelo de Control de Acceso Basado en Roles (RBAC) para limitar los privilegios de los usuarios según sus funciones dentro de la organización; en la siguiente tabla se detalla los mecanismos técnicos de

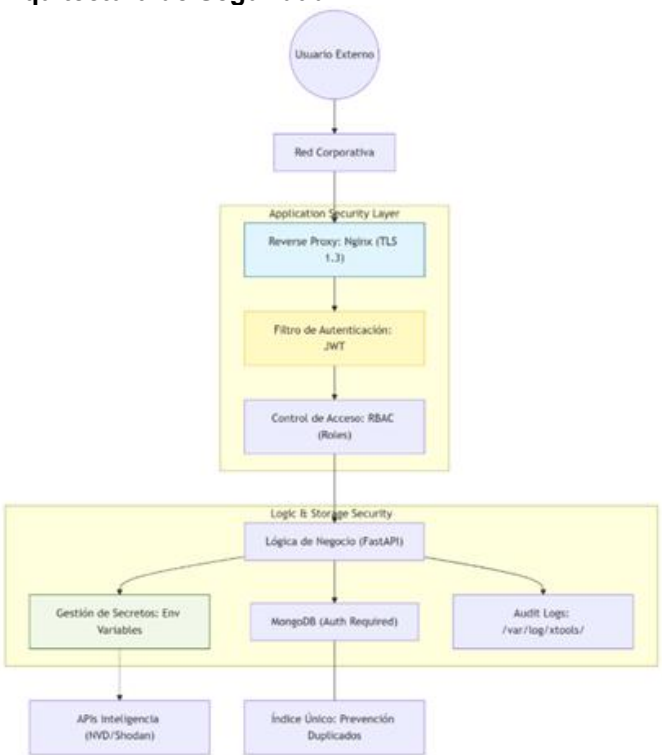
autenticación y autorización empleados para ejecutar el RBAC:

Tabla 15
Mecanismos de Autenticación y Autorización

Mecanismo	Descripción Técnica	Garantía de Seguridad
Hashing de Contraseñas	Algoritmo bcrypt con sal (salt) aleatoria.	Protege las credenciales de los usuarios contra ataques de diccionario y tablas de arcoíris en caso de una filtración de la BD.
Sesiones Stateless	JSON Web Tokens (JWT).	Permite una gestión de sesiones escalable y segura, donde el estado no se guarda en el servidor, reduciendo la superficie de ataque.
Expiración de Tokens	Tiempo de vida limitado (15 a 30 minutos).	Mitiga el riesgo de secuestro de sesión (session hijacking), forzando una re-autenticación periódica.
Autorización RBAC	Roles diferenciados (Admin, Analyst, Viewer).	Asegura el principio de "mínimo privilegio", impidiendo que roles de visualización realicen cambios en la configuración o ejecuten escaneos.

Más allá del blindaje perimetral convencional, el prototipo adopta un modelo de seguridad por capas donde la confidencialidad y la integridad convergen de forma sistémica. Esta arquitectura articula el cifrado en tránsito mediante TLS 1.3 con una validación granular de identidades basada en JWT y políticas RBAC, garantizando que el acceso a la plataforma permanezca estrictamente controlado

Ilustración 35
Arquitectura de Seguridad



Nota: Esquema de seguridad basado en TLS 1.3, JWT y RBAC.

La ilustración técnica adjunta expone la transición lógica desde el ingreso en la red corporativa hasta el almacenamiento protegido y la auditoría de los hallazgos técnicos.

ADMINISTRACIÓN DE USUARIOS

El sistema implementa un modelo de Control de Acceso Basado en Roles (RBAC) diseñado para garantizar que cada usuario interactúe únicamente con las funciones estrictamente necesarias para su cargo dentro de la empresa de telecomunicaciones. Este enfoque mitiga riesgos de manipulación interna y asegura la integridad de los resultados de las auditorías técnicas. Bajo este esquema de seguridad, la siguiente tabla detalla las capacidades operativas habilitadas para los roles de Administrador, Analista y Visualizador:

Tabla 16
Matriz de Permisos por Rol de Usuario (RBAC)

Funcionalidad	Administrator	Analyst	Viewer
Gestión de Cuentas (CRUD)	Sí	No	No
Configuración de API Keys (NVD/Shodan)	Sí	No	No
Ejecución de Escaneos (Nmap/Nikto)	No	Sí	No
Cálculo y Análisis VPR	No	Sí	Solo lectura
Gestión de Estados de Remediación	No	Sí	No
Visualización de Dashboards y KPIs	Sí	Sí	Sí
Eliminación de Historial de Escaneos	Sí	No	No

Control del Ciclo de Vida: El módulo de administración permite al rol 'Administrator' supervisar la creación, desactivación y modificación de perfiles. El estado active (booleano) en la base de datos permite revocar accesos de forma inmediata sin necesidad de eliminar los registros históricos vinculados al usuario.

AUTENTICACIÓN DE USUARIOS Y GESTIÓN DE SESIONES

La seguridad del acceso se fundamenta en un mecanismo de autenticación robusto implementado en el backend, diseñado bajo el principio de sesiones sin estado (stateless) para mejorar la escalabilidad y reducir la superficie de ataque.

Proceso de Validación y Hashing: Para la protección de las credenciales, el sistema no almacena contraseñas en texto plano. Se utiliza el algoritmo bcrypt con un factor de trabajo (salt) aleatorio para generar un hash irreversible. En base a esto, cuando se inicia sesión, el sistema comienza a realizar una comparación entre el hash almacenado en la colección users de MongoDB y la contraseña proporcionada por el usuario.

Gestión de Sesiones (JSON Web Tokens): Para la gestión de identidades

digitales se trabaja bajo un procedimiento de autenticación híbrida, que valida las credenciales y define el alcance operativo del usuario a través de roles. Lo que garantiza que cada sesión tenga un periodo temporal estricto que son necesarios para el control de acceso basado en roles (RBAC) dentro de la red corporativa. Una vez validada la identidad, el sistema emite un token JWT firmado digitalmente que contiene los claims del usuario (ID, username y rol). Definiendo los siguientes puntos:

- **Temporalidad:** Para mitigar riesgos de secuestros de sesión, los tokens caducan automáticamente cada 15 minutos.
- **Autorización en cada petición:** El cliente (Frontend) incluye este token en el encabezado Authorization de cada solicitud HTTP. El backend valida la firma del token antes de realizar cualquier actividad dentro del prototipo.

El desarrollo de esta práctica de autenticación influye en la centralización verificable de identidades mediante un soporte híbrido de validaciones locales; este flujo implementa un control de errores como la de una excepción HTTP 401 ante credenciales inválidas. Tras la validación, el sistema genera un token JWT que considera el rol del usuario para establecerle una expiración al sistema de 30 minutos para reducir riesgos de persistencia no deseada.

Finalmente, la API retorna un token de tipo Bearer, permitiendo una comunicación sin estado y segura entre la interfaz y el backend. En la siguiente ilustración se detalla este proceso llevado en código.}

Ilustración 36
Código: Api /login

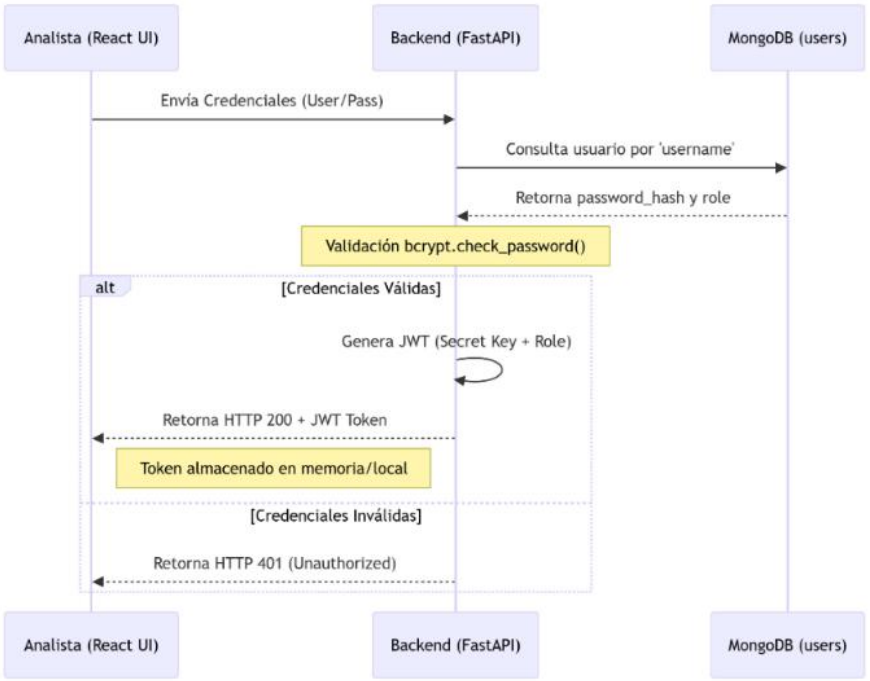
```python
@router.post("/login", response_model=TokenResponse)
async def login(request: LoginRequest):
    user = None

    if request.auth_type == "local":
        user = await auth_service.authenticate_local(request.username, request.password)
    elif request.auth_type == "entraid":
        user = auth_service.authenticate_entraid(request.password)  # token in password field

    if not user:
        logger.warning(f"Authentication failed for {request.username} via {request.auth_type}")
        raise HTTPException(status_code=401, detail="Invalid credentials")

    access_token_expires = timedelta(minutes=30)
    access_token = auth_service.create_access_token(
        data={"sub": user["username"], "role": user["role"]},
        expires_delta=access_token_expires
    )

    # Log successful login to audit
    from app.services.audit_service import AuditService
    audit_service = AuditService()
    await audit_service.log_login(user["username"], "127.0.0.1", True)

    logger.info(f"User {user['username']} logged in successfully via {request.auth_type}")

    return {
        "access_token": access_token,
        "token_type": "bearer",
        "expires_in": 1800
    }
```

Nota: Implementación de generación de JWT y registro automático de eventos de acceso.

Bajo esta lógica operativa se aplica el uso de sesiones persistentes hacia un modelo dinámico de tokens JWT, garantizando que cada petición sea validada de forma independiente y segura. El diagrama de secuencia expuesto a continuación sintetiza la interacción técnica entre la interfaz y el backend durante el ciclo de autenticación.

Ilustración 37

Diagrama de Flujo de Autenticación



Nota: Esquema de intercambio de tokens JWT y manejo de estados de autorización.

Combinando los elementos expuestos en este capítulo se establece una arquitectura sólida, concebida precisamente para cubrir las exigencias de alta disponibilidad de una infraestructura de telecomunicaciones. Esta combinación entre persistencia híbrida, el procesamiento asíncrono de tareas y el modelo de seguridad de fondo no solo garantiza la eficiencia en la priorización de las vulnerabilidades por medio del algoritmo VPR, sino que también asegura la integridad y la trazabilidad de cada evento. Con la consolidación de esta base tecnológica, el prototipo queda validado conceptualmente para transitar hacia la fase de experimentación y evaluación de resultados, donde se medirá su impacto real en la reducción de riesgos operativos

CONCLUSIONES

Se logró desarrollar e implementar el prototipo de framework bajo una arquitectura de microservicios, integrando tecnologías como FastAPI, MongoDB, escaneo activo y fuentes de inteligencia de amenazas; esta implementación técnica permitió alcanzar cada objetivo específico mediante la construcción de un backend y frontend funcional, lo que resultó consolidar el Objetivo General al entregar una herramienta capaz de disponer escaneo activo y priorización de las vulnerabilidades. De esta forma, se comprueba que el alcance propuesto fue completamente alcanzado, asegurando un modelo que responde a los requisitos operativos definidos en la propuesta.

Como resultado a la pregunta de investigación planteada, los resultados obtenidos confirman que la integración de escaneo activo con análisis contextual es la solución efectiva para la gestión de vulnerabilidades en la empresa de telecomunicaciones. La evidencia recolectada demuestra que el prototipo resuelve la problemática de priorización que afectaba a los servidores virtualizados Linux. Por lo tanto, se respalda que la hipótesis de una arquitectura dinámica y automatizada es de mayor valor ante los métodos manuales, conectando de forma directa cada resultado con el problema estudiado.

La integración de los componentes técnicos desarrollados incluyendo backend, frontend, APIs y módulos de escaneo; demuestra que el framework no solo cumple con los requisitos establecidos en los objetivos del proyecto, sino que constituye una herramienta operativa para ser utilizada como referente en la gestión proactiva de riesgos de seguridad.

El prototipo se adapta a una escalabilidad hacia entornos en la nube, es decir no sólo considera servidores virtualizados del entorno local, sino a una cobertura de escaneo a infraestructuras híbridas y de diferentes sectores productivos. Esto asegura que se logra adaptar a las necesidades de la organización, garantizando una amplia seguridad en el entorno Linux original y a la complejidad de las redes corporativas actuales.

RECOMENDACIONES

Una vez finalizado el prototipo y puesto en producción, se proponen las siguientes recomendaciones:

- La empresa auspiciante sustituya la autenticación local por una integración nativa con servicios de identidad centralizada (como Microsoft Entra ID), lo que permitiría estandarizar el control de acceso, implementar Single Sign-On (SSO) y gestionar permisos de forma más robusta y ágil.
- La empresa auspiciante migre la orquestación de los workers asíncronos a un entorno de contenedores elásticos (como Kubernetes o AWS ECS); esto optimizaría las ejecuciones en infraestructuras masivas, permitiendo que el sistema asigne recursos dinámicamente según la demanda de escaneo sin afectar la interfaz del usuario.
- Se sugiere al Equipo de Seguridad (SOC) de la empresa auspiciante defina e incorpore nuevas capas de detección y telemetría de amenazas que vayan más allá de la disponibilidad de exploits, incluyendo tendencias de ataques globales; esto permitiría al analista correlacionar la vulnerabilidad técnica con el contexto dinámico del entorno para un diagnóstico de riesgo más granular.
- Se recomienda a los estudiantes de la “Universidad Católica Santiago de Guayaquil” y nuevos investigadores utilizar este prototipo como base investigativa y/o referencia técnica para implementar módulos avanzados de respuesta automática o inteligencia artificial predictiva, aprovechando la arquitectura modular ya establecida en los servicios de la API.

REFERENCIAS BIBLIOGRÁFICAS

Al, B. e., Brazil, E., & Chodorow, K. (2022). MongoDB: The Definitive Guide: Powerful and Scalable Storage in the Cloud. (3ra ed.). O'Reilly Media.

Alfredo Estraño, A. F. (2002). FEDUPEL.

Arias, F. (2020). El Proyecto de Investigación: Introducción a la metodología científica. Editorial Episteme.

Foundation., O. (2025). OWASP Top 10:2025 - Introduction.

Hernández-Sampieri, R., & Mendoza, C. (2018). Metodología de la investigación: Las rutas cuantitativa, cualitativa y mixta. McGraw-Hill Education.

Horváth, M., Sakhnenko, V., & GurbáĬ, F. (2024). IEEEXPLORE. Obtenido de https://ieeexplore.ieee.org/abstract/document/10900892

Latin Science. (2024). Seguridad y autenticación en arquitecturas de microservicios mediante JSON Web Tokens (JWT). Revista de Tecnología y Ciencia Aplicada.

Lubanovic, B. (2024). FastAPI Modern Python Web Development. Sebastopol, CA: O'Reilly Media.

Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Oliveira, H. (2023). Redis Depth Guide: High-Performance Data Structures and Real-Time Architectures. Packt Publishing.

Perkins Santiago Haro Parra, L. A. (2024). Medidas de seguridad para la protección a los servidores de institutos. Polo del Conocimiento, 23.

Pressman, R. (2021). Software Engineering: A Practitioner's Approach (9.ª ed. ed.). McGraw-Hill Education.

Pressman, R., & Maxim, B. (2020). Ingeniería de software: Un enfoque práctico. (9. ed., Ed.) McGraw-Hill Education.

Programación Web del Frontend al Backend. (2023). En R. J. Celi Párraga. Grupo AEA.

Provos, N. (2023). Evolving Bcrypt: Adaptive Hashing for Modern Security and Brute-Force Resistance. No Starch Press.

Ramirez, S. (2024). FastAPI Documentation: High performance, easy to learn, fast to code, ready for production. Obtenido de Tiangolo: https://fastapi.tiangolo.com/

Raymond. (2025). SERVER SPACE. Obtenido de https://serverspace.io/es/support/help/how-to-deploy-and-configurate-reverse-proxy-server/

Retek. (2019). Service LayerTM 11.0.2 Programmer's Guide. Retek Inc.

Sampieri, R. H. (2014). Metodología de la Investigación. McGRAW-HILL.

Sánchez Carlessi, H., Reyes Romero, C., & Mejía Sáenz, M. (2018). Influencia en la elaboración de prototipos para el desarrollo de proyectos. Revista de Investigación en Psicología, 21(2), 235-248.

Seitz, J., & Arnold, T. (2021). Black Hat Python: Python Programming for Hackers and Pentesters. (2da ed.). No Starch Press.

SENDOYA, B. A. (2024). UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA – UNAD. Obtenido de https://repository.unad.edu.co/jspui/bitstream/10596/65205/1/bacruzs.pdf

Sistemas operativos : Linux teoría y práctica. (2019). En F. A. Sandra Liliana Allende, Sistemas operativos : Linux teoría y práctica.

Vargas-Cordero, Z. (2021). La investigación aplicada: una forma de conocer las realidades con evidencia científica. Revista Educación.

Ward , B. (2021). How Linux Works: What Every Superuser Should Know. (3ra ed.). No Starch Press.

Universidad Pedagógica Experimental Libertador. (2021). Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales. Caracas: FEDUPEL.

ANEXOS

Anexo 1
Encuesta de Satisfacción utilizada

ENCUESTA DE SATISFACCIÓN

Universidad Católica Santiago de Guayaquil
Facultad de Ingeniería
Empresa de estudio: Servicio de Telecomunicaciones SETEL .S.A

OBJETIVO Evaluar la percepción del personal técnico respecto a la funcionalidad, eficiencia y usabilidad del prototipo de framework propuesto para la gestión y priorización de vulnerabilidades, en comparación con el proceso manual actual.

INSTRUCCIONES

- Lea detenidamente cada pregunta antes de responder.
- Marque la casilla que mejor represente su criterio.
- Responda con total sinceridad; la información es anónima y será utilizada únicamente con fines académicos para el procesamiento estadístico.

1. ¿Considera usted que el proceso manual actual de gestión de vulnerabilidades presenta retrasos que afectan la seguridad de la red? *

- ◯ SÍ
- ◯ NO

2. ¿Cree usted que la implementación del framework propuesto es necesaria para optimizar las operaciones del departamento de TI? *

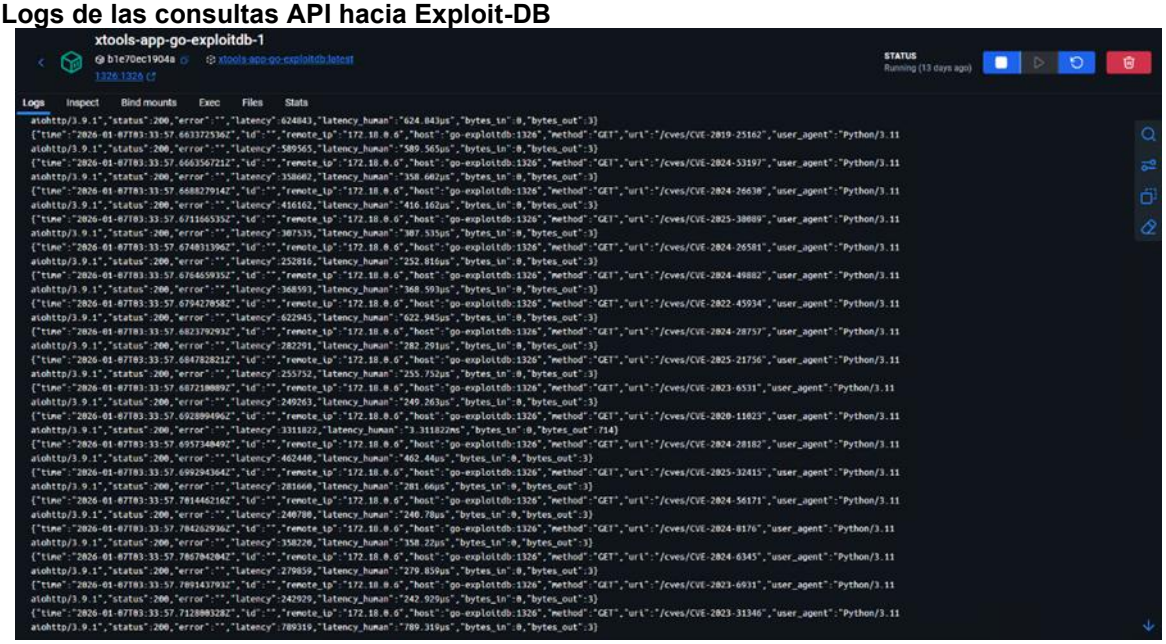
- ◯ SÍ
- ◯ NO

3. *Evalúe los siguientes enunciados marcando el nivel de acuerdo según la escala:*
1: Totalmente en Desacuerdo
2: En Desacuerdo
3: Indiferente
4: De Acuerdo
5: Totalmente de Acuerdo *

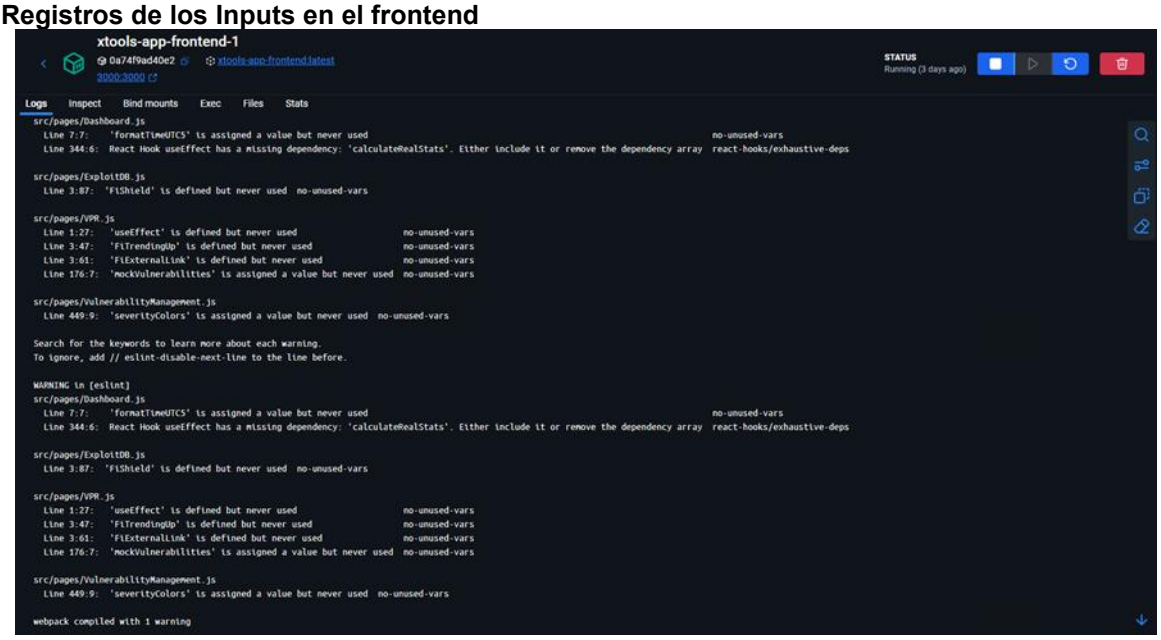
	1	2	3	4	5
La integración de escaneo activo y herramientas como Nmap en el framework facilita la detección de puertos y servicios expuestos en comparación con el método anterior.	◯	◯	◯	◯	◯
El uso de la inteligencia de Exploit-DB dentro del sistema permite identificar con mayor rapidez qué vulnerabilidades representan un riesgo real y explotable.	◯	◯	◯	◯	◯
El análisis de contexto proporcionado por el prototipo ayuda a priorizar eficazmente las vulnerabilidades críticas sobre aquellas de menor relevancia.	◯	◯	◯	◯	◯
Los reportes y el Dashboard generados por el sistema son claros, precisos y facilitan la toma de decisiones para la remediación.	◯	◯	◯	◯	◯
El framework contribuye a mejorar los tiempos de respuesta (MTTP) ante incidentes de seguridad en los servidores Linux virtualizados	◯	◯	◯	◯	◯
En general, el prototipo funcional cumple con los requerimientos necesarios para ser implementado en el entorno de producción de la empresa.	◯	◯	◯	◯	◯

Anexo 2
Contenedor levantado con los servicios indicados

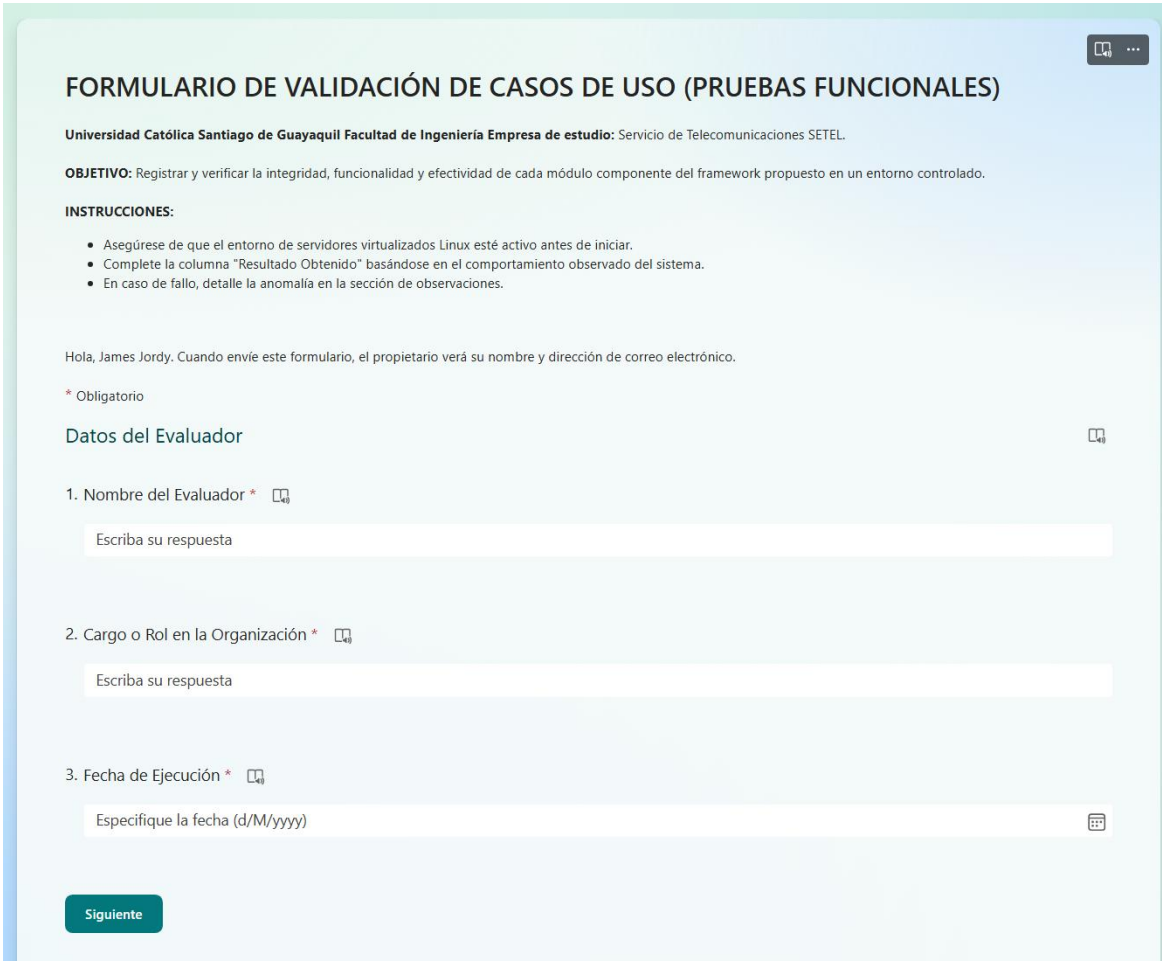
Anexo 3
Logs de las consultas API hacia Exploit-DB



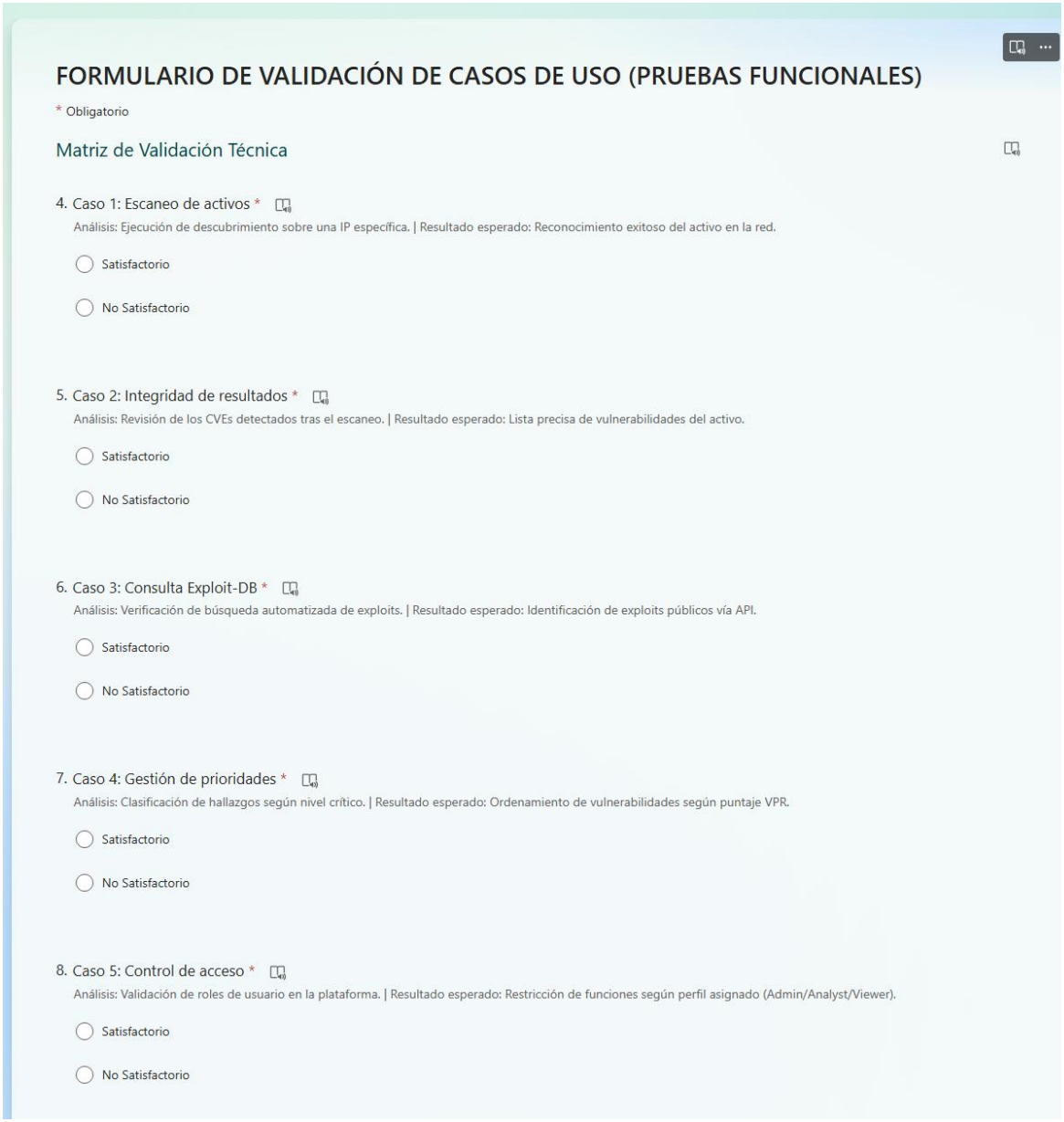
Anexo 4
Registros de los Inputs en el frontend



Anexo 5
Sección 1 del Formulario de Validación de Casos de Uso



Anexo 6
Matriz de Validación Técnica de Casos de Uso



FORMULARIO DE VALIDACIÓN DE CASOS DE USO (PRUEBAS FUNCIONALES)

* Obligatorio

Matriz de Validación Técnica

4. Caso 1: Escaneo de activos *
Análisis: Ejecución de descubrimiento sobre una IP específica. | Resultado esperado: Reconocimiento exitoso del activo en la red.
- Satisfactorio
- No Satisfactorio

5. Caso 2: Integridad de resultados *
Análisis: Revisión de los CVEs detectados tras el escaneo. | Resultado esperado: Lista precisa de vulnerabilidades del activo.
- Satisfactorio
- No Satisfactorio

6. Caso 3: Consulta Exploit-DB *
Análisis: Verificación de búsqueda automatizada de exploits. | Resultado esperado: Identificación de exploits públicos vía API.
- Satisfactorio
- No Satisfactorio

7. Caso 4: Gestión de prioridades *
Análisis: Clasificación de hallazgos según nivel crítico. | Resultado esperado: Ordenamiento de vulnerabilidades según puntaje VPR.
- Satisfactorio
- No Satisfactorio

8. Caso 5: Control de acceso *
Análisis: Validación de roles de usuario en la plataforma. | Resultado esperado: Restricción de funciones según perfil asignado (Admin/Analyst/Viewer).
- Satisfactorio
- No Satisfactorio

DECLARACIÓN Y AUTORIZACIÓN

Nosotros, **Baque Ortiz, Lizbeth Melissa** con C.C: # **0929460467** y **Ruiz Panezo, James Jordy** con C.C: # **0953299831** autores del proyecto de tecnología de información, **Desarrollo de un Prototipo de Framework que integre escaneo activo, inteligencia de exploit-DB y análisis de contexto que priorice y optimice la gestión de vulnerabilidades en servidores virtualizados LINUX a ser aplicado en una empresa de telecomunicaciones de la ciudad de Guayaquil,** requerido para la obtención del título de **Ingeniero en Ciencias de la Computación** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 27 de febrero de 2026

Nombre: **Baque Ortiz, Lizbeth Melissa**
C.C: **0929460467**

Nombre: **Ruiz Panezo, James Jordy**
C.C: **0953299831**

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

TEMA Y SUBTEMA:	**Desarrollo de un Prototipo de Framework que integre escaneo activo, inteligencia de exploit-DB y análisis de contexto que priorice y optimice la gestión de vulnerabilidades en servidores virtualizados LINUX a ser aplicado en una empresa de telecomunicaciones de la ciudad de Guayaquil**		
AUTOR(ES)	**Baque Ortiz, Lizbeth Melissa** **Ruiz Panezo, James Jordy**		
REVISOR(ES)/TUTOR(ES)	**Yong Yong, Byron Severo**		
INSTITUCIÓN:	**Universidad Católica de Santiago de Guayaquil**		
FACULTAD:	**Ingeniería**		
CARRERA:	**De Computación**		
TITULO OBTENIDO:	**Ingeniero en Ciencias de la Computación**		
FECHA DE PUBLICACIÓN:	27 de **febrero** de **2026**	**No. DE PÁGINAS:**	**67 p.**
ÁREAS TEMÁTICAS:	**Seguridad informática, Tecnología de la información, Gestión de riesgos, Telecomunicaciones, Análisis de datos, Automatización.**		
PALABRAS CLAVES/ KEYWORDS:	**Virtualización, Ciberseguridad, Framework, Telecomunicaciones.**		

RESUMEN/ABSTRACT: La propuesta de nuestro proyecto de titulación se basó en el desarrollo de un framework para la gestión y priorización de vulnerabilidades en los servidores virtualizados Linux, empleado para una empresa de telecomunicaciones de la ciudad de Guayaquil. Debido al progresivo avance de la dependencia tecnológica, se desea robustecer la seguridad para los activos críticos de la organización como su continuidad operativa. La investigación se enfocó en un análisis descriptivo y aplicado, la cual se realizó bajo el proceso actual de la gestión de vulnerabilidades mediante una metodología mixta, la cual implicó un análisis cualitativo para comprender el contexto operacional y un análisis cuantitativo para observar las métricas de los indicadores de avance. Para el desarrollo tecnológico se aplica una metodología de prototipado que permite el control ordenado de las etapas diseño, implementación, prueba y validación del prototipo. El estudio inicia con la revisión de antecedentes y conceptos claves en seguridad informática, orientado en la gestión de vulnerabilidades en entornos virtualizados Linux. Se examinan las limitaciones del enfoque actual y se identifican las razones vinculadas a la falta de automatización y priorización contextual adecuada; Se desarrolla una propuesta para un sistema que combina escaneo activo, inteligencia sobre amenazas y análisis contextual con el fin de clasificar las vulnerabilidades de acuerdo con su criticidad y su importancia para la empresa. El framework propuesto facilita la elaboración de reportes confiables y efectivos, mejora la supervisión operativa de los técnicos y optimiza la utilización de recursos en el campo de la ciberseguridad. El prototipo presentado resulta para mejorar el manejo de vulnerabilidades en los servidores virtualizados de la compañía de telecomunicaciones, ofreciendo ventajas operativas y estratégicas. Este estudio fundamenta de manera consistente futuras investigaciones y avances en la protección digital empresarial.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:**+593-99-5646838 **Teléfono:**+593-99-0480411	E-mail: **meli.13baque@gmail.com** E-mail: **jamesruizpanezo@gmail.com**
CONTACTO CON LA INSTITUCIÓN (C00RDINADOR DEL PROCESO UTE)::	**Toala Quimí, Edison José** **Teléfono:** +593-990-976776 **E-mail:** edison.toala@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	