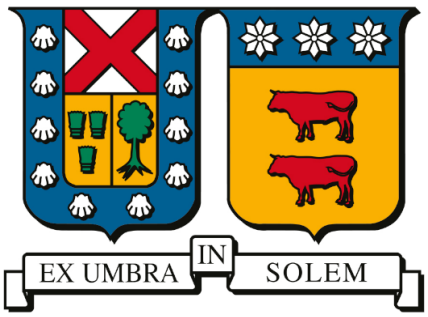
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE



DISEÑO E IMPLEMENTACIÓN DE UN FRAMEWORK INTEGRAL DE CIBERSEGURIDAD PARA LA EVALUACIÓN Y PRIORIZACIÓN CONTEXTUAL DE VULNERABILIDADES EN ENTORNOS DIGITALIZADOS

PAOLA DEL CARMEN SILVESTRE CANDIA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERIO EN INFORMATICA

PROFESOR GUÍA : JAVIER ALEXIS MALDONADO CARMONA

MARZO 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): X Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: DISEÑO E IMPLEMENTACIÓN DE UN FRAMEWORK INTEGRAL DE CIBERSEGURIDAD PARA LA EVALUACIÓN Y PRIORIZACIÓN CONTEXTUAL DE VULNERABILIDADES EN ENTORNOS DIGITALIZADOS

Nombre del candidato(a): Paola Del Carmen Silvestre Candia Carrera /

Grado: Ingeniería Informatica , Pregrado

Campus: Concepcion Departamento: Electrónica e Informática 2.-

VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Javier Alexis Maldonado Carmona, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente DEJO CONSTANCIA que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo NO contiene información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo CONTIENE información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Digitally signed by Javier Maldonado C.
Date: 2026.04.23 10:24:32 -04'00'

Fecha: 23/04/2026 Firma: _____

Estudiante o Candidato(a):

Fecha: _____24/04/2026_____Firma: _____

AGRADECIMIENTOS

Toda investigación parte de una pregunta, pero rara vez se desarrolla en solitario. Detrás de cada página escrita, de cada corrección y de cada idea que logró compilar sin errores fatales, hubo personas, aprendizajes y momentos que hicieron posible este trabajo. Por eso, quiero agradecer a quienes fueron parte de este proceso, directa o indirectamente.

A mi profesor guía, por su orientación, su paciencia y sus observaciones precisas a lo largo de esta investigación. Gracias por ayudarme a debuggear ideas, identificar vulnerabilidades teóricas y fortalecer la arquitectura de este trabajo hasta llevarlo a una versión mucho más sólida y estable.

A mis docentes, por compartir conocimientos, experiencias y herramientas que fueron fundamentales en mi formación. Gracias por enseñarme que investigar no es solo buscar respuestas, sino también aprender a formular mejores preguntas, revisar supuestos, corregir fallos y volver a intentar, incluso cuando el sistema parece negarse a cooperar.

A mis amigos, y personas cercanas, gracias por estar presentes en este proceso, por el apoyo, el ánimo y la contención en esos momentos donde el estrés consumía demasiados recursos, el sueño se iba a modo suspensión y mi estabilidad emocional dependía peligrosamente de un hilo... o de un café.

También quiero agradecerme a mí. Porque, a pesar de todo lo que ha pasado en mi vida, aquí sigo. Con bugs, sí; con parches improvisados a veces también; pero avanzando. Gracias a mí por no apagar el sistema, por seguir incluso en modo ahorro de energía, por reiniciarme las veces que hizo falta y por demostrarme que, aun después de varias caídas, siempre fui capaz de volver a levantar el servidor y seguir ejecutando mis metas.

Finalmente, agradezco a la institución por brindar el espacio académico y los recursos necesarios para que esta tesis pudiera desarrollarse y llegar a su versión final. Porque el proceso no estuvo libre de errores, pero después de muchas pruebas, ajustes y validaciones, esta entrega finalmente logró compilar.

RESUMEN EJECUTIVO

La digitalización del transporte público ha modificado de manera importante la gestión operacional, la trazabilidad de los servicios y la interacción con las personas usuarias, mediante la incorporación de aplicaciones web, medios de pago electrónicos, servicios en red, mecanismos de autenticación, plataformas de monitoreo y componentes de administración remota [1], [7]. Este proceso ha incrementado la dependencia de infraestructura tecnológica conectada y, con ello, la exposición a amenazas cibernéticas que pueden afectar la disponibilidad de los servicios, la integridad de la información y la continuidad operacional [1], [2], [18].

Este proyecto de grado desarrolla un framework integral de ciberseguridad orientado a entornos digitalizados del transporte público, concebido como una plataforma capaz de articular, dentro de una misma arquitectura, funciones de descubrimiento de superficie, escaneo de vulnerabilidades, revisión de configuraciones, consolidación de hallazgos, evaluación de impacto, validación controlada, mitigación, monitoreo y apoyo a la toma de decisiones. La propuesta organiza estas capacidades dentro de un flujo metodológico común, con el propósito de transformar evidencia técnica dispersa en información útil para la gestión del riesgo [18], [20], [25].

Uno de los problemas principales abordados en este proyecto de grado corresponde a la dificultad de priorizar adecuadamente el tratamiento de vulnerabilidades cuando los hallazgos provienen de múltiples herramientas y se presentan con criterios de severidad heterogéneos. Aunque la severidad técnica constituye una referencia relevante, su capacidad de orientar decisiones es limitada cuando no se incorpora el contexto del activo afectado. La criticidad real de un hallazgo puede variar según la exposición del recurso, el tipo de servicio comprometido, la necesidad de autenticación, la criticidad operacional del activo y el impacto potencial asociado a una eventual explotación [18], [19], [48], [49]. A partir de este problema, el framework incorpora una capa de priorización contextual basada en aprendizaje automático, apoyada en algoritmos de clasificación supervisada, orientada a complementar la clasificación tradicional y fortalecer la decisión sobre qué vulnerabilidades deben tratarse primero [59], [60], [64], [65].

Para viabilizar este enfoque, la solución integra un módulo de normalización de hallazgos que traduce resultados provenientes de distintas fuentes a una estructura común de análisis. Esta capa permite registrar atributos comparables, como herramienta de origen, tipo de hallazgo, severidad técnica, activo afectado, evidencia, exposición y estado de tratamiento, lo que facilita la consolidación, comparación y análisis transversal de la información. Sobre esta base, el sistema incorpora además señales provenientes de checklist de controles, auditoría, logs, alertas y ejecución de scripts de mitigación, con el fin de enriquecer el contexto de priorización y reducir el ruido operativo generado por salidas heterogéneas [20], [22], [41], [43].

Desde el punto de vista técnico, la solución se implementa mediante una arquitectura ligera y modular apoyada en Python, FastAPI y Uvicorn como núcleo backend de orquestación, integración y procesamiento [50], [51], [52]. La persistencia se resuelve mediante SQLite, mientras que la visualización se desarrolla a través de un dashboard analítico en Streamlit y una interfaz unificada construida con Tailwind CSS [53], [54], [57]. El entorno de validación se apoya en Proxmox VE como base de virtualización para laboratorio y pruebas controladas [56]. Asimismo, el framework integra herramientas como OWASP ZAP, Gobuster y Nikto para el descubrimiento y análisis técnico, y considera componentes como Burp Suite, Nessus, Metasploit, Wazuh y Snort dentro del ecosistema general de seguridad y proyección metodológica de la solución [33], [34], [35], [36], [37], [38], [39], [41], [42].

La propuesta se sustenta en marcos y principios reconocidos de ciberseguridad, en particular NIST Cybersecurity Framework, ISO/IEC 27001, ISO/IEC 27005 y CIS Controls, los cuales proporcionan la base conceptual para estructurar el framework desde una perspectiva de gestión del riesgo [13], [15], [18], [25]. La validación de la solución se aborda en términos funcionales, de integración, analíticos y operacionales, con el objetivo de demostrar que el sistema es capaz de ejecutar pruebas, registrar evidencia, consolidar resultados, contextualizar hallazgos y apoyar la priorización de vulnerabilidades dentro de un entorno de pruebas controlado [20], [28], [29].

Palabras clave: ciberseguridad, transporte público, gestión de vulnerabilidades, priorización contextual, aprendizaje automático, evaluación de riesgos, infraestructura crítica.

ABSTRACT

The digitization of public transportation has significantly altered operational management, service traceability, and user interaction through the incorporation of web applications, electronic payment methods, network services, authentication mechanisms, monitoring platforms, and remote management components [1], [7]. This process has increased dependence on connected technological infrastructure and, consequently, exposure to cyber threats that can affect service availability, data integrity, and operational continuity [1], [2], [18].

This undergraduate thesis develops a comprehensive cybersecurity framework for digitized public transportation environments. It is conceived as a platform capable of integrating, within a single architecture, functions such as surface discovery, vulnerability scanning, configuration review, findings consolidation, impact assessment, controlled validation, mitigation, monitoring, and decision support. The proposal organizes these capabilities within a common methodological flow, aiming to transform dispersed technical evidence into useful information for risk management [18], [20], [25].

One of the main problems addressed in this thesis project is the difficulty of properly prioritizing vulnerability treatment when findings come from multiple tools and are presented with heterogeneous severity criteria. Although technical severity is a relevant reference, its ability to guide decisions is limited when the context of the affected asset is not incorporated. The actual criticality of a finding can vary depending on the exposure of the resource, the type of service compromised, the need for authentication, the operational criticality of the asset, and the potential impact associated with a possible exploitation [18], [19], [48], [49]. Based on this problem, the framework incorporates a contextual prioritization layer based on machine learning, supported by supervised classification algorithms, designed to complement traditional classification and strengthen the decision on which vulnerabilities should be addressed first [59], [60], [64], [65].

To enable this approach, the solution integrates a findings normalization module that translates results from different sources into a common analysis structure. This layer allows for the recording of comparable attributes, such as source tool, type of finding, technical severity, affected asset, evidence, exposure, and treatment status, facilitating the consolidation, comparison, and cross-sectional analysis of information. Building on this foundation, the system also incorporates signals from control checklists, audits, logs, alerts, and the execution of mitigation scripts, enriching the prioritization context and reducing operational noise generated by heterogeneous outputs [20], [22], [41], [43].

From a technical perspective, the solution is implemented using a lightweight and modular architecture supported by Python, FastAPI, and Uvicorn as the core backend for orchestration, integration, and processing [50], [51], [52]. Persistence is handled by SQLite, while visualization is developed through an analytical dashboard in Streamlit and a unified interface built with Tailwind CSS [53], [54], [57]. The validation environment relies on Proxmox VE as a virtualization platform for laboratory and controlled testing [56]. The framework also integrates tools such as OWASP ZAP, Gobuster, and Nikto for discovery and technical analysis, and incorporates components like Burp Suite, Nessus, Metasploit, Wazuh, and Snort within the overall security ecosystem and methodological framework of the solution [33], [34], [35], [36], [37], [38], [39], [41], [42].

The proposal is based on recognized cybersecurity frameworks and principles, particularly the NIST Cybersecurity Framework, ISO/IEC 27001, ISO/IEC 27005, and CIS Controls, which provide the conceptual basis for structuring the framework from a risk management perspective [13], [15], [18], [25]. The validation of the solution is addressed in functional, integration, analytical and operational terms, with the objective of demonstrating that the system is capable of running tests, recording evidence, consolidating results, contextualizing findings and supporting the prioritization of vulnerabilities within a controlled testing environment [20], [28], [29].

Keywords: cybersecurity, public transport, vulnerability management, contextual prioritization, machine learning, risk assessment, critical infrastructure.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

GLOSARIO

Activo: Recurso que posee valor para la organización y que debe ser protegido. En esta tesis, puede corresponder a aplicaciones, servicios, bases de datos, endpoints, credenciales, infraestructura o componentes operativos del sistema.

Alerta de seguridad: Registro o señal generada a partir de eventos, logs o hallazgos que indica una posible condición de riesgo, anomalía o incidente que requiere revisión.

Aprendizaje Automático (Machine Learning): Rama de la inteligencia artificial que permite construir modelos capaces de identificar patrones en los datos y apoyar tareas de clasificación, predicción o priorización.

API (Application Programming Interface): Conjunto de reglas y mecanismos que permiten la comunicación entre distintos programas o servicios.

AppSec (Application Security): Disciplina orientada a proteger aplicaciones durante su diseño, desarrollo, despliegue y operación, mediante controles, pruebas y buenas prácticas de seguridad.

Arquitectura modular: Forma de diseño en la que un sistema se organiza en componentes separados, pero integrados, cada uno con responsabilidades específicas y claramente definidas.

Auditoría: Proceso de revisión y registro de actividades, eventos o configuraciones relevantes para verificar trazabilidad, cumplimiento o comportamiento del sistema.

Autenticación: Proceso mediante el cual un sistema verifica la identidad de un usuario, servicio o entidad antes de permitir el acceso.

Backend: Capa lógica de un sistema encargada del procesamiento, la orquestación de funciones, la persistencia de datos y la exposición de servicios o endpoints.

Base de datos: Sistema utilizado para almacenar, organizar y consultar información estructurada. En esta tesis se utiliza para persistir hallazgos, alertas, logs, controles y resultados analíticos.

Burp Suite: Plataforma profesional de pruebas de seguridad para aplicaciones web, orientada al análisis manual y semiautomatizado de peticiones, respuestas, sesiones y vulnerabilidades.

Checklist de cumplimiento: Lista estructurada de controles o medidas de seguridad que se revisan para verificar el estado de cumplimiento técnico u operativo de un sistema.

Ciberseguridad: Conjunto de prácticas, tecnologías, procesos y criterios destinados a proteger sistemas, redes, servicios y datos frente a amenazas, ataques, fallos o accesos no autorizados.

CIS Controls: Conjunto priorizado de controles de seguridad publicado por el Center for Internet Security, orientado a reducir riesgos frecuentes mediante prácticas verificables.

Confidencialidad: Principio de seguridad que busca asegurar que la información solo sea accesible para usuarios, sistemas o entidades autorizadas.

Contenedorización: Técnica que permite ejecutar aplicaciones y servicios dentro de entornos aislados y portables, incluyendo sus dependencias y configuraciones necesarias.

Continuidad operativa: Capacidad de un sistema o servicio para mantenerse funcionando, o recuperarse adecuadamente, frente a fallos, incidentes o interrupciones.

Control de seguridad: Medida técnica, administrativa u operativa orientada a reducir riesgos, limitar exposición o proteger activos relevantes.

CORS (Cross-Origin Resource Sharing): Mecanismo de seguridad del navegador que regula cómo un recurso web puede ser solicitado desde un origen distinto al del servidor que lo expone.

Correlación de eventos: Proceso de relacionar múltiples registros, alertas o señales para identificar patrones de comportamiento, incidentes o condiciones de riesgo.

Criticidad contextual: Estimación de la relevancia real de un hallazgo dentro del sistema evaluado, considerando exposición, tipo de activo, entorno, impacto operativo y otras variables de contexto.

CSRF (Cross-Site Request Forgery): Ataque que induce a un usuario autenticado a ejecutar acciones no deseadas en una aplicación web sin su consentimiento explícito.

CVSS (Common Vulnerability Scoring System): Sistema estandarizado utilizado para medir la severidad técnica de una vulnerabilidad.

Dashboard: Panel visual que presenta métricas, hallazgos, alertas y estados del sistema de forma centralizada para facilitar su interpretación.

DevSecOps: Enfoque que integra la seguridad en todo el ciclo de vida del software, desde el desarrollo hasta la operación, promoviendo automatización y mejora continua.

DDoS (Distributed Denial of Service): Ataque que intenta saturar un servicio mediante múltiples solicitudes simultáneas para degradar o impedir su funcionamiento.

Digitalización: Proceso de incorporación de tecnologías digitales a servicios, procesos y operaciones con el fin de mejorar eficiencia, trazabilidad y capacidad de gestión.

Docker: Plataforma de contenedorización que permite empaquetar y ejecutar aplicaciones con sus dependencias dentro de entornos aislados.

Endpoint: Punto de acceso o recurso concreto de un sistema, servicio o API al que se puede acceder mediante una petición o conexión.

EPSS (Exploit Prediction Scoring System): Sistema que estima la probabilidad de explotación de una vulnerabilidad conocida a partir de modelos estadísticos y datos públicos.

Escaneo de vulnerabilidades: Proceso automatizado o asistido por herramientas que busca identificar debilidades técnicas, configuraciones inseguras o componentes expuestos.

Explotación: Materialización técnica de una vulnerabilidad por medio de acciones que aprovechan una debilidad para comprometer un sistema o activo.

FastAPI: Framework de desarrollo backend en Python utilizado para construir APIs y servicios web de manera rápida, clara y eficiente.

Framework: Estructura metodológica y técnica que organiza componentes, procesos y herramientas bajo una lógica común de funcionamiento.

Gobuster: Herramienta de enumeración utilizada para descubrir directorios, rutas, archivos y recursos accesibles en un servicio web.

Hallazgo: Resultado relevante producido por una herramienta, un análisis o una verificación, que evidencia una condición de exposición, vulnerabilidad, incumplimiento o riesgo.

Hardening: Conjunto de medidas aplicadas a sistemas, servicios o configuraciones con el fin de reducir su superficie de exposición y fortalecer su seguridad.

HTTP Header (Cabecera HTTP): Información adicional intercambiada entre cliente y servidor en una comunicación HTTP. Algunas cabeceras cumplen funciones importantes de seguridad.

IDS/IPS: Sistemas de detección y prevención de intrusiones que permiten identificar o bloquear actividades maliciosas en redes o servicios.

Impacto: Consecuencia potencial que podría producir la explotación de una vulnerabilidad sobre la operación, la información, los activos o la continuidad del servicio.

Infraestructura crítica: Conjunto de sistemas, servicios o recursos cuya interrupción, degradación o compromiso puede afectar gravemente funciones esenciales de una organización o de la sociedad.

Integridad: Principio de seguridad que busca asegurar que la información, las configuraciones y los procesos no sean alterados sin autorización.

ISO/IEC 27001: Norma internacional que establece requisitos para implementar, mantener y mejorar un sistema de gestión de seguridad de la información.

ISO/IEC 27005: Norma internacional que proporciona directrices para la gestión de riesgos en seguridad de la información.

KEV (Known Exploited Vulnerabilities): Catálogo de vulnerabilidades conocidas como explotadas, publicado por CISA, utilizado como referencia de priorización y riesgo real.

Log: Registro generado por un sistema, servicio o aplicación que documenta eventos, errores, acciones o cambios ocurridos durante su operación.

Metasploit: Framework orientado a validación ofensiva controlada, explotación en laboratorio y pruebas de penetración sobre vulnerabilidades seleccionadas.

MITRE ATT&CK: Base de conocimiento que clasifica tácticas y técnicas utilizadas por adversarios en campañas y ataques cibernéticos reales.

Mitigación: Conjunto de acciones orientadas a reducir el riesgo asociado a una vulnerabilidad, exposición o condición insegura.

Monitoreo continuo: Proceso permanente de observación del estado de seguridad de un entorno a través de eventos, logs, alertas y otros indicadores operativos.

Nessus: Escáner de vulnerabilidades utilizado para identificar fallos de seguridad en infraestructura, red, servicios y configuraciones.

Nikto: Herramienta utilizada para detectar configuraciones inseguras, exposición de archivos, cabeceras faltantes y problemas de hardening en servicios web.

NIST CSF (NIST Cybersecurity Framework): Marco de referencia desarrollado por el National Institute of Standards and Technology para estructurar la gestión del riesgo en ciberseguridad.

Normalización de hallazgos: Proceso mediante el cual resultados heterogéneos provenientes de distintas herramientas se traducen a una estructura común y comparable.

OWASP ZAP: Herramienta de código abierto utilizada para identificar vulnerabilidades y configuraciones inseguras en aplicaciones web y servicios HTTP.

Persistencia: Capacidad de almacenar información de manera estructurada para su posterior consulta, análisis o trazabilidad.

Phishing: Técnica de engaño orientada a obtener credenciales o información sensible mediante correos, mensajes o sitios fraudulentos.

Priorización contextual: Proceso de ordenamiento de hallazgos considerando no solo severidad técnica, sino también variables operativas y contextuales que afectan su urgencia real.

Proxmox VE: Plataforma de virtualización que permite ejecutar y administrar máquinas virtuales y contenedores dentro de un mismo entorno.

Prueba de penetración (Penetration Testing): Simulación controlada de un ataque con el objetivo de identificar, validar o medir vulnerabilidades en un sistema.

Random Forest: Algoritmo de aprendizaje automático basado en múltiples árboles de decisión, utilizado para tareas de clasificación y predicción.

Ransomware: Tipo de malware que cifra o bloquea datos y exige un pago para restaurar el acceso a la información o al sistema.

Regresión Logística: Algoritmo supervisado de clasificación utilizado para estimar probabilidades y separar observaciones en categorías.

Resiliencia: Capacidad de un sistema para resistir, adaptarse y recuperarse frente a incidentes, fallos o ataques sin perder de forma crítica su funcionalidad.

Riesgo: Resultado de la relación entre una amenaza, una vulnerabilidad, la exposición de un activo y el impacto potencial de su explotación.

SGSI (Sistema de Gestión de Seguridad de la Información): Conjunto de políticas, procesos, controles y recursos destinados a gestionar de forma sistemática la seguridad de la información.

SIEM (Security Information and Event Management): Solución orientada a centralizar, correlacionar y analizar eventos de seguridad provenientes de múltiples fuentes.

SQLite: Sistema de gestión de base de datos relacional, ligero y embebido, utilizado en esta tesis para persistir hallazgos, alertas y evidencia operativa.

Snort: Herramienta IDS/IPS utilizada para detección y análisis de tráfico malicioso o patrones de intrusión en red.

Snapshot: Copia exacta del estado de un sistema o entorno virtual en un momento determinado, útil para restauración y pruebas controladas.

Streamlit: Herramienta utilizada para construir interfaces y dashboards analíticos de manera rápida sobre aplicaciones desarrolladas en Python.

STRIDE: Modelo de análisis de amenazas que clasifica riesgos en seis categorías: suplantación, manipulación, repudio, divulgación de información, denegación de servicio y elevación de privilegios.

Superficie de ataque: Conjunto de puntos, interfaces, rutas, servicios y recursos a través de los cuales un sistema puede ser observado, alcanzado o comprometido.

Suricata: Motor IDS/IPS y de monitoreo de tráfico de red orientado a detección de amenazas y análisis de eventos.

Tailwind CSS: Framework de estilos utilitario utilizado para construir interfaces visuales modernas, limpias y mantenibles.

Amenaza: Posibilidad de que un actor, evento o condición intente explotar una vulnerabilidad y comprometer un activo o sistema.

Transporte público digitalizado: Entorno de operación del transporte público que integra servicios tecnológicos, aplicaciones, plataformas y componentes conectados para su funcionamiento.

Transformación digital: Proceso mediante el cual una organización incorpora tecnologías digitales a sus operaciones, procesos y servicios para mejorar su funcionamiento.

Uvicorn: Servidor ASGI liviano utilizado para ejecutar aplicaciones backend desarrolladas con FastAPI.

Validación ofensiva controlada: Verificación técnica de la explotabilidad de un hallazgo en un entorno de laboratorio o pruebas acotadas, sin comprometer el entorno productivo.

Vulnerabilidad: Debilidad de diseño, implementación, configuración o mantenimiento que puede ser aprovechada para afectar la seguridad de un sistema.

Wazuh: Plataforma orientada a monitoreo, gestión de eventos, detección y análisis de seguridad, utilizada como referencia de SIEM/XDR de código abierto.

XDR (Extended Detection and Response): Enfoque de detección y respuesta extendida que integra señales de distintos dominios de seguridad para mejorar visibilidad y respuesta.

XSS (Cross-Site Scripting): Vulnerabilidad que permite inyectar scripts maliciosos en páginas web para su ejecución en el navegador de otros usuarios.

Introduccion

La transformación digital ha modificado de manera profunda la forma en que operan los sistemas de transporte público, incorporando tecnologías orientadas a optimizar la gestión operacional, mejorar la trazabilidad de los servicios y ampliar las formas de interacción con las personas usuarias. En este proceso, aplicaciones web, sistemas de pago electrónico, plataformas de monitoreo, servicios en red, mecanismos de autenticación y componentes de administración remota han pasado a formar parte de la infraestructura tecnológica que sostiene el funcionamiento cotidiano del sistema [1], [7], [8], [9]. Sin embargo, esta digitalización también ha ampliado la superficie de exposición frente a amenazas cibernéticas, configurando un escenario en el que la eficiencia tecnológica convive con riesgos crecientes para la seguridad, la continuidad operacional y la confianza institucional [1], [2], [3], [18].

Esta situación adquiere especial relevancia en entornos asociados al transporte público, ya que la afectación de un sistema digital no se limita al compromiso de información o recursos informáticos. También puede repercutir en la disponibilidad del servicio, en la estabilidad de procesos esenciales y en la percepción de seguridad de la ciudadanía. En este contexto, la ciberseguridad se entiende como una capacidad continua de evaluación, interpretación y tratamiento del riesgo técnico, particularmente necesaria en infraestructuras digitalizadas que sostienen funciones críticas para la operación [6], [9], [15], [18], [19].

A partir de este problema, este proyecto de grado propone el diseño e implementación de un framework integral de ciberseguridad orientado a entornos digitalizados del transporte público. El framework integra capacidades de descubrimiento de superficie, escaneo de vulnerabilidades, revisión de configuraciones, consolidación de hallazgos, evaluación de impacto, validación controlada, mitigación, monitoreo y apoyo a la toma de decisiones. Su aporte consiste en organizar estas capacidades dentro de una arquitectura común, de modo que la evidencia técnica obtenida pueda registrarse, compararse, contextualizarse y utilizarse de manera útil para la remediación [18], [20], [22], [25], [29].

Uno de los problemas centrales que motivan este proyecto de grado es que, en la práctica, las herramientas de seguridad producen una cantidad considerable de hallazgos, pero no siempre ofrecen una forma clara de priorizarlos. La severidad técnica constituye una referencia importante, aunque en numerosos casos resulta insuficiente para decidir qué debilidad debe tratarse primero. La criticidad real de un hallazgo depende también de factores como la exposición del activo, el tipo de servicio afectado, la necesidad de autenticación, la relevancia operativa del recurso comprometido, la evidencia adicional disponible y el impacto potencial que su explotación podría producir sobre el sistema [15], [17], [19], [45], [47], [48]. A partir de esta constatación, el framework incorpora una capa de priorización contextual basada en aprendizaje automático, cuyo propósito es complementar la clasificación tradicional y aportar un criterio más cercano a la realidad operativa del entorno evaluado [45], [46], [59], [60], [64], [65].

La investigación se desarrolla bajo una perspectiva aplicada. Para ello, articula fundamentos conceptuales de ciberseguridad, metodologías de evaluación, marcos normativos y herramientas técnicas relevantes dentro de una sola propuesta metodológica y operativa. En esta línea, el framework se apoya en principios de gestión del riesgo y mejora continua presentes en referencias como NIST Cybersecurity Framework, ISO/IEC 27001, ISO/IEC 27005 y CIS Controls [13], [15], [18], [25], los cuales se traducen a una arquitectura concreta orientada a la observación, el análisis, la priorización y el tratamiento de hallazgos. De este modo, la solución se presenta como una propuesta funcional diseñada para responder a necesidades reales de evaluación y apoyo a la decisión en infraestructura crítica digitalizada [6], [16], [18], [22].

Desde el punto de vista técnico, la solución se implementa mediante una arquitectura ligera y modular, apoyada en Python, FastAPI y Uvicorn como núcleo backend de orquestación, consolidación y análisis [50], [51], [52]. A ello se suma el uso de SQLite para persistencia [53], Streamlit y una interfaz unificada con Tailwind CSS para visualización [54], [57], y un entorno de virtualización basado en Proxmox VE para laboratorio y validación controlada [56]. Asimismo, se integran herramientas como OWASP ZAP, Gobuster y Nikto para el descubrimiento y la evaluación técnica del entorno [26], [29], [33], [34], [38], [39], y se consideran componentes de validación, monitoreo y correlación como Burp Suite, Nessus, Metasploit, Wazuh y Snort dentro del ecosistema general del framework [35], [36], [37], [41], [42]. Esta estructura permite que los hallazgos dejen de permanecer aislados en la salida de cada herramienta y pasen a formar parte de una base común de evidencia orientada al análisis y la decisión.

En este contexto, el objetivo central del proyecto de grado es demostrar que es posible diseñar e implementar un framework integral de ciberseguridad capaz de identificar vulnerabilidades, integrar hallazgos provenientes de múltiples fuentes, relacionarlos con evidencia operativa y priorizarlos de forma contextual para apoyar la toma de decisiones de remediación. Para ello, se desarrolla una metodología aplicada que combina levantamiento técnico, ejecución de pruebas, normalización de resultados, análisis del riesgo, validación funcional del sistema y evaluación del comportamiento del framework en un entorno de pruebas controlado [18], [19], [20], [21], [29].

El aporte esperado de este trabajo se sitúa en dos niveles complementarios. En el plano técnico, se busca ofrecer una solución funcional y coherente para apoyar la evaluación de seguridad en entornos digitalizados del transporte público. En el plano académico, se propone una forma más estructurada de comprender la ciberseguridad aplicada como un proceso continuo de observación, interpretación y priorización del riesgo basado en evidencia organizada. Desde esta perspectiva, el proyecto de grado presenta una respuesta metodológicamente fundamentada, técnicamente viable y pertinente frente a un problema real de seguridad en infraestructura crítica digitalizada.

1 DEFINICION DEL PROBLEMA

La transformación digital del transporte público en el Gran Concepción ha impulsado la incorporación de tecnologías orientadas a optimizar la operación, mejorar la trazabilidad de los servicios y fortalecer la experiencia de las personas usuarias. En este contexto, sistemas de pago electrónico, aplicaciones móviles, servicios web, APIs, plataformas de monitoreo, mecanismos de autenticación y herramientas de administración remota han pasado a formar parte del ecosistema tecnológico que sostiene el funcionamiento del servicio [1], [7], [8], [9]. Sin embargo, esta misma incorporación tecnológica ha ampliado de forma considerable la superficie de exposición frente a amenazas cibernéticas, al multiplicar los puntos de acceso, aumentar la dependencia de componentes interconectados y complejizar la administración segura del entorno [1], [2], [3], [18].

En este escenario, el problema no se limita a la existencia de sistemas digitalizados, sino a la necesidad de proteger adecuadamente el entorno tecnológico que soporta la operación del transporte público. El foco de este proyecto de grado se sitúa en la dimensión digital del servicio, es decir, en las plataformas, aplicaciones, servicios, componentes expuestos y fuentes de evidencia técnica que participan en su funcionamiento o gestión. Por tanto, la propuesta no se orienta al monitoreo integral del transporte como sistema físico u operacional, sino al análisis de ciberseguridad del ecosistema digital asociado al transporte público, entendido como un conjunto de activos, servicios y evidencias que requieren evaluación, correlación y priorización para apoyar decisiones de remediación.

Esta delimitación es relevante porque la afectación de un entorno digital vinculado al transporte público no se restringe al compromiso de información o recursos informáticos. También puede repercutir en la disponibilidad del servicio, en la estabilidad de procesos esenciales y en la percepción de seguridad de la ciudadanía. En consecuencia, la ciberseguridad se entiende en este trabajo como una capacidad continua de evaluación, interpretación y tratamiento del riesgo técnico, particularmente necesaria en infraestructuras digitalizadas que sostienen funciones relevantes para la operación [6], [9], [15], [18], [19].

En la práctica, el problema no radica únicamente en la existencia de vulnerabilidades. Desde un punto de vista técnico, es esperable que un entorno digitalizado presente servicios expuestos, configuraciones mejorables, dependencias inseguras o fallas de implementación en algún grado. La dificultad principal aparece cuando no se dispone de un mecanismo estructurado para descubrir esas debilidades, analizarlas de forma periódica, relacionarlas entre sí y determinar cuáles deben tratarse primero. Detectar fallas es necesario, pero no suficiente. El problema central consiste en transformar hallazgos técnicos heterogéneos en evidencia útil para la toma de decisiones [18], [19], [20], [25], [29].

A partir de esta necesidad, este proyecto de grado propone un framework integral de ciberseguridad orientado al entorno digital del transporte público del Gran Concepción. Este framework articula distintas funciones dentro de una misma lógica operativa: descubrimiento de superficie, análisis de vulnerabilidades, revisión de configuraciones inseguras, monitoreo, registro de eventos, verificación de controles, ejecución de scripts de mitigación, consolidación de hallazgos y priorización contextual. Su propósito es organizar técnicamente la información necesaria para sostener decisiones de remediación, sobre la base de evidencia obtenida desde múltiples fuentes [18], [20], [22], [25].

La base conceptual del sistema se apoya en marcos reconocidos como NIST Cybersecurity Framework, ISO/IEC 27001, ISO/IEC 27005 y CIS Controls, utilizados como criterios para estructurar el análisis de activos, la valoración del riesgo, la mitigación y el seguimiento [13], [15], [18], [25]. Bajo esta lógica, el framework integra herramientas de análisis web y descubrimiento de superficie como OWASP ZAP, Gobuster y Nikto [26], [29], [33], [34], [38], [39], dentro de una arquitectura más amplia que también contempla módulos de monitoreo, registro, checklist de controles, scripts de hardening, visualización de resultados y un mecanismo de priorización contextual basado en aprendizaje automático [41], [42], [54].

Precisamente en este punto se encuentra una de las dimensiones más relevantes del problema. Las herramientas de seguridad generan una gran cantidad de información, pero esa información suele llegar en formatos distintos, con taxonomías diferentes y con criterios de severidad que no siempre son comparables entre sí. Un escáner web puede reportar vulnerabilidades; una herramienta de enumeración puede descubrir rutas expuestas; un módulo de checklist puede evidenciar incumplimientos de configuración; mientras que los registros y alertas del sistema pueden sugerir comportamientos operativamente relevantes. Cada una de estas fuentes aporta evidencia valiosa, pero si esa evidencia no se normaliza ni se contextualiza, el resultado es una sobrecarga de señales difícil de interpretar y de utilizar en un proceso real de remediación [18], [20], [22], [25].

Esta dificultad se vuelve especialmente crítica en entornos con capacidad limitada de remediación. En la práctica, no todas las vulnerabilidades pueden tratarse de manera simultánea, ni todos los hallazgos presentan la misma urgencia operativa. Por ello, una estrategia que dependa exclusivamente de la severidad técnica resulta insuficiente. Métricas como CVSS entregan una referencia útil, pero no incorporan con suficiente profundidad variables como la exposición real del servicio, la criticidad del activo afectado, la necesidad de autenticación, la existencia de alertas correlacionadas, la evidencia de explotación conocida, el estado de controles de seguridad o la presencia de validaciones técnicas adicionales [45], [46], [47], [48], [49]. Como consecuencia, una vulnerabilidad de severidad media puede requerir tratamiento prioritario si afecta un activo crítico y expuesto, mientras que otra de severidad alta puede tener menor urgencia si se encuentra en un contexto aislado o de bajo impacto operativo.

Por esta razón, el problema de investigación no se limita a diseñar un proceso de evaluación técnica, sino a construir un framework capaz de integrar análisis, contexto y priorización dentro de un mismo sistema. El interés del proyecto de grado no consiste únicamente en demostrar la existencia de vulnerabilidades, sino en demostrar que un framework bien diseñado puede convertir resultados dispersos y heterogéneos en una base priorizada de decisión, más útil que la severidad aislada y más cercana a las necesidades reales de la operación digital evaluada.

En este contexto, el aprendizaje automático no se incorpora como un componente externo o accesorio, sino como una función integrada dentro del framework. Su finalidad es fortalecer la priorización de hallazgos mediante el análisis de variables técnicas, operativas y de cumplimiento que no quedan adecuadamente representadas por un criterio único de severidad. Su función no consiste en reemplazar el juicio experto ni en automatizar de manera absoluta la toma de decisiones. Su aporte se orienta a complementar el análisis con una estimación contextual de criticidad que permita ordenar mejor los esfuerzos de remediación y reducir el ruido operativo generado por múltiples fuentes de evidencia [45], [46], [59], [60], [64], [65].

Desde esta perspectiva, la investigación responde a un problema real de la ciberseguridad aplicada: la dificultad de articular descubrimiento, evaluación, monitoreo, verificación y priorización dentro de un solo sistema coherente. En muchos entornos, las herramientas existen, pero operan de forma fragmentada. En cambio, la propuesta de este proyecto de grado plantea un framework unificado, donde los distintos módulos se entienden como componentes de una misma arquitectura orientada a observar, interpretar y priorizar el riesgo [18], [20], [25], [29].

En consecuencia, el problema de investigación se formula del siguiente modo:

> **¿Cómo diseñar e implementar un framework integral de ciberseguridad para el entorno digital del transporte público que permita descubrir vulnerabilidades, integrar hallazgos provenientes de múltiples herramientas, relacionarlos con evidencia de monitoreo y cumplimiento, y priorizarlos de forma contextual e inteligente para apoyar eficazmente la toma de decisiones de remediación?**

La respuesta propuesta en este proyecto de grado consiste en construir un framework único y articulado, capaz de combinar módulos de evaluación técnica, descubrimiento de superficie, checklist de controles, registros operativos, monitoreo, scripts de hardening, visualización y priorización contextual basada en aprendizaje automático. De este modo, la ciberseguridad deja de operar únicamente como un mecanismo de detección y pasa a funcionar como un sistema de apoyo técnico a la decisión, orientado a ordenar, justificar y fortalecer el tratamiento del riesgo en un entorno digital crítico.

A partir de ello, el problema se define de manera integral del siguiente modo: el entorno digital asociado al transporte público del Gran Concepción enfrenta una exposición creciente a ciberamenazas en un contexto donde la sola detección de vulnerabilidades no basta para sostener una remediación eficaz. Se requiere un framework capaz de integrar hallazgos heterogéneos, verificar condiciones de seguridad, registrar evidencia operativa y priorizar vulnerabilidades según su criticidad contextual, de modo que la evaluación de ciberseguridad contribuya directamente a la toma de decisiones y a la resiliencia del sistema.

1.0.1. Objetivo general

Diseñar e implementar un framework integral de ciberseguridad para el entorno digital del transporte público, capaz de descubrir vulnerabilidades, integrar hallazgos provenientes de múltiples herramientas, relacionarlos con evidencia de monitoreo y cumplimiento, y priorizarlos de forma contextual mediante aprendizaje automático, con el fin de apoyar la toma de decisiones de remediación.

1.0.2. Objetivos específicos

1. Estructurar un framework de evaluación de ciberseguridad que contemple identificación de activos, análisis de vulnerabilidades, valoración de riesgo, mitigación y seguimiento continuo bajo una lógica metodológica coherente.
2. Integrar herramientas complementarias de análisis como OWASP ZAP, Gobuster y Nikto, de modo que el sistema amplíe su cobertura sobre vulnerabilidades web, descubrimiento de superficie y configuraciones inseguras.
3. Diseñar un mecanismo de normalización de hallazgos que permita consolidar en una estructura común los resultados obtenidos desde herramientas de escaneo, descubrimiento, monitoreo y verificación de controles.
4. Implementar un módulo de checklist con base en controles de seguridad relevantes, capaz de producir evidencia operativa a partir del estado del entorno y no solo de criterios declarativos.
5. Incorporar mecanismos de registro de eventos, logs, alertas y ejecución de scripts de mitigación, con el propósito de enriquecer la trazabilidad y la interpretación del contexto de riesgo.
6. Desarrollar un módulo de priorización contextual basado en aprendizaje automático, que estime la criticidad de los hallazgos considerando variables técnicas, operativas y de cumplimiento.
7. Comparar la priorización tradicional basada en severidad técnica con la priorización contextual generada por el framework, evaluando su utilidad como apoyo a la decisión.
8. Implementar una interfaz unificada de visualización que permita representar hallazgos, alertas, estados de controles, ejecuciones de mitigación y rankings de vulnerabilidades de forma comprensible para distintos perfiles de usuario.

2 | MARCO CONCEPTUAL

2.1. Ciberseguridad como disciplina de gestión del riesgo

En este proyecto de grado, la ciberseguridad se entiende como una disciplina de gestión del riesgo y no únicamente como una función técnica orientada a proteger componentes aislados. Esta perspectiva resulta especialmente pertinente en entornos digitalizados, donde el funcionamiento continuo del sistema depende tanto de la robustez de cada componente como de la capacidad para identificar, evaluar y tratar oportunamente las debilidades que aparecen en la relación entre aplicaciones, servicios, infraestructuras y procesos operativos [15], [17], [18], [19].

Desde esta perspectiva, la seguridad no se reduce a una lógica reactiva de corrección posterior al incidente. Su propósito comprende la identificación anticipada de condiciones de exposición, la disminución de la probabilidad de explotación y la reducción del impacto que una falla podría producir sobre la operación. En consecuencia, la ciberseguridad se configura como una práctica continua de reducción del riesgo técnico, sostenida mediante evaluación periódica, análisis contextual, verificación de controles y decisiones de tratamiento fundamentadas [13], [15], [18], [19], [22].

Esta concepción adquiere una relevancia mayor cuando el entorno evaluado se vincula con servicios de carácter esencial. En estos casos, una afectación sobre la confidencialidad, la integridad o la disponibilidad no repercute únicamente sobre la plataforma tecnológica, sino también sobre la continuidad funcional del servicio, la estabilidad operacional y la confianza de las personas usuarias [6], [9], [16], [18]. Por ello, en este proyecto de grado la seguridad no se trata como un conjunto aislado de herramientas de diagnóstico, sino como una capacidad transversal de resiliencia que debe incorporarse al sistema completo.

En términos fundamentales, esta disciplina continúa sosteniéndose sobre los principios clásicos de confidencialidad, integridad y disponibilidad. La confidencialidad busca que la información sensible sea accesible solo por entidades autorizadas. La integridad procura preservar la consistencia, exactitud y fiabilidad de la información, las configuraciones y los procesos. La disponibilidad se orienta a mantener los sistemas accesibles y operativos frente a fallos, incidentes o ataques [12], [13], [22].

En el marco de este proyecto de grado, estos principios deben traducirse a mecanismos concretos de observación, evaluación, correlación y priorización. Esa es una de las bases conceptuales del framework propuesto: integrar dichos principios dentro de una arquitectura capaz de evaluarlos a partir del comportamiento real del sistema, de sus configuraciones, de sus hallazgos técnicos y de las decisiones de tratamiento que estos exigen [18], [22], [25].

2.2. Digitalización, complejidad tecnológica y superficie de ataque

La digitalización introduce eficiencia, automatización y trazabilidad, pero también expande la superficie de ataque. A medida que un sistema incorpora aplicaciones web, APIs, servicios móviles, mecanismos de autenticación, componentes de administración remota y recursos conectados, aumentan los puntos de acceso legítimo y también las posibilidades de observación, interacción y explotación potencial por parte de un atacante [1], [2], [26], [27], [29].

La superficie de ataque puede entenderse como el conjunto de interfaces, rutas, servicios, configuraciones y activos a través de los cuales un sistema puede ser alcanzado o comprometido. En entornos complejos, esta superficie rara vez permanece estática. Cambia con el despliegue de nuevas versiones, la incorporación de dependencias, la publicación de nuevos endpoints, la habilitación de servicios auxiliares o la modificación de configuraciones. Por ello, cualquier enfoque serio de ciberseguridad debe asumir que la visibilidad de la superficie expuesta constituye una condición previa para la evaluación efectiva del riesgo [20], [29], [30], [31], [32].

Este aspecto adquiere una importancia particular cuando conviven múltiples capas tecnológicas dentro de una misma operación. Un hallazgo no puede interpretarse adecuadamente si se desconoce el contexto en el que aparece. Una ruta descubierta, un puerto abierto o una configuración insegura no representan automáticamente el mismo nivel de riesgo en todos los casos; su relevancia depende de la función del activo, de su accesibilidad real, de su relación con otros componentes y del papel que cumple dentro de la operación [15], [17], [19]. Por esta razón, una lectura aislada del hallazgo técnico resulta insuficiente y debe complementarse con la comprensión del entorno donde dicho hallazgo existe.

A partir de ello, la evaluación no puede limitarse al escaneo tradicional de vulnerabilidades. Debe incorporar también reconocimiento, enumeración de recursos, observación de configuraciones, revisión de dependencias y análisis del entorno operativo. Esta ampliación del foco permite pasar desde una visión reducida del hallazgo técnico hacia una comprensión más completa del riesgo. En coherencia con ello, el framework propuesto integra módulos de descubrimiento de superficie, análisis web, revisión de configuraciones, monitoreo, checklist de controles y priorización contextual dentro de una sola arquitectura [20], [26], [29], [33], [38], [39].

2.3. Vulnerabilidad, amenaza, explotación y riesgo

Para sostener conceptualmente esta investigación, resulta indispensable distinguir nociones que suelen aparecer juntas, pero que no significan lo mismo. Una vulnerabilidad corresponde a una debilidad de diseño, implementación, configuración o mantenimiento que puede ser aprovechada para comprometer un sistema. Una amenaza representa la posibilidad de que un actor, un evento o una condición intente explotar esa debilidad. La explotación es la materialización técnica de esa posibilidad. El riesgo surge de la relación entre la vulnerabilidad identificada, la amenaza asociada, el nivel de exposición del activo y el impacto potencial que produciría su aprovechamiento [15], [17], [18], [19], [48], [49].

Esta distinción no es meramente teórica; define la forma en que se estructura el sistema propuesto. Si todos los hallazgos se trataran como equivalentes, la evaluación se reduciría a una simple enumeración de fallas. Sin embargo, en la práctica, la existencia de una vulnerabilidad no implica automáticamente que el riesgo sea máximo. Del mismo modo, una debilidad aparentemente menor puede adquirir una importancia mucho mayor si afecta un activo crítico, un servicio públicamente expuesto o un componente vinculado a funciones operativas sensibles [15], [17], [19].

Desde esta perspectiva, el valor de un hallazgo no puede establecerse únicamente a partir de su descripción técnica. Es necesario comprender dónde aparece, qué activo compromete, qué condiciones habilitan su explotación y qué consecuencias tendría sobre la operación. Esta lógica justifica que el framework integre módulos de detección, evaluación, correlación y priorización. Detectar una vulnerabilidad constituye el primer paso; estimar su relevancia real y su prioridad relativa frente a otros hallazgos es lo que convierte la evaluación en una herramienta útil para la decisión [18], [19], [25].

2.4. Infraestructura crítica y continuidad operacional

La noción de infraestructura crítica resulta fundamental para comprender el alcance del problema analizado. Un sistema no se considera crítico únicamente por su complejidad técnica, sino por las consecuencias que generaría su degradación, interrupción o manipulación sobre la operación y sobre el entorno al que sirve. En este sentido, el entorno digital asociado al transporte público puede entenderse como parte de una infraestructura crítica, ya que su funcionamiento depende de plataformas tecnológicas que sostienen procesos relevantes para la movilidad urbana y la continuidad de un servicio esencial [6], [9], [16], [18].

Este carácter crítico modifica la forma de abordar la seguridad. En un entorno convencional, una vulnerabilidad podría interpretarse solo desde su severidad técnica. En cambio, en un entorno crítico, esa lectura resulta insuficiente. Es necesario incorporar el impacto operacional del activo afectado, su exposición real, su relación con otros componentes del sistema y la posibilidad de que una debilidad puntual desencadene efectos en cascada [15], [16], [17], [19]. Por ello, la evaluación desarrollada en este proyecto de grado no se limita a identificar fallas aisladas, sino que las interpreta en función de su capacidad de alterar procesos esenciales.

A partir de ello, la continuidad operacional deja de ser una preocupación exclusivamente funcional y pasa a integrarse directamente con la lógica de ciberseguridad. Proteger sistemas críticos implica reducir la posibilidad de compromiso técnico y preservar la estabilidad del servicio que depende de ellos. Esta es una de las razones por las que el framework incorpora una capa de priorización contextual: en infraestructura crítica, decidir qué tratar primero constituye una necesidad operativa [16], [18], [21].

2.5. Evaluación de seguridad como proceso estructurado

En este proyecto de grado, la evaluación de seguridad se entiende como un proceso estructurado, continuo y metodológicamente articulado. No se concibe como una auditoría aislada ni como una actividad puntual ejecutada al final del ciclo. En entornos con exposición dinámica, donde los servicios cambian, se incorporan dependencias y se amplían las superficies accesibles, la seguridad exige una lógica cíclica de observación, análisis y tratamiento [18], [20], [23], [24], [32].

Por ello, el framework se organiza bajo una secuencia de etapas que responden a funciones diferenciadas: identificación de activos, reconocimiento de superficie, escaneo de vulnerabilidades, análisis de hallazgos, valoración del riesgo, verificación de controles, definición de mitigaciones y seguimiento continuo. Cada etapa cumple una función específica dentro del sistema. La identificación de activos establece qué debe protegerse; el reconocimiento amplía la visibilidad del perímetro expuesto; el escaneo revela debilidades técnicas; el análisis permite interpretar los resultados; la valoración del riesgo ayuda a definir prioridad; y la mitigación conecta la evaluación con acciones concretas de tratamiento [18], [19], [20], [22], [25].

Lo relevante no es solo la existencia de estas etapas, sino su integración dentro de una sola arquitectura. Cuando estas funciones operan de manera aislada, la seguridad tiende a fragmentarse y la información pierde valor operativo. En cambio, cuando forman parte de un mismo flujo metodológico, el sistema adquiere continuidad, coherencia y capacidad de mejora. Esa es la lógica central del framework propuesto: convertir múltiples evidencias en una lectura ordenada del riesgo [18], [20], [25].

Esta visión es coherente con los principios de seguridad por diseño y con enfoques contemporáneos como DevSecOps y AppSec, donde la seguridad deja de ser una revisión tardía y pasa a integrarse a lo largo del ciclo de vida del sistema [24], [28], [31], [32]. Desde esta lógica, el framework no se limita a ejecutar pruebas, sino que organiza evidencia, mantiene trazabilidad y fortalece la toma de decisiones.

2.6. Herramientas técnicas relevantes para la evaluación de seguridad

La práctica de la ciberseguridad aplicada exige reconocer que no existe una herramienta única capaz de ofrecer una visión completa del sistema. Cada herramienta responde a una función distinta y produce un tipo particular de evidencia. Por ello, dentro del framework propuesto no se adopta una lógica reducida basada en una sola utilidad, sino una aproximación integradora en la que distintas herramientas se articulan según el aporte específico que realizan al proceso de evaluación [20], [25], [29].

2.7. Herramientas de escaneo de aplicaciones web

Dentro de esta categoría, OWASP ZAP ocupa un lugar central. Su principal valor radica en la automatización del análisis de aplicaciones web y servicios HTTP, permitiendo detectar vulnerabilidades frecuentes, errores de configuración y patrones inseguros de interacción [26], [29], [33], [34]. Además, su capacidad de integración mediante API resulta adecuada para una arquitectura backend orientada a la orquestación programática de pruebas.

Burp Suite cumple una función distinta. Aunque no constituye el eje principal de la automatización del sistema, resulta relevante como herramienta de inspección detallada, contraste manual y validación avanzada de hallazgos [35]. Su fortaleza se encuentra en el examen fino de peticiones, parámetros, sesiones o flujos de autenticación, por lo que complementa el análisis automatizado cuando se requiere revisión experta. En este sentido, su incorporación conceptual refuerza la idea de que la seguridad depende de la articulación entre automatización y validación especializada.

2.8. Herramientas de infraestructura y exposición de servicios

En el plano de infraestructura, Nessus se incorpora como una referencia relevante para comprender la seguridad más allá de la capa web. Su valor metodológico reside en la detección de servicios inseguros, software desactualizado, configuraciones peligrosas y exposición de infraestructura [36]. Aunque la implementación principal del MVP prioriza una arquitectura más liviana y centrada en servicios web, la inclusión de Nessus en el marco conceptual sitúa al framework dentro de un ecosistema más amplio de evaluación y proyecta una extensión natural hacia revisiones de infraestructura más completas.

2.9. Herramientas de descubrimiento y enumeración

Una etapa especialmente relevante es la de reconocimiento y descubrimiento de superficie. En esta dimensión, Gobuster aporta un valor concreto al permitir enumerar rutas, directorios, archivos y recursos accesibles que pueden no ser visibles en una revisión superficial [38]. Su función no consiste en detectar vulnerabilidades en sentido estricto, sino en ampliar la visibilidad sobre la superficie de ataque real.

Este aporte es metodológicamente importante porque un sistema no puede evaluar correctamente aquello que no ha descubierto. Por ello, Gobuster se integra como parte del mismo framework y no como una pieza secundaria. El descubrimiento de superficie fortalece la calidad del análisis posterior y mejora la capacidad del sistema para interpretar hallazgos dentro de una visión más completa del entorno [20], [29], [38].

2.10. Herramientas de configuración y hardening

Nikto se ubica dentro del framework como una herramienta complementaria orientada a la detección de configuraciones débiles, archivos sensibles expuestos, cabeceras inseguras y patrones comunes de hardening insuficiente en servicios web [39]. Su función no compite con ZAP ni con Gobuster. Su aporte corresponde a una capa distinta de evidencia: la asociada al endurecimiento, la exposición innecesaria y las deficiencias de configuración.

Esta distinción resulta importante porque permite reforzar la idea de que los hallazgos de seguridad no provienen únicamente de vulnerabilidades explotables, sino también de condiciones de configuración que incrementan la exposición del sistema. La integración de Nikto amplía la capacidad del framework para captar señales que afectan el riesgo aunque no siempre se manifiesten como una vulnerabilidad clásica [20], [23], [39].

2.11. Herramientas de validación ofensiva

Metasploit representa una categoría distinta dentro del marco conceptual. No se entiende como un motor de descubrimiento ni como un escáner masivo, sino como un entorno orientado a la validación ofensiva controlada. Su valor radica en la posibilidad de confirmar, bajo condiciones acotadas y justificadas, la explotabilidad de ciertas debilidades detectadas por el sistema [20], [37].

Por ello, dentro del framework no se sitúa como núcleo operativo, sino como un recurso de validación avanzada para escenarios específicos. Su función refuerza una distinción central: detectar una vulnerabilidad, priorizarla y confirmar su explotabilidad son actividades relacionadas, pero no equivalentes. La presencia de Metasploit en el marco conceptual permite sostener esa diferenciación y subraya que el sistema propuesto contempla evidencia de validación controlada cuando esta resulta metodológicamente pertinente [20], [37].

2.12. Herramientas de monitoreo y correlación

Wazuh y otras soluciones de monitoreo cumplen una función distinta a la de un escáner o una herramienta de enumeración. Su valor no radica en descubrir vulnerabilidades de manera directa, sino en observar eventos, centralizar registros y correlacionar señales de seguridad [41], [42], [43]. En este proyecto de grado, su incorporación conceptual refuerza la idea de que la evaluación no termina en el hallazgo inicial, sino que puede extenderse hacia mecanismos de observación continua y seguimiento del entorno.

Esto resulta coherente con la arquitectura del framework, ya que permite relacionar resultados de evaluación con evidencia operativa posterior. De este modo, el sistema no solo identifica debilidades, sino que también incorpora contexto adicional desde eventos, logs y alertas, fortaleciendo así la interpretación del riesgo [21], [22], [41], [42], [43].

2.13. Heterogeneidad de resultados y necesidad de normalización

Uno de los desafíos más importantes al integrar múltiples herramientas dentro de un mismo framework es la heterogeneidad de los resultados. Cada herramienta reporta con su propio formato, su propia taxonomía y su propio criterio de severidad. Algunas entregan alertas estructuradas, otras listados de recursos descubiertos, otras evidencias de misconfiguración y otras señales relacionadas con monitoreo o cumplimiento. Si estos resultados se almacenan y analizan tal como salen de origen, el sistema se vuelve difícil de interpretar y pierde utilidad para la toma de decisiones [20], [25], [29].

Por esta razón, se considera indispensable una capa de normalización de hallazgos. En este enfoque, normalizar no significa únicamente traducir datos a un formato común. Significa construir una gramática analítica compartida que permita comparar, agrupar, filtrar y correlacionar resultados provenientes de fuentes distintas. Esta capa es la que transforma una colección dispersa de salidas técnicas en una base integrada de evidencia.

La normalización cumple, por tanto, una función estructural dentro del framework. Gracias a ella es posible representar hallazgos de distinta naturaleza mediante atributos comparables, como tipo de hallazgo, herramienta de origen, severidad técnica, activo afectado, evidencia disponible, exposición, estado de tratamiento y señales adicionales de contexto. Sin esta capa, el sistema sería únicamente una suma de herramientas. Con ella, en cambio, se convierte en una plataforma analítica capaz de sostener procesos reales de priorización [18], [20], [25].

2.14. Severidad técnica y criticidad contextual

La severidad técnica constituye un punto de partida importante en cualquier evaluación de seguridad. Métricas como CVSS ofrecen un lenguaje estandarizado para expresar gravedad y establecer comparaciones generales entre vulnerabilidades [48], [49]. Sin embargo, en términos operativos, la severidad no siempre coincide con la prioridad de tratamiento. Esta diferencia constituye uno de los fundamentos centrales del proyecto de grado.

Una vulnerabilidad técnicamente severa no es necesariamente la más urgente si afecta un entorno de baja exposición o de reducido impacto. A la inversa, una debilidad con severidad media puede requerir atención inmediata si aparece en un componente crítico, accesible desde el exterior o vinculado a procesos sensibles. Esto lleva a sostener que la prioridad real depende de una combinación de factores que excede la métrica técnica pura [15], [17], [19], [45], [46], [47].

La noción de criticidad contextual permite capturar esta diferencia. Mientras la severidad describe la gravedad inherente del hallazgo, la criticidad contextual expresa su relevancia concreta dentro del sistema evaluado. Esta idea justifica que el framework no se limite a reportar severidades, sino que incorpore una capa adicional de análisis capaz de estimar prioridad de manera más rica y operativamente útil.

2.15. Priorización de vulnerabilidades como problema técnico-operativo

La priorización de vulnerabilidades constituye uno de los problemas más complejos de la ciberseguridad aplicada. No se trata solamente de ordenar hallazgos por gravedad, sino de establecer un criterio que permita decidir qué corregir primero cuando el volumen de alertas supera la capacidad real de tratamiento. En contextos operativos, la remediación siempre ocurre bajo restricciones de tiempo, personal, ventanas de mantenimiento, criticidad del servicio y dependencia entre componentes [15], [17], [18], [23]. Por ello, priorizar constituye una necesidad estructural.

Desde esta perspectiva, la priorización no puede entenderse como una operación meramente administrativa. Se concibe como una función técnico-operativa que traduce evidencia dispersa en decisiones justificadas. Para que esta función resulte útil, debe incorporar variables que representen tanto la naturaleza del hallazgo como el contexto en que aparece. Entre ellas destacan el tipo de vulnerabilidad, la severidad técnica, la exposición externa, la necesidad de autenticación, la sensibilidad del activo, la existencia de alertas correlacionadas y la evidencia adicional que surge desde controles, scripts o validaciones [15], [19], [22], [45], [47].

Este enfoque es coherente con la lógica del framework propuesto. A medida que el sistema integra más herramientas y más módulos operativos, la calidad de la defensa depende menos de la cantidad de hallazgos que produce y más de la calidad del criterio con que esos hallazgos se ordenan. En este sentido, la priorización se sitúa como un problema central del sistema y no como una tarea secundaria posterior al escaneo.

2.16. Aprendizaje automático como mecanismo de apoyo a la priorización

La incorporación de aprendizaje automático dentro del framework responde a una necesidad concreta: mejorar la capacidad del sistema para priorizar hallazgos en contextos donde la severidad técnica resulta insuficiente como criterio único. Su función no consiste en sustituir el juicio experto ni en automatizar completamente la toma de decisiones. Su propósito es incorporar una capa analítica capaz de estimar criticidad a partir de patrones observables en los hallazgos y en el contexto que los rodea [45], [46], [59], [60], [64], [65].

En este marco, los modelos supervisados resultan especialmente apropiados. Algoritmos como Regresión Logística o Random Forest permiten trabajar con variables bien definidas, producir clasificaciones interpretables y evaluar el desempeño mediante métricas cuantitativas [59], [60], [64], [65]. Esto resulta metodológicamente adecuado para un proyecto de grado aplicado, ya que el interés no está en construir un modelo opaco o innecesariamente complejo, sino en demostrar que una capa inteligente puede aportar valor real y medible a la priorización.

El aporte conceptual del aprendizaje automático dentro del framework radica en que permite formalizar una intuición operativa ampliamente reconocida: no todas las vulnerabilidades deben tratarse del mismo modo, y ciertas combinaciones de factores pueden anticipar una criticidad mayor que la sugerida por la severidad aislada. Al integrar esta capa, el sistema deja de limitarse a detectar y comienza también a sugerir un orden de tratamiento, fortaleciendo así su utilidad como soporte para la decisión.

2.17. Marcos normativos fundamentales

El marco metodológico y técnico de esta investigación se apoya en referencias normativas reconocidas, no como una formalidad documental, sino como una base para dotar de coherencia, trazabilidad y legitimidad al framework propuesto.

NIST Cybersecurity Framework ofrece una arquitectura flexible basada en identificar, proteger, detectar, responder y recuperar. Su valor principal radica en que permite organizar la seguridad como proceso y no como un conjunto aislado de controles [18]. Esta lógica resulta plenamente coherente con la arquitectura integrada desarrollada en el proyecto de grado.

ISO/IEC 27001 aporta una perspectiva de gestión y gobernanza de la seguridad de la información. Su relevancia no está únicamente en el cumplimiento, sino en la forma en que estructura responsabilidades, controles y evidencias dentro de una visión organizacional más amplia [13], [14]. En este proyecto de grado, esa perspectiva evita que el framework quede restringido a una dimensión puramente instrumental.

ISO/IEC 27005 resulta especialmente pertinente porque profundiza en la lógica de análisis y tratamiento del riesgo. Esta norma respalda conceptualmente la importancia de valorar no solo hallazgos aislados, sino su relación con impacto, probabilidad y contexto, lo que dialoga directamente con la priorización contextual integrada en el sistema [15].

CIS Controls, finalmente, ofrece una traducción práctica de buenas prácticas a controles verificables. Su principal aporte dentro del framework es facilitar la conexión entre hallazgo técnico, evidencia operativa y estado de cumplimiento, permitiendo incorporar un módulo de checklist alimentado por verificaciones concretas del entorno [25].

Desde esta perspectiva, estos marcos no compiten con el framework propuesto; lo sostienen. Permiten situar el proyecto de grado dentro de un conjunto de buenas prácticas reconocidas y evitan que el sistema quede reducido a una implementación aislada o puramente instrumental. En conjunto, funcionan como soporte conceptual para la arquitectura técnica, los mecanismos de evaluación y la lógica de priorización desarrollados a lo largo del trabajo.

3 PROPUESTA DE SOLUCIÓN

3.1. Enfoque general de la solución

Este proyecto de grado propone un framework integral de ciberseguridad orientado al entorno digital asociado al transporte público, diseñado como una plataforma capaz de articular, dentro de un mismo flujo metodológico, tareas de descubrimiento, evaluación, consolidación, monitoreo y priorización del riesgo. La solución no se plantea como un escáner aislado, ni como una interfaz de visualización independiente, ni como un conjunto de herramientas acopladas sin criterio común. Se concibe como un sistema unificado cuyo propósito es transformar evidencia técnica dispersa en información priorizada y útil para la toma de decisiones de remediación [18], [20], [22], [25].

La necesidad de esta solución surge del contexto analizado en los capítulos anteriores. A medida que el transporte público incorpora sistemas de pago electrónico, aplicaciones web, APIs, mecanismos de autenticación, plataformas de monitoreo y servicios remotos, aumentan simultáneamente sus capacidades operativas y su superficie de exposición [1], [2], [7], [8], [9]. En este escenario, la dificultad no consiste únicamente en detectar vulnerabilidades, sino en determinar cuáles de ellas comprometen de forma más directa la continuidad del servicio y, por tanto, requieren atención prioritaria [15], [17], [18], [19].

Desde esta perspectiva, el valor del sistema no radica en una única herramienta ni en una única técnica de análisis. Su aporte consiste en integrar descubrimiento de superficie, escaneo de vulnerabilidades, revisión de configuraciones, evaluación de controles, registro de eventos, monitoreo, ejecución de scripts de mitigación y priorización contextual mediante aprendizaje automático dentro de una sola arquitectura [20], [25], [29], [33], [38], [39], [41]. Esta integración reduce la fragmentación que habitualmente existe entre los resultados técnicos de seguridad y el proceso real de decisión, acercando el análisis a una lógica operativa más útil para la remediación.

En términos prácticos, la solución combina una base tecnológica ligera, replicable y controlable con una lógica metodológica sustentada en referencias como NIST Cybersecurity Framework, ISO/IEC 27001, ISO/IEC 27005 y CIS Controls [13], [15], [18], [25]. De este modo, el framework se entiende como la materialización técnica de una estrategia de evaluación estructurada, orientada a mejorar la capacidad del sistema para identificar exposición, interpretar hallazgos y sostener decisiones de remediación justificadas.

3.2. Principios de diseño de la solución

El framework se apoya en cinco principios de diseño que orientan tanto su estructura técnica como su coherencia metodológica:

1. **Modularidad.** Cada función relevante del sistema se organiza como un componente delimitado, de modo que el reconocimiento de superficie, el escaneo, la normalización, la priorización, el checklist de controles, el registro de eventos, la gestión de alertas y la ejecución de scripts puedan mantenerse o ampliarse sin comprometer el conjunto. Esta modularidad no persigue fragmentar el sistema, sino permitir que un framework integral crezca de manera ordenada y justificable [18], [22], [25].
2. **Integrabilidad.** Desde el diseño se asume que ninguna herramienta aislada puede ofrecer una visión completa del entorno. Por ello, el sistema se construye para interactuar con distintos componentes de análisis y combinar evidencias técnicas diversas. Este principio es fundamental porque el riesgo real no se manifiesta en una sola dimensión, sino en la relación entre exposición, configuración, comportamiento operativo y criticidad del activo [15], [17], [19], [20].
3. **Normalización.** Todo hallazgo técnico, ya provenga de escaneo, enumeración, monitoreo o verificación de controles, debe traducirse a una estructura común. Este principio evita que el framework se convierta en una colección de salidas inconexas y permite analizar resultados bajo una misma gramática operativa [20], [25], [29].
4. **Priorización contextual.** La solución asume desde su diseño que la severidad técnica no basta para ordenar el tratamiento del riesgo. Por ello, el sistema incorpora variables adicionales, como exposición, tipo de servicio, criticidad del activo, señales de monitoreo, estado de controles y evidencia de validación, con el fin de estimar una prioridad más cercana a la realidad operativa [45], [46], [47], [48].
5. **Continuidad operativa.** La solución no se concibe como una auditoría puntual, sino como un framework capaz de sostener evaluaciones repetibles, registrar resultados, comparar estados y apoyar procesos de mejora continua. Esta lógica le permite acompañar un entorno digital dinámico, donde la postura de seguridad debe observarse y ajustarse de manera constante [16], [18], [21], [25].

3.3. Arquitectura general del sistema

La arquitectura general del framework se organiza en cuatro capas principales: recolección de evidencia, procesamiento y normalización, análisis y priorización, y visualización orientada a la decisión. Esta organización responde a una secuencia lógica de tratamiento de la información y permite explicar con claridad el rol de cada componente dentro del sistema completo.

La primera capa, correspondiente a la recolección de evidencia, integra los mecanismos encargados de ampliar la visibilidad sobre el entorno evaluado y generar hallazgos preliminares. Aquí se incorporan herramientas de escaneo web, enumeración de recursos, revisión de configuraciones y módulos de verificación de controles. Esta capa cumple la función de observación inicial y constituye la base a partir de la cual se construye el resto del análisis [20], [29], [33], [38], [39].

La segunda capa es la de procesamiento y normalización. Su función consiste en recibir los resultados emitidos por los distintos módulos y traducirlos a una estructura homogénea. Esta operación no se limita al almacenamiento, sino que permite construir una gramática común de análisis en torno a atributos como herramienta de origen, tipo de hallazgo, severidad técnica, activo afectado, evidencia, exposición y estado de tratamiento [20], [25].

La tercera capa corresponde al análisis y la priorización. Sobre la base ya normalizada, el sistema aplica una lógica de *scoring* y un modelo supervisado de aprendizaje automático orientado a estimar criticidad contextual. Esta capa no genera hallazgos nuevos, pero reinterpreta los existentes a partir de variables técnicas y operativas, acercando la evaluación al problema real de la remediación [45], [46], [59], [60], [64], [65].

La cuarta capa es la de visualización y apoyo a la decisión. A través de una interfaz unificada, el framework presenta hallazgos, métricas, estados de controles, alertas, registros de actividad, scripts ejecutados y rankings de

vulnerabilidades. De esta forma, la arquitectura no solo produce datos, sino que los transforma en evidencia legible, trazable y útil para distintos perfiles de usuario [54], [57].

3.4. Stack tecnológico y base operativa

La base tecnológica del sistema fue seleccionada con un criterio pragmático, considerando eficiencia operativa, bajo acoplamiento, facilidad de despliegue, replicabilidad y coherencia con un entorno académico de prueba controlada. Esta selección permite asegurar compatibilidad entre componentes, mantener un prototipo reproducible y sostener el análisis sobre tecnologías suficientemente robustas y viables dentro del alcance del proyecto de grado.

3.4.1. Python y FastAPI como núcleo de orquestación

El núcleo lógico del framework está implementado en Python, utilizando FastAPI como backend de orquestación. Esta elección responde a varios factores. En primer lugar, Python ofrece una base flexible para integrar herramientas de análisis, procesamiento de datos, lógica de *scoring* y aprendizaje automático dentro de un mismo lenguaje [50], [64], [65]. En segundo lugar, FastAPI permite estructurar *endpoints* de forma clara, coordinar solicitudes relacionadas con escaneo, consulta de resultados, entrenamiento del modelo, generación de alertas y ejecución de scripts, y mantener una arquitectura backend liviana y comprensible [51], [52].

El backend no se utiliza únicamente como expositor de servicios. Su función principal es unificar el sistema. Desde esta capa se coordina la interacción entre herramientas externas, base de datos, módulos de checklist, auditoría, análisis contextual y visualización. En ese sentido, Python y FastAPI constituyen la base que permite materializar el framework como un sistema único y coherente [50], [51], [52].

3.4.2. Frontend unificado y visualización analítica

La interfaz principal del sistema se apoya en Tailwind CSS, acompañado por una capa frontend liviana orientada a consolidar la interacción con los distintos módulos del framework. Esta elección responde a la necesidad de construir una interfaz visual clara, moderna, mantenible y suficientemente flexible para representar hallazgos, métricas, estados de controles, módulos operativos y acciones de remediación sin introducir complejidad innecesaria en el prototipo [57].

Complementariamente, se implementa una capa de visualización analítica con Streamlit, que permite construir *dashboards* orientados a la exploración de hallazgos, la distribución de severidades, la comparación entre severidad técnica y prioridad contextual, y la observación del comportamiento general del sistema [54]. Mientras el frontend unificado actúa como consola operativa del framework, Streamlit profundiza la lectura analítica de los datos.

En esta versión funcional del MVP se privilegia, por tanto, una interfaz desacoplada y ligera basada en Tailwind CSS, JavaScript y visualización analítica en Python. Esta decisión no impide una evolución posterior hacia frameworks frontend más estructurados, como Nuxt.js [58], pero evita introducir una complejidad adicional que no resulta necesaria para demostrar el funcionamiento del sistema dentro del alcance del proyecto de grado.

3.4.3. SQLite como repositorio de persistencia

Para la persistencia de hallazgos, *logs*, alertas, controles, scripts ejecutados y otros artefactos operativos se utiliza SQLite [53]. Esta decisión se justifica por su ligereza, su facilidad de despliegue y su suficiencia para un entorno controlado de evaluación académica. SQLite permite sostener una estructura unificada de almacenamiento sin requerir infraestructura adicional compleja, lo que favorece la reproducibilidad del prototipo.

La base de datos no se emplea solo como repositorio pasivo. Su diseño cumple una función analítica. Al almacenar hallazgos provenientes de fuentes heterogéneas bajo una estructura común, SQLite se convierte en la base que sostiene la normalización, la correlación y la priorización del riesgo dentro del framework [20], [25], [53].

3.4.4. Proxmox VE como entorno de virtualización y laboratorio controlado

El entorno de despliegue y pruebas se apoya en Proxmox Virtual Environment (VE), plataforma adecuada para este trabajo por su capacidad de combinar máquinas virtuales y contenedores dentro de una misma infraestructura de laboratorio [56]. Su valor en este proyecto de grado no radica únicamente en la virtualización, sino en la posibilidad de construir entornos controlados, reproducibles y aislados para ejecutar análisis, pruebas técnicas y validaciones sin comprometer sistemas productivos.

Proxmox permite definir laboratorios donde es posible ejecutar escaneos, validar comportamientos, recrear escenarios de exposición y restaurar rápidamente estados seguros mediante *snapshots*. Esta capacidad resulta particularmente útil en el contexto de una investigación aplicada a infraestructura crítica, donde la validación de ciertos escenarios requiere aislamiento, control y reversibilidad [16], [56].

3.4.5. Docker como soporte de contenedorización y despliegue reproducible

Complementariamente, Docker se considera una tecnología de apoyo para la contenedorización de servicios y para la proyección del framework hacia despliegues más controlados y reproducibles [55]. En un entorno como este, Docker ofrece ventajas claras: segmentación lógica entre componentes, control sobre dependencias, facilidad de replicación y mayor consistencia entre entornos de prueba.

Aunque la versión funcional del MVP se apoya principalmente en una arquitectura ligera desplegada sobre el entorno controlado del laboratorio, la incorporación conceptual y operativa de Docker fortalece la capacidad del sistema para avanzar hacia escenarios de integración continua, automatización y despliegues más consistentes. En este sentido, Docker funciona como una extensión natural de la base operativa del framework [24], [32], [55].

3.5. Integración de herramientas de seguridad

La integración de herramientas dentro del framework responde a una lógica funcional. Cada una se incorpora según el tipo de evidencia que aporta al proceso de evaluación y no como una utilidad aislada sin relación con el resto de la arquitectura.

3.5.1. OWASP ZAP como motor principal de escaneo

OWASP ZAP constituye el núcleo del análisis automatizado sobre aplicaciones y servicios web. Se integra mediante API, lo que permite iniciar escaneos, recuperar resultados y procesarlos directamente desde el backend [33], [34]. Su valor principal dentro del framework reside en la detección de vulnerabilidades web, errores de configuración y patrones comunes de exposición, convirtiéndose en una de las principales fuentes estructuradas de hallazgos para la capa de análisis técnico [26], [29], [33].

3.5.2. Gobuster como módulo de descubrimiento de superficie

Gobuster se integra para fortalecer la visibilidad sobre el perímetro expuesto. Su función es enumerar rutas, directorios, archivos y recursos accesibles que pueden no estar claramente documentados ni visibles en una observación superficial [38]. Este aporte es metodológicamente importante porque permite conocer mejor el entorno antes de interpretar sus debilidades. Dentro del framework, Gobuster no compite con el escaneo de vulnerabilidades; lo complementa al ampliar la comprensión de la superficie de ataque [20], [29], [38].

3.5.3. Nikto como revisión de configuración y hardening

Nikto se incorpora como una herramienta complementaria orientada a identificar configuraciones inseguras, cabeceras faltantes, archivos sensibles expuestos y problemas comunes de *hardening* en servicios web [39]. Su función dentro del framework es ampliar el espectro de hallazgos hacia condiciones de exposición que no siempre aparecen como vulnerabilidades clásicas, pero que afectan de manera concreta la postura general de seguridad [23], [39].

3.5.4. Herramientas de apoyo y validación avanzada

Se mantienen además herramientas como Burp Suite, Nessus y Metasploit dentro del marco técnico de la solución como referencias coherentes de validación y extensión. Burp Suite representa la dimensión de análisis manual avanzado sobre aplicaciones web y APIs [35]. Nessus sitúa el framework dentro de una proyección más amplia hacia evaluación de infraestructura y servicios de red [36]. Metasploit, por su parte, cumple una función de validación ofensiva controlada sobre hallazgos específicos en entornos de laboratorio [37].

No todas estas herramientas participan con el mismo nivel de automatización en el MVP, pero sí forman parte del horizonte técnico del sistema y refuerzan la idea de que el framework puede crecer sin abandonar su diseño conceptual.

3.5.5. Monitoreo, correlación y observación continua

Junto con los módulos de evaluación, se integra una capa de monitoreo y observación continua que permite relacionar hallazgos técnicos con señales operativas derivadas de *logs*, eventos y alertas. En esta dimensión, herramientas y enfoques cercanos a Wazuh, Snort o Suricata resultan conceptualmente pertinentes porque muestran que el framework no se limita a descubrir debilidades, sino que también puede conectarlas con evidencia observada durante la operación [21], [41], [42], [43].

3.6. Módulo de normalización de hallazgos

Uno de los componentes más importantes del framework es el módulo de normalización de hallazgos. La necesidad de este módulo surge directamente de la heterogeneidad producida por las distintas herramientas y mecanismos del sistema. Un framework que integra escaneo, enumeración, checklist, eventos, *logs* y alertas no puede depender de formatos aislados. Necesita una estructura común que permita analizar los resultados bajo un mismo criterio [20], [25], [29].

El módulo de normalización transforma la evidencia técnica a una representación uniforme compuesta por atributos como herramienta de origen, tipo de hallazgo, severidad técnica, activo afectado, evidencia, exposición estimada, estado de tratamiento y señales contextuales adicionales. Gracias a esta estructura, el sistema puede comparar hallazgos de distinta naturaleza y tratarlos como parte de una misma base analítica.

Este componente es el que otorga unidad real al framework. Sin él, la solución sería solo una acumulación de salidas técnicas. Con él, se convierte en una arquitectura coherente de evaluación, donde el descubrimiento, el escaneo, la verificación de controles, el monitoreo y la priorización operan sobre una base compartida de evidencia [20], [25].

3.7. Módulo de priorización contextual basado en aprendizaje automático

El núcleo analítico del framework se materializa en el módulo de priorización contextual. Este componente recibe los hallazgos ya normalizados y estima su criticidad relativa a partir de variables técnicas y operativas. Entre estas variables se consideran severidad CVSS, tipo de hallazgo, tipo de servicio, nivel de exposición, necesidad de autenticación, criticidad del activo, alertas asociadas, estado de controles, evidencias de scripts ejecutados y señales de validación [45], [46], [47], [48], [49].

La incorporación de aprendizaje automático se justifica porque permite construir un criterio de priorización más flexible que la severidad aislada. En lugar de asumir que el orden de tratamiento depende solo del *score* técnico, el sistema puede aprender patrones asociados a combinaciones de variables y producir una clasificación más cercana a la criticidad operativa real [45], [46], [59], [60], [64], [65].

Para esta tarea se privilegian modelos supervisados interpretables, como Regresión Logística o *Random Forest*, debido a que permiten trabajar con datos estructurados, generar salidas comprensibles y evaluar el desempeño mediante métricas cuantitativas [59], [60], [64], [65]. El objetivo no es construir una capa opaca, sino demostrar que una capa analítica bien delimitada mejora la utilidad del framework como soporte para la decisión.

Este apartado describe el propósito y la función del módulo dentro de la solución. La explicación detallada del proceso de entrenamiento, construcción del *dataset*, validación del modelo y resultados obtenidos se desarrolla posteriormente en el capítulo de validación, donde corresponde presentar la evidencia experimental del sistema.

3.8. Flujo operativo y metodología técnica aplicada

El funcionamiento del framework puede describirse como una secuencia de etapas articuladas que reflejan la lógica metodológica del sistema.

En una primera fase se realiza la identificación y el mapeo de activos, buscando reconocer servidores, APIs, aplicaciones, servicios y recursos expuestos que forman parte del entorno evaluado. Esta visibilidad inicial es necesaria para establecer qué debe analizarse y con qué profundidad [18], [19], [20].

En una segunda fase se ejecutan procesos de descubrimiento de superficie y análisis de vulnerabilidades, utilizando herramientas como ZAP, Gobuster y Nikto para producir hallazgos técnicos, ampliar el conocimiento del perímetro expuesto y observar configuraciones inseguras [29], [33], [38], [39]. Según el escenario de laboratorio, esta etapa puede complementarse con herramientas de apoyo para infraestructura o validación manual.

En una tercera fase se desarrolla la evaluación de riesgos y la normalización de hallazgos. Aquí se consolida la información recolectada bajo una estructura común y se la relaciona con criterios inspirados en NIST SP 800-30, ISO/IEC 27005 y métricas como CVSS, de modo que el sistema no solo acumule resultados, sino que empiece a interpretarlos [15], [19], [48].

En una cuarta fase se incorporan pruebas de validación controlada, especialmente en entornos aislados, cuando resulta necesario comprobar la explotabilidad de ciertos hallazgos o fortalecer la interpretación de su criticidad [20], [37].

En una quinta fase se activan procesos de mitigación y *hardening*, apoyados tanto en decisiones analíticas como en scripts ejecutables que permiten realizar tareas controladas de parcheo, revisión de configuración, endurecimiento y verificación técnica [23], [24], [25].

Finalmente, en una sexta fase se mantiene un esquema de monitoreo continuo y revisión cíclica, donde eventos, *logs*, alertas y estados de controles alimentan nuevamente al sistema y enriquecen la comprensión del riesgo. Esta última fase es especialmente importante porque permite que el framework no termine en el hallazgo, sino que integre observación posterior, trazabilidad y retroalimentación analítica [21], [22], [41], [42], [43].

3.9. Valor diferencial de la propuesta

El valor diferencial de la propuesta no reside en el uso aislado de herramientas conocidas ni en la mera incorporación de aprendizaje automático. Su aporte principal se encuentra en la articulación de ambos elementos dentro de una arquitectura metodológica coherente.

Por una parte, el framework mantiene el rigor de una evaluación estructurada, alineada con principios de NIST, ISO/IEC 27001, ISO/IEC 27005 y CIS Controls [13], [15], [18], [25]. Por otra, introduce una capa analítica y operativa que permite superar una limitación frecuente en la práctica: depender únicamente de la severidad técnica como criterio de prioridad. Esta combinación da lugar a un sistema más útil para la realidad operativa, porque reduce la distancia entre el hallazgo técnico y la decisión de tratamiento.

El valor de la propuesta también se expresa en su capacidad para unificar evaluación técnica, evidencia operativa, estado de controles y priorización contextual dentro de un único flujo. Esta unificación constituye el aporte diferenciador del framework frente a aproximaciones centradas exclusivamente en el escaneo o en la visualización aislada de resultados.

3.10. Proyección y escalabilidad de la solución

Aunque el prototipo desarrollado en este proyecto de grado se implementa en un entorno experimental y controlado, la lógica del framework permite proyectar su crecimiento hacia escenarios de mayor cobertura. La incorporación de herramientas adicionales, la ampliación del modelo de datos, el fortalecimiento de la contenedorización, la integración más profunda con sistemas de monitoreo y la adopción de componentes frontend más estructurados pueden aumentar gradualmente la capacidad del sistema sin alterar su diseño conceptual [24], [31], [32], [55], [58].

Esta proyección demuestra que la solución no se limita a una demostración puntual. Se sostiene sobre una base metodológica y tecnológica escalable, preparada para incorporar nuevas fuentes de evidencia y nuevas capacidades analíticas sin perder el principio que organiza toda la propuesta: evaluar, consolidar, interpretar y priorizar el riesgo dentro de un único framework integral.

4 | VALIDACION DE LA SOLUCIÓN

4.1. Enfoque de validación

La validación de la solución que propongo en esta tesis no puede reducirse a una comprobación superficial de funcionamiento ni a una simple demostración visual de la interfaz. Dado que el framework fue diseñado para responder a un problema aplicado de ciberseguridad en un entorno digitalizado y operacionalmente sensible, considero que su validación debe centrarse en demostrar que el sistema es capaz de cumplir, de manera integrada, las funciones para las cuales fue concebido. En consecuencia, la validación que desarrollo se orienta a comprobar que el framework permite descubrir superficie expuesta, recolectar hallazgos técnicos desde distintas fuentes, normalizarlos, enriquecerlos con contexto adicional, priorizarlos de forma útil para la remediación y representar esa información dentro de una arquitectura operativa coherente.

Desde esta perspectiva, no valido únicamente la corrección de componentes individuales, sino la capacidad del sistema para funcionar como una plataforma unificada de evaluación y apoyo a la decisión. Esto resulta importante porque el valor de la propuesta no está en la existencia aislada de herramientas como ZAP, Gobuster, Nikto o el módulo de priorización, sino en su articulación dentro de un solo framework capaz de traducir evidencia técnica heterogénea en una salida analítica más útil que la severidad aislada. Por ello, la validación que presento se estructura sobre una lógica funcional e integradora, donde cada módulo se evalúa no solo por lo que hace por separado, sino por la forma en que contribuye al comportamiento global del sistema.

En términos metodológicos, la validación se centra en cuatro dimensiones principales. La primera corresponde a la validación funcional, orientada a comprobar que los módulos del framework ejecutan correctamente las tareas para las cuales fueron diseñados. La segunda corresponde a la validación de integración, destinada a verificar que los distintos componentes intercambian datos de forma consistente dentro de una misma arquitectura. La tercera dimensión es la validación analítica, donde examino si la capa de normalización, evaluación de riesgo y priorización contextual realmente añade valor interpretativo al sistema. Finalmente, la cuarta dimensión es la validación operativa, mediante la cual compruebo que el framework puede desplegarse, ejecutarse y observarse en un entorno de pruebas replicable y controlado.

Bajo esta lógica, la validación no pretende demostrar que el framework reemplaza herramientas comerciales o que constituye una solución definitiva para cualquier escenario productivo. Lo que busco demostrar es algo distinto y metodológicamente más adecuado al alcance de la tesis: que el framework propuesto es técnicamente viable, operativamente consistente y suficientemente robusto para responder al problema que motivó la investigación en el contexto donde fue planteado.

4.2. Criterios de validación

Para validar la solución definí un conjunto de criterios directamente vinculados al objetivo general y a los objetivos específicos de la tesis. Estos criterios no se formulan como simples verificaciones de software, sino como condiciones que el framework debe cumplir para demostrar que efectivamente resuelve el problema de investigación.

El primer criterio consiste en comprobar que el sistema puede descubrir y recolectar evidencia desde múltiples fuentes. Esto implica verificar que el framework no depende de una única herramienta, sino que es capaz de ejecutar análisis sobre aplicaciones y servicios web, enumerar recursos expuestos, revisar configuraciones y registrar información relevante desde el entorno operativo.

El segundo criterio consiste en demostrar que el sistema puede normalizar hallazgos heterogéneos dentro de una estructura común. Este punto es esencial, porque una parte central del problema identificado en la tesis era precisamente la dificultad de interpretar resultados inconexos producidos por fuentes distintas. En consecuencia, considero validado este criterio si el framework logra representar hallazgos diversos bajo una misma base analítica y con atributos comparables.

El tercer criterio corresponde a la capacidad de evaluación y priorización contextual. Aquí no basta con mostrar que el sistema ordena vulnerabilidades. Lo que debo demostrar es que la solución incorpora variables adicionales a la severidad técnica y produce una jerarquización más cercana al contexto operativo del entorno evaluado.

El cuarto criterio se relaciona con la integración operativa del framework. Esto implica validar que los módulos de backend, persistencia, visualización, checklist, auditoría, scripts y priorización funcionan dentro de una sola arquitectura, sin quedar reducidos a piezas aisladas o manualmente desconectadas.

El quinto criterio consiste en verificar que el sistema mantiene trazabilidad y capacidad de observación. Para ello resulta necesario comprobar que el framework registra eventos, logs, alertas, estados de controles y ejecuciones de scripts, de forma que el proceso de evaluación no termine en el hallazgo puntual, sino que conserve evidencia operativa del ciclo completo.

Finalmente, el sexto criterio corresponde a la replicabilidad del entorno de validación. Dado que se trata de una tesis aplicada, considero importante demostrar que el framework puede desplegarse y probarse dentro de un entorno controlado, reproducible y metodológicamente justificable, apoyado en virtualización y servicios desacoplados.

4.3. Entorno técnico de validación

La validación del framework se desarrolla sobre un entorno técnico controlado, diseñado para permitir la ejecución coordinada de los módulos del sistema sin comprometer infraestructuras productivas. Para ello utilizo una base tecnológica compuesta por Python, FastAPI, Uvicorn, SQLite, Streamlit, Tailwind CSS y un entorno de virtualización soportado por Proxmox VE. Esta elección responde a criterios de viabilidad, mantenibilidad, bajo acoplamiento y capacidad de replicación.

El backend del framework se ejecuta en Python, utilizando FastAPI como capa de orquestación y Uvicorn como servidor ASGI. Esta combinación me permite exponer endpoints claros para escaneo, entrenamiento del modelo, priorización, consulta de hallazgos, ejecución de scripts, revisión de checklist, lectura de eventos, logs y alertas. Sobre esta base, el sistema integra la lógica principal de procesamiento y coordinación de los módulos.

La persistencia se resuelve mediante SQLite, que cumple la función de almacenar hallazgos normalizados, inteligencia pública, estados de controles, registros de auditoría, alertas derivadas del sistema y ejecuciones de scripts. En el contexto de la tesis, esta base resulta suficiente para sostener el flujo analítico del prototipo y demostrar la consistencia del framework sin requerir una infraestructura más compleja.

La capa de visualización se construye mediante dos componentes complementarios. Por una parte, utilizo Streamlit para el dashboard analítico, donde puedo representar tablas de hallazgos, distribuciones de severidad, rankings de prioridad y comparaciones entre severidad técnica y prioridad contextual. Por otra, incorporo una interfaz unificada estilizada con Tailwind CSS, que funciona como consola operativa del framework y me permite exponer módulos, estados, acciones y salidas del sistema dentro de una experiencia visual coherente con el MVP.

Como soporte para la validación en laboratorio, utilizo Proxmox VE como entorno de virtualización. Esta plataforma me permite aislar escenarios de prueba, levantar máquinas virtuales o entornos controlados para validación técnica y mantener la posibilidad de restaurar estados seguros mediante snapshots. En una investigación aplicada a ciberseguridad, esta capacidad resulta especialmente relevante, porque permite ejecutar pruebas sin trasladar el riesgo al entorno operativo real.

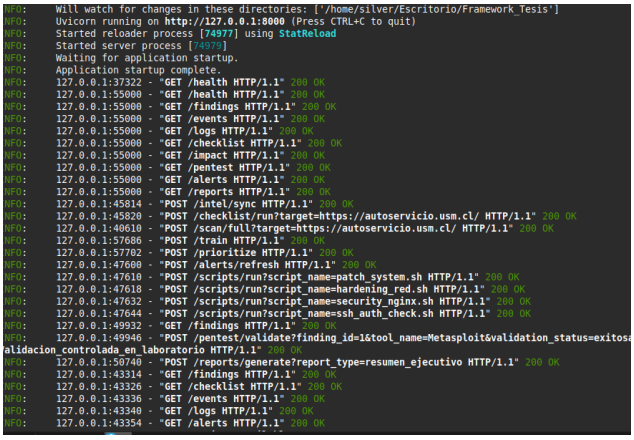


Figura 4.1: Entorno de validación del framework desplegado sobre backend, dashboard y laboratorio virtualizado
(Fuente: Elaboración propia)

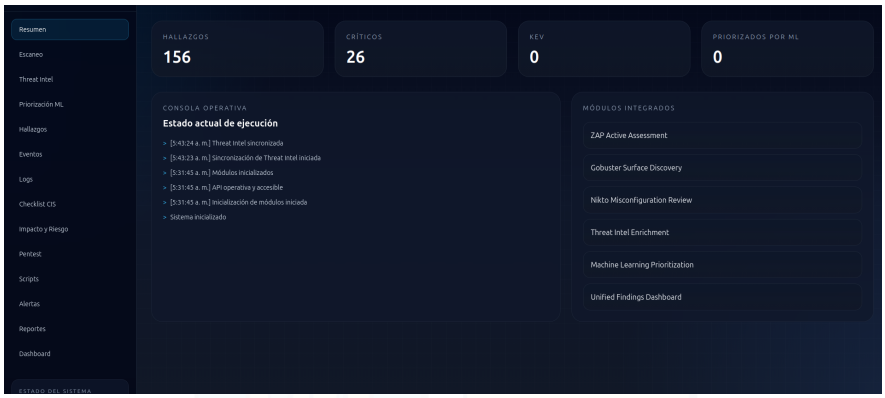


Figura 4.2:
Entorno del Sistema funcionando
(Fuente: Elaboración propia)

4.4. Validación funcional del framework

La primera dimensión de validación que desarrollo es la funcional. En ella compruebo que cada una de las capacidades relevantes del framework ejecuta correctamente la función para la cual fue diseñada dentro del sistema.

4.4.1. Validación del módulo de descubrimiento y escaneo

La validación comienza en la capa de recolección de evidencia. En esta fase compruebo que el framework puede iniciar procesos de análisis sobre objetivos definidos, ejecutar reconocimiento de superficie y registrar hallazgos desde herramientas integradas como OWASP ZAP, Gobuster y Nikto. Lo importante en esta etapa no es únicamente que las herramientas se ejecuten, sino que sus resultados sean capturados por el backend y preparados para su posterior tratamiento.

Desde el punto de vista del funcionamiento, esta validación permite demostrar que el framework efectivamente observa el entorno evaluado y no se limita a representar información precargada. Asimismo, confirma que la plataforma puede operar sobre un flujo real de entrada, donde los hallazgos provienen de análisis técnicos ejecutados desde el propio sistema.

Figura 4.3: Módulo de escaneo integrado en ejecución
(Fuente: Elaboración propia)

4.4.2. Validación del módulo de normalización

Una vez generados los hallazgos, valido la capacidad del framework para normalizarlos. Este punto tiene una importancia central, porque sin una capa de normalización la solución quedaría reducida a una suma de herramientas con salidas incompatibles. La validación de este módulo consiste en comprobar que los hallazgos provenientes de distintas fuentes son traducidos a una estructura común que conserva atributos comparables, tales como herramienta de origen, tipo de hallazgo, activo afectado, severidad técnica, evidencia, exposición estimada y estado.

Desde una perspectiva metodológica, esta validación confirma que el framework no solo ejecuta herramientas, sino que convierte sus resultados en una base analítica integrada. Esta capacidad es precisamente la que permite sostener la evaluación posterior y la priorización contextual.

ID	Tool	Tipo	Objetivo	CVSS	EPSS	KEV	Score Final
1	ZAP	Respuesta de Gestión de Sesión Identificada	https://autoservicio.usm.cl	1.0	0.00	0	15.03
165	ZAP	Cabecera Content Security Policy (CSP) no configurada	https://autoservicio.usm.cl	5.0	0.00	0	12.15
161	ZAP	Cabecera Content Security Policy (CSP) no configurada	https://autoservicio.usm.cl	5.0	0.00	0	12.15
157	ZAP	Cabecera Content Security Policy (CSP) no configurada	https://autoservicio.usm.cl	5.0	0.00	0	12.15
156	ZAP	Format String Error (Error de formato de cadena)	https://autoservicio.usm.cl	5.0	0.00	0	12.15
155	ZAP	Buffer Overflow	https://autoservicio.usm.cl	5.0	0.00	0	12.15
134	ZAP	Falta atributo de integridad de recursos secundarios	https://autoservicio.usm.cl	5.0	0.00	0	12.15

Figura 4.4: Esquema de normalización de hallazgos provenientes de múltiples herramientas
(Fuente: Elaboración propia)

4.4.3. Validación del módulo de evaluación de impacto y riesgo

En una etapa posterior valido el módulo encargado de interpretar los hallazgos desde la lógica del impacto y la probabilidad. En esta capa compruebo que el framework puede organizar la evidencia recolectada en categorías de riesgo comprensibles, utilizando CVSS v3.1 como referencia de severidad técnica e incorporando criterios adicionales para estimar criticidad contextual.

La relevancia de esta validación radica en demostrar que el sistema ya no se limita a mostrar fallas, sino que las interpreta dentro de una lógica de riesgo. Esto representa un cambio importante, porque convierte la salida del framework en una herramienta de análisis y no solo de observación.

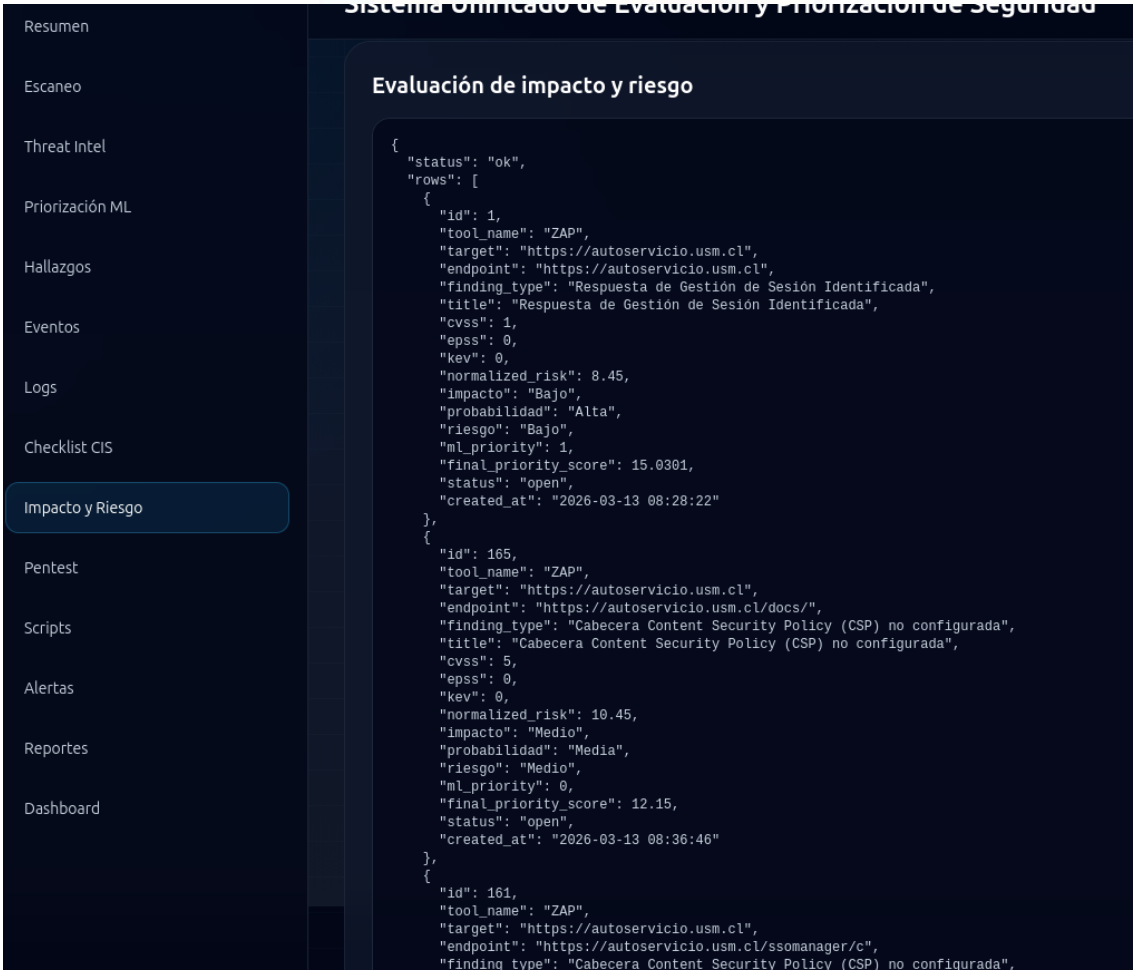


Figura 4.5: Vista del módulo de evaluación de impacto y riesgo

(Fuente: Elaboración propia)

4.4.4. Validación del módulo de priorización contextual

La validación del módulo de priorización contextual constituye una de las comprobaciones más importantes de la tesis. En esta fase verifico que el sistema es capaz de procesar hallazgos ya consolidados y asignarles una prioridad relativa utilizando variables que exceden la severidad técnica pura. Entre estas variables se encuentran la exposición externa, la criticidad del activo, la necesidad de autenticación, el tipo de servicio, la existencia de alertas asociadas, el estado de controles y otras señales que enriquecen el contexto del hallazgo.

El propósito de esta validación no es demostrar una autonomía absoluta del modelo, sino comprobar que la capa analítica efectivamente aporta un criterio adicional de ordenamiento. Desde esta perspectiva, el valor del módulo no reside en reemplazar la evaluación humana, sino en fortalecerla mediante una estimación de criticidad más próxima al riesgo operativo.

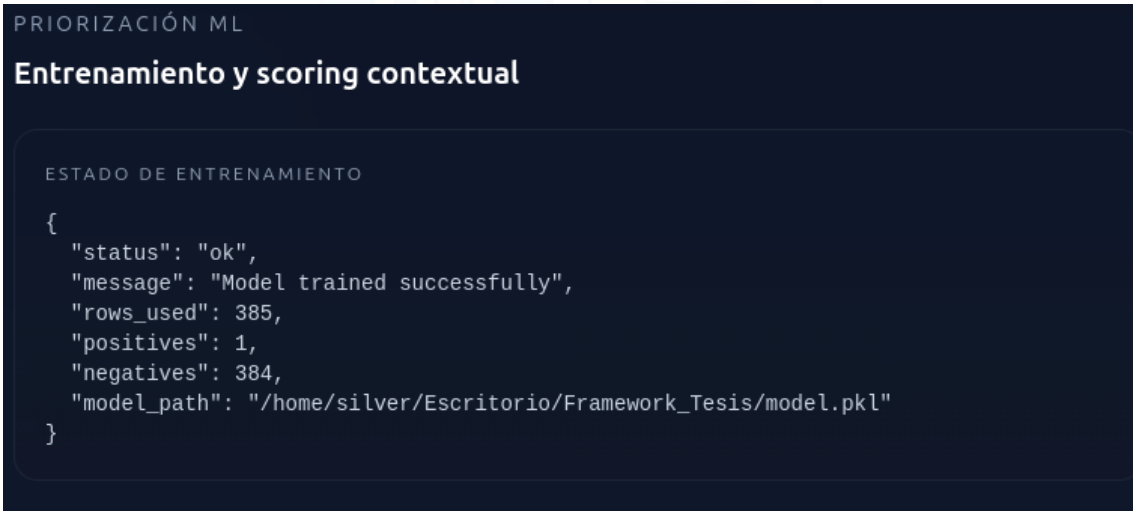


Figura 4.6: Arquitectura del módulo inteligente de priorización contextual de vulnerabilidades
(Fuente: Elaboración propia)

Figura 4.7: Comparación entre priorización basada en severidad técnica y priorización contextual
(Fuente: Elaboración propia)

4.4.5. Validación del módulo de pruebas de penetración controladas

También valido el componente de pruebas de penetración controladas, cuyo propósito es verificar la explotabilidad de hallazgos seleccionados dentro de un entorno de laboratorio. En esta capa no busco alimentar masivamente la base de datos, sino demostrar que el framework contempla un mecanismo para distinguir entre debilidades detectadas y condiciones cuya explotación puede ser confirmada de manera técnica y controlada.

La existencia de este módulo fortalece la validez de la solución, porque muestra que el sistema no se limita a inferir criticidad desde el hallazgo nominal, sino que puede incorporar evidencia adicional de validación cuando las condiciones del entorno lo permiten.

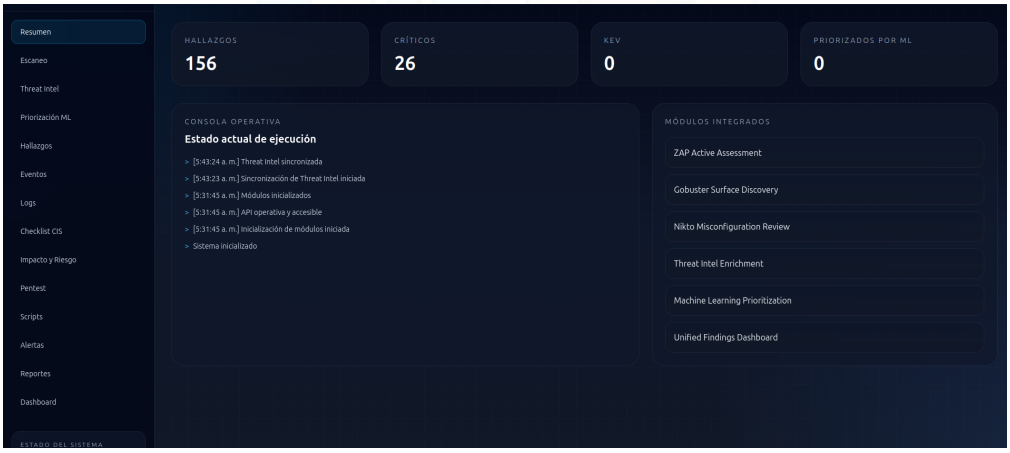


Figura 4.8:
Vista del módulo de pruebas de penetración controladas
(Fuente: Elaboración propia)

4.4.6. Validación del módulo de mitigación y hardening

El framework también incorpora un módulo de mitigación y hardening, cuya función es cerrar el ciclo entre evaluación y tratamiento. La validación de este componente consiste en comprobar que el sistema puede ejecutar acciones concretas, tales como verificación de configuración, endurecimiento básico, comprobación de autenticación y ejecución de scripts asociados a medidas correctivas.

Este punto resulta especialmente importante porque evita que la solución quede en una etapa puramente diagnóstica. En términos metodológicos, valida que el framework no solo observa y prioriza, sino que también puede apoyar la aplicación de medidas concretas orientadas a reducir la exposición del entorno.

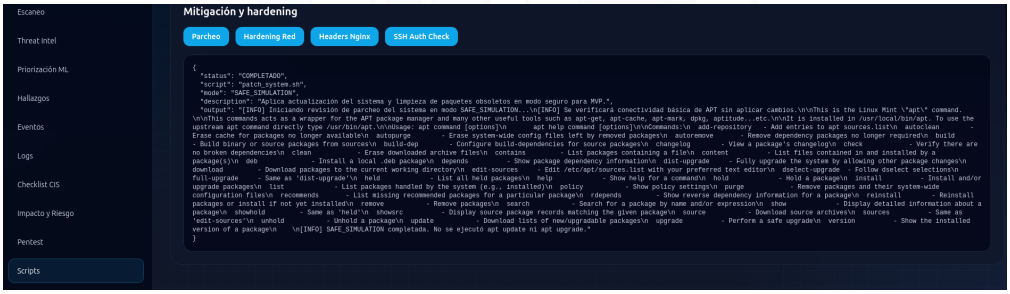


Figura 4.9:
Ejecución de script de remediación automática para aplicar parches y actualizaciones críticas
(Fuente: Elaboración propia)

Figura 4.10:
Vista del módulo de ejecución de scripts de mitigación y hardening
(Fuente: Elaboración propia)

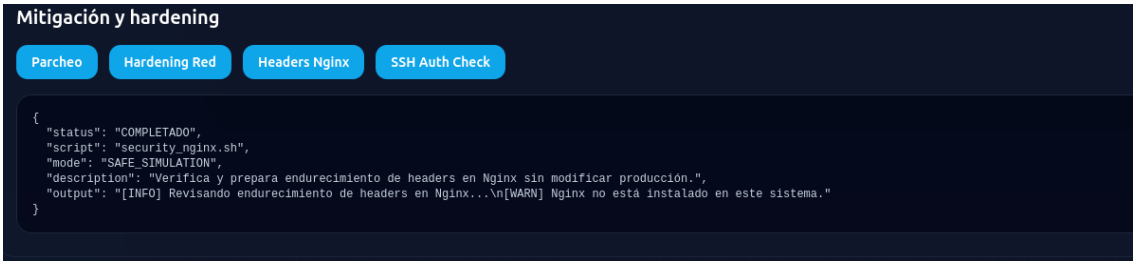


Figura 4.11:
Ejecución de script de remediación automática para cierre de puertos o endurecimiento de red
(Fuente: Elaboración propia)

Figura 4.12:
Ejecución de script de remediación automática para verificación de autenticación SSH
(Fuente: Elaboración propia)

Figura 4.13:
Vista del módulo de ejecución de scripts de mitigación y hardening
(Fuente: Elaboración propia)

4.4.7. Validación del módulo de monitoreo y respuesta

Finalmente, valido la capacidad del framework para registrar eventos, logs y alertas derivadas de su propia operación y del comportamiento observado en el sistema. Esta validación resulta relevante porque demuestra que la solución no termina en el hallazgo inicial, sino que extiende su lógica hacia la trazabilidad y la observación continua.

Aunque este módulo no constituye el núcleo del proceso de priorización, sí fortalece la visión integral del framework. Su validación me permite sostener que el sistema no actúa solo como un ejecutor de pruebas, sino como una arquitectura de evaluación que conserva evidencia operativa del ciclo completo.

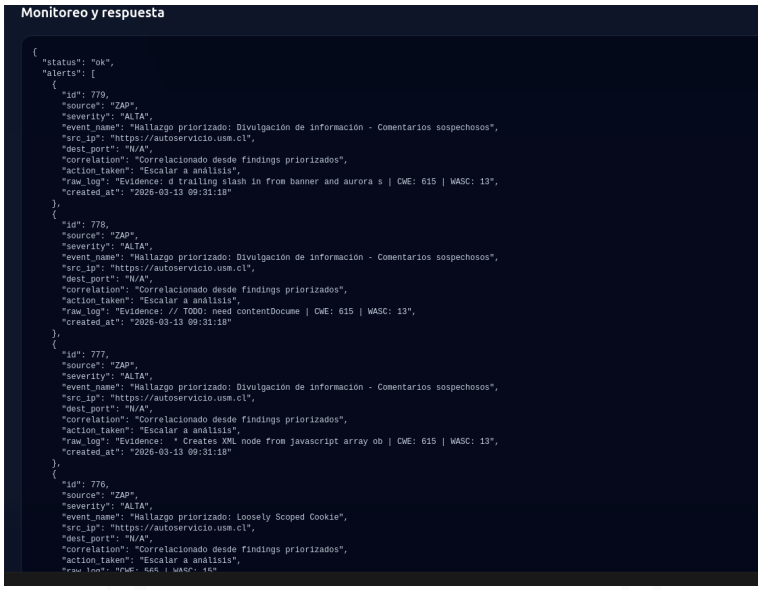


Figura 4.14:
Vista de alertas del Framework
(Fuente: Elaboración propia)

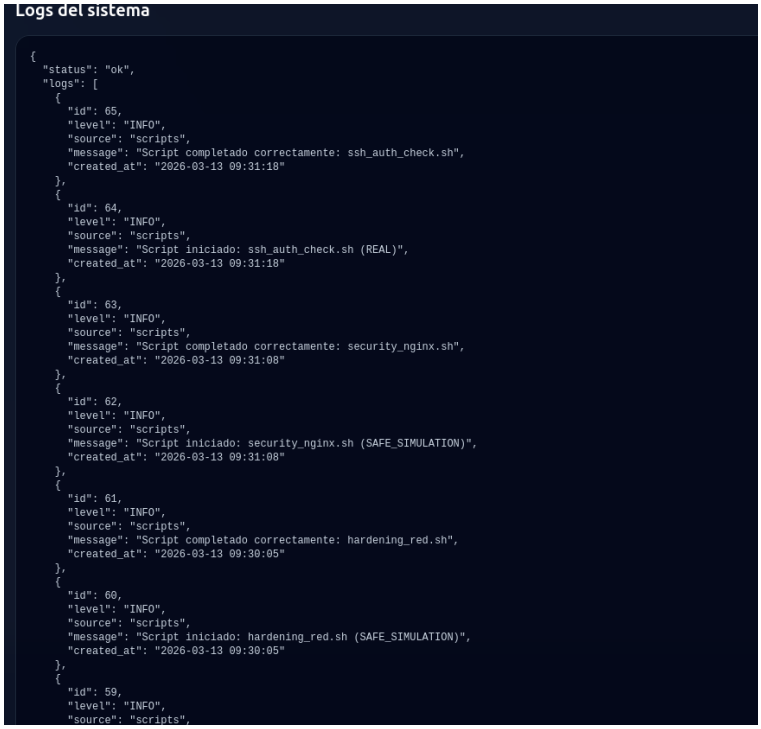


Figura 4.15:
Panel de eventos, logs y alertas del sistema
(Fuente: Elaboración propia)

4.5. Validación de integración

Más allá de la validación funcional por módulos, considero indispensable validar la integración del framework como sistema completo. Esta dimensión busca comprobar que los módulos no operan de forma aislada, sino que intercambian información de manera consistente dentro de una misma arquitectura.

La integración se considera satisfactoria cuando el backend recibe resultados de herramientas externas, los normaliza, los almacena en la base de datos, activa la lógica analítica correspondiente y los presenta en la interfaz de forma coherente. Asimismo, verifico que acciones como la ejecución de scripts, la actualización del checklist, la generación de alertas o el entrenamiento del modelo de priorización no quedan desconectadas, sino que participan dentro del mismo ciclo de observación y decisión.

Desde esta perspectiva, la validación de integración es una de las pruebas más importantes de la tesis, porque demuestra que el valor del framework no está en módulos individuales, sino en la forma en que estos cooperan para sostener una lectura unificada del riesgo.

4.6. Validación del sistema en funcionamiento

La validación del sistema en funcionamiento se orienta a comprobar que el framework puede operar de extremo a extremo sobre un flujo real de trabajo. En esta etapa verifico que la plataforma permite iniciar escaneos, descubrir recursos expuestos, consolidar hallazgos, calcular prioridad contextual, registrar eventos, generar alertas y presentar los resultados en una interfaz unificada.

Este punto resulta central, porque demuestra que el framework deja de ser una agregación conceptual de módulos y pasa a comportarse como una arquitectura operativa. La interfaz principal cumple aquí una función decisiva: muestra conteos de hallazgos, distribuciones de severidad, detalle por herramienta, estados del sistema y rankings de vulnerabilidades. Además, permite comparar la severidad técnica con la prioridad contextual generada por el modelo, lo que transforma el dashboard en una herramienta de análisis y no solo de visualización pasiva.

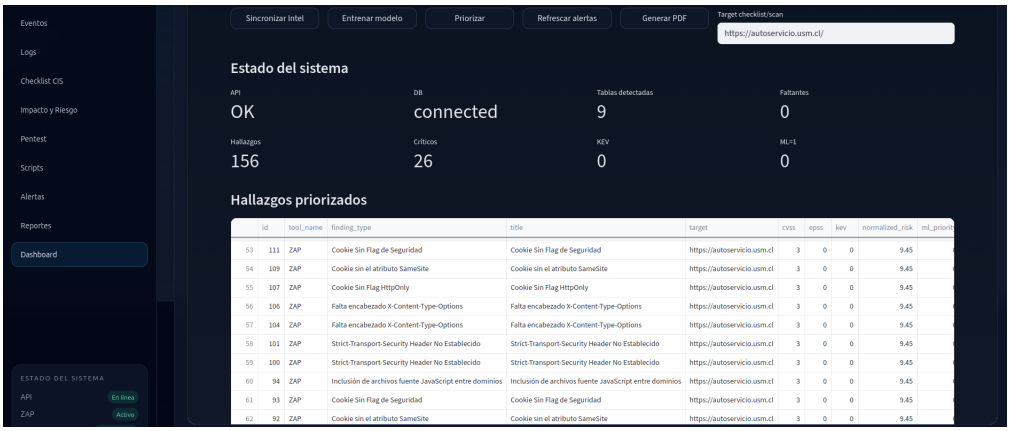


Figura 4.16:
Dashboard principal del framework en funcionamiento
(Fuente: Elaboración propia)

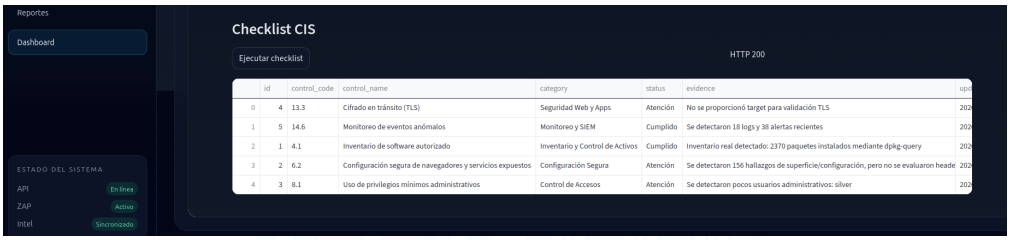


Figura 4.17:
Vista del módulo de checklist y estado de controles
(Fuente: Elaboración propia)

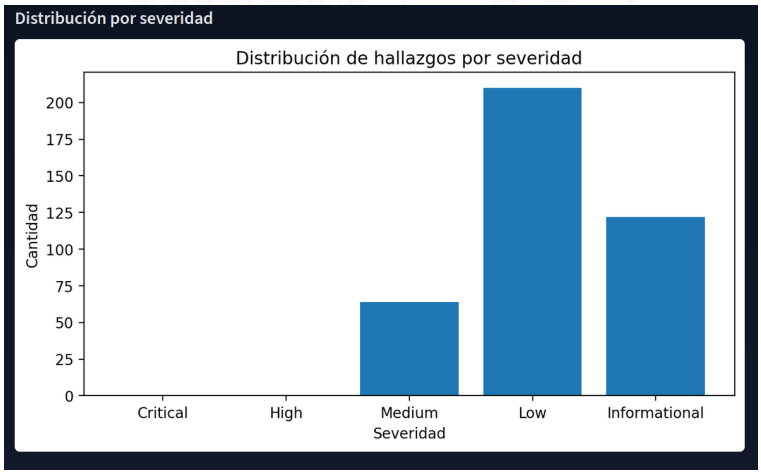


Figura 4.18:
Grafico de logs y alertas del sistema
(Fuente: Elaboración propia)

4.7. Análisis de la validez de la solución

Con base en las dimensiones anteriores, sostengo que la solución propuesta es válida para el entorno en el que fue planteada, en la medida en que cumple con las funciones centrales que definí en el problema de investigación. El framework demuestra capacidad para recolectar evidencia desde múltiples fuentes, normalizar hallazgos heterogéneos, interpretarlos dentro de una lógica de riesgo, priorizarlos de forma contextual y presentarlos mediante una arquitectura integrada de apoyo a la decisión.

La validez de la propuesta no depende de que cada componente alcance una madurez industrial completa, sino de que el sistema, como conjunto, responda de manera coherente al problema planteado. En ese sentido, la validación confirma que el framework no se limita a ejecutar pruebas, sino que articula de forma efectiva descubrimiento, evaluación, normalización, priorización, trazabilidad y mitigación dentro de un solo sistema. Esto constituye precisamente el aporte central de la tesis.

Al mismo tiempo, reconozco que la validez del framework debe entenderse dentro del alcance del prototipo. La solución fue diseñada y validada en un entorno controlado de tesis, por lo que su extrapolación a escenarios productivos requeriría fortalecer ciertos aspectos, como una mayor automatización del monitoreo externo, una ampliación del número de controles evaluados, una integración más profunda con infraestructura de producción y una mayor cobertura de validación ofensiva. Sin embargo, estas limitaciones no invalidan la propuesta; simplemente delimitan el alcance de la validación desarrollada.

4.8. Arquitectura del Sistema

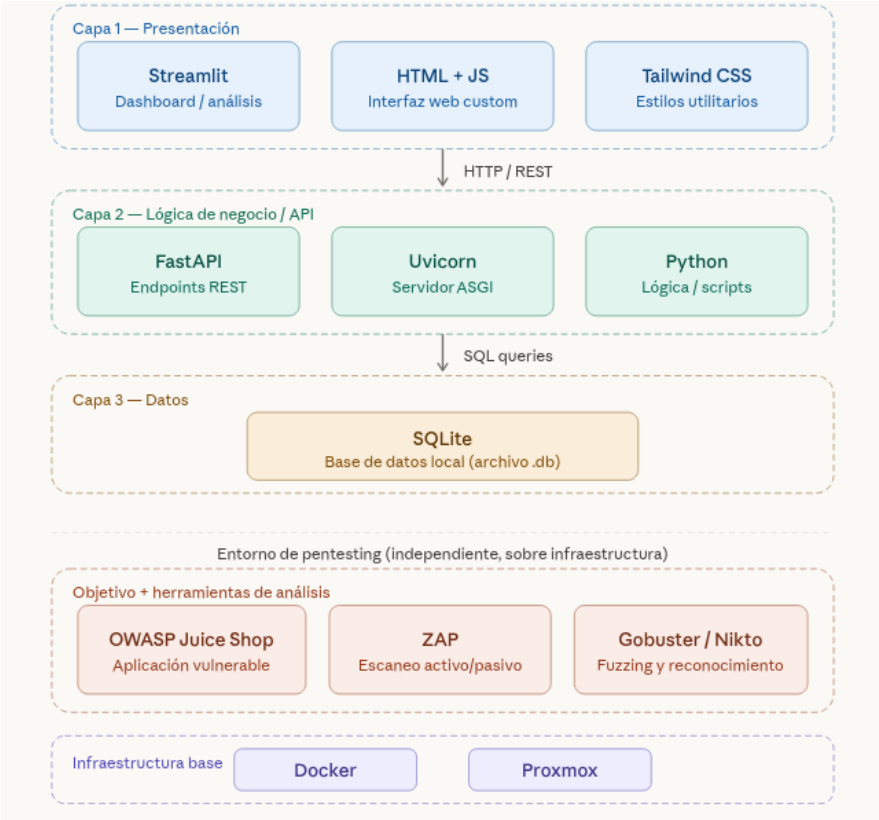


Figura 4.19:
Arquitectura del Sistema
(Fuente: Elaboración propia)

4.9. Stack Tecnológico

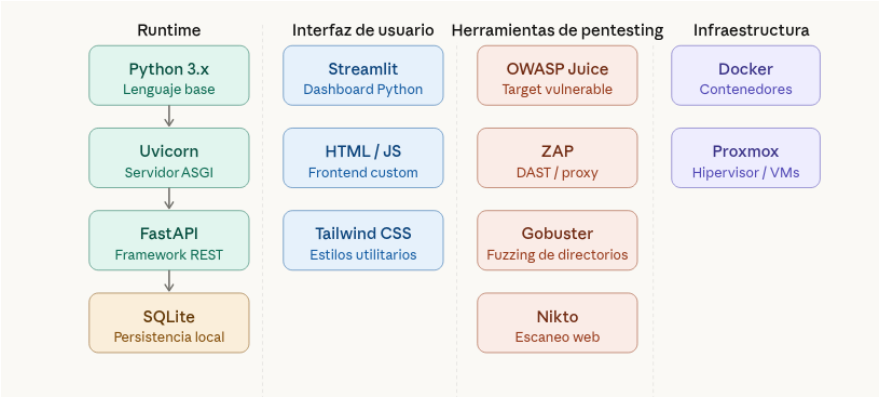


Figura 4.20:
Stack tecnologico del Sistema
(Fuente: Elaboración propia)

4.10. Casos de uso

4.10.1. Caso de Uso 1: Autenticación en la plataforma

Nombre: Autenticación en la plataforma

Actores: Administrador, Usuario normal, Sistema

Descripción: Este caso de uso permite que un usuario autorizado acceda al framework mediante credenciales válidas, de modo que pueda utilizar los módulos disponibles según su perfil.

Tabla 4.1: Caso de uso 1. Autenticación en la plataforma

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El actor accede a la plataforma.	2. El sistema muestra la pantalla de inicio de sesión.
3. El actor ingresa sus credenciales.	4. El sistema valida usuario y contraseña.
5. El actor confirma el inicio de sesión.	6. El sistema habilita el acceso según el rol asignado.
7. El actor ingresa datos incorrectos.	8. El sistema rechaza el acceso y registra el intento.
9. El actor cierra sesión.	10. El sistema finaliza la sesión activa y redirige al inicio.

4.10.2. Caso de Uso 2: Registrar objetivo de evaluación

Nombre: Registrar objetivo de evaluación

Actores: Administrador, Sistema

Descripción: Este caso de uso permite definir el recurso objetivo que será sometido a evaluación, como una URL, host o servicio.

Tabla 4.2: Caso de uso 2. Registrar objetivo de evaluación

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador accede al módulo de evaluación.	2. El sistema habilita el formulario de ingreso del objetivo.
3. El administrador ingresa la URL o recurso objetivo.	4. El sistema valida formato y disponibilidad básica.
5. El administrador confirma el registro.	6. El sistema almacena el objetivo para el proceso de evaluación.
7. El administrador ingresa un objetivo inválido.	8. El sistema muestra un mensaje de error.
9. El administrador modifica el objetivo.	10. El sistema actualiza el objetivo asociado a la sesión.

4.10.3. Caso de Uso 3: Ejecutar escaneo integrado de seguridad

Nombre: Ejecutar escaneo integrado de seguridad

Actores: Administrador, Sistema

Descripción: Este caso de uso permite iniciar el análisis completo del objetivo utilizando los módulos integrados de descubrimiento, escaneo y revisión de configuraciones.

Tabla 4.3: Caso de uso 3. Ejecutar escaneo integrado de seguridad

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona la opción de escaneo integrado.	2. El sistema inicia la secuencia de evaluación.
3. El administrador confirma el proceso.	4. El sistema activa los módulos correspondientes.
5. El sistema ejecuta análisis sobre el objetivo.	6. El sistema registra el avance y los resultados obtenidos.
7. El administrador solicita detener el proceso.	8. El sistema registra la solicitud de detención.
9. El escaneo finaliza correctamente.	10. El sistema almacena los hallazgos y actualiza el estado general.

4.10.4. Caso de Uso 4: Descubrir superficie expuesta

Nombre: Descubrir superficie expuesta

Actores: Administrador, Sistema

Descripción: Este caso de uso permite identificar rutas, directorios, endpoints o recursos visibles desde el objetivo evaluado.

Tabla 4.4: Caso de uso 4. Descubrir superficie expuesta

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador inicia el reconocimiento del objetivo.	2. El sistema activa el módulo de descubrimiento.
3. El sistema enumera recursos accesibles.	4. El sistema registra rutas y directorios encontrados.
5. El administrador consulta la superficie descubierta.	6. El sistema muestra los recursos identificados.
7. No se detectan recursos adicionales.	8. El sistema registra el resultado sin hallazgos relevantes.
9. El reconocimiento falla por indisponibilidad del objetivo.	10. El sistema informa la incidencia y registra el evento.

4.10.5. Caso de Uso 5: Detectar vulnerabilidades web

Nombre: Detectar vulnerabilidades web

Actores: Administrador, Sistema

Descripción: Este caso de uso permite identificar vulnerabilidades, errores de configuración y patrones inseguros sobre servicios y aplicaciones web.

Tabla 4.5: Caso de uso 5. Detectar vulnerabilidades web

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador activa el módulo de escaneo web.	2. El sistema ejecuta el análisis del objetivo.
3. El sistema detecta debilidades web.	4. El sistema registra vulnerabilidades y evidencias asociadas.
5. El administrador consulta los hallazgos.	6. El sistema presenta la información obtenida.
7. El objetivo no responde al análisis.	8. El sistema informa el error y registra el evento.
9. El escaneo no detecta vulnerabilidades.	10. El sistema registra el resultado como evaluación sin hallazgos críticos.

4.10.6. Caso de Uso 6: Revisar configuraciones inseguras y hardening

Nombre: Revisar configuraciones inseguras y hardening

Actores: Administrador, Sistema

Descripción: Este caso de uso permite identificar condiciones de hardening insuficiente, cabeceras faltantes, exposición de archivos o configuraciones débiles.

Tabla 4.6: Caso de uso 6. Revisar configuraciones inseguras y hardening

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador solicita la revisión de configuración del servicio.	2. El sistema ejecuta el módulo correspondiente.
3. El sistema analiza cabeceras, archivos expuestos y configuraciones.	4. El sistema registra observaciones y evidencias.
5. El administrador revisa el resultado.	6. El sistema muestra hallazgos de configuración y postura de seguridad.
7. No se detectan debilidades de configuración.	8. El sistema registra el resultado sin observaciones.
9. La revisión no puede completarse.	10. El sistema notifica la limitación y conserva trazabilidad.

4.10.7. Caso de Uso 7: Normalizar hallazgos heterogéneos

Nombre: Normalizar hallazgos heterogéneos

Actores: Sistema

Descripción: Este caso de uso permite transformar resultados provenientes de distintas herramientas a una estructura común de análisis.

Tabla 4.7: Caso de uso 7. Normalizar hallazgos heterogéneos

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El sistema recibe resultados desde distintas herramientas.	2. El sistema identifica atributos comparables.
3. El sistema traduce los hallazgos a una estructura uniforme.	4. El sistema asigna campos comunes de análisis.
5. El sistema consolida la evidencia normalizada.	6. El sistema almacena los registros en la base de datos.
7. Un hallazgo carece de información suficiente.	8. El sistema completa con valores controlados o marca datos faltantes.
9. El proceso finaliza.	10. El sistema deja disponible la base analítica para evaluación posterior.

4.10.8. Caso de Uso 8: Evaluar impacto y riesgo

Nombre: Evaluar impacto y riesgo

Actores: Sistema, Administrador

Descripción: Este caso de uso permite interpretar los hallazgos normalizados mediante una lógica de severidad, impacto y probabilidad.

Tabla 4.8: Caso de uso 8. Evaluar impacto y riesgo

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador solicita la evaluación de riesgo.	2. El sistema recupera hallazgos normalizados.
3. El sistema calcula severidad e impacto.	4. El sistema clasifica los hallazgos según riesgo.
5. El administrador consulta la evaluación generada.	6. El sistema muestra categorías de riesgo comprensibles.
7. Un hallazgo carece de contexto suficiente.	8. El sistema lo marca como evaluación parcial.
9. El proceso termina.	10. El sistema actualiza los registros de análisis.

4.10.9. Caso de Uso 9: Ejecutar priorización contextual de vulnerabilidades

Nombre: Ejecutar priorización contextual de vulnerabilidades

Actores: Administrador, Sistema

Descripción: Este caso de uso permite ordenar hallazgos según criticidad contextual, utilizando variables técnicas y operativas adicionales a la severidad técnica.

Tabla 4.9: Caso de uso 9. Ejecutar priorización contextual de vulnerabilidades

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador activa la opción de priorización.	2. El sistema carga hallazgos consolidados.
3. El sistema incorpora variables de contexto.	4. El sistema calcula el score final de prioridad.
5. El administrador revisa el ranking resultante.	6. El sistema muestra el orden de tratamiento sugerido.
7. El modelo no está disponible.	8. El sistema informa el error y registra la incidencia.
9. La priorización finaliza correctamente.	10. El sistema actualiza el ranking y lo deja disponible en la interfaz.

4.10.10. Caso de Uso 10: Entrenar el modelo de aprendizaje automático

Nombre: Entrenar el modelo de aprendizaje automático

Actores: Administrador, Sistema

Descripción: Este caso de uso permite entrenar el modelo de priorización usando datos del framework y señales contextuales disponibles.

Tabla 4.10: Caso de uso 10. Entrenar el modelo de aprendizaje automático

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador solicita el entrenamiento del modelo.	2. El sistema recupera datos desde la base analítica.
3. El sistema construye el conjunto de entrenamiento.	4. El sistema procesa variables técnicas y contextuales.
5. El sistema ejecuta el entrenamiento.	6. El sistema guarda el modelo actualizado.
7. No existen datos suficientes.	8. El sistema informa que no es posible entrenar correctamente.
9. El entrenamiento finaliza.	10. El sistema deja el modelo listo para priorización.

4.10.11. Caso de Uso 11: Ejecutar checklist de controles de seguridad

Nombre: Ejecutar checklist de controles de seguridad

Actores: Administrador, Sistema

Descripción: Este caso de uso permite verificar controles relevantes del entorno a partir de evidencia real y actualizar su estado dentro del framework.

Tabla 4.11: Caso de uso 11. Ejecutar checklist de controles de seguridad

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador activa el módulo de checklist.	2. El sistema ejecuta verificaciones sobre controles definidos.
3. El sistema revisa evidencias del entorno.	4. El sistema asigna estado a cada control.
5. El administrador consulta el resultado.	6. El sistema muestra cumplimiento, atención o no conformidad.
7. Un control no puede ser evaluado automáticamente.	8. El sistema lo marca como revisión parcial o pendiente.
9. El checklist finaliza.	10. El sistema actualiza los controles y conserva la evidencia.

4.10.12. Caso de Uso 12: Registrar eventos y logs del sistema

Nombre: Registrar eventos y logs del sistema

Actores: Sistema

Descripción: Este caso de uso permite conservar trazabilidad operativa sobre accesos, ejecuciones, errores, cambios y procesos relevantes del framework.

Tabla 4.12: Caso de uso 12. Registrar eventos y logs del sistema

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El sistema detecta una acción relevante.	2. El sistema genera un evento o log correspondiente.
3. El sistema registra inicio de procesos.	4. El sistema almacena el detalle en auditoría.
5. El sistema registra errores o fallos.	6. El sistema conserva evidencia del incidente.
7. El sistema registra finalización de tareas.	8. El sistema actualiza la trazabilidad operativa.
9. El usuario consulta registros históricos.	10. El sistema muestra eventos y logs disponibles.

4.10.13. Caso de Uso 13: Generar alertas a partir de evidencia operativa

Nombre: Generar alertas a partir de evidencia operativa

Actores: Sistema, Administrador

Descripción: Este caso de uso permite convertir logs, hallazgos o eventos priorizados en alertas relevantes para el análisis de seguridad.

Tabla 4.13: Caso de uso 13. Generar alertas a partir de evidencia operativa

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El sistema revisa logs y hallazgos recientes.	2. El sistema detecta condiciones de alerta.
3. El sistema clasifica la severidad de la señal.	4. El sistema registra la alerta generada.
5. El administrador consulta el módulo de alertas.	6. El sistema presenta el detalle disponible.
7. No se detectan señales relevantes.	8. El sistema finaliza sin nuevas alertas.
9. Una fuente de datos no está disponible.	10. El sistema ejecuta el análisis sobre las fuentes restantes.

4.10.14. Caso de Uso 14: Ejecutar scripts de mitigación y hardening

Nombre: Ejecutar scripts de mitigación y hardening

Actores: Administrador, Sistema

Descripción: Este caso de uso permite aplicar o verificar medidas técnicas de remediación desde el propio framework.

Tabla 4.14: Caso de uso 14. Ejecutar scripts de mitigación y hardening

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona un script autorizado.	2. El sistema valida que el script esté permitido.
3. El administrador confirma la ejecución.	4. El sistema ejecuta la acción correspondiente.
5. El sistema procesa la salida del script.	6. El sistema registra estado, resultado y trazabilidad.
7. El script falla en su ejecución.	8. El sistema conserva la salida y registra el error.
9. El administrador revisa el resultado.	10. El sistema presenta la evidencia de ejecución.

4.10.15. Caso de Uso 15: Validar hallazgo mediante prueba de penetración controlada

Nombre: Validar hallazgo mediante prueba de penetración controlada

Actores: Administrador, Sistema

Descripción: Este caso de uso permite confirmar la explotabilidad de un hallazgo específico dentro de un entorno controlado de laboratorio.

Tabla 4.15: Caso de uso 15. Validar hallazgo mediante prueba de penetración controlada

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona un hallazgo para validación.	2. El sistema carga la información asociada.
3. El administrador inicia la validación controlada.	4. El sistema registra la prueba.
5. El administrador documenta el resultado obtenido.	6. El sistema actualiza la evidencia del hallazgo.
7. La validación no puede ejecutarse.	8. El sistema registra el intento fallido.
9. El proceso finaliza.	10. El sistema deja trazabilidad de la validación realizada.

4.10.16. Caso de Uso 16: Visualizar dashboard unificado del framework

Nombre: Visualizar dashboard unificado del framework

Actores: Administrador, Usuario normal, Sistema

Descripción: Este caso de uso permite consultar de manera integrada el estado del sistema, los hallazgos, las alertas, los controles y la priorización generada.

Tabla 4.16: Caso de uso 16. Visualizar dashboard unificado del framework

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El actor accede al dashboard.	2. El sistema carga métricas y módulos disponibles.
3. El actor revisa hallazgos, alertas y prioridades.	4. El sistema presenta la información consolidada.
5. El actor navega entre módulos.	6. El sistema actualiza la vista según la selección.
7. No existen datos cargados.	8. El sistema informa que no hay resultados disponibles.
9. Un módulo no responde.	10. El sistema mantiene visibles los módulos restantes y registra la incidencia.

4.10.17. Caso de Uso 17: Generar reporte de seguridad

Nombre: Generar reporte de seguridad

Actores: Administrador, Sistema

Descripción: Este caso de uso permite consolidar en un reporte la información relevante del framework para documentación, revisión o toma de decisiones.

Tabla 4.17: Caso de uso 17. Generar reporte de seguridad

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El administrador solicita la generación del reporte.	2. El sistema recopila hallazgos, alertas, controles y prioridades.
3. El sistema consolida la información.	4. El sistema construye una salida estructurada.
5. El administrador consulta el reporte generado.	6. El sistema presenta el resultado disponible.
7. No existen datos suficientes.	8. El sistema informa que el reporte será parcial.
9. El proceso finaliza.	10. El sistema registra el reporte en la plataforma.

4.10.18. Caso de Uso 18: Revisar estado histórico de evaluación

Nombre: Revisar estado histórico de evaluación

Actores: Administrador, Usuario normal, Sistema

Descripción: Este caso de uso permite consultar el historial de eventos, logs, scripts, alertas y resultados previos del framework.

Tabla 4.18: Caso de uso 18. Revisar estado histórico de evaluación

Fuente: Elaboración propia.

Acción del Actor	Respuesta del Sistema
1. El actor accede al módulo histórico.	2. El sistema recupera registros persistidos.
3. El actor consulta eventos, logs y resultados previos.	4. El sistema muestra la trazabilidad almacenada.
5. El actor compara estados o revisa ejecuciones anteriores.	6. El sistema presenta los datos ordenados.
7. No existe historial disponible.	8. El sistema informa que no hay registros previos.
9. El proceso concluye.	10. El sistema mantiene la información disponible para análisis comparativo.

5 | CONCLUSIONES

La transformación digital del transporte público ha incrementado la dependencia de plataformas, servicios, aplicaciones y componentes conectados que sostienen parte relevante de su operación. Este proceso ha mejorado capacidades de gestión, trazabilidad e interacción, pero también ha ampliado la superficie de exposición frente a amenazas cibernéticas. A partir de este escenario, el presente proyecto de grado se orientó a abordar un problema específico: la dificultad de evaluar, integrar y priorizar de manera útil la evidencia de ciberseguridad generada en el entorno digital asociado al transporte público, especialmente cuando dicha evidencia proviene de múltiples fuentes y presenta formatos, taxonomías y niveles de severidad heterogéneos.

Uno de los principales resultados del trabajo consiste en demostrar que es viable diseñar e implementar un framework integral de ciberseguridad capaz de articular, dentro de una misma arquitectura, funciones de descubrimiento de superficie, escaneo de vulnerabilidades, revisión de configuraciones, consolidación de hallazgos, evaluación de impacto, monitoreo, mitigación y priorización contextual. Este resultado es relevante porque responde directamente a una limitación frecuente en la práctica: la fragmentación de la seguridad entre herramientas que generan resultados válidos de forma individual, pero difíciles de relacionar entre sí dentro de un proceso coherente de análisis y remediación.

El desarrollo del framework permitió confirmar que la detección de vulnerabilidades, aunque necesaria, no es suficiente para sostener una gestión eficaz del riesgo. Las herramientas de seguridad producen evidencia técnica valiosa, pero esa evidencia pierde gran parte de su utilidad cuando permanece dispersa, heterogénea o desvinculada del contexto operativo del activo evaluado. En este sentido, uno de los aportes centrales del proyecto de grado fue la incorporación de una capa de normalización de hallazgos capaz de traducir resultados provenientes de distintas fuentes a una estructura común de análisis. Esta decisión permitió consolidar, comparar y correlacionar evidencia técnica dentro de una misma base analítica, transformando al framework en algo más que una suma de utilidades independientes.

La investigación también permitió confirmar que la severidad técnica no siempre coincide con la prioridad operativa. A lo largo del diseño, implementación y validación del framework se observó que dos hallazgos con características técnicas similares pueden tener implicancias distintas según el activo afectado, su nivel de exposición, la necesidad de autenticación, la existencia de alertas asociadas, el estado de los controles y la función que dicho activo cumple dentro del sistema. Esta constatación justifica la incorporación de una capa de priorización contextual basada en aprendizaje automático. Su aporte no consiste en reemplazar el juicio experto, sino en enriquecerlo mediante un criterio adicional de ordenamiento, más cercano al riesgo operativo que el proporcionado por la severidad aislada. Desde esta perspectiva, uno de los aportes más significativos del proyecto de grado fue demostrar que el framework puede apoyar decisiones de remediación de manera más razonada, estructurada y trazable.

Asimismo, se verificó que la integración de herramientas complementarias dentro de una sola arquitectura fortalece la calidad del análisis. La incorporación de módulos como OWASP ZAP, Gobuster y Nikto permitió ampliar la cobertura del framework sobre distintas dimensiones del riesgo, incluyendo vulnerabilidades web, descubrimiento de superficie y configuraciones inseguras. A ello se suman componentes de checklist, auditoría, monitoreo, scripts de hardening y validación controlada, que en conjunto enriquecen la comprensión del entorno evaluado. Este resultado reafirma una idea central del trabajo: la seguridad adquiere mayor valor cuando distintas fuentes de evidencia dejan de operar por separado y pasan a contribuir a una lectura integrada del sistema.

Otro aspecto relevante del trabajo fue la verificación de coherencia entre la solución desarrollada y marcos normativos y metodológicos ampliamente reconocidos. La arquitectura propuesta no fue construida como un ejercicio aislado de desarrollo de software, sino como una traducción aplicada de principios provenientes de NIST Cybersecurity

Framework, ISO/IEC 27001, ISO/IEC 27005 y CIS Controls. Esta articulación entrega consistencia conceptual al framework y permite relacionar los hallazgos técnicos con criterios más amplios de gestión, cumplimiento y tratamiento del riesgo. En consecuencia, la propuesta no queda reducida a una implementación puramente instrumental, sino que se sitúa dentro de una lógica metodológica reconocible y defendible.

En el plano técnico, se verificó que fue posible materializar esta propuesta mediante un stack ligero, pero suficientemente expresivo para sostener la lógica del framework. El uso de Python, FastAPI y Uvicorn permitió concentrar la orquestación, la integración de herramientas, el procesamiento de resultados y la ejecución del módulo analítico dentro de un backend coherente. Del mismo modo, SQLite ofreció una base suficiente para persistencia y análisis en el contexto del prototipo, mientras que Streamlit y la interfaz unificada basada en tecnologías web permitieron representar la información de forma clara y útil para la evaluación. En conjunto, esta arquitectura demostró ser adecuada para el alcance del proyecto de grado, al priorizar el valor analítico y metodológico del sistema sin introducir complejidad innecesaria.

En cuanto a la validación, el framework demostró ser técnicamente viable y metodológicamente consistente para el entorno en el que fue planteado. La solución fue capaz de ejecutar pruebas, recolectar evidencia desde múltiples fuentes, consolidar resultados, registrar eventos, generar alertas, representar estados de control y priorizar hallazgos dentro de una misma arquitectura operativa. Esto permite sostener que la propuesta no quedó en el plano teórico, sino que efectivamente produjo información útil para el análisis y tratamiento del riesgo dentro de un entorno de pruebas controlado y replicable.

Al mismo tiempo, resulta necesario reconocer con claridad los límites del trabajo. Este proyecto de grado no busca demostrar el despliegue de una plataforma industrial completa ni sustituir soluciones comerciales consolidadas. Su alcance corresponde al de un prototipo funcional, metodológicamente fundamentado y técnicamente defendible. Del mismo modo, el módulo de aprendizaje automático debe entenderse como una capa de apoyo a la priorización y no como un sistema autónomo de seguridad. Esta delimitación no debilita la propuesta, sino que fortalece su rigor, porque permite situar con precisión el aporte real del trabajo y evita atribuirle un alcance que no le corresponde.

En síntesis, se concluye que este proyecto de grado demuestra la viabilidad de diseñar e implementar un framework integral de ciberseguridad para el entorno digital asociado al transporte público, capaz de integrar herramientas diversas, consolidar evidencia técnica y avanzar hacia una priorización contextual de vulnerabilidades. Su principal contribución consiste en articular, dentro de una misma arquitectura, la dimensión técnica, la dimensión metodológica y la dimensión operativa de la seguridad. De este modo, el trabajo aporta una solución funcional y, al mismo tiempo, una forma más madura de comprender la evaluación de seguridad: no como una lista de hallazgos aislados, sino como un proceso de apoyo a la decisión fundamentado en evidencia organizada y contextualizada.

Bibliografía

[1] European Union Agency for Cybersecurity. (2023). *ENISA transport threat landscape*. https://www.enisa.europa.eu/publications/enisa-transport-threat-landscape

[2] European Union Agency for Cybersecurity. (2022). *ENISA threat landscape 2022*. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022

[3] Verizon. (2023). *2023 data breach investigations report (DBIR)*. https://www.verizon.com/business/resources/reports/2023-data-breach-investigations-report-dbir.pdf

[4] KPMG. (2022). *Cyber trust insights 2022*. https://assets.kpmg.com/content/dam/kpmg/xx/pdf/2022/10/kpmg-cyber-trust-insights-2022.pdf

[5] Asociación de Municipalidades de Chile. (2024). *El 71 % de los municipios en Chile no ha capacitado a su personal en ciberseguridad en los últimos 18 meses*. https://amuch.cl/el-71-de-los-municipios-en-chile-no-ha-capacitado-a-su-personal-en-ciberseguridad-en-los-ultimos-1

[6] Biblioteca del Congreso Nacional de Chile. (2024). *Ley N° 21.663: Ley marco sobre ciberseguridad e infraestructura crítica de la información*. https://www.bcn.cl/leychile/navegar?idNorma=1202434

[7] Ministerio del Interior y Seguridad Pública. (2025, 25 de marzo). *Cámaras de buses RED se integran al sistema SITIA con inteligencia artificial*. https://www.interior.gob.cl/noticias/2025/03/25/camaras-de-buses-red-sitia

[8] Red Movilidad. (2025). *Young people from Chilean universities innovated to continue improving safety on Red Movilidad buses*. https://www.red.cl/en/red-communicates/jovenes-de-universidades-chilenas-innovaron-para-seguir-mejorando-la-seguridad-en-buses-red-movili

[9] U.S. Department of Transportation. (n.d.). *Cybersecurity*. https://www.transportation.gov/priorities/cybersecurity

[10] European Union. (2018). *Regulation (EU) 2016/679 (General Data Protection Regulation)*. https://gdpr-info.eu/

[11] European Union. (2022). *Directive (EU) 2022/2555 on measures for a high common level of cybersecurity across the Union (NIS2 Directive)*. https://eur-lex.europa.eu/eli/dir/2022/2555

[12] International Organization for Standardization. (2013). *ISO/IEC 27001:2013 information technology—Security techniques—Information security management systems—Requirements*. https://www.iso.org/standard/54534.html

[13] International Organization for Standardization. (2022). *ISO/IEC 27001:2022 information security, cybersecurity and privacy protection—Information security management systems—Requirements*. https://www.iso.org/standard/27001

[14] International Organization for Standardization. (2022). *ISO/IEC 27002:2022 information security, cybersecurity and privacy protection—Information security controls*. https://www.iso.org/standard/75652.html

[15] International Organization for Standardization. (2018). *ISO/IEC 27005:2018 information security risk management*. https://www.iso.org/standard/75281.html

[16] International Organization for Standardization. (2019). *ISO 22301:2019 security and resilience—Business continuity management systems—Requirements*. https://www.iso.org/standard/75106.html

[17] International Organization for Standardization. (2018). *ISO 31000:2018 risk management—Guidelines*. https://www.iso.org/standard/65694.html

[18] National Institute of Standards and Technology. (2024). *The NIST cybersecurity framework (CSF) 2.0* (NIST CSWP 29). https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf

[19] National Institute of Standards and Technology. (2012). *Guide for conducting risk assessments* (NIST Special Publication 800-30 Rev. 1). https://csrc.nist.gov/pubs/sp/800/30/r1/final

[20] Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008). *Technical guide to information security testing and assessment* (NIST Special Publication 800-115). National Institute of Standards and Technology. https://csrc.nist.gov/pubs/sp/800/115/final

[21] National Institute of Standards and Technology. (2025). *Incident response recommendations and considerations for cybersecurity risk management* (NIST Special Publication 800-61 Rev. 3). https://csrc.nist.gov/pubs/sp/800/61/r3/final

[22] National Institute of Standards and Technology. (2020). *Security and privacy controls for information systems and organizations* (NIST Special Publication 800-53 Rev. 5). https://csrc.nist.gov/pubs/sp/800/53/r5/final

[23] National Institute of Standards and Technology. (2022). *Guide to enterprise patch management planning* (NIST Special Publication 800-40 Rev. 4). https://csrc.nist.gov/pubs/sp/800/40/r4/final

[24] National Institute of Standards and Technology. (2022). *Secure software development framework (SSDF) version 1.1* (NIST Special Publication 800-218). https://csrc.nist.gov/pubs/sp/800/218/final

[25] Center for Internet Security. (2021). *CIS critical security controls v8*. https://www.cisecurity.org/controls/v8

[26] OWASP Foundation. (2021). *OWASP top 10: The ten most critical web application security risks*. https://owasp.org/Top10/2021/

[27] OWASP Foundation. (2023). *OWASP API security top 10: 2023*. https://owasp.org/API-Security/editions/2023/en/0x00-header/

[28] OWASP Foundation. (n.d.). *OWASP application security verification standard (ASVS)*. https://owasp.org/www-project-application-security-verification-standard/

[29] OWASP Foundation. (n.d.). *OWASP web security testing guide*. https://owasp.org/www-project-web-security-testing-guide/

[30] OWASP Foundation. (n.d.). *Threat modeling*. https://owasp.org/www-community/Threat_Modeling

[31] OWASP Foundation. (n.d.). *OWASP SAMM (Software assurance maturity model)*. https://owasp.org/www-project-samm/

[32] OWASP Foundation. (n.d.). *OWASP DevSecOps guideline*. https://owasp.org/www-project-devsecops-guideline/

[33] ZAP Project. (n.d.). *ZAP by Checkmarx*. https://www.zaproxy.org/

[34] ZAP Project. (n.d.). *ZAP API reference*. https://www.zaproxy.org/docs/api/

[35] PortSwigger. (n.d.). *Burp Suite*. https://portswigger.net/burp

[36] Tenable. (n.d.). *Nessus vulnerability scanner*. https://www.tenable.com/products/nessus

[37] Rapid7. (n.d.). *Metasploit framework*. https://docs.rapid7.com/metasploit/msf-overview/

[38] Reeves, O. (n.d.). *Gobuster*. GitHub. https://github.com/OJ/gobuster

[39] Sullo, C. (n.d.). *Nikto web server scanner*. https://cirt.net/nikto/

[40] Lyon, G. F. (n.d.). *Nmap network scanning*. https://nmap.org/book/

[41] Wazuh Project. (n.d.). *Wazuh documentation*. https://documentation.wazuh.com/current/index.html

[42] Cisco. (n.d.). *Snort*. https://www.snort.org/

[43] Open Information Security Foundation. (n.d.). *Suricata documentation*. https://docs.suricata.io/

[44] The MITRE Corporation. (2025). *MITRE ATT&CK*. https://attack.mitre.org/

[45] Forum of Incident Response and Security Teams. (n.d.). *Exploit prediction scoring system (EPSS)*. https://www.first.org/epss/

[46] Forum of Incident Response and Security Teams. (n.d.). *The EPSS model*. https://www.first.org/epss/model

[47] Cybersecurity and Infrastructure Security Agency. (n.d.). *Known exploited vulnerabilities catalog*. https://www.cisa.gov/known-exploited-vulnerabilities-catalog

[48] Forum of Incident Response and Security Teams. (2019). *Common vulnerability scoring system version 3.1: Specification document*. https://www.first.org/cvss/v3.1/specification-document

[49] Forum of Incident Response and Security Teams. (2019). *Common vulnerability scoring system version 3.1: User guide*. https://www.first.org/cvss/v3.1/user-guide

[50] Python Software Foundation. (n.d.). *Python 3 documentation*. https://docs.python.org/3/

[51] Ramírez, S. (n.d.). *FastAPI documentation*. https://fastapi.tiangolo.com/

[52] Encode OSS Ltd. (n.d.). *Uvicorn documentation*. https://www.uvicorn.org/

[53] SQLite Consortium. (n.d.). *SQLite documentation*. https://sqlite.org/docs.html

[54] Snowflake Inc. (n.d.). *Streamlit documentation*. https://docs.streamlit.io/

[55] Docker, Inc. (n.d.). *Docker documentation*. https://docs.docker.com/

[56] Proxmox Server Solutions GmbH. (2025). *Proxmox VE administration guide for 8.x*. https://www.proxmox.com/en/downloads/proxmox-virtual-environment/documentation/proxmox-ve-admin-guide-for-8-x

[57] Tailwind Labs. (n.d.). *Tailwind CSS documentation*. https://tailwindcss.com/docs

[58] Nuxt Labs. (n.d.). *Nuxt documentation*. https://nuxt.com/docs

[59] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://link.springer.com/article/10.1023/A:1010933404324

[60] Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). Wiley.

[61] Shostack, A. (2014). *Threat modeling: Designing for security*. Wiley.

[62] Anderson, R. (2020). *Security engineering: A guide to building dependable distributed systems* (3rd ed.). Wiley. https://www.cl.cam.ac.uk/archive/rja14/book.html

[63] Schneier, B. (1996). *Applied cryptography: Protocols, algorithms, and source code in C* (2nd ed.). Wiley.

[64] scikit-learn developers. (n.d.). *RandomForestClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[65] scikit-learn developers. (n.d.). *LogisticRegression*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html