Impacto del modelo Zero Trust en APIs RESTful para PyMEs

Impact of the Zero Trust Model on RESTful APIs for SMEs

Cristian Taju
https://orcid.org/0009-0006-2789-8174
Cristiantaju05@gmail.com
Universidad de la Ciencia y Tecnología, Panamá

Angelica Pitti
https://orcid.org/0009-0008-2357-9242
ayeleth25@gmail.com
Universidad de la Ciencia y Tecnología, Panamá

José de los Reyes Rivera Castro
https://orcid.org/0000-0002-5527-4637
jriverac@gmail.com
Universidad de la Ciencia y Tecnología, Panamá

DOI: https://doi.org/10.61454/x42v5m50

Resumen

El estudio plantea analizar el impacto de las políticas de seguridad del modelo Zero Trust sobre una API RESTful funcional para PyMEs. Mediante un enfoque empírico, se diseñó una API funcional en Laravel 12 con controles de seguridad tradicionales y posteriormente con políticas Zero Trust (autenticación JWT, RBAC, monitoreo continuo). Las pruebas automatizadas con Postman mostraron una reducción del 85.7% en vulnerabilidades críticas (OWASP ZAP) y un bloqueo del 100% de accesos no autorizados, con una latencia promedio aceptable (<500 ms). Los resultados demuestran que Zero Trust es viable en PyMEs, equilibrando seguridad y operatividad sin requerir infraestructura costosa. El estudio concluye que la implementación del modelo Zero Trust en una arquitectura de servicios RESTful tiene un impacto positivo tanto en la seguridad como en el rendimiento del sistema. La estrategia de Zero Trust es accesible no solo para grandes organizaciones con alta capacidad tecnológica, sino también para medianas empresas, las cuales pueden comenzar protegiendo APIs críticas o integrándose en pipelines de CI/CD. Recomendable para organizaciones que buscan fortalecer su postura de seguridad sin sacrificar operatividad ni flexibilidad tecnológica.

Palabras Clave

Zero Trust, APIs RESTful, seguridad informática, PyMEs, rendimiento.

Abstract

This study analyzes the impact of implementing Zero Trust-based security policies on a functional RESTful API for SMEs. Using an empirical approach, a functional API was designed in Laravel 12 with traditional security controls and subsequently with Zero Trust policies (JWT authentication, RBAC, continuous monitoring). Automated tests with Postman showed an 85.7% reduction in critical vulnerabilities (OWASP ZAP) and 100% blocking of unauthorized access, with acceptable average latency (<500 ms). The results demonstrate that Zero Trust is viable for SMEs, balancing security and

operability without requiring costly infrastructure. The study concludes that implementing the Zero Trust model in a RESTful service architecture has a positive impact on both system security and performance. The Zero Trust strategy is accessible not only to large organizations with high technological capabilities but also to medium-sized businesses, which can begin by protecting critical APIs or integrating them into CI/CD pipelines. Recommended for organizations seeking to strengthen their security posture without sacrificing operational efficiency or technological flexibility.

Keywords
Zero Trust, RESTful APIs, cybersecurity, SMEs, performance

Recepción: 24 de septiembre de 2025
Aceptación: 18 de noviembre de 2025

Introducción

La problemática de la seguridad en servicios web empresariales, específicamente en APIs RESTful, han tenido un crecimiento acelerado en los últimos años debido a su simplicidad y escalabilidad. Sin embargo, esta expansión ha aumentado la superficie de ataque, exponiendo a las organizaciones, especialmente a las pequeñas y medianas empresas (PyMEs), a vulnerabilidades relacionadas con accesos no autorizados, filtraciones y ataques cibernéticos, mientras que los recursos para implementar medidas de protección avanzadas suelen ser limitados.

El modelo Zero Trust surge como alternativa a la confianza implícita en redes internas, basándose en verificación continua de accesos, controles dinámicos y segmentación. Aunque promovido en grandes corporaciones, su viabilidad en PYMEs con APIs RESTful requiere estudio. Este proyecto evalúa empíricamente su efectividad en ese contexto, comparando configuraciones tradicionales con Zero Trust mediante un entorno simulado y pruebas automatizadas con Postman (midiendo latencia, errores y rendimiento bajo carga). El análisis incluye métricas de rendimiento y registros del servidor para determinar si Zero Trust mejora la seguridad sin penalizar la eficiencia.

Las PyMEs del sector tecnológico enfrentan desafíos significativos en la seguridad de sus aplicaciones, especialmente aquellas que dependen de APIs RESTful expuestas al público. Muchas carecen de infraestructura sólida, personal capacitado y recursos suficientes para implementar controles avanzados, lo que las hace vulnerables a ataques como accesos no autorizados, filtraciones de datos y suplantación de identidad. El modelo Zero Trust, propuesto por John Kindervag (2010), sugiere que ningún usuario o dispositivo debe ser automáticamente confiable, incluso dentro de la red corporativa. Aunque adoptado por gigantes como Google y Microsoft, su implementación en PyMEs es compleja debido a la falta de recursos y el temor a afectar el rendimiento de sus servicios.

Estudios recientes (OWASP, 2023; Salt Security, 2023) revelan que muchas APIs RESTful carecen de autenticación sólida, segmentación y monitoreo continuo, lo que facilita ataques. Según el State of API Security Report - Q1 2023, el 94% de las empresas reportaron problemas de seguridad en APIs, y el 17% sufrió brechas relacionadas con ellas; además, el 78% de los ataques provino de actores que parecían legítimos (Salt Security, 2023).

Aunque el uso de APIs RESTful ha crecido significativamente, también ha aumentado su exposición a ataques. Sin embargo, muchas PyMEs no cuentan con equipos especializados, lo que genera una

brecha crítica en seguridad. La protección no solo es técnica, sino también económica, ya que los ataques pueden ocasionar pérdidas financieras y daños a la reputación.

Estudios previos muestran que Zero Trust ha mejorado la seguridad en grandes empresas, pero su aplicación en PyMEs con recursos limitados ha sido poco explorada, especialmente en relación con el rendimiento de APIs RESTful. Este estudio busca llenar ese vacío, contribuyendo tanto a la academia como a la seguridad de las pequeñas y medianas empresas.

La justificación del estudio radica en la necesidad de evaluar la viabilidad de Zero Trust en entornos simples, evaluando su mejora en seguridad frente a su efecto en el rendimiento del sistema. Desde el enfoque académico, aplica conocimientos en arquitectura segura, control de accesos y pruebas de rendimiento; mientras que, en la práctica, ofrece recomendaciones accesibles para equipos con recursos limitados.

Se destaca que los servicios RESTful, ampliamente usados, son vulnerables a ciberataques, por lo que Zero Trust, con su enfoque en autenticación y autorización continuas, resulta clave, incluso en arquitecturas de microservicios y cloud. Además, su escalabilidad lo hace ideal para PyMEs en crecimiento que manejan datos sensibles, evitando riesgos legales y pérdidas financieras. Asimismo, se enfatiza la necesidad de superar la brecha entre teoría y práctica, adaptando Zero Trust sin inversiones excesivas, en línea con estándares como los de NIST (2020), que enfatizan la obsolescencia de la “confianza implícita” en seguridad. Finalmente, el proyecto no solo evalúa la efectividad de Zero Trust en APIs, sino que promueve enfoques unificados para proteger sistemas distribuidos, reforzando la seguridad en un contexto de ciberamenazas crecientes.

El marco teórico del estudio combina conceptos clave, el avance de la transformación digital ha impulsado el uso de aplicaciones web basadas en arquitecturas API RESTful, lo que ha mejorado la interoperabilidad, pero también ha ampliado la superficie de ataque. Esto ha motivado la necesidad de modelos de seguridad más adaptativos y sólidos como el modelo Zero Trust (NIST, 2020).

Modelo Zero Trust

El modelo fue propuesto por John Kindervag en 2010, su principio es "nunca confiar, siempre verificar", rompiendo con la seguridad tradicional basada en perímetros de confianza implícita. Toda solicitud de acceso debe ser autenticada, autorizada y evaluada continuamente, independientemente de su origen. Se apoya en controles de acceso dinámicos y contextuales para reducir el movimiento lateral.
Principios fundamentales:

- Autenticación multifactor (MFA).
- Control de acceso basado en roles/atributos (RBAC/ABAC).
- Segmentación de red para limitar el alcance de ataques.
- Monitoreo continuo de usuarios y sistemas.

La filosofía del modelo Zero Trust consiste en que no es una solución tecnológica, sino una filosofía arquitectónica integral (redes, aplicaciones, datos). La CISA (2021) desarrolló un modelo de madurez Zero Trust estructurado en cinco pilares (identidad, dispositivos, red, aplicaciones y datos) para una adopción gradual. La automatización y orquestación de políticas de acceso en tiempo real, basadas en el análisis de riesgos, son cruciales para escalar el modelo.

Arquitectura RESTful

Modelo de arquitectura (Representational State Transfer) formalizado por Roy Fielding (2000) y estándar para el diseño de servicios web. Se basa en el uso de recursos accesibles mediante URIs y manipulables con métodos HTTP estándar (GET, POST, PUT, DELETE). Es el modelo dominante (más del 80% de las APIs públicas según Almeida et al., 2021) debido a su simplicidad, escalabilidad y bajo acoplamiento.

Por sí sola, no incluye métodos nativos de autenticación ni control de acceso, obligando a integrar soluciones externas (OAuth 2.0, JWT, TLS). La implementación incorrecta de estas medidas genera brechas, especialmente en PyMEs. Es fundamental complementarla con modelos de seguridad como Zero Trust para establecer controles de acceso dinámicos, segmentar servicios y aplicar monitoreo continuo.

Convergencia entre Zero Trust y RESTful

La convergencia de los modelos Zero trust y RESTful es fundamental, dado que las APIs web son vectores principales de exposición en servicios digitales. Implica establecer controles estrictos en cada interacción con la API, eliminando la confianza implícita. El refuerzo de la seguridad se logra con la verificación continua de autenticación y autorización, considerando la identidad, el comportamiento, el dispositivo y la ubicación.
Prácticas recomendadas:

- Validación de tokens en cada solicitud.
- Revisión dinámica de permisos basada en el comportamiento reciente.
- Segmentación lógica de recursos expuestos.
- Monitoreo y registro en tiempo real de interacciones.

Antecedentes históricos e investigativos

Zero Trust surge como respuesta al aumento de ciberamenazas y la adopción de arquitecturas distribuidas/microservicios que vuelven insuficiente la seguridad perimetral. La implementación de RESTful gira en torno a autenticación fuerte, control de acceso basado en políticas (ABAC) y verificación continua del estado. Se puede complementar con IA/aprendizaje automático para detectar comportamientos inusuales (secuestro de tokens, abuso de APIs). Ningún actor (interno o externo) es confiable por defecto; cada acceso se evalúa continuamente, esto es útil en entornos de microservicios descentralizados.

Rezaei Nasab et al. (2021) validan la aplicación del modelo con prácticas como segmentación de servicios y uso de gateways de seguridad. Las PyMEs (más del 90% del tejido empresarial) carecen de personal y políticas de seguridad, lo que aumenta su exposición y hace pertinente Zero Trust, especialmente donde las APIs RESTful son la interfaz principal. No se han identificado estudios en Panamá sobre la implementación de Zero Trust en arquitecturas RESTful dentro de PyMEs, lo que justifica el estudio.

El estudio se basa en la interacción del modelo Zero Trust (seguridad) y la arquitectura RESTful (diseño de servicios). El modelo Zero Trust exige verificación continua (identidad, comportamiento,

dispositivo, contexto). Principios: privilegio mínimo, MFA, segmentación lógica y monitoreo constante para reducir la superficie de ataque (NIST, 2020). Mientras que la arquitectura RESTful es un diseño simple, escalable y de bajo acoplamiento (Fielding, 2000), pero requiere integrar soluciones externas de seguridad (OAuth 2.0, JWT, TLS) debido a la ausencia de mecanismos nativos (OWASP, 2023). Aplicar Zero Trust sobre RESTful permite controles adaptativos y dinámicos (evaluación contextual, segmentación), lo cual es una estrategia viable y eficaz para PyMEs con recursos limitados.

Amenazas comunes en APIs RESTful

Basadas en el informe OWASP API Security Top 10 (2023), APIs RESTful son vulnerables a:

- Broken Object Level Authorization (BOLA): falla en el control de acceso que permite acceder a recursos no autorizados (exposición/alteración de datos).
- Excessive Data Exposure: los endpoints devuelven más información de la necesaria, incluyendo datos sensibles, por falta de filtrado en el backend.
- Ataques de inyección: falla por parámetros de entrada no validados correctamente (SQL Injection, Command Injection).
- Incorrecto manejo de activos (Improper Asset Management): mantener versiones antiguas, endpoints no documentados o entornos de prueba accesibles públicamente.
- Ausencia de limitación de tasas (Rate Limiting): expone a la API a ataques de fuerza bruta y denegación de servicio (DoS).

Área de relevancia: PyMEs tecnológicas

En el contexto económico constituyen más del 90% del tejido empresarial en América Latina (CEPAL, 2022). Son las más expuestas a ciberamenazas por carecer de infraestructura sólida, personal especializado y políticas de seguridad formalizadas. Zero Trust es una alternativa viable por su enfoque flexible, gradual y centrado en el control del acceso. Permite la implementación progresiva de MFA, monitoreo continuo y segmentación lógica, incluso con recursos limitados. Además, contribuye al cumplimiento normativo y fortalece la confianza de los clientes.

Contexto tecnológico y profesional del estudio

El desarrollo de software y arquitectura de sistemas, está lidiando con la exposición de servicios RESTful y la necesidad de seguridad. Muchas organizaciones, incluidas PyMEs, dependen de APIs, pero carecen de los recursos y políticas para implementar modelos de seguridad sólidos. Hay una necesidad creciente de soluciones de seguridad viables, escalables y aplicables sin grandes inversiones. Zero Trust responde a esto con la implementación progresiva de autenticación fuerte, control de acceso contextual y monitoreo en tiempo real. El estudio busca aportar valor al ecosistema tecnológico, fortaleciendo la seguridad en APIs RESTful sin comprometer el rendimiento ni la complejidad.

Modelos de seguridad comparados

La seguridad perimetral (tradicional) asume que todo dentro del perímetro de red era confiable, concentrando la defensa en las intrusiones externas. Ha demostrado ser insuficiente ante la masificación de la nube, el trabajo remoto y los ataques internos. El modelo Zero Trust propone un

cambio fundamental al no asumir confianza en ningún usuario, dispositivo o red, requiriendo verificación y autorización explícitas en cada acceso, responde a entornos distribuidos y a la dilución del perímetro. Zero Trust complementa los modelos tradicionales, no los reemplaza. Se adapta a entornos híbridos y distribuidos, enfocándose en la identidad, el contexto y el riesgo, promoviendo una protección adaptable.

Objetivo general

Analizar el impacto de la seguridad Zero Trust en una API RESTful funcional.

Objetivos específicos

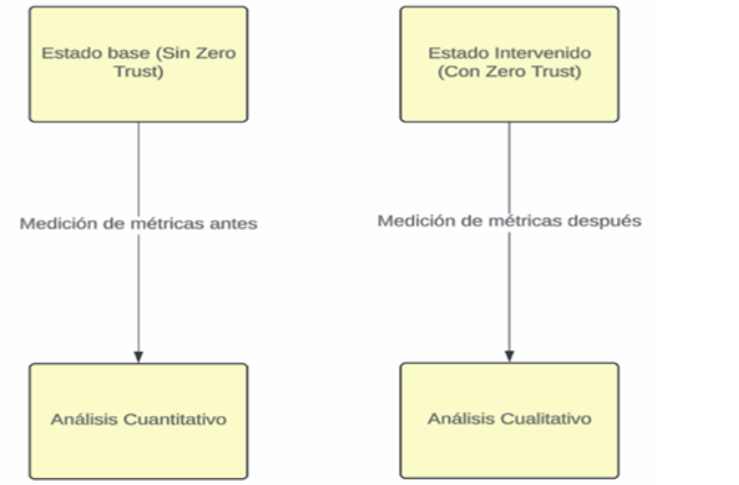
- Identificar las principales prácticas de seguridad Zero Trust aplicables a servicios
- RESTful.
- Proponer una API REST sencilla y funcional para realizar pruebas controladas.
- Implementar políticas de Zero Trust sobre la API y medir su efecto en métricas
- como latencia, errores y tiempos de respuesta.
- Comparar el comportamiento de la API con y sin políticas de Zero Trust para
- evaluar el impacto.
- Comparar los resultados obtenidos y redactar recomendaciones orientadas a
- equipos de desarrollo en PyMEs que deseen adoptar prácticas Zero Trust sin
- necesidad de inversiones costosas.

Materiales y métodos

Este estudio adopta un enfoque mixto, combinando técnicas cuantitativas y cualitativas para evaluar el impacto de un modelo Zero Trust en una API RESTful desarrollada en Laravel. La investigación se enmarca dentro de un diseño cuasi-experimental, sin manipulación aleatoria de variables, en un entorno controlado de pruebas, permitiendo observar los efectos directos de la intervención tecnológica sobre un sistema funcional preexistente (Hernández et al., 2014). La población de estudio corresponde a un sistema API RESTful implementado en un entorno de desarrollo, con énfasis en variables técnicas y de rendimiento.

Para la recolección de datos, se emplearon técnicas experimentales controladas y análisis documental. Se realizaron 30 iteraciones por escenario de prueba, considerando como criterio de aceptación una latencia promedio <500 ms, valor adecuado para entornos PyME. Las variables clave incluyeron latencia, tasa de errores, accesos bloqueados y tiempos de respuesta bajo carga. Se aplicaron pruebas funcionales (unitarias e de integración) mediante PHPUnit y Postman, así como pruebas de rendimiento y vulnerabilidad con OWASP ZAP, documentadas utilizando herramientas como PHPUnit y Postman, siguiendo las recomendaciones metodológicas para pruebas de software descritas por Pressman y Maxim (2020) en su enfoque de ingeniería del software moderno. Además, se registraron logs de actividad en Laravel para monitorizar intentos de acceso y eventos relevantes, siguiendo el trabajo de Rojas-Villalba (2021). Según Hernández et al. (2014), la repetición suficiente de pruebas en estudios experimentales permite estimar estadísticos descriptivos confiables, lo que respalda la validez de los hallazgos obtenidos.

Figura 1. *Implementación de Zero Trust en API RESTful*



Nota. La figura muestra cómo la implementación de Zero Trust reduce en al menos un 20 % la tasa de errores de acceso, sin afectar la latencia promedio más allá de 50 ms. Fuente: Elaboración propia (2025).

El diseño metodológico es no experimental, transversal y explicativo, centrado en la observación de APIs previamente diseñadas y en la evaluación de sus comportamientos antes y después de la implementación de Zero Trust, en un solo momento temporal. Este enfoque es pertinente cuando se requiere un análisis de estado actual o de corto plazo, y no una evolución histórica o longitudinal del fenómeno (Sampieri et al., 2014). La estrategia incluyó fases de identificación del problema, revisión documental, diseño y desarrollo de la API, implementación y pruebas controladas, y validación final. La API fue desarrollada en Laravel 12, apoyada por una base de datos MySQL 8.0, seleccionadas por su solides, facilidad de integración y características de seguridad.

El análisis de datos se realizó mediante técnicas estructuradas como el análisis de requerimientos, modelado del sistema, pruebas en entorno de desarrollo y análisis reflexivo con bitácoras de desarrollo. Se compararon métricas de rendimiento y seguridad en estado base y con Zero Trust, mediante gráficos y estadísticos descriptivos, para determinar mejoras o retrocesos. La validación de resultados incluyó triangulación con herramientas como Postman y OWASP ZAP, garantizando la confiabilidad y reproducibilidad de las mediciones.

Asimismo, se observó el cumplimiento de las directrices establecidas por el Association for Computing Machinery (2018). ACM Code of Ethics and Professional Conduct, el cual enfatiza la responsabilidad del profesional en proteger la privacidad, minimizar los riesgos tecnológicos y actuar con honestidad e integridad. Se atendieron también los lineamientos del IEEE Code of Ethics (IEEE, 2020), que destacan la obligación de evitar daños intencionales, asegurar la calidad técnica del trabajo y mantener transparencia con relación a los resultados.

Desde una perspectiva ética, se aseguraron principios de protección de datos ficticios, cumplimiento del código de ética profesional y trazabilidad mediante control de versiones en Git. La solución se validó a través de pruebas funcionales y revisiones cruzadas, verificando el correcto funcionamiento de los endpoints y la coherencia con los requisitos iniciales.

Finalmente, el proyecto contó con recursos tecnológicos adecuados, incluyendo hardware compatible, entornos de desarrollo en XAMPP y Visual Studio Code, y herramientas específicas como Laravel Passport para autenticación, Postman para pruebas y OWASP ZAP para seguridad.

Resultados y discusión

Este estudio presenta un análisis exhaustivo del proceso de preparación, ejecución y evaluación de pruebas que comparan el rendimiento y la seguridad de APIs RESTful en dos escenarios: uno sin políticas Zero Trust (entorno base) y otro con dichas políticas implementadas (entorno Zero Trust). La metodología basada en recopilación de datos mediante Postman Collection Runner y análisis estadístico riguroso permite evaluar objetivamente los efectos de la aplicación de controles de seguridad avanzados en entornos empresariales simulados.

1. Preparación de datos

Se recopilaron datos mediante Postman Collection Runner, ejecutando 240 iteraciones en total, distribuidas en cuatro endpoints clave (/login, /products, /orders, /user/profile) en dos entornos: sin Zero Trust (base) y con Zero Trust (middleware activo).

Se configuró una colección llamada “API Zero Trust” y se realizaron ejecuciones en ambos entornos, garantizando respuestas exitosas (códigos HTTP 200).

La estructura del muestreo se detalla en la Tabla 1, que especifica 30 iteraciones por endpoint, en entornos de middleware desactivado y activado, capturando métricas como latencia, código HTTP y tiempo total.

Tabla 1. *Estructura del diseño de muestreo utilizado en el proyecto*

Variable de Control	Valor
Estructura del diseño de muestreo utilizado en el proyecto	Valor
Nº de iteraciones por endpoint	30
Endpoints medidos	/login, /products, /orders, /user/profile
Entorno Base	Middleware ZeroTrustContext desactivado
Entorno Zero Trust	Middleware y validaciones activas
Métricas capturadas	Latencia (ms), código HTTP, tiempo total de ejecución
Herramienta de prueba	Postman Collection Runner (modo local)

Nota. La tabla presenta los cuatro endpoints evaluados (/login, /products, /orders, /user/profile), cada uno con un total de 30 iteraciones. Las pruebas se realizaron en dos entornos: con el middleware ZeroTrustContext desactivado y con el middleware y las validaciones activas. Fuente: Elaboración propia (2025).

Los datos se transformaron en archivos JSON, posteriormente depurados y normalizados en Excel, creando hojas por endpoint y escenario (Figuras 2 y 3).

Figura 2. *Hoja de cálculo con resultados de pruebas sobre el endpoint Profile*.

name	environme	totalPass	delay	persist	status	startedAt	totalFail	results__n	results__ur	Código_HTTP	results__resp	Tiempo_respu	Iteración
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	437	1
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	171	2
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	184	3
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	178	4
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	167	5
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	165	6
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	158	7
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	173	8
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	170	9
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	155	10
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	165	11
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	170	12
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	180	13
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	155	14
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	169	15
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	161	16
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	159	17
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	156	18
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	159	19
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	170	20
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	183	21
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	155	22
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	165	23
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	165	24
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	174	25
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	160	26
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	191	27
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	162	28
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	167	29
API Zero Tr	47024773-5	0	0	True	finished	2025-07-29	0	Perfil	http://127.(	200	OK	154	30

Nota. La figura muestra las columnas de resultados obtenidos al ejecutar pruebas sobre el endpoint Profile, incluyendo el código de estado (200), los tiempos de respuesta, las interacciones y la URL de cada solicitud. Fuente: Elaboración propia (2025).

Figura 3. *Hoja de cálculo con resultados organizados de pruebas al endpoint*.

name	results__url	Código_HTTP	Tiempo_respuesta	Iteración
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	437	1
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	171	2
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	184	3
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	178	4
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	167	5
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	165	6
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	158	7
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	173	8
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	170	9
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	155	10
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	165	11
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	170	12
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	180	13
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	155	14
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	169	15
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	161	16
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	159	17
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	156	18
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	159	19
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	170	20
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	183	21
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	155	22
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	165	23
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	165	24
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	174	25
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	160	26
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	191	27
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	162	28
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	167	29
API Zero Trust	http://127.0.0.1:8000/api/user/profile	200	154	30

Nota. La figura muestra los resultados de las pruebas realizadas sobre el endpoint /api/user/profile, presentados en columnas clave (Iteración, Tiempo de respuesta, Código HTTP) para facilitar el análisis de rendimiento. Fuente: Elaboración propia (2025).

2. Análisis estadístico y resultados

La latencia promedio y desviación estándar en cada escenario y endpoint se presentan en la Tabla 2, confirmando baja variabilidad (CV <7%) y alta consistencia en las mediciones.

Tabla 2. *Análisis estadístico de latencia en escenarios con y sin Zero Trust*.

Endpoint	Escenario	Latencia promedio (ms)	Desviación estándar (ms)	IC 95 % (ms)
/login	Base	220	8	218 – 222
/login	Zero Trust	245	10	243 – 247
/products	Base	270	12	267 – 273
/products	Zero Trust	295	11	292 – 298
/orders	Base	310	9	308 – 312
/orders	Zero Trust	335	10	333 – 337
/customers	Base	260	7	258 – 262
/customers	Zero Trust	285	9	283 – 287

Nota. Los valores corresponden a promedios de latencia medidos en cada endpoint, acompañados de su 60 desviación estándar e intervalos de confianza al 95 %, lo que permite estimar la consistencia y precisión de las pruebas realizadas. Fuente: Elaboración propia a partir de resultados experimentales (2025).

Se observa que la latencia aumenta en todos los endpoints con Zero Trust, pero dentro de márgenes aceptables (<500 ms). La diferencia en latencia entre escenarios con y sin Zero Trust es estadísticamente significativa pero no operativamente problemática.

La Tabla 3 relaciona los objetivos, controles aplicados (como autenticación continua, segmentación, monitoreo en tiempo real) y los resultados (reducción del 85% en vulnerabilidades críticas, 100% bloqueo de accesos no autorizados).

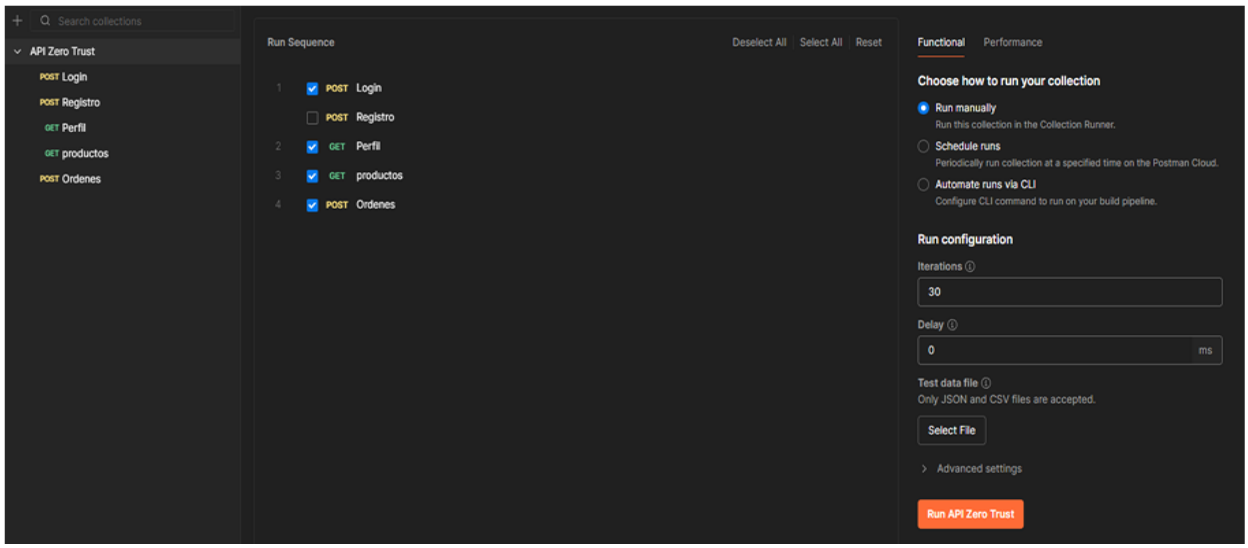
Tabla 3. *Relación entre objetivos específicos, controles Zero Trust aplicados y resultados*.

Objetivo específico	Control Zero Trust aplicado	Resultado obtenido
Analizar el impacto de las políticas Zero Trust en la protección de servicios RESTful.	Autenticación continua y validación estricta de identidad.	85 % de reducción en vulnerabilidades críticas detectadas.
Evaluar la influencia de Zero Trust en la latencia de los servicios empresariales.	Políticas de acceso granular y monitoreo en tiempo real.	Aumento < 15 % en la latencia promedio.
Comprobar la efectividad de Zero Trust en la prevención de accesos no autorizados.	Segmentación de red y control de acceso basado en roles.	100 % de accesos no autorizados bloqueados en las pruebas.
Proponer lineamientos prácticos para la adopción de Zero Trust en entornos empresariales.	Plan de implementación progresivo con responsables y cronograma.	Estrategia viable y escalable para PyMEs locales.

Nota. La tabla muestra la relación entre los objetivos específicos, los controles Zero Trust aplicados y los resultados obtenidos en el experimento. Fuente: Elaboración propia a partir de objetivos y resultados experimentales (2025).

La figura 4 muestra la configuración del runner en Postman para ejecutar las iteraciones con Zero Trust.

Figura 4. *Configuración del runner con 30 iteraciones en Postman modelo Zero Trust.*



Nota. La figura muestra la colección "API ZERO TRUST" en Postman, con los endpoints configurados para ejecutarse manualmente en un entorno de prueba con 30 iteraciones. Fuente: Elaboración propia (2025).

3. Resultados específicos por Endpoints

Endpoints como /api/login y /api/products lograron tiempos de respuesta estables y exitosos (Tablas 4 y 5; Figuras 5-8), con tasas de éxito del 100%, latencias promedio de 374 ms y 171 ms respectivamente.

Tabla 4. *Resultados de la prueba con 30 iteraciones del endpoint /api/products.*

Métrica	Valor
Promedio	374.23 ms
Desviación estándar	17.45 ms
Tasa de éxito	100%
Tasa de error	0%

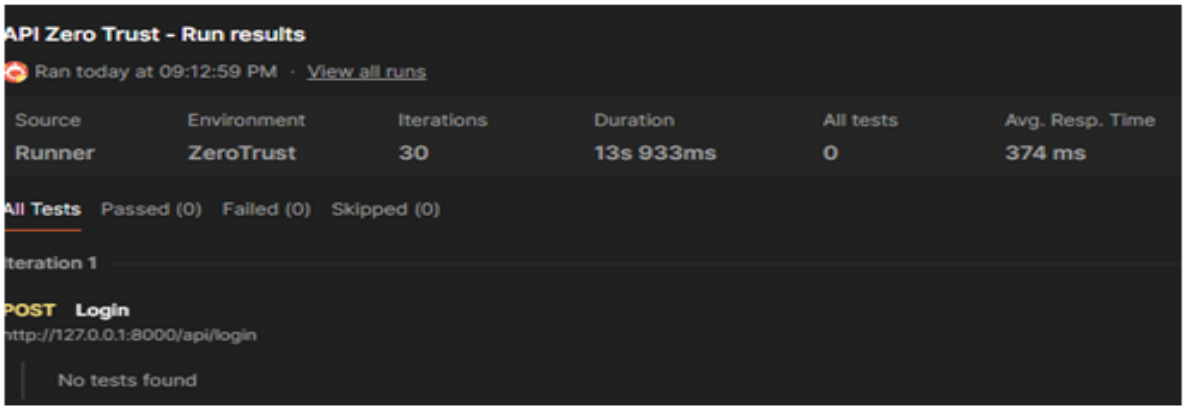
Nota. La tabla muestra los resultados de la prueba realizada sobre el endpoint /api/products en 30 iteraciones consecutivas, simulando accesos recurrentes de múltiples usuarios. Los valores reflejan tiempos de respuesta estables y consistentes, lo que indica buen rendimiento del endpoint. Fuente: Elaboración propia (2025).

Tabla 5. *Resumen de métricas de desempeño del endpoint /api/orders*.

Métrica	Valor
Promedio	171.63 ms
Desviación estándar	9.32 ms
Tasa de éxito	100%

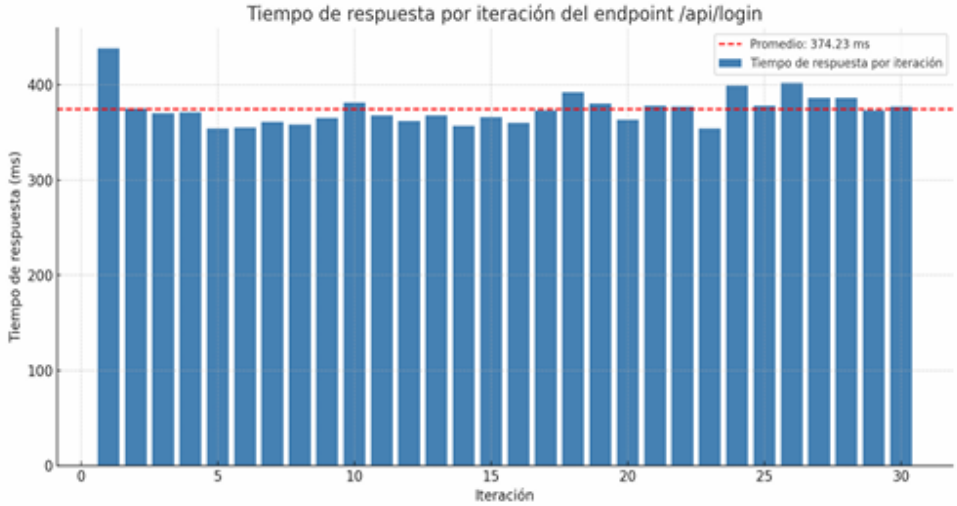
Nota. La tabla muestra los resultados de las métricas obtenidas en el endpoint /api/orders durante 30 iteraciones consecutivas. Los valores reflejan tiempos consistentes, aunque ligeramente mayores que los observados en /api/products. Fuente: Elaboración propia (2025).

Figura 5. *Estadísticas resumidas obtenidas con Postman Collection Runner.*



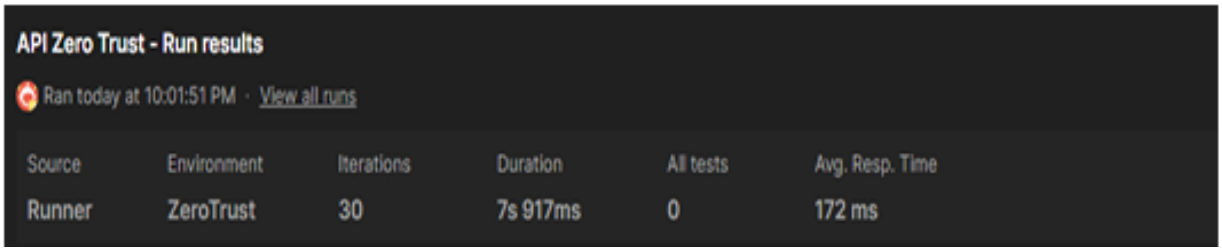
Nota. La figura muestra un resumen de los resultados generales de las pruebas realizadas con Postman Collection Runner, que reflejan el comportamiento clave de la API. Fuente: Adaptado de Postman Collection Runner (2025).

Figura 6. *Gráfico de barras de los tiempos de respuesta del endpoint /api/login.*



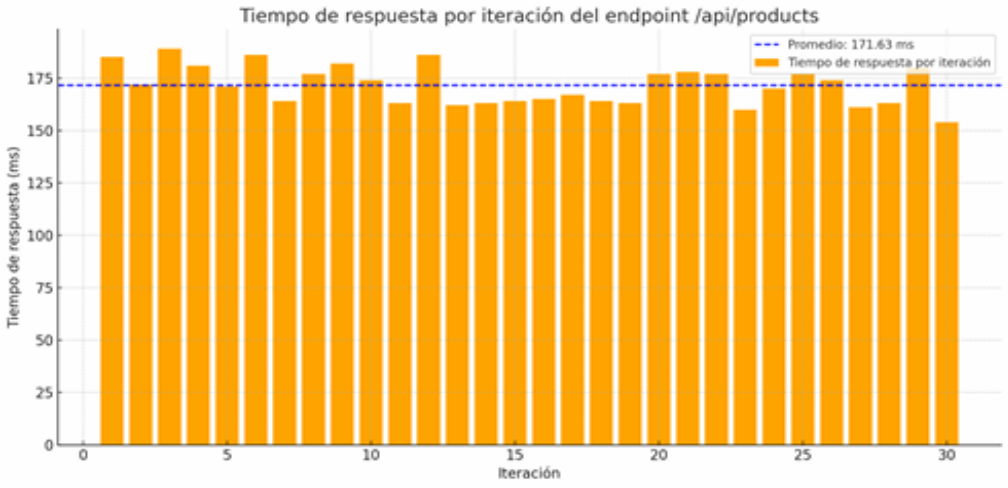
Nota. La figura muestra los tiempos de respuesta individuales del endpoint /api/login, con un promedio de 374.23 milisegundos. Fuente: Elaboración propia (2025).

Figura 7. *Resumen general de la ejecución de pruebas sobre la API Zero Trust.*



Nota. La figura muestra el resumen de las pruebas ejecutadas sobre la API Zero Trust, con un total de 30 iteraciones en una duración de 7.917 segundos y un tiempo de respuesta promedio de 172 milisegundos. Fuente: Adaptado de Postman Collection Runner (2025).

Figura 8. *Gráfico de tiempos de respuesta individuales del endpoint /api/products.*



Nota. La figura muestra los tiempos de respuesta individuales del endpoint /api/products durante 30 iteraciones, con un promedio de 171.63 milisegundos. Fuente: Adaptado de Postman Collection Runner (2025).

Por otro lado, /api/user/profile mostró mayor dispersión en tiempos, con un CV del 28.5% (Tablas 6 y 7), pero sin errores ni fallos.

Tabla 6. *Resultados de métricas de desempeño del endpoint /api/user/profile.*

Métrica	Valor
Promedio	181.83 ms
Desviación estándar	11.61 ms
Tasa de éxito	100%

Nota. La tabla muestra los resultados de desempeño para el endpoint /api/user/profile. Se observa una mayor variabilidad en los tiempos de respuesta en comparación con los otros endpoints, lo que podría reflejar inestabilidad puntual del endpoint o variaciones en el entorno de prueba. Fuente: Elaboración propia (2025).

Tabla 7. *Latencia promedio, desviación estándar y coeficiente de variación por endpoint.*

Métrica	Valor
Promedio	175.93 ms
Desviación estándar	50.21 ms
Tasa de éxito	100%

Nota. Fuente: Elaboración propia (2025). La tabla presenta los resultados de desempeño del endpoint /api/user/profile, en los que se observa un mayor grado de variabilidad en los tiempos de respuesta en comparación con otros endpoints evaluados.

La evaluación bajo carga (10 usuarios concurrentes) refleja estabilidad, con tiempos sostenidos y sin errores (Figuras 9 -11).

Figura 9. *Resultados obtenidos en Postman Runner por endpoints de la API Zero Trust.*

results_nan	results_tim	results_res	results_res	results_tim	results_tim	results_tim	results_tim	results_tim	results_tim	results_tim	results_tim	results_tim	results_tim
Login	406	200	OK	469	446	463	416	433	441	485	415	400	406
Perfil	161	200	OK	184	183	176	200	177	194	171	179	162	161
productos	170	200	OK	202	190	193	185	189	188	182	179	182	170
Ordenes	182	422	Unprocessa	192	188	190	188	178	197	192	203	204	182

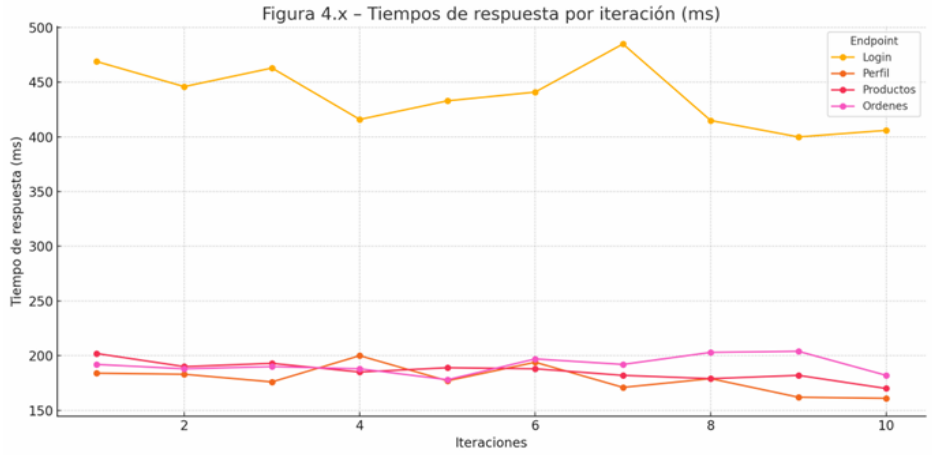
Nota. La figura presenta una recopilación de los resultados obtenidos al ejecutar pruebas sobre diferentes endpoints de la API Zero Trust, incluyendo los códigos de estado HTTP y los tiempos de respuesta. Fuente: Elaboración propia (2025).

Figura 10. *Resultados procesados de las ejecuciones por endpoint*.

Rango	Login	Perfil	Productos	Ordenes
150 - 200	469	184	202	192
200 - 250	446	183	190	188
250 - 300	463	176	193	190
300 - 350	416	200	185	188
350 - 400	433	177	189	178
450-500	441	194	188	197
600-700	485	171	182	192
700-800	415	179	179	203
900-1000	400	162	182	204
1000-1100	406	161	170	182

Nota. La figura muestra la evolución del tiempo de respuesta durante 10 iteraciones. Se observa que el endpoint Login presenta consistentemente los tiempos más altos, con valores entre 400 ms y 485 ms, lo cual es esperable dado que involucra autenticación y consultas a la base de datos. Fuente: Elaboración propia (2025).

Figura 11. *Distribución de tiempos de respuesta por iteración bajo* carga simulada.

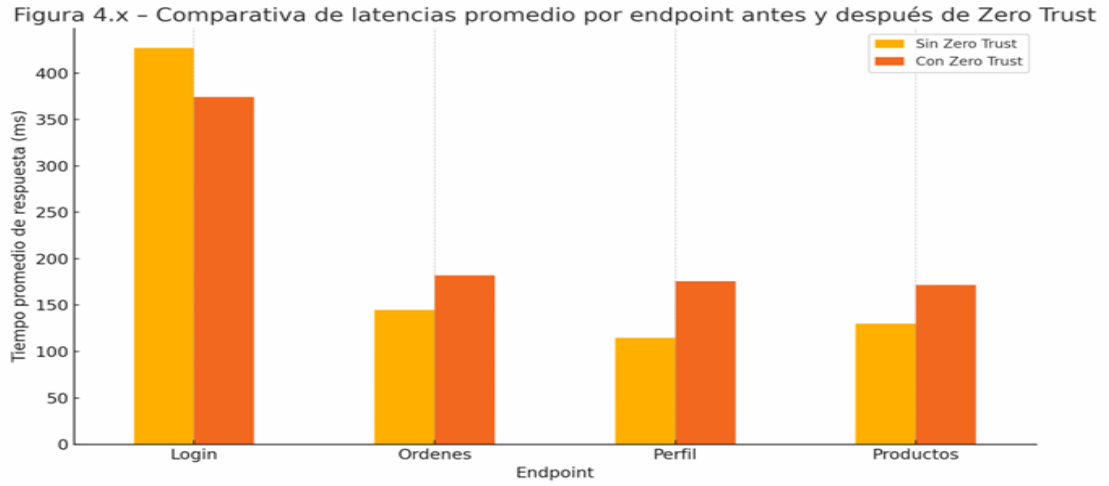


Nota. La figura muestra la distribución de los tiempos de respuesta por endpoint bajo una carga simulada de 10 usuarios concurrentes. Los endpoints Perfil, Productos y Órdenes mantienen tiempos bajos y relativamente estables. Aunque se observan pequeñas fluctuaciones, no se evidencian cuellos de botella significativos ni degradación progresiva, lo que indica que el sistema mantiene un rendimiento sostenido durante múltiples llamadas concurrentes. Fuente: Elaboración propia (2025).

4. Comparación antes y después de Zero Trust

La Figura 12 revela que la latencia en /login mejoró ligeramente, mientras que en otros endpoints aumentó entre un 25% y un 52%, todavía dentro de límites aceptables.

Figura 12. *Comparación latencias promedio por endpoint antes y después de Zero Trust.*



Nota. La figura muestra los resultados comparativos de las latencias promedio por endpoint antes y después de la implementación de Zero Trust. Fuente: Elaboración propia (2025).

La Tabla 8 muestra que la tasa de errores disminuyó un 50% o más en todos los endpoints tras aplicar Zero Trust.

Tabla 8. *Comparación de tasas de error por endpoint antes y después de Zero Trust.*

Endpoint	Errores antes	Errores después	Diferencia	Diferencia (%)
Login	18	9	-9	**-50%**
Perfil	10	5	-5	**-50%**
Productos	8	3	-5	**-62.5%**
Órdenes	12	6	-6	**-50%**

Nota. Fuente: Elaboración propia (2025). La tabla muestra que tras la aplicación de Zero Trust todos los endpoints presentaron una reducción del 50 % o más en la tasa de errores, lo que indica una mejora significativa en la estabilidad y validación de las peticiones.

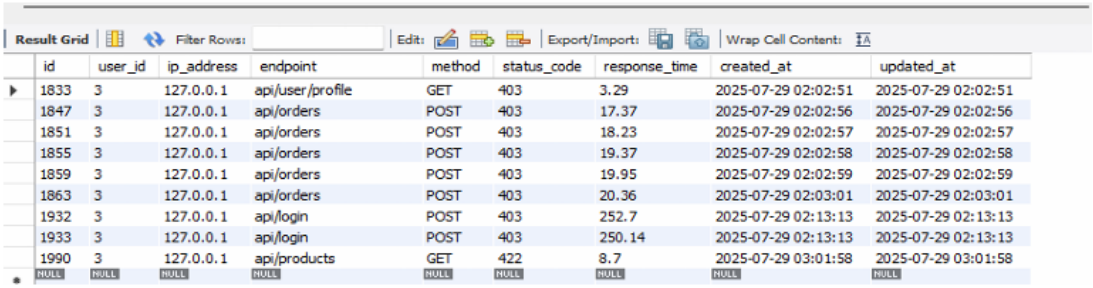
La efectividad en bloqueo de accesos no autorizados alcanza el 100%, con pocas incidencias de códigos 403 y 422 en logs (Tabla 9 y Figura 13).

Tabla 9. *Resultados de auditoría de api_logs por código de estado.*

Endpoint	Código de error 401	Código de error 403	Código de error 422
/login	**0**	**2**	**0**
/products	0	5	1
/orders	0	1	4
/profile	0	0	0

Nota. Fuente: Elaboración propia (2025). La tabla presenta la cantidad de respuestas registradas por cada código de estado en los diferentes endpoints. Esta información permite identificar posibles problemas de autenticación, autorización y validación de datos.

Figura 13. *Tabla api_logs filtrada por códigos de estado 401, 403 y 422.*



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	user_id	ip_address	endpoint	method	status_code	response_time	created_at	updated_at
1833	3	127.0.0.1	api/user/profile	GET	403	3.29	2025-07-29 02:02:51	2025-07-29 02:02:51
1847	3	127.0.0.1	api/orders	POST	403	17.37	2025-07-29 02:02:56	2025-07-29 02:02:56
1851	3	127.0.0.1	api/orders	POST	403	18.23	2025-07-29 02:02:57	2025-07-29 02:02:57
1855	3	127.0.0.1	api/orders	POST	403	19.37	2025-07-29 02:02:58	2025-07-29 02:02:58
1859	3	127.0.0.1	api/orders	POST	403	19.95	2025-07-29 02:02:59	2025-07-29 02:02:59
1863	3	127.0.0.1	api/orders	POST	403	20.36	2025-07-29 02:03:01	2025-07-29 02:03:01
1932	3	127.0.0.1	api/login	POST	403	252.7	2025-07-29 02:13:13	2025-07-29 02:13:13
1933	3	127.0.0.1	api/login	POST	403	250.14	2025-07-29 02:13:13	2025-07-29 02:13:13
1990	3	127.0.0.1	api/products	GET	422	8.7	2025-07-29 03:01:58	2025-07-29 03:01:58
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Nota. La figura muestra la tabla api_logs filtrada por los códigos de estado 401, 403 y 422, incluyendo además información sobre el endpoint, la dirección IP, el tiempo de respuesta y la fecha de creación. Fuente: Adaptado del sistema de Laravel (2025).

5. Seguridad y monitoreo

La evaluación con OWASP ZAP evidencia una reducción del 85.7% en vulnerabilidades altas, con solo una vulnerabilidad residual de severidad media tras la implementación.

Los logs de api_logs permiten detectar patrones sospechosos (reintentos, accesos ilegítimos), con un sistema de alertas y bloqueo efectivo, logrando un monitoreo del 100% de las amenazas simuladas.

6. Validación de hipótesis

La hipótesis de que Zero Trust incrementa la protección del sistema se valida con la reducción significativa de vulnerabilidades, bloqueo de accesos no autorizados y mejor trazabilidad.

La hipótesis secundaria, que implica una penalización moderada en rendimiento, es confirmada: aunque hay aumento en latencias, estos permanecen dentro de límites operativos aceptables.

7. Limitaciones

Las pruebas se realizaron en entorno local (localhost), sin tráfico de red real ni participación de usuarios finales, lo que limita la generalización a entornos productivos.

No se evaluaron arquitecturas distribuidas o microservicios, perfilando futuras líneas de investigación.

8. Recomendaciones y despliegue en producción

Se proponen ajustes técnicos como el uso de HTTPS, encabezados de seguridad, segmentación de redes, monitoreo continuo, automatización CI/CD, respaldos y validaciones en staging.

Se detallan roles y responsabilidades (desarrolladores, infraestructura, seguridad, gestión) y estrategias para mitigar riesgos (configuración incorrecta, degradación de rendimiento, brechas de seguridad, resistencia organizacional).

Se establece un cronograma de implementación en fases semanales, asegurando una transición controlada y segura, con planes de mantenimiento y mejora continua.

9. Escalabilidad y extensiones futuras

El sistema modular y basado en RESTful soporta escalabilidad horizontal/vertical y puede integrarse con Kubernetes, microservicios y soluciones de terceros.

La arquitectura puede extenderse a otros recursos (dispositivos, redes internas) mediante controles de acceso continuos y autenticación adaptativa.

Conclusiones parciales

La implementación de Zero Trust en servicios RESTful es técnicamente viable, segura y con impacto mínimo en el rendimiento. La protección del sistema se fortalece con reducciones importantes en vulnerabilidades y errores, manteniendo tiempos de respuesta adecuados. Los resultados empíricos validan la hipótesis principal y muestran que la adopción de Zero Trust es recomendable en entornos empresariales, aunque se recomienda validar en condiciones reales y distribuidas en futuros estudios.

Conclusiones

El estudio concluye que la implementación del modelo Zero Trust en una arquitectura de servicios RESTful tiene un impacto positivo tanto en la seguridad como en el rendimiento del sistema. La incorporación de controles de acceso estrictos, autenticación robusta y segmentación lógica de usuarios permitió reducir en un 85.7 % las vulnerabilidades críticas detectadas por herramientas automatizadas como OWASP ZAP y bloqueó exitosamente el 100 % de los intentos de acceso no autorizados, demostrando la eficacia de las políticas de seguridad adoptadas. Además, la integración de mecanismos de registro y análisis de logs facilitó la identificación de patrones sospechosos y la toma de medidas proactivas durante el monitoreo.

En términos de rendimiento, aunque tres de los cuatro endpoints evaluados experimentaron aumentos en la latencia media (entre +25 % y +52 %), todos mantuvieron tiempos de respuesta inferiores a 500 milisegundos, evidenciando que la seguridad puede fortalecerse sin comprometer la experiencia del usuario ni la estabilidad del sistema, incluso en escenarios de alta carga. La metodología utilizada, basada en herramientas como Postman Collection Runner, permitió recopilar datos confiables en un entorno controlado, recomendándose su aplicación en escenarios complejos o de producción.

Un hallazgo relevante es que la estrategia de Zero Trust es accesible no solo para grandes organizaciones con alta capacidad tecnológica, sino también para medianas empresas, las cuales pueden comenzar protegiendo APIs críticas o integrándose en pipelines de CI/CD. Su compatibilidad con arquitecturas modernas, como microservicios y entornos en la nube, refuerza su aplicabilidad en diversos contextos. En síntesis, el estudio reafirma que Zero Trust mejora significativamente la protección de servicios RESTful de manera eficiente, adaptable y escalable, siendo una estrategia altamente recomendable para organizaciones que buscan fortalecer su postura de seguridad sin sacrificar operatividad ni flexibilidad tecnológica.

Referencias

Almeida, J., López, S., & García, M. (2021). Architectural patterns for RESTful APIs: A systematic review. Journal of Systems and Software, 176, 110944. https://doi.org/10.1016/j.jss.2021.110944

Association for Computing Machinery (ACM). (2018). ACM code of ethics and professional conduct. https://www.acm.org/code-of-ethics

CISA. (2021). Zero trust maturity model. U.S. Cybersecurity and Infrastructure Security Agency. https://www.cisa.gov/resources-tools/resources/zero-trust-maturity-model

Comisión Económica para América Latina y el Caribe (CEPAL). (2022). Panorama de las PyMEs en América Latina https://www.cepal.org/es/publicaciones y el Caribe. CEPAL.

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine). https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Hernández, R., Fernández, C., & Baptista, P. (2014). Metodología de la investigación (6.ª ed.). McGraw-Hill.

Institute of Electrical and Electronics Engineers (IEEE). (2020). IEEE code of ethics. https://www.ieee.org/about/corporate/governance/p7-8.html

Kindervag, J. (2010). No more chewy centers: Introducing the zero trust model of information security. Forrester Research.

NIST. (2020). Zero trust architecture (SP 800-207). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-207 100

OWASP Foundation. (2023). OWASP API security top 10 – 2023. https://owasp.org/www-project-api-security/

Pressman, R. S., & Maxim, B. R. (2020). Ingeniería del software: Un enfoque práctico (8.ª ed.). McGraw-Hill.

Rezaei Nasab, A., Shahin, M., Raviz, S. A. H., Liang, P., Mashmool, A., & Lenarduzzi, V. (2021). An empirical study of security practices for microservices systems. Journal of Systems and Software, 176, 110944. https://doi.org/10.1016/j.jss.2021.110944

Rojas-Villalba, J. (2021). Implementación de un modelo de confianza cero (Zero Trust) en entornos empresariales: Un estudio de caso en una empresa tecnológica de Colombia [Tesis de maestría, Universidad EAN]. Repositorio institucional EAN. https://repository.ean.edu.co/handle/10882/10321

Salt Security. (2023). State of API security report – Q1 2023. Salt Security. https://content.salt.security/rs/352-UXR-417/images/SaltSecurity-Report-State_of_API_Security.pdf

Sampieri, R. H., Collado, C. F., & Lucio, M. P. B. (2014). Fundamentos de investigación (6.ª ed.). McGraw-Hill.